User Manual

wireless LoRa 用户手册

V0.0.0 Date:2022/03/03

产品用户手册

类别	内容
关键词	wireless LoRa、通用接口、移植说明、应用说明、常见问题
摘　要	描述关于 wireless LoRa 产品软件通用接口的使用说明

修订历史

版本	日期	原因
发布 0.0.0	2022/3/3	创建文档

©2022 Guangzhou ZHIYUAN Electronics Co., Ltd.

目 录

1. 更改记录 …… 1
2. ZM4xx 软件通用接口概述 …… 1
2.1 概述 …… 1
2.2 结构框架 …… 2
2.3 SDK 软件包结构说明 …… 2
3. DEMO 板测试例程使用说明 …… 10
3.1 使用调试串口助手发送数据进行通信测试 …… 10
3.2 使用按键 SW1 发送数据进行通信测试 …… 11
3.3 使用按键 SW2 调节发射功率 …… 12
4. 无线驱动代码移植 …… 13
4.1 移植前注意事项 …… 13
4.2 文件的移植 …… 14
4.3 文件的修改 …… 14
5. 无线收发测试例程说明 …… 18
5.1 无线模块参数配置 …… 18
5.2 轮询接收测试例程说明 …… 19
5.3 中断接收测试例程说明 …… 21
6. radio API 介绍 …… 24
6.1 设置模式 …… 24
6.2 数据包空中时间获取 …… 24
6.3 发送数据 …… 25
6.4 接收数据 …… 26
6.5 设置频率 …… 26
6.6 复位模块 …… 27
6.7 设置前导码长度 …… 27
6.8 获取前导码长度 …… 28
6.9 设置功率 …… 28
6.10 获取运行状态 …… 29
6.11 无线模块控制 …… 30
7. 无线模块使用过程中常见问题总结 …… 31
7.1 客户板子发送，DEMO 板接收不到数据 …… 31
7.2 初始化失败，句柄返回为 NULL …… 31
7.3 通信距离短 …… 31
7.4 接收不到数据 …… 31
7.5 接收一次数据之后就再也接收不到了 …… 32

广州致远电子 wireless LoRa 无线模块

©2022 Guangzhou ZHIYUAN Electronics Co., Ltd.

7.6 使用 CAD 需要注意的问题 ····· 32
7.7 CAD 只能检测到 CAD Done 中断而没有 CAD Detect 中断 ····· 32
7.8 发现频率相差 1Mhz、5Mhz，CAD 还能产生 CAD Detect 标志 ····· 32
7.9 丢包问题 ····· 32
7.10 发送时间长 ····· 32
7.11 什么时候接收数据 ····· 33
7.12 A 发送 B 能接收，反之无法通信 ····· 33
7.13 开启了硬件 CRC 但是收到了错误的数据包 ····· 33
7.14 没有发送数据，LoRa 却收到了乱码 ····· 33
8. 免责声明 ····· 33

广州致远电子 wireless LoRa 无线模块

©2022 Guangzhou ZHIYUAN Electronics Co., Ltd.

1. 更改记录

1.1.3 <2022-02-23>

- 将原本的 Wireless_module-EVB 评估板工程目录名修改为 RF_module-EVB 评估板工程目录名
- 添加无线模块型号说明表格
- 更新各个评估板图片并添加评估板部件描述表格
- 完善代码移植部分内容
- 添加中断接收测试例程说明

1.1.2 <2021-09-02>

- 添加 ZSL64 系列和 ZSL42x 系列无线模块支持
- 添加自动或手动开启低数据速率优化功能

1.1.1 <2021-08-10>

- 添加设备型号选择
- 添加获取数据包空中时间接口使用说明

1.1.0 <2021-03-03>

- 添加 CPU 锁函数接口说明
- 修改操作流程介绍
- 添加用户 API 使用说明

1.0.0 <2020-09-10>

- 首次提交

2. ZM4xx 软件通用接口概述

2.1 概述

ZM4xx 软件通用接口可以使该系列的不同产品通过统一的一套软件接口操作设备，用户在更换产品时只需修改少量的应用层代码就可以直接使用，大大降低了用户重新开发软件的成本。用户使用该通用接口可以在不查看芯片手册的基础上完成模块收发数据的基本功能，大大降低了用户的学习成本和上手难度。ZM4xx 软件通用接口提供了模块收发数据，频率设置，发射功率设置和模式设置等等一些基本操作，通过这些操作就可以实现一些简单的收发功能。

广州致远电子 wireless LoRa 无线模块

©2022 Guangzhou ZHIYUAN Electronics Co., Ltd.

2.2 结构框架



图 2.1 ZM4xx 软件结构框图

ZM4xx 通用软件接口结构如 图 2.1 所示。用户可以直接在应用层调用 radio 层接口即可操作该系列无线模块。

2.3 SDK 软件包结构说明

wireless_lora_sdk v1.0.0 （v1.0.0 为 SDK 的版本号，以具体的版本号为准）SDK 包目录结构如 图 2.2 所示。

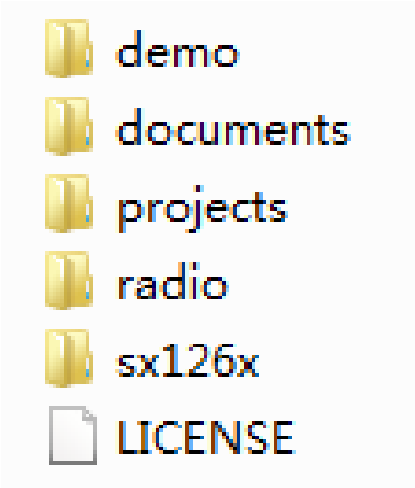


图 2.2 wireless lora SDK 包目录结构图

- demo 文件夹提供了一些简单的测试例程。

©2022 Guangzhou ZHIYUAN Electronics Co., Ltd.

- doccument 文件夹用于存放文档，如用户手册。
- projects 文件夹存放不同的评估板对应的 keil 工程文件，具体内容会在后续进行说明。
- radio 文件夹提供了操作无线模块的标准接口文件。
- sx126x 文件夹包含了无线模块的驱动文件。sx126x_radio_differ.h 定义了 sx126x 独有的功能。
- LICENSE 文件为 ZM4xx 软件免责声明

进入 wireless_lora_sdk v1.0.0\projects\wireless_lora 目录后，可以看到如 图 2.3 所示的几个目录，这几个目录分别对应不同评估板的 keil 工程。在打开 keil 工程之前，需要先确认手上的评估板的型号，根据评估板型号打开对应的 keil 工程，具体对应关系在下方给出。注意，假如使用的 keil 工程与所用的评估板型号不对应，可能会出现无线初始化失败而无法正常使用无线模块的情况。

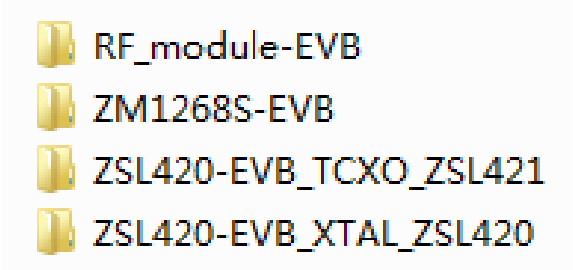


图 2.3 各个评估板 keil 工程

RF_module-EVB 目录对应的评估板如 图 2.4 所示，RF_module-EVB 评估板是为帮助用户快速上手 ZSL64 系列与 ZM68S 无线模块而开发的评估套件。ZSL64 系列与 ZM68S 无线模块各个型号说明见下表。

表 2.1 ZSL64 系列与 ZM68S 无线模块各个型号说明

芯片型号	工作频段	内置晶振
ZSL64A1ALHA	470MHz ~ 510MHz	XTAL
ZSL64A2ALHA	470MHz ~ 510MHz	TCXO
ZSL64B2ALHA	863MHz ~ 870MHz	XTAL
ZSL64B3ALHA	863MHz ~ 870MHz	TCXO
ZSL64C3ALHA	902MHz ~ 928MHz	XTAL
ZM68S	470MHz ~ 510MHz	XTAL

广州致远电子 wireless LoRa 无线模块

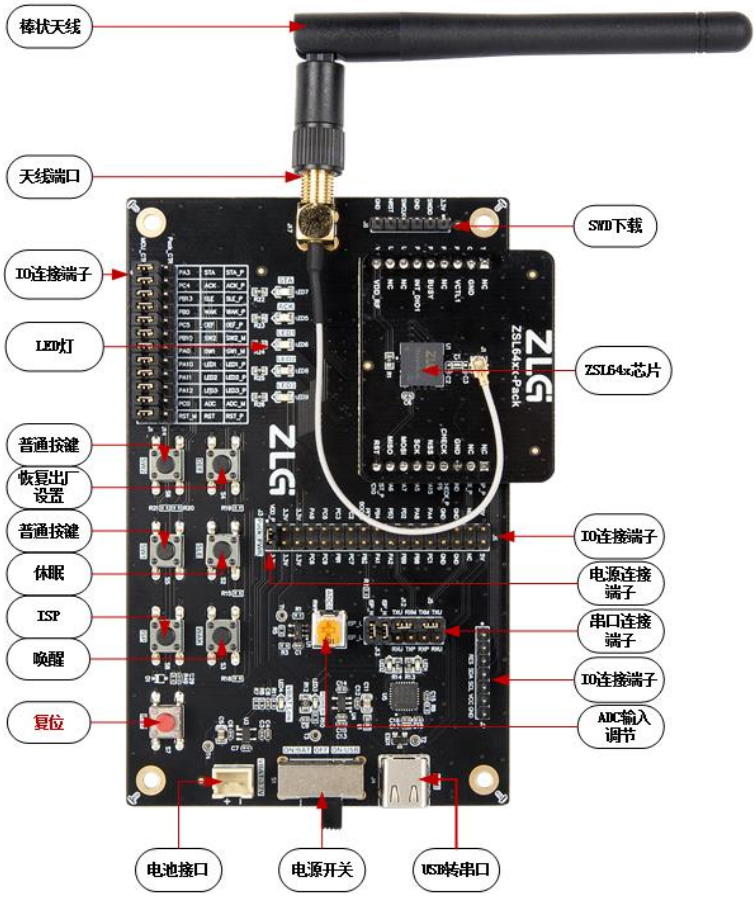


图 2.4 RF_module-EVB 评估板

所有的评估板的供电方式都有两种，USB 或者电池，正面有电池输入接口，评估板套件不带电池，需要客户自备电池。RF_module-EVB 评估板部件相关描述见下表。

广州致远电子 wireless 无线模块

©2022 Guangzhou ZHIYUAN Electronics Co., Ltd.

表 2.2 RF_module-EVB 评估板部件描述

部件	描述
电池接口	外部电池输入端
电源开关	选择电源供电方式：USB 或电池
USB 转串口	供电和 USB 转串口功能，用 USB 电缆直接连到电脑，电脑需要装相应的驱动，可用来收发数据
ADC 输入调节	通过电位器调节，使模块 ADC 采集到不同的电压值
串口连接端子	需要用短路器短接，否则电脑串口与芯片串口断开
电源连接端子	必须用短路器短接，否则 MCU 与 LoRa 模块无法供电
LED 灯	指示灯
IO 连接端子	可选短路器短接，否则外设引脚与芯片引脚断开
天线端口	用于外接棒状天线
ZSL64 系列芯片	LoRa 系统级芯片
SWD 下载口	可通过 SWD 接口下载固件
恢复出厂设置按键	用户可配置
唤醒按键	用户可配置
休眠按键	用户可配置
普通按键	用户可配置
ISP 按键	用户可配置

ZM1268S-EVB 目录对应的评估板如 图 2.5 所示，ZM1268S-EVB 评估板是为帮助用户快速上手 ZM1268S 无线模块而开发的评估套件。ZM1268S 无线模块说明见下表。

表 2.3 ZM1268S 无线模块说明

芯片型号	工作频段	内置晶振
ZM1268S	470MHz～510MHz	TCXO

广州致远电子 wireless LoRa 无线模块

©2022 Guangzhou ZHIYUAN Electronics Co., Ltd.

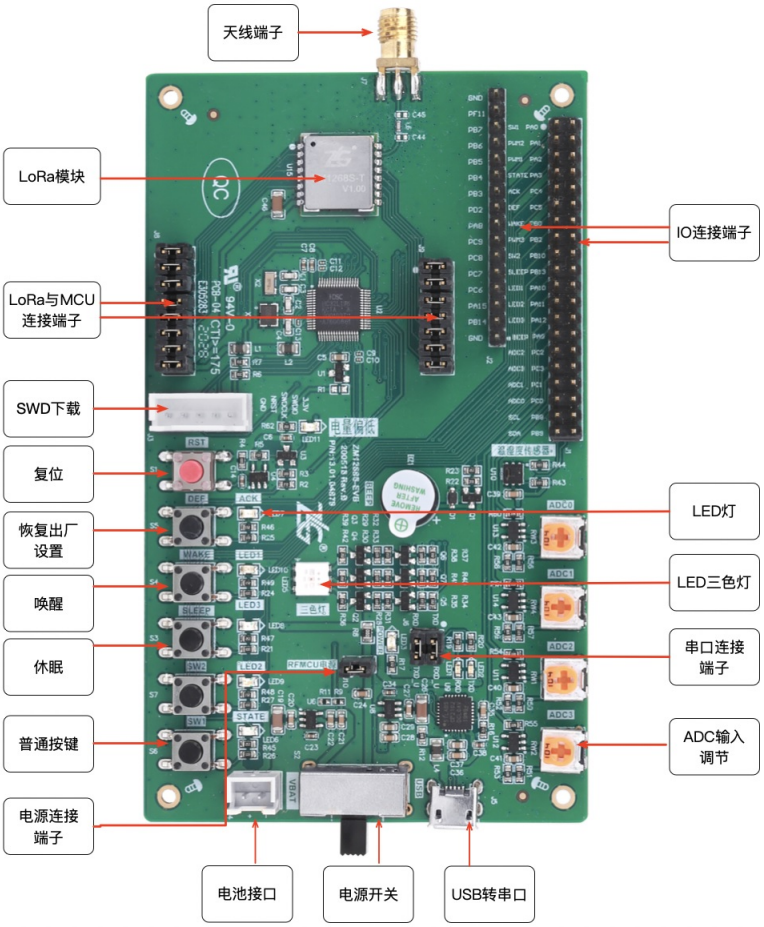


图 2.5 ZM1268S-EVB 评估板

ZM1268S-EVB 评估板部件相关描述见下表。

广州致远电子 wireless 无线模块

©2022 Guangzhou ZHIYUAN Electronics Co., Ltd.

表 2.4 ZM1268S-EVB 评估板部件描述

部件	描述
电池接口	外部电池输入端
电源开关	选择电源供电方式：USB 或电池
USB 转串口	供电和 USB 转串口功能，用 USB 电缆直接连到电脑，电脑需要装相应的驱动，可用来收发数据
ADC 输入调节	通过电位器调节，使模块 ADC 采集到不同的电压值
串口连接端子	需要用短路器短接，否则电脑串口与芯片串口断开
LED 三色灯	通过三色灯展示 PWM 输出
LED 灯	指示灯
IO 连接端子	可选短路器短接，否则外设引脚与芯片引脚断开
天线端口	用于外接棒状天线
LoRa 模块	ZM1268S LoRa 无线模块
LoRa 与 MCU 连接端子	必须用短路器短接，否则 LoRa 模块与 MCU 引脚断开
SWD 下载口	可通过 SWD 接口下载固件
恢复出厂设置按键	用户可配置
唤醒按键	用户可配置
休眠按键	用户可配置
普通按键	用户可配置
电源连接端子	必须用短路器短接，否则 MCU 与 LoRa 模块无法供电

ZSL420-EVB_XTAL_ZSL420 目录与 ZSL420-EVB_TCXO_ZSL421 目录对应的评估板如 图 2.6 所示，ZSL420-EVB 评估板是为帮助用户快速上手 ZSL42x 系列无线芯片而开发的评估套件。ZSL42x 系列无线芯片型号说明见下表。

表 2.5 ZSL42x 系列无线芯片型号说明

芯片型号	工作频段	内置晶振
ZSL420	470MHz ~ 510MHz	XTAL
ZSL421	470MHz ~ 510MHz	TCXO

广州致远电子 wireless LoRa 无线模块

©2022 Guangzhou ZHIYUAN Electronics Co., Ltd.

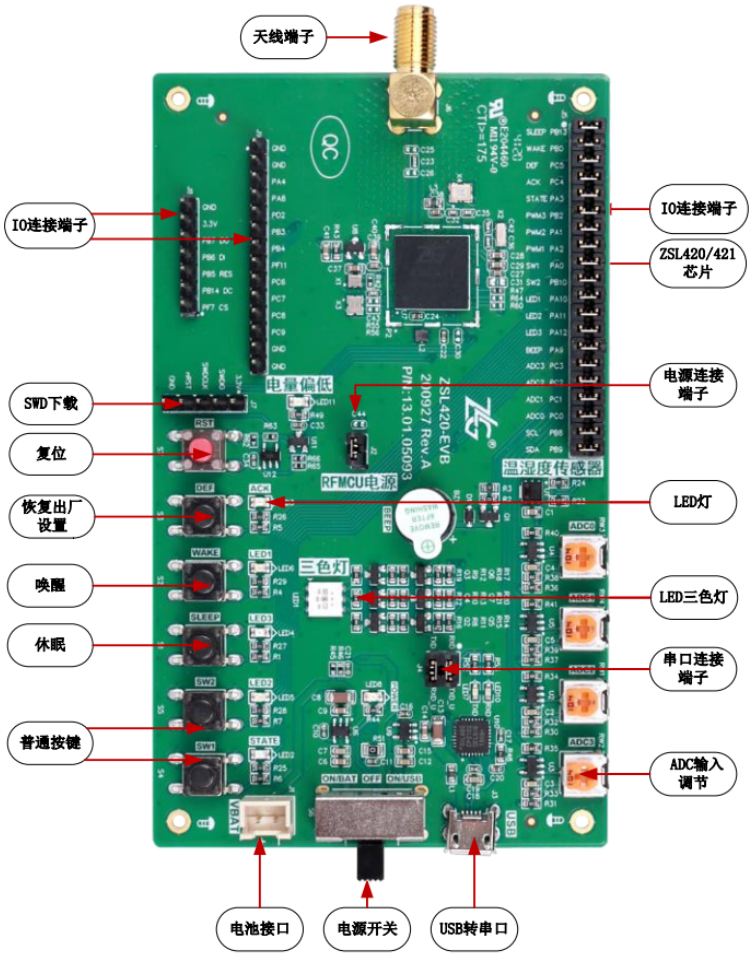


图 2.6 ZSL420-EVB 评估板

ZSL420-EVB 评估板部件相关描述见下表。

©2022 Guangzhou ZHIYUAN Electronics Co., Ltd.

表 2.6 ZSL420-EVB 评估板部件描述

部件	描述
电池接口	外部电池输入端
电源开关	选择电源供电方式：USB 或电池
USB 转串口	供电和 USB 转串口功能，用 USB 电缆直接连到电脑，电脑需要装相应的驱动，可用来收发数据
ADC 输入调节	通过电位器调节，使模块 ADC 采集到不同的电压值
串口连接端子	需要用短路器短接，否则电脑串口与芯片串口断开
LED 三色灯	通过三色灯展示 PWM 输出
LED 灯	指示灯
IO 连接端子	可选短路器短接，否则外设引脚与芯片引脚断开
天线端口	用于外接棒状天线
ZSL420/421 芯片	LoRa 智能组网芯片
SWD 下载口	可通过 SWD 接口下载固件
恢复出厂设置按键	用户可配置
唤醒按键	用户可配置
休眠按键	用户可配置
普通按键	用户可配置
电源连接端子	必须用短路器短接，否则 MCU 与 LoRa 模块无法供电

注意，ZSL420-EVB_XTAL_ZSL420 目录与 ZSL420-EVB_TCXO_ZSL421 目录使用的评估板都是 ZSL420-EVB 评估板，但由于使用的无线芯片不同，因此评估板上会有细微的差异。可通过如 图 2.7 所示进行区分，首先是芯片上丝印不同，会分别标明”ZSL420” 或”ZSL421” 字样，另外，假如是 ZSL420 无线芯片评估板，则 ZSL420 芯片左侧没有焊接 32MHz 的 TCXO 晶振，而 ZSL421 无线芯片评估板上是有焊接的。

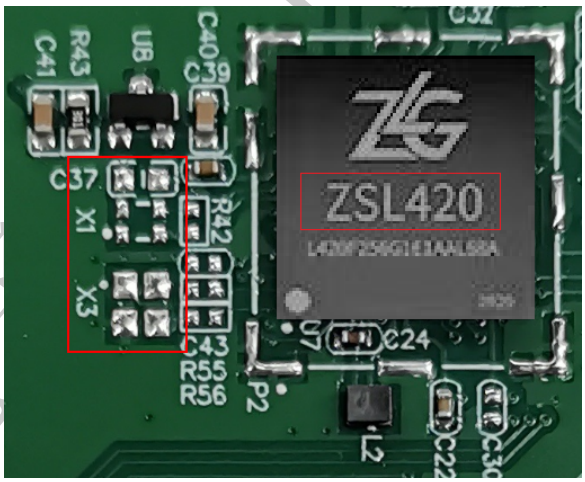


图 2.7 ZSL420 芯片评估板上没有焊接 TCXO 晶振

进入 keil 工程界面 {路径：wireless_lora_sdk v1.0.0\projects\wireless_lora}，打开对应评估板的 keil 工程。目录结构如 图 2.8 所示。

©2022 Guangzhou ZHIYUAN Electronics Co., Ltd.

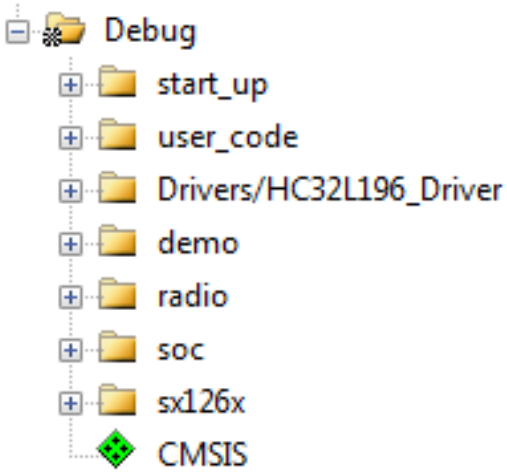


图 2.8 keil 工程目录结构

各个文件包含的内容如下：

- start_up 文件夹包含 MCU 相关的启动文件和内核文件。
- user_code 文件夹包含主函数文件 main.c，主函数中实例初始化了无线模块，初始化成功之后运行 demo 例程。另外还包含了跟评估板对应的无线模块配置文件 sx126x_radio_lora_cfg.c。
- Drivers/HC32L196_Driver 文件夹提供 MCU 相关外设底层驱动文件。
- demo 文件夹提供了 demo 板的简单测试例程。
- radio 文件夹主要提供用户操作无线模块的标准接口文件。
- soc 文件夹提供了适配无线模块所需的 MCU 外设驱动文件。
- sx126x 文件夹包含了无线模块的驱动文件。

3. DEMO 板测试例程使用说明

3.1 使用调试串口助手发送数据进行通信测试

用户可以同时打开两个串口助手工具，配置相应的串口号，波特率为 115200，数据位为 8，停止位为 1，校验位为无，流控制为无。在串口助手的发送框输入内容并按发送，DEMO 板就会将数据发送出去，另外一个 DEMO 板收到数据后会在串口助手打印接收到的数据包内容、RSSI 值和 SNR 值。发送端在发送完成时，红色的 LED 灯 (ACK) 会进行翻转显示。接收端在接收到数据后，绿色的 LED 灯 (STATE) 会进行翻转显示。

广州致远电子 wireless Lora 无线模块

©2022 Guangzhou ZHIYUAN Electronics Co., Ltd.

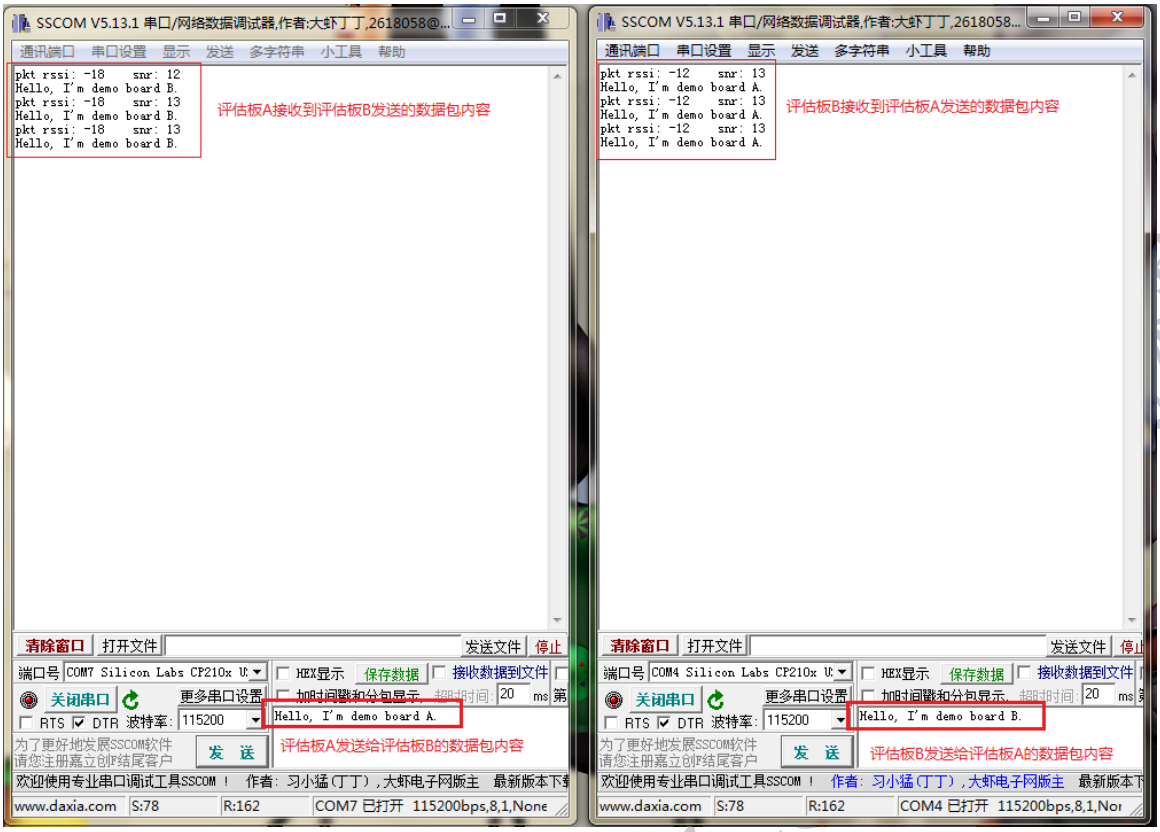


图 3.1 使用调试串口助手发送数据进行通信测试

3.2 使用按键 SW1 发送数据进行通信测试

按下 DEMO 板的 SW1 按键，即可发送一包数据，发送数据包内容固定字符串”WIRELESS SX126X TEST !”，同样的，DEMO 板在收到数据后会在串口助手打印接收到的数据包内容、RSSI 值和 SNR 值。发送端在发送完成时，红色的 LED 灯 (ACK) 会进行翻转显示。接收端在接收到数据后，绿色的 LED 灯 (STATE) 会进行翻转显示。

广州致远电子 wireless LoRa 模块

©2022 Guangzhou ZHIYUAN Electronics Co., Ltd.

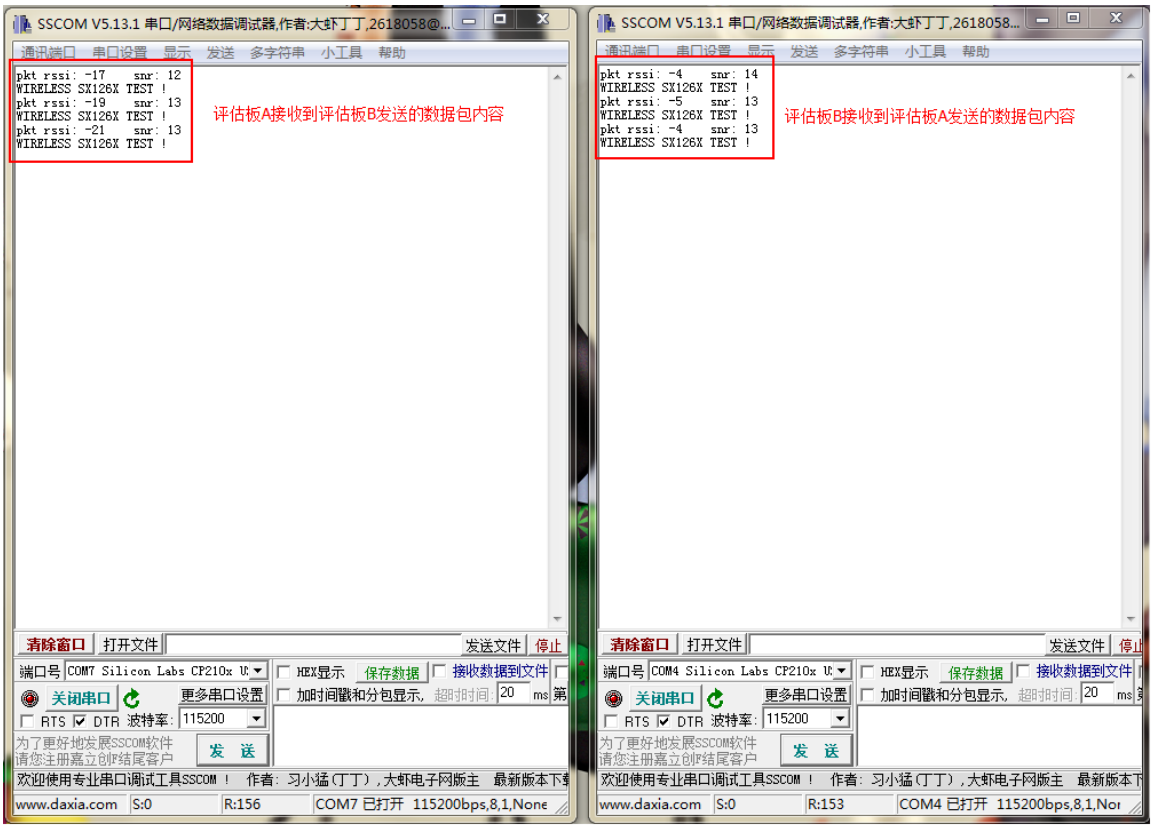


图 3.2 使用按键 SW1 发送数据进行通信测试

3.3 使用按键 SW2 调节发射功率

按下 SW2 可循环调节发射功率，默认 22dBm，如果超出最大值则回到最小值，可调整的功率档位为：”-9dBm、-5dBm, -1dBm, 3dBm, 7dBm, 12dBm, 17dBm, 22dBm”，同时，串口也会打印对应当前设置的功率值。

广州致远电子 wireless LoRa 模块

©2022 Guangzhou ZHIYUAN Electronics Co., Ltd.

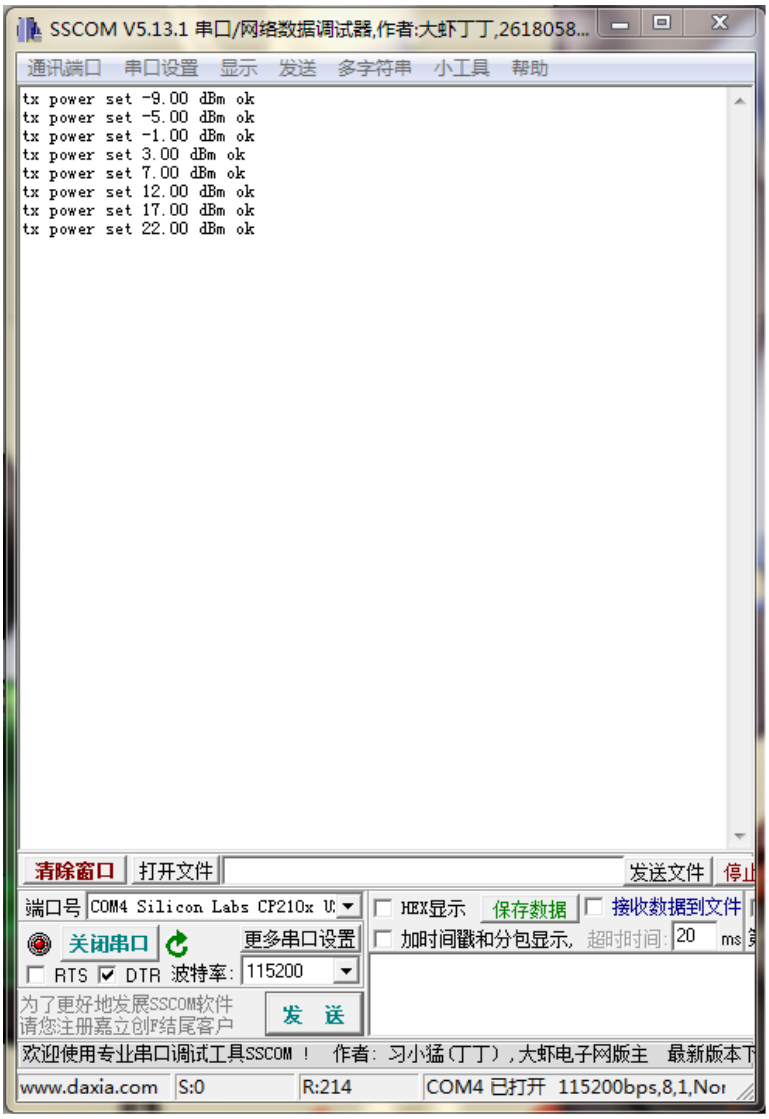


图 3.3 使用按键 SW2 调节发射功率

4. 无线驱动代码移植

对于使用该系列无线模块，用户可能不是使用 DEMO 板的 MCU 而是用自己 MCU 平台或者使用不同的编译环境，为此，用户需要将模块的驱动代码移植到自己平台上，下面将介绍无线驱动软件移植的方法。

4.1 移植前注意事项

1. 客户在使用无线模块 + 自己 MCU 平台时，建议先使用我们 DEMO 板作为接收端，这样在调试过程中可以排除接收端的问题，客户可以专注于调试发送端，发送功能调通，基本接收功能也不会有大的问题，大大缩短开发周期。
2. 建议使用 32 位的 MCU，以往有客户使用 8 位单片机时导致在 uint8_t val = 0x01; if(val == 0x01) {} 时会认为条件不匹配而无法执行相应的程序。
3. 客户在移植驱动时，除非是编译问题，否则不能修改 sx126x_lora.c 里面的内容，也不能把 spi 函数写在 sx126x_lora.c 里面，我们只需要用户从 sx126x_radio_lora_cfg.c 文件

©2022 Guangzhou ZHIYUAN Electronics Co., Ltd.

将 spi 函数传入即可。

4.2 文件的移植

1. 将标准接口移植到用户的平台中，即 radio 文件夹里的所有文件。
2. 将 sx126x 无线驱动相关文件移植到用户的平台中，即 sx126x 里所有文件 {路径：wireless_lora_sdk v1.0.0\sx126x}。
3. 将无线模块配置文件移植到用户的平台中，即 user_code 文件夹中的 sx126x_radio_lora_cfg.c 文件。
4. 默认情况下无线模块使用轮询接收模式，假如用户需要使用中断接收模式，则还需要对 DIO1 引脚进行中断功能适配并设置相应的无线模块接收中断服务函数。详细内容可查看” 中断接收测试例程说明” 章节。

4.3 文件的修改

为了适配用户的平台，只需要修改 sx126x_radio_cfg.c 文件内容即可。首先，用户需要提供以下函数。

1. 准备 SPI 驱动

该系列无线模块都是使用 SPI 接口，SPI 驱动的正常运行对无线模块的正常运行至关重要，用户仅需提供 SPI 读字节和写字节函数即可，函数格式和简单范例如 程序清单 4.1 所示，MCU 的主机 SPI 特性如下：

- SPI 主机采用模式 0，CPOL=0 和 CPHA=0;
- 全双工通信；
- 数据长度 8 位，MSB 通信；
- SPI 速度最高为 16MHz；

程序清单 4.1 SPI 读写函数

```
typedef struct sx126x_lora_spi_funcs {
    /** \brief SPI 写一个字节函数【该接口必须提供】  */
    void    (*pfn_spi_write_byte) (uint8_t byte);

    /** \brief SPI 读一个字节函数【该接口必须提供】 */
    uint8_t (*pfn_spi_read_byte)  (void);
} sx126x_lora_spi_funcs_t;

// 简单范例

// SPI 发送一个字节
void mcu_spi_send_byte (uint8_t byte)
{
    //发送一个字节
}
// SPI 接收一个字节
uint8_t mcu_spi_recv_byte (void)
{
    //返回接收一个字节
}
```


©2022 Guangzhou ZHIYUAN Electronics Co., Ltd.

注意：用户不需要在函数内部操作 CS 片选引脚，函数内部只需单纯的发送一个字节数据或读取一字节数据。

2. 准备 GPIO 操作驱动

除了 SPI 的 SCK、MISO 和 MOSI 引脚外，还有 RST、SEL、BUSY、TXEN 和 RXEN（VCTL_1）引脚，驱动中需要对这个几个引脚进行操作，因此需要用户提供这几个引脚的 GPIO 操作函数：读引脚电平函数和设置引脚电平函数。函数格式如 程序清单 4.2 所示。

程序清单 4.2 GPIO 操作函数集

```
typedef struct sx126x_lora_gpio_funcs {

    /*
     * SX_NRESET 复位引脚电平设置【该接口必须提供】
     * val=0 表示设置为低电平，val =1 表示设置为高电平
     */
    void    (*pfn_reset_pin_set) (uint8_t val);

    /*
     * SPI_NSS 片选引脚电平设置【该接口必须提供】
     * val=0 表示设置为低电平，val =1 表示设置为高电平
     */
    void    (*pfn_sel_pin_set)   (uint8_t val);

    /*
     * 射频开关发射控制引脚电平设置【该接口可根据无线模块的选择进行提供】
     * 假如为 ZSL420、ZSL421、ZM1268S、 ZM68S 无线模块，必须提供此接口
     * val=0 表示设置为低电平，val =1 表示设置为高电平
     * 假如为 ZSL64 系列无线模块，可直接设置此接口为 NULL，或者接口实现内容为空
     */
    void    (*pfn_txen_pin_set)   (uint8_t val);

    /*
     * VCTL_1(RXEN) 射频开关接收控制引脚电平设置【该接口必须提供】
     * val=0 表示设置为低电平，val =1 表示设置为高电平
     */
    void    (*pfn_rxen_pin_set)   (uint8_t val);

    /*
     * BUSY 繁忙引脚电平读取【该接口必须提供】
     * 返回 0 表示引脚为低电平，返回 1 表示引脚为高电平
     */
    uint8_t (*pfn_busy_pin_read)  (void);


} sx126x_lora_gpio_funcs_t;

// GPIO 操作函数的简单范例。

/* 复位引脚电平设置 */
void mcu_gpio_rst_pin_set (uint8_t val)
{
    if (val == 1) {
        //引脚输出高电平;
    } else if (val == 0) {
        //引脚输出低电平;
    }
}
```


©2022 Guangzhou ZHIYUAN Electronics Co., Ltd.

```
/* 读 BUSY 引脚电平 */
uint8_t mcu_gpio_busy_pin_read (void)
{
   //返回引脚电平，返回 1 表示高电平，返回 0 表示低电平
}
```

注意：为了能有效的控制 GPIO，用户需要自行初始化引脚配置。

3. 准备延时函数

无线模块内部读写寄存器有一定的时序要求，所以需要用户提供毫秒延时函数。函数格式如 程序清单 4.3 所示。

程序清单 4.3 延时函数

```
/*
 * \brief 延时函数集
 */
typedef struct sx126x_lora_delay_funcs {
    /** \brief 毫秒延时函数【该接口必须提供】 */
    void (*pfn_delay_ms) (uint32_t ms);

     /** \brief 微秒延时函数 */
    void (*pfn_delay_us) (uint32_t us);
} sx126x_lora_delay_funcs_t;

// 使用范例
void delay1ms(uint32_t ms)
{
     // ms 为延时的毫秒数
}
```

4. 准备 CPU 锁函数

在 SPI 传输过程中，需要添加 CPU 锁对其进行保护，防止 SPI 传输过程中被打断，导致传输数据丢失或者模块死机等问题。函数格式如 程序清单 4.4 所示。

程序清单 4.4 CPU 锁函数

```
/**
 * \brief CPU 锁函数集
 * 这里的函数用于保护 SPI 传输，不加锁可能会导致模块死机
 */
typedef struct sx126x_lora_cpu_lock_funcs {
    /** \brief CPU 关中断【该接口必须提供】 */
    uint32_t (*pfn_cpu_lock) (void);

    /** \brief CPU 开中断【该接口必须提供】 */
    void (*pfn_cpu_unlock) (uint32_t cpu_key);
} sx126x_lora_cpu_lock_funcs_t;

// 使用范例
uint32_t cpu_lock (void)
{
    // cpu 关中断
}

void cpu_unlock (uint32_t cpu_key)
```


©2022 Guangzhou ZHIYUAN Electronics Co., Ltd.

```
{
    // cpu 开中断
}
```

准备好以上函数之后，用户需要将这些函数注册到驱动里面。sx126x_radio_cfg.c 文件提供了一个模板，如 程序清单 4.5 所示。用户只需要将上述说的四种函数对应的替换成自己的函数即可，其他地方不必修改。

程序清单 4.5 函数注册

```
#include "ddl.h"

#include "sx126x_lora.h"

#include "board_gpio.h"
#include "board_spi.h"
#include "radio_int.h"


static sx126x_lora_spi_funcs_t      __g_spi_dev;
static sx126x_lora_gpio_funcs_t     __g_gpio_dev;
static sx126x_lora_delay_funcs_t    __g_delay;
static sx126x_lora_cpu_lock_funcs_t __g_cpu_lock;
static radio_sx126x_lora_dev_t      __g_sx126x;
static radio_sx126x_lora_devinfo_t  __g_sx126x_devinfo;


radio_handle_t radio_zm4xx_inst_init (void)
{
    /********** SPI 读写函数设置 **********/

    /* SPI 发送一字节函数【该接口必须提供】 */
    __g_spi_dev.pfn_spi_read_byte  = mcu_spi_recv_byte;

    /* SPI 接收一字节函数【该接口必须提供】 */
    __g_spi_dev.pfn_spi_write_byte = mcu_spi_send_byte;



    /********** GPIO 操作函数设置 **********/

    /* SX_NRESET 复位引脚电平设置【该接口必须提供】 */
    __g_gpio_dev.pfn_reset_pin_set = mcu_gpio_rst_pin_set;

    /* SPI_NSS 片选引脚电平设置【该接口必须提供】 */
    __g_gpio_dev.pfn_sel_pin_set = mcu_gpio_sel_pin_set;

    /*
     * 射频开关发射控制引脚电平设置【该接口可根据无线模块的选择进行提供】
     * 假如为 ZSL420、ZSL421、ZM1268S、 ZM68S 无线模块，必须提供此接口
     * 假如为 ZSL64 系列无线模块，可直接设置此接口为 NULL，或者接口实现内容为空
     */
    __g_gpio_dev.pfn_txen_pin_set = mcu_gpio_txen_pin_set;

    /* VCTL_1(RXEN) 射频开关接收控制引脚电平设置【该接口必须提供】 */
    __g_gpio_dev.pfn_rxen_pin_set = mcu_gpio_rxen_pin_set;

    /*  BUSY 繁忙引脚电平读取【该接口必须提供】 */
    __g_gpio_dev.pfn_busy_pin_read = mcu_gpio_busy_pin_read;
```


©2022 Guangzhou ZHIYUAN Electronics Co., Ltd.

```
    /********** 延时函数设置 **********/

    /* 毫秒延时函数【该接口必须提供】 */
    __g_delay.pfn_delay_ms = delay1ms;



    /********** CPU 锁函数设置 **********/
    /* 这里的函数用于保护 SPI 传输，不加锁可能会导致模块死机 */

    /* CPU 关中断【该接口必须提供】 */
    __g_cpu_lock.pfn_cpu_lock   = cpu_lock;
    /* CPU 开中断【该接口必须提供】 */
    __g_cpu_lock.pfn_cpu_unlock = cpu_unlock;


    /* 设置设备信息 */
    __g_sx126x_devinfo.p_events     = &__g_sx126x_events;
    __g_sx126x_devinfo.p_lora_param = &__g_lora_param;

    return radio_sx126x_lora_init(&__g_sx126x,
                                  &__g_spi_dev,
                                  &__g_gpio_dev,
                                  &__g_delay,
                                  &__g_cpu_lock,
                                  &__g_sx126x_devinfo);
}
```

5. 无线收发测试例程说明

5.1 无线模块参数配置

在运行无线收发测试例程之前，用户可根据自己的使用场景在 sx126x_radio_lora_cfg.c 文件的模块配置参数结构体中进行参数配置，如 程序清单 5.1 所示。

程序清单 5.1 LoRa 参数配置

```
static sx126x_lora_param_set_t __g_lora_param =
{
    ZSL64A1ALHA, /* 设备型号选择 */
                 /* [ZSL420       ZSL421
                     ZM1268S      ZM68S
                     ZSL64A1ALHA  ZSL64A2ALHA
                     ZSL64B2ALHA  ZSL64B3ALHA
                     ZSL64C3ALHA  ZSL64C4ALHA] */
    470000000, /* 频率 */
               /* ZSL420      [470000000 ~ 510000000]  */
               /* ZSL421      [470000000 ~ 510000000]  */
               /* ZM1268S     [470000000 ~ 510000000]  */
               /* ZM68S       [470000000 ~ 510000000]  */
               /* ZSL64A1ALHA [470000000 ~ 510000000]  */
               /* ZSL64A2ALHA [470000000 ~ 510000000]  */
               /* ZSL64B2ALHA [863000000 ~ 870000000]  */
               /* ZSL64B3ALHA [863000000 ~ 870000000]  */
               /* ZSL64C3ALHA [902000000 ~ 928000000]  */
               /* ZSL64C4ALHA [902000000 ~ 928000000]  */
    22,        /* 发射功率     [-9 ~ +22]  */
    8,  /* 带宽  */
        /* ZSL420      [6: 62.5 kHz, 7: 125 kHz, 8: 250 kHz,  9: 500 kHz] */
        /* ZSL421      [6: 62.5 kHz, 7: 125 kHz, 8: 250 kHz,  9: 500 kHz] */
```


©2022 Guangzhou ZHIYUAN Electronics Co., Ltd.

```
        /* ZM1268S     [6: 62.5 kHz, 7: 125 kHz, 8: 250 kHz,  9: 500 kHz] */
        /* ZM68S       [             7: 125 kHz, 8: 250 kHz,  9: 500 kHz] */
        /* ZSL64A1ALHA [6: 62.5 kHz, 7: 125 kHz, 8: 250 kHz,  9: 500 kHz] */
        /* ZSL64A2ALHA [6: 62.5 kHz, 7: 125 kHz, 8: 250 kHz,  9: 500 kHz] */
        /* ZSL64B2ALHA [6: 62.5 kHz, 7: 125 kHz, 8: 250 kHz,  9: 500 kHz] */
        /* ZSL64B3ALHA [6: 62.5 kHz, 7: 125 kHz, 8: 250 kHz,  9: 500 kHz] */
        /* ZSL64C3ALHA [6: 62.5 kHz, 7: 125 kHz, 8: 250 kHz,  9: 500 kHz] */
        /* ZSL64C4ALHA [6: 62.5 kHz, 7: 125 kHz, 8: 250 kHz,  9: 500 kHz] */
    7,  /* 扩频因子 */
        /* ZSL420  [5: 32, 6: 64, 7: 128, 8: 256, 9: 512,
                    10: 1024, 11: 2048, 12: 4096 chips] */
        /* ZSL421  [5: 32, 6: 64, 7: 128, 8: 256, 9: 512,
                    10: 1024, 11: 2048, 12: 4096 chips] */
        /* ZM1268S [5: 32, 6: 64, 7: 128, 8: 256, 9: 512,
                    10: 1024, 11: 2048, 12: 4096 chips] */
        /* ZM68S   带宽 125kHz [5: 32, 6: 64, 7: 128, 8: 256, 9: 512 chips]*/
        /*         带宽 250kHz [5: 32, 6: 64, 7: 128, 8: 256, 9: 512,
                                10: 1024 chips]   */
        /*         带宽 500kHz [5: 32, 6: 64, 7: 128, 8: 256, 9: 512,
                                10: 1024, 11: 2048 chips] */
        /* ZSL64A1ALHA [5: 32, 6: 64, 7: 128, 8: 256, 9: 512,
                        10: 1024, 11: 2048, 12: 4096 chips] */
        /* ZSL64A2ALHA [5: 32, 6: 64, 7: 128, 8: 256, 9: 512,
                        10: 1024, 11: 2048, 12: 4096 chips] */
        /* ZSL64B2ALHA [5: 32, 6: 64, 7: 128, 8: 256, 9: 512,
                        10: 1024, 11: 2048, 12: 4096 chips] */
        /* ZSL64B3ALHA [5: 32, 6: 64, 7: 128, 8: 256, 9: 512,
                        10: 1024, 11: 2048, 12: 4096 chips] */
        /* ZSL64C3ALHA [5: 32, 6: 64, 7: 128, 8: 256, 9: 512,
                        10: 1024, 11: 2048, 12: 4096 chips] */
        /* ZSL64C4ALHA [5: 32, 6: 64, 7: 128, 8: 256, 9: 512,
                        10: 1024, 11: 2048, 12: 4096 chips] */
    1,   /* 编码率         [1: 4/5, 2: 4/6, 3: 4/7, 4: 4/8]    */
    1,   /* 硬件 CRC 检验   [0: 关闭, 1: 开启]                   */
    0,   /* 报头模式       [0: 显式报头模式, 1: 隐式报头模式]  */
    0,   /* 接收模式       [0: Continuous, 1: Single]          */
    100, /* 单次接收模式下接收超时时间 (ms)                     */
    0,   /* 发送超时使能 [0: 禁能 1: 使能]                      */
    10,  /* 负载长度                                            */
    16,  /* 前导码长度                                          */
    0,   /* 接收 IQ 翻转    [0: 正常, 1: 翻转]                   */
    0,   /* 发送 IQ 翻转    [0: 正常, 1: 翻转]                   */
    1,   /* 软件 CRC 检验   [0: 关闭, 1: 开启]                   */
    1,   /* 低数据速率优化开启方式选择 [0: 手动开启, 1: 自动开启] */
    1,   /* 手动控制低数据速率优化的开启 (需提前将开启方式选择为" 手动开启")[0: 关闭,
↪1: 开启] */
};
```

无线收发测试例程分为轮询接收测试例程（默认）与中断接收测试例程，下面将对这两种无线接收方式进行说明。

5.2 轮询接收测试例程说明

现在以 程序清单 5.2 的例程来讲解无线模块的数据收发流程，例程首先将模块设置为待机模式，然后配置模块的通信频率，最后配置为接收模式，当模块接收到数据 radio_data_recv() 接口返回值为 0，此时 LoRa 接收到的数据已经拷贝到 __g_data_buf，接收到的数据长度也会设置到 len，然后评估板会将数据从串口发送出去 (如果是单次接收模式还需要重新配置一下接收模式，否则无法继续接收数据)。当串口接收到数据后 uart_recv_pkt() 接口返回

©2022 Guangzhou ZHIYUAN Electronics Co., Ltd.

值为 0，串口数据会拷贝到 __g_data_buf，数据长度设置到 pkt_size，然后串口数据会通过 radio_data_send() 接口发送出去，发送完成后执行 radio_mode_set(handle, RX_MODE) 配置模块重新进入接收模式。

程序清单 5.2 轮询测试例程

```
static uint8_t        __g_data_buf[512] = {0};
static uint16_t       __g_send_byte     = 0;
static radio_handle_t __gp_handle;


void demo_zm4xx_polling_entry (radio_handle_t handle)
{
    int      ret;
    uint16_t len;
    uint16_t pkt_size = 0;
    uint16_t i = 0;

    __gp_handle = handle;

    /*
     * 设置频率等参数需要先进入待机模式，否则有可能设置不成功或
     * 模块无法工作（多线程使用请加锁保护该接口）
     */
    radio_mode_set(handle, STANDBY_MODE);

    /* 设置为接收模式（多线程使用请加锁保护该接口） */
    radio_mode_set(handle, RX_MODE);

    while (1) {

        /* 轮询检查是否接收到数据（多线程使用请加锁保护该接口） */
        ret = radio_data_recv(handle, __g_data_buf, &len);
        if (ret == RADIO_RET_OK) { /* 返回值为 RADIO_RET_OK 表示接收到数据 */

            /* 把数据从串口发送出去 */
            for (i = 0; i < len; i++) {
                uart_byte_send(__g_data_buf[i]);
            }

            /*
             * 假如为单次接收模式，则需要再次进入接收模式才能重新进行接收
             * 假如为连续接收模式，则不需要再次进入接收模式，在接收之后会保持接收模式
             * （多线程使用请加锁保护该接口）
             */
            radio_mode_set(handle, RX_MODE);

        }

        /* 发送 */
        if (__g_send_byte > 0) {

            /* 无线发送数据（多线程使用请加锁保护该接口） */
            radio_data_send(handle, __g_data_buf, __g_send_byte);

            /*
             * 所有型号的模块只要发送了数据就必须切换回接收模式，否则无法接收数据
             * （多线程使用请加锁保护该接口）
             */
            radio_mode_set(handle, RX_MODE);

            __g_send_byte = 0;
```


©2022 Guangzhou ZHIYUAN Electronics Co., Ltd.

```
        }

        /* 从串口获取要发送的数据包 */
        if (uart_recv_pkt(__g_data_buf, &pkt_size) == 0) {
            __g_send_byte = pkt_size;
        }
    }
}
```

5.3 中断接收测试例程说明

现在以 程序清单 5.3 的例程来讲解无线模块的数据收发流程。中断接收测试例程与轮询接收测试例程的执行流程大体相同，唯一不同的是使用中断的方式进行无线数据的接收。在无线模块设置为接收模式之后，假如接收到数据，则无线模块的 DIO1 引脚会产生上升沿，该上升沿信号可以被 MCU 设置的中断检测到。在 MCU 检测到上升沿之后，会产生对应引脚的中断并执行中断服务函数里的内容。对于中断服务函数执行的内容，测试例程的应用中填充了“无线模块接收中断服务函数”即 __radio_recv_int_handle() 函数作为中断回调函数。因此，在无线接收到数据后，产生中断，就会执行应用层的 __radio_recv_int_handle() 函数，对数据进行接收。关于 MCU 对 DIO1 引脚中断功能的配置与相关函数设置，可以参考 程序清单 5.4 。

程序清单 5.3 轮询测试例程

```
static uint8_t        __g_data_buf[512] = {0};
static uint16_t       __g_send_byte     = 0;
static radio_handle_t __gp_handle;


/**
 * \brief  无线模块接收中断服务函数
 */
static void __radio_recv_int_handle (void)
{
    int      ret = 0;
    uint8_t  len = 0;

    ret = radio_data_recv(__gp_handle, __g_data_buf, &len);
    if (ret == RADIO_RET_OK) {
        __g_recv_byte = len;
    }
}


void demo_zm4xx_int_entry (radio_handle_t handle)
{
    uint32_t key;
    uint16_t pkt_size = 0;
    uint8_t  recv_buf[512];
    uint16_t recv_data_size = 0;
    uint16_t i = 0;

    __gp_handle = handle;

    /*
     * 设置频率等参数需要先进入待机模式，否则有可能设置不成功或
     * 模块无法工作（多线程使用请加锁保护该接口）
```


©2022 Guangzhou ZHIYUAN Electronics Co., Ltd.

```
 */
radio_mode_set(handle, STANDBY_MODE);

/* 设置为接收模式 (多线程使用请加锁保护该接口)   */
radio_mode_set(handle, RX_MODE);


/*
 * 无线模块中断初始化
 * 无线模块使用中断接收模式，则必须实现该接口，
 * 实现内容包含对 DIO1 引脚进行中断功能配置和 DIO1 引脚中断服务函数准备
 */
radio_int_init(handle,__radio_recv_int_handle);

/* 初始化按键中断 */
mcu_sw1_int_init(__sw1_int_handle);
mcu_sw2_int_init(__sw2_int_handle);

while (1) {

    /*
     * 无线模块在连续接收模式下随时可能产生中断导致数据被覆盖，
     * 所以最好将接收到的数据拷贝到新的缓冲区并加锁保护拷贝的过程
     */
    key = cpu_lock();
    if (__g_recv_byte > 0) {
        memcpy(recv_buf, __g_data_buf, __g_recv_byte);
        recv_data_size = __g_recv_byte;
        __g_recv_byte = 0;
    }
    cpu_unlock(key);

    /*  接收  */
    if (recv_data_size > 0) {

        mcu_gpio_pin_toggle(GpioPortA, GpioPin3);

        /*  把数据从串口发送出去  */
        for (i = 0; i < recv_data_size; i++) {
            mcu_uart_byte_send(__g_data_buf[i]);
        }

        recv_data_size = 0;
        /*
         * 假如为单次接收模式，则需要再次进入接收模式才能重新进行接收
         * 假如为连续接收模式，则不需要再次进入接收模式，在接收之后会保持接收模式
         * (多线程使用请加锁保护该接口)
         */
        radio_mode_set(handle, RX_MODE);

    }

    /*  发送  */
    if (__g_send_byte > 0) {

        mcu_gpio_pin_toggle(GpioPortC, GpioPin4);

        /* 无线发送数据 (多线程使用请加锁保护该接口) */
        radio_data_send(handle, __g_data_buf, __g_send_byte);

        /*
         * 所有型号的模块只要发送了数据就必须切换回接收模式，否则无法接收数据
```


©2022 Guangzhou ZHIYUAN Electronics Co., Ltd.

```
             * (多线程使用请加锁保护该接口)
             */
            radio_mode_set(handle, RX_MODE);

            __g_send_byte = 0;


        }

        /* 从串口获取要发送的数据包 */
        if (mcu_uart_recv_pkt(__g_data_buf, &pkt_size) == 0) {
            __g_send_byte = pkt_size;
        }

    }
}
```

程序清单 5.4 MCU 配置 DIO1 引脚中断功能

```
/*
 * DIO1 引脚中断功能配置
 * 默认配置为上升沿中断模式
 */
static void __radio_dio1_int_init (void)
{
    stc_gpio_cfg_t stcGpioCfg;
    Sysctrl_SetPeripheralGate(SysctrlPeripheralGpio, TRUE);

    stcGpioCfg.enDir      = GpioDirIn;
    stcGpioCfg.enDrv      = GpioDrvL;
    stcGpioCfg.enPu       = GpioPuDisable;
    stcGpioCfg.enPd       = GpioPdEnable;
    stcGpioCfg.enCtrlMode = GpioAHB;
    Gpio_Init(GpioPortC, GpioPin12, &stcGpioCfg);

    /* 配置 DIO1 引脚为上升沿中断并使能 */
    Gpio_EnableIrq(GpioPortC, GpioPin12, GpioIrqRising);
    /* 使能端口 PORTC 系统中断 */
    EnableNvic(PORTC_E_IRQn, IrqLevel0, TRUE);
}

/*
 * DIO1 引脚中断服务函数
 * 当 DIO1 引脚产生上升沿时产生中断并调用应用层传入的无线接收函数进行数据接收
 */
void PortC_IRQHandler (void)
{
    if(TRUE == Gpio_GetIrqStatus(GpioPortC, GpioPin12)) {
        if(__g_radio_dio1_int_handle) {
            /* 调用应用层的无线接收函数进行数据接收 */
            __g_radio_dio1_int_handle();
        }
        Gpio_ClearIrq(GpioPortC, GpioPin12);
    }
}

/*
 * 无线模块中断初始化
 * 无线模块使用中断接收模式，则必须实现该接口，
 * 实现内容包含对 DIO1 引脚进行中断功能配置和 DIO1 引脚中断服务函数准备
 */
void radio_int_init (radio_handle_t handle, pfn_radio_int_cb radio_int_cb)
```


©2022 Guangzhou ZHIYUAN Electronics Co., Ltd.

```
{
    __radio_dio1_int_init();
    __g_radio_dio1_int_handle = radio_int_cb;
}
```

6. radio API 介绍

下面介绍的 API 入口参数 handle 句柄都来自 radio_zm4xx_inst_init() 初始化时的返回值。

6.1 设置模式

该接口用于配置模块的工作模式。

程序清单 6.1 radio_mode_set 接口介绍

```
/*
 * \brief 设置无线模块模式
 *
 * \param[in] handle : 无线模块操作句柄
 * \param[in] mode   : 模式
 *
 * \retval RADIO_RET_OK                 : 设置成功
 * \retval RADIO_RET_PARAM_INVALID      : 参数无效
 * \retval RADIO_RET_MODULE_NOT_SUPPORT : 该模块不支持该模式
 */
int radio_mode_set (radio_handle_t handle, uint8_t mode);
```

使用方法如下：

程序清单 6.2 radio_mode_set 用法介绍

```
int ret;
/* 设置接收模式 */
ret = radio_mode_set(handle, RX_MODE);
if (ret == RADIO_RET_OK) {
    //设置成功
}
/* 设置待机模式 */
ret = radio_mode_set(handle, STANDBY_MODE);
if (ret == RADIO_RET_OK) {
    //设置成功
}
/* 设置接收模式 */
ret = radio_mode_set(handle, RX_MODE);
if (ret == RADIO_RET_OK) {
    //设置成功
}
```

注意： 发送数据不需要设置发送模式，用户调用 radio_data_send() 即可发送数据。

6.2 数据包空中时间获取

该接口用于获取数据包的空中传输时间。

©2022 Guangzhou ZHIYUAN Electronics Co., Ltd.

程序清单 6.3 radio_time_on_air_get 接口介绍

```
/**
 * \brief 数据包空中时间获取
 *
 * \param[in]  handle       : 无线模块操作句柄
 * \param[in]  preamble_len : 前导码长度，如需使用当前已配置前导码长度请使用宏
                             RADIO_TIME_ON_AIR_USE_CUR_PREAMBLE_LEN
 * \param[in]  pkt_len      : 包长度
 * \param[out] p_time       : 空中时间
 *
 * \retval  RADIO_RET_OK                  : 获取成功
 * \retval  RADIO_RET_PARAM_INVALID       : 参数无效
 * \retval  RADIO_RET_MODULE_NOT_SUPPORT : 该模块不支持该模式
 */
int radio_time_on_air_get (radio_handle_t handle,
                           uint16_t       preamble_len,
                           uint8_t        pkt_len,
                           uint32_t      *p_time);
```

使用方法如下：

程序清单 6.4 radio_time_on_air_get 用法介绍

```
int ret;
uint32_t time_ms;
uint8_t  pkt_len = 10;
uint32_t preamble_len = 32;
ret = radio_time_on_air_get (handle, preamble_len, pkt_len, &time_ms);
if (ret == RADIO_RET_OK) {
    //获取成功
    printf("pkt time on air:%d\r\n", time_ms);
}
```

6.3 发送数据

该接口用于发送无线数据。

程序清单 6.5 radio_data_send 接口介绍

```
/**
* \brief 无线模块发送数据
*
* \param[in] handle : 无线模块操作句柄
* \param[in] p_buf  : 发送数据缓冲区
* \param[in] size   : 数据长度
*
* \retval RADIO_RET_OK               : 发送成功
* \retval RADIO_RET_PARAM_INVALID    : 参数无效
* \retval RADIO_RET_CFG_OUT_OF_RANGE : 包长度超出范围
*/
int radio_data_send (radio_handle_t handle, uint8_t *p_buf, uint16_t size);
```

使用方法如下：

程序清单 6.6 radio_data_send 用法介绍

```
int ret;
uint8_t buf[5] = {0x01, 0x02, 0x03, 0x04, 0x05};
ret = radio_data_send(handle, buf, sizeof(buf));
if (ret == RADIO_RET_OK) {
    //发送成功
}
```


©2022 Guangzhou ZHIYUAN Electronics Co., Ltd.

6.4 接收数据

该接口用于判断是否有接收到数据，如果有则将数据拷贝到缓冲区并且返回 RADIO_RET_OK。

程序清单 6.7 radio_data_recv 接口介绍

```
/**
* \brief 无线模块接收数据
*
* \param[in] handle : 无线模块操作句柄
* \param[in] p_buf  : 接收数据缓冲区
* \param[in] size   : 接收到数据的长度
*
* \retval RADIO_RET_OK             : 接收成功
* \retval RADIO_RET_PARAM_INVALID : 参数无效
* \retval RADIO_RET_NOT_RECV_PKT  : 没有接收到数据包
*/
int radio_data_recv (radio_handle_t handle, uint8_t *p_buf, uint16_t *p_size);
```

使用方法如下：

程序清单 6.8 radio_data_recv 用法介绍

```
int ret;
uint8_t buf[255]; //缓冲区请设置为 255 字节以上，避免出现拷贝数据时缓冲区溢出的情况
uint16_t size;
ret = radio_data_recv(handle, buf, &size);
if (ret == RADIO_RET_OK) {
    //接收成功，数据已拷贝到 buf， size 为数据长度
}
```

6.5 设置频率

该接口用于设置无线通信频率。

程序清单 6.9 radio_freq_set 接口介绍

```
/**
* \brief 设置无线模块频率
*
* \note 设置前请先把模块切换到 STANDBY 模式
*
* \param[in] handle : 无线模块操作句柄
* \param[in] freq   : 频率
*
* \retval RADIO_RET_OK                : 设置成功
* \retval RADIO_RET_PARAM_INVALID     : 参数无效
* \retval RADIO_RET_CFG_OUT_OF_RANGE : 频率超出范围
* \retval RADIO_RET_FREQ_NOT_SUPPORT : 不支持该频率
*/
int radio_freq_set (radio_handle_t handle, uint32_t freq);
```

使用方法如下：

程序清单 6.10 radio_freq_set 用法介绍

```
int ret;
/* 配置前请先把模块设置为 STANDBY 模式 */
radio_mode_set(handle, STANDBY_MODE);
```


©2022 Guangzhou ZHIYUAN Electronics Co., Ltd.

```
ret = radio_freq_set(handle, 470000000);
if (ret == RADIO_RET_OK) {
    //设置成功
}
```

6.6 复位模块

该接口会操作模块的复位引脚进行复位，复位之后模块的寄存器参数会丢失，所以需要用户重新对模块初始化。

程序清单 6.11 radio_reset 接口介绍

```
/**
* \brief 无线模块复位
*
* \note 复位后需要对模块重新初始化
*
* \param[in] handle : 无线模块操作句柄
*
* \retval RADIO_RET_OK            : 复位成功
* \retval RADIO_RET_PARAM_INVALID : 参数无效
*/
int radio_reset (radio_handle_t handle);
```

使用方法如下：

程序清单 6.12 radio_reset 用法介绍

```
int ret;
radio_handle_t handle;
ret = radio_reset(handle);
if (ret == RADIO_RET_OK) {
    //复位完成
} else {
    //复位失败
    //返回值为 RADIO_RET_PARAM_INVALID 表示 handle 可能为 NULL
}
/* 重新对模块初始化 */
handle = radio_zm4xx_inst_init();
if (handle != NULL) {
    //初始化成功
}
```

6.7 设置前导码长度

该接口用于设置模块的前导码长度，前导码长度范围为 0~65535，前导码越长数据包发送时间越长。

程序清单 6.13 radio_preamble_length_set 接口介绍

```
/**
* \brief 设置无线模块前导码长度
*
* \note 设置前请先把模块切换到 STANDBY 模式
*
* \param[in] handle : 无线模块操作句柄
* \param[in] size   : 前导码长度
*
* \retval RADIO_RET_OK            : 设置成功
* \retval RADIO_RET_PARAM_INVALID : 参数无效
```


©2022 Guangzhou ZHIYUAN Electronics Co., Ltd.

```
* \retval RADIO_RET_CFG_OUT_OF_RANGE : 前导码长度超出范围
*/
int radio_preamble_length_set (radio_handle_t handle, uint16_t size);
```

使用方法如下：

程序清单 6.14 radio_preamble_length_set 用法介绍

```
int ret;
/* 配置前请先把模块设置为 STANDBY 模式 */
radio_mode_set(handle, STANDBY_MODE);
ret = radio_preamble_length_set(handle, 20);
if (ret == RADIO_RET_OK) {
    //设置成功
} else {
    //设置失败，失败原因请查看接口返回值说明
}
```

6.8 获取前导码长度

该接口用于获取模块当前设置的前导码长度。

程序清单 6.15 radio_preamble_length_get 接口介绍

```
/**
* \brief 获取无线模块前导码长度
*
* \param[in] handle : 无线模块操作句柄
* \param[out] p_len : 前导码长度
*
* \retval RADIO_RET_OK            : 获取成功
* \retval RADIO_RET_PARAM_INVALID : 参数无效
*/
int radio_preamble_length_get (radio_handle_t handle, uint16_t *p_len);
```

使用方法如下：

程序清单 6.16 radio_preamble_length_get 用法介绍

```
int ret;
uint16_t len;
ret = radio_preamble_length_get(handle, &len);
if (ret == RADIO_RET_OK) {
    //获取成功
    printf("preamble lenght:%d\r\n", len);
}
```

6.9 设置功率

设置模块的发射功率，档位为 0~7，对应功率如下表。

表 6.1 功率档位说明表

功率档位	0	1	2	3	4	5	6	7
发射功率	-9	-5	-1	3	7	12	17	22

©2022 Guangzhou ZHIYUAN Electronics Co., Ltd.

程序清单 6.17 radio_tx_power_set 接口介绍

```
/**
* \brief 设置无线模块发射功率档位（0~7 档）
*
* \note 设置前请先把模块切换到 STANDBY 模式
*
* \param[in] handle   : 无线模块操作句柄
* \param[in] level    : 功率档位
* \param[out] p_power : 返回实际的设置功率
*
* \retval RADIO_RET_OK             : 设置成功
* \retval RADIO_RET_ERR            : 设置失败
* \retval RADIO_RET_PARAM_INVALID : 参数无效
*/
int radio_tx_power_set (radio_handle_t handle, uint8_t level, float *p_power);
```

使用方法如下：

程序清单 6.18 radio_tx_power_set 用法介绍

```
int ret;
float power;
/* 配置前请先把模块设置为 STANDBY 模式 */
radio_mode_set(handle, STANDBY_MODE);
ret = radio_tx_power_set(handle, 0, &power);
if (ret == RADIO_RET_OK) {
    //设置成功
    printf("tx power:%d\r\n", power);
}
```

6.10 获取运行状态

该接口用于获取模块当前的运行状态，运行状态有以下几种：

表 6.2 运行状态说明表

状态	值	说明
IDLE_ST	0	空闲状态。当模块配置为待机模式，睡眠模式，或单次接收模式接收超时，模块为空闲状态
RX_RUNING_ST	1	接收状态。当模块处于接收模式时，模块为接收状态
TX_RUNING_ST	2	发送状态。当模块正在发送数据时，模块为发送状态
CAD_ST	3	CAD 状态。当模块开启 CAD 检测时，模块为 CAD 状态

程序清单 6.19 radio_state_get 接口介绍

```
/**
* \brief 获取无线模块当前状态
*
* \param[in] handle   : 无线模块操作句柄
* \param[out] p_state : 模块状态
*
* \retval RADIO_RET_OK                  : 获取成功
* \retval RADIO_RET_PARAM_INVALID       : 参数无效
* \retval RADIO_RET_MODULE_NOT_SUPPORT : 该模块不支持该模式
*/
int radio_state_get (radio_handle_t handle, radio_state_t *p_state);
```


©2022 Guangzhou ZHIYUAN Electronics Co., Ltd.

使用方法如下：

程序清单 6.20 radio_state_get 用法介绍

```
int ret;
radio_state_t state;
ret = radio_state_get(handle, &state);
if (ret == RADIO_RET_OK) {
    //获取成功
    if (state == IDLE_ST) {
        //空闲状态
    } else if (state == RX_RUNING_ST) {
        //接收状态
    } else if (state == TX_RUNING_ST) {
        //CAD 状态
    }
}
```

6.11 无线模块控制

用户可以使用无线模块控制接口配置模块参数。无线模块控制接口以命令 + 参数的形式来使用。

程序清单 6.21 radio_ioctl 接口介绍

```
/**
* \brief 无线模块控制函数
*
* \note 设置前请先把模块切换到 STANDBY 模式
*
* \param[in] handle : 无线模块操作句柄
* \param[in] cmd    : 控制命令
* \param[in] p_arg  : 该命令对应的参数
*
* \retval RADIO_RET_OK              : 命令执行成功
* \retval RADIO_RET_ERR             : 命令执行失败
* \retval RADIO_RET_PARAM_INVALID   : 参数无效
* \retval RADIO_RET_CMD_NOT_SUPPORT : 不支持该命令
*/
int radio_ioctl (radio_handle_t handle, int cmd, void *p_arg);
```

使用方法如下：

程序清单 6.22 radio_ioctl 用法介绍

```
/* 配置前请先把模块设置为 STANDBY 模式 */
radio_mode_set(handle, STANDBY_MODE);
/* 设置带宽 */
ret = radio_ioctl(handle, BW_SET, (void*)8);
if (ret == RADIO_RET_OK) {
    //设置成功
}
/* 设置扩频因子 */
ret = radio_ioctl(handle, SF_SET, (void*)8);
if (ret == RADIO_RET_OK) {
    //设置成功
}
```


©2022 Guangzhou ZHIYUAN Electronics Co., Ltd.

7. 无线模块使用过程中常见问题总结

7.1 客户板子发送，DEMO 板接收不到数据

1. 首先确认在调用 radio_buf_send() 函数发送完成时是否有出现卡死在该接口的情况。如果卡死了请参考步骤 2，没有卡死请参考步骤 3。

2. 确认初始化是否成功，确认该问题最直接的方式是判断初始化时 radio_zm4xx_inst_init() 返回的 handle 地址是否为 NULL，如果为 NULL 则初始化不成功。初始化不成功最主要的问题是 SPI 无法正常读写，可以先确认 SPI 的速率是否超过了 10Mhz，确认 SPI 配置是否符合要求。可以通过逻辑分析仪抓取 SPI 的 SEL、SCK、MOSI 和 MISO 四条线的数据，通过波形查看 SCK 的速率，SEL 是否能正常拉高拉低，MOSI 是否有按照程序输出了想要的数据，MISO 是否有返回了数据。如果 MOSI 输出的数据都是正确的，MISO 依然没有任何数据返回，可以更换模块试试，模块可能出现了硬件问题。

3. 确认是否有调用 radio_xxx 等接口配置了其它参数，如果要往 DEMO 板发送数据，基本只需要初始化 radio_zm4xx_inst_init() 完成后配置一下 radio_freq_set() 频率即可往 DEMO 板发送数据，切勿再配置其它参数。

4. 如果以上都没有问题，请使用逻辑分析仪抓取 RST 引脚，RST 引脚只会在 radio_zm4xx_inst_init() 初始化时有一次拉低，之后都应处于高电平。有客户遇到 RST 引脚在没有程序的操作时也会有电平不断跳变，因为 RST 是复位引脚，所以该引脚不能在初始化完成后还有电平变化，这是不正常的，会影响到模块正常工作。

5. 请测试 mcu_gpio_rst_pin_set()，mcu_gpio_sel_pin_set() 是否能正常运行。测试方法可以在外部写个例程，例如运行 mcu_gpio_rst_pin_set(1) 然后用万用表测试该引脚是否是高电平，运行 mcu_gpio_rst_pin_set(0) 是否是低电平。

6. 请测试无线模块的 VCC 引脚电压是否过低。

7. 每次发送不能超过 255byte，否则驱动会忽略该数据包。

7.2 初始化失败，句柄返回为 NULL

这种情况很可能是 SPI 和 GPIO(引脚操作) 问题。1. 确认 SPI 初始化是否有问题。SPI 速率要确保在要求范围内，SPI 模式为 CPOL=0 和 CPHA=0，SPI 速率 <=10Mhz。2. 确认 SPI 读写函数内部没有操作 CS 引脚，SPI 写函数内部要判断数据传输完成才能退出。3. 确认 CS 引脚操作函数能否能使引脚输出高低电平。

7.3 通信距离短

1. 请确认天线是否匹配，天线对通信的距离影响较大。2. 确认模块的频点，如果是 470 模块，最佳频点为 470Mhz。3. 确认模块的发射功率是否已经配置为最大。

7.4 接收不到数据

1. 确认双方的配置参数是否一致，如频率、带宽、扩频因子和前导码长度等。2. 确认接收端是否已进入接收模式。3. 如果采用在引脚中断服务函数里接收数据，要确认中断引脚是否正确，引脚中断触发条件是否正确，无线模块的 DIO1 引脚是否配置为上升沿中断。4. 确

©2022 Guangzhou ZHIYUAN Electronics Co., Ltd.

认当前是否是单次接收模式，单次接收模式如果超时接收不到数据将切换为待机模式，此时需要用户重新切换到接收模式。

7.5 接收一次数据之后就再也接收不到了

确认是否设置了单次接收模式，单次接收模式需要在 RX timeout 的回调函数里重新切换到接收模式或者在接收完成一包数据之后重新切换到接收模式才能使它持续处于接收状态。如果有持续接收的需要，建议使用连续接收模式。

7.6 使用 CAD 需要注意的问题

1. 如果设置为连续接收模式，当检测到一次 CAD 检测成功之后模块将一直处于接收模式，无法自动回到待机模式以达到降低功耗的目的，所以使用 CAD 请设置为单次接收模式。2.CAD 的开启周期由用户决定，而 CAD 开启的窗口大小由扩频因子和带宽决定。3.CAD 检测不到有效信号时会自动回到待机模式。

7.7 CAD 只能检测到 CAD Done 中断而没有 CAD Detect 中断

CAD Done 表示一次 CAD 检测完成，如果检测到有效前导码则会产生 CAD Detect 信号，所以在没有检测到 CAD Detect 信号，应该重新开启 CAD。很多用户误以为只要一开始开启 CAD 模就一直处于 CAD 模式，其实开启一次检测完一次模块就自动恢复为 Standby 模式，所以如果要做到定时检测就要间隔一定时间就重新打开 CAD，提高 CAD 检测到有效数据的概率。如果 CAD 一直无 detect 中断应检查是否发送端发送数据包的间隔是否太长，导致空中大多数时间处于无信号状态。

7.8 发现频率相差 1Mhz、5Mhz，CAD 还能产生 CAD Detect 标志

因为 Lora 芯片信号的衰减和信道隔离能力有限，如果收发双方距离较近则会出现 CAD Detect 标志被误触发。有时即使不发送数据，CAD 也会因为空气中其它干扰而出现这种情况。这些 CAD 被误触发的情况芯片都不会认为是有效数据而把其接收下来，所以用户不必做其它处理，芯片只是被短暂唤醒之后会马上回到待机模式以降低功耗。如果是 CAD Detect 被频繁误触发，可以做以下处理：1. 把收发两端距离拉远；2. 适当降低发送功率；3. 把频率间隔加大。

7.9 丢包问题

考虑到接收端处理数据需要一定时间，所以发送端应间隔一定时间再发送下一包数据，防止接收端无法及时数据包而导致丢失。

7.10 发送时间长

1. 确认是否前导码过长。2. 确认带宽设置是否太小。3. 确认扩频因子是否太大。4. 确认编码率是否太大。

©2022 Guangzhou ZHIYUAN Electronics Co., Ltd.

7.11 什么时候接收数据

当模块处于 RX 模式，DIO1 中断发生时，接收到的数据就已经被读取到内部的缓冲区，此时可以调用 radio_buf_recv() 来拷贝数据。

7.12 A 发送 B 能接收，反之无法通信

1. 可能是前导码不一致，长前导码配置能兼容短前导码配置，反之无法识别。2. 可能是 A 未开启 CRC，B 开启了 CRC。

7.13 开启了硬件 CRC 但是收到了错误的数据包

LoRa 的硬件 CRC 并不能做到 100% 的检测出错误的数据包，需要用户自己对数据再做一个软件的 CRC 校验，也可以开启驱动里自带的软件 CRC 校验。

7.14 没有发送数据，LoRa 却收到了乱码

LoRa 灵敏度较高，再加上不完全可靠的硬件 CRC，所以导致了会将空中的干扰信号误认为是有效信号而接收下来，可以使用开启软件 CRC 来解决。

8. 免责声明

ZM4xx 软件免责声明

2.1 版本

广州致远电子有限公司（以下简称“ZLG 公司”）为了更好地服务用户，针对 ZM4xx 软件的使用进行如下说明：

1、ZM4xx 软件由本目录下的 ZLG 公司自主开发的软件和第三方软件组成（如有），除了需要单独授权的 ZLG 公司自主开发的软件外，用户可以免费在商业产品中使用本目录下的 ZLG 公司自主开发的软件，且无需开源用户产品代码；对于第三方软件，用户需要自行理解及满足其权利要求，第三方软件权利要求以其版权文件及源代码中的版权声明为准，ZLG 公司不承担用户使用第三方软件相关的任何责任。

2、ZLG 公司是 ZM4xx 软件中由 ZLG 公司自主开发的软件的知识产权权利人，本软件的一切软件著作权、商标权、专利权、商业秘密等知识产权和其他权利，以及与本软件相关的所有信息内容（包括但不限于文字、图片、音频、视频、图表、界面设计、版面框架、有关数据或电子文档等）均受中华人民共和国法律法规和相应的国际条约保护，未经权利人书面同意，用户不得为任何商业或非商业目的自行或许可任何第三方实施、利用、转让上述知识产权或其他权利，以及与本软件相关的所有信息内容。

3、ZLG 公司建议用户搭配由 ZLG 公司官方发布的硬件产品使用 ZM4xx 软件，ZLG 公司不保证 ZM4xx 软件与 ZLG 公司官方发布的硬件产品之外的其他品牌或型号的硬件适配；若用户搭配其他品牌或型号的硬件使用，应自行承担相应的风险、后果及法律责任。

4、用户因第三方如电信部门的通讯线路故障、技术问题、网络问题、电脑故障、系统不稳定性及与使用 ZM4xx 软件相关的任何设备、系统或网络受到入侵或攻击而遭受损失的，ZLG 公司不承担任何责任。

广州致远电子有限公司 wireless LoRa 无线模块

©2022 Guangzhou ZHIYUAN Electronics Co., Ltd.

5、除法律另有明确规定外，ZLG 公司不保证 ZM4xx 软件无错误或不间断地运行，也不保证 ZLG 公司一次性纠正所有错误。对于 ZLG 公司官方发布的 ZM4xx 相关软件产品，在其产品生命周期内，ZLG 公司会积极修复发现的 ZM4xx 相关软件缺陷问题（如有），并及时更新发布。在技术支持有效期内，ZLG 公司会积极协助用户分析产品问题和故障原因，并及时修复发现的 ZM4xx 相关软件的缺陷问题。

6、ZLG 公司对 ZM4xx 软件不作任何类型的承诺或保证，包括但不限于 ZM4xx 软件的适销性、适用性、无病毒、无疏忽或无技术瑕疵问题、所有权以及无侵权的明示或默示担保。对用户在任何情况下因使用或不能使用 ZM4xx 软件所产生的直接、间接、偶然、特殊及后续的损害及风险，ZLG 公司不承担任何责任。

7、用户违反本说明书的规定，导致或产生的任何第三方主张的任何索赔、要求或损失，包括但不限于合理的诉讼费用和律师费用等，用户同意赔偿 ZLG 公司由此导致的一切损失。ZLG 公司有权视用户的行为性质，采取包括但不限于中断使用许可、停止提供服务、限制使用、追究法律责任等措施。

广州致远电子 wireless LoRa 无线模块

©2022 Guangzhou ZHIYUAN Electronics Co., Ltd.

销售与服务网络

广州致远电子有限公司

地址：广州市天河区车陂路黄洲工业区 7 栋 2 楼
邮编：510660
网址：www.zlg.cn

全国服务热线电话：400-888-4005

广州总公司
广州市天河区车陂路黄洲工业区 7 栋 2 楼

上海分公司
上海市北京东路 668 号科技京城东楼 12E 室

北京分公司
北京市丰台区马家堡路 180 号蓝光云鼎 208 室

深圳分公司
深圳市宝安区新安街道海秀路 21 号龙光世纪大厦 A 座 1205

武汉分公司
武汉市洪山区民族大道江南家园 1 栋 3 单元 602 室

南京分公司
南京市秦淮区汉中路 27 号友谊广场 17 层 F、G 区

杭州分公司
杭州市西湖区紫荆花路 2 号杭州联合大厦 A 座 4 单元 508 室

成都分公司
四川省成都市高新技术开发区天府大道中段 500 段 1 号东方希望天祥广场 1 栋 C 座 3521 室（地铁世纪城站 B 出口）

郑州分公司
河南省郑州市中原区建设西路 118 号 1 号楼 3 单元 13 层 1302 室（华亚广场）

重庆分公司
重庆市渝北区龙溪街道新溉大道 18 号山顶国宾城 11 幢 4-14

西安办事处
西安市长安北路 54 号太平洋大厦 1201 室

天津办事处
天津市河东区津塘路与十一经路交口鼎泰大厦 1004

青岛办事处
山东省青岛市李沧区枣园路 11 号银座华府 1 号楼 2 单元 1901 室

请您用以上方式联系我们，我们会为您安排样机现场演示，感谢您对我公司产品的关注！

广州致远电子 wireless LoRa 无线模块

©2022 Guangzhou ZHIYUAN Electronics Co., Ltd.