AN1304

应用笔记

UM2010 API 使用指南

版本：V1.0

UNICMICRO

广芯微电子

广芯微电子（广州）股份有限公司

http://www.unicmicro.com/

条款协议

本文档的所有部分，其著作产权归广芯微电子（广州）股份有限公司（以下简称广芯微电子）所有，未经广芯微电子授权许可，任何个人及组织不得复制、转载、仿制本文档的全部或部分组件。本文档没有任何形式的担保、立场表达或其他暗示，若有任何因本文档或其中提及的产品所有资讯所引起的直接或间接损失，广芯微电子及所属员工恕不为其担保任何责任。除此以外，本文档所提到的产品规格及资讯仅供参考，内容亦会随时更新，恕不另行通知。

1. 本文档中所记载的关于电路、软件和其他相关信息仅用于说明半导体产品的操作和应用实例。用户如在设备设计中应用本文档中的电路、软件和相关信息，请自行负责。对于用户或第三方因使用上述电路、软件或信息而遭受的任何损失，广芯微电子不承担任何责任。
2. 在准备本文档所记载的信息的过程中，广芯微电子已尽量做到合理注意，但是，广芯微电子并不保证这些信息都是准确无误的。用户因本文档中所记载的信息的错误或遗漏而遭受的任何损失，广芯微电子不承担任何责任。
3. 对于因使用本文档中的广芯微电子产品或技术信息而造成的侵权行为或因此而侵犯第三方的专利、版权或其他知识产权的行为，广芯微电子不承担任何责任。本文档所记载的内容不应视为对广芯微电子或其他人所有的专利、版权或其他知识产权作出任何明示、默示或其它方式的许可及授权。
4. 使用本文档中记载的广芯微电子产品时，应在广芯微电子指定的范围内，特别是在最大额定值、电源工作电压范围、热辐射特性、安装条件以及其他产品特性的范围内使用。对于在上述指定范围之外使用广芯微电子产品而产生的故障或损失，广芯微电子不承担任何责任。
5. 虽然广芯微电子一直致力于提高广芯微电子产品的质量和可靠性，但是，半导体产品有其自身的具体特性，如一定的故障发生率以及在某些使用条件下会发生故障等。此外，广芯微电子产品均未进行防辐射设计。所以请采取安全保护措施，以避免当广芯微电子产品在发生故障而造成火灾时导致人身事故、伤害或损害的事故。例如进行软硬件安全设计（包括但不限于冗余设计、防火控制以及故障预防等）、适当的老化处理或其他适当的措施等。

目录

1 UM2010 移植

UM2010 的示例代码已封装相关驱动接口，把 UM2010 文件夹的所有文件拷贝到项目工程文件下，在移植过程中仅需要移植实现 um2010_hal 硬件接口，其他文件无需修改。

注：所有的示例代码下的 UM2010 驱动文件接口都一样，仅可能存在 rfconfig.h 不一样，rfconfig.h 文件可由 RFOCT 上位机导出，可直接替换。

1.1 文件准备

1) UM2010 驱动文件

路径：/ UM2010/*

描述：UM2010 文件夹包含以下文件:

- radio.h : 包括初始化、发射数据包、接收数据包（Mode2）
- radio.c : 实现初始化、发射数据包、接收数据包（Mode2）
- um2010.h : UM2010 驱动文件接口
- um2010.c : 实现 UM2010 驱动
- rfconfig.h : 配置文件，修改配置直接替换，由上位机 RFOCT 导出
- typedefs.h : 类型定义
- um2010_defs.h : UM2010 的宏定义
- um2010_hal.h : UM2010 硬件接口
- um2010_hal.c : UM2010 硬件接口实现

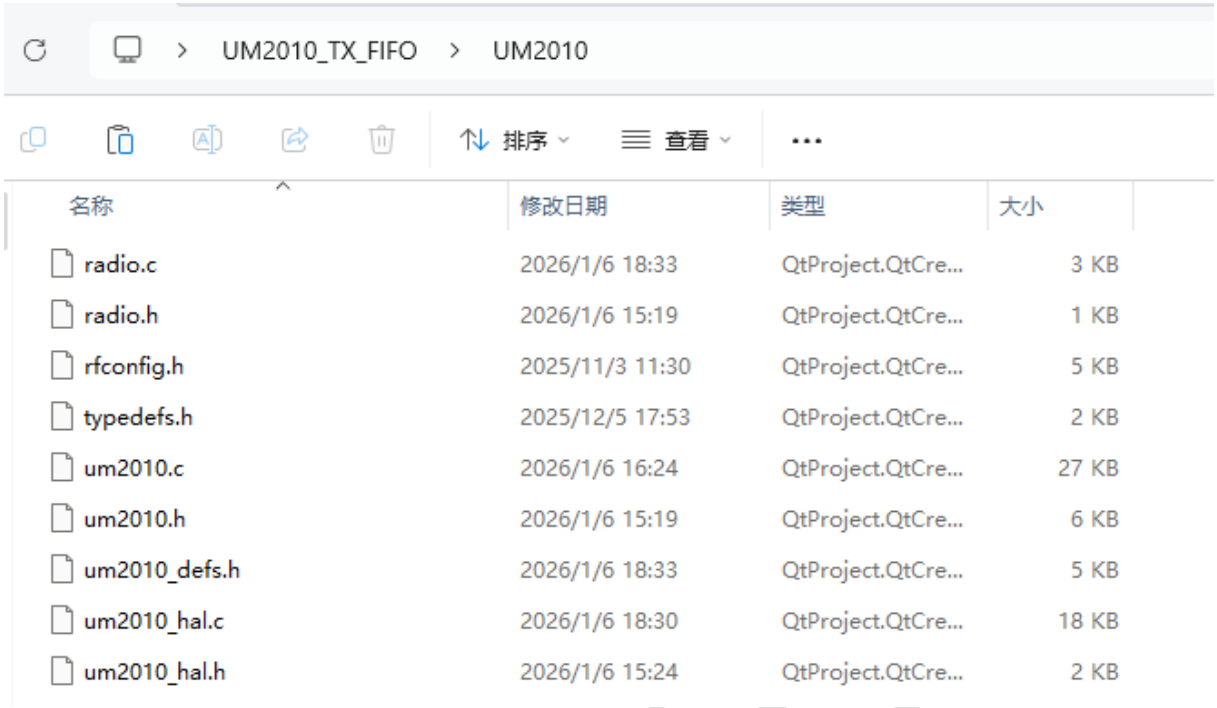

图 1-1：UM2010 驱动文件夹的文件

1.2 硬件接口

1.2.1 um2010_hal 介绍

移植文件（um2010_hal）的硬件接口介绍：

1. void um2010_hal_delay_us(u16_t us)

   UM2010 延时，实现μs 延时。

2. void um2010_hal_set_sdn(em_level_t level)

   SDN 控制，实现 SDN 的拉高关断芯片和拉低使芯片正常工作。

3. void um2010_hal_set_rst(em_level_t level)

   RST 控制，实现 RST 的拉低使芯片处于复位状态和拉高使芯片正常工作。

4. void um2010_hal_init(void)

   硬件初始化，实现以下功能：

   - 实现配置中断输出连接的 GPIO 初始化，并使能中断，采用单边沿的上升沿中断。

- 实现控制 SDN 的 GPIO 的初始化功能，并拉低 SDN 启动让芯片处于工作中。
- 实现控制 RST 的 GPIO 的初始化功能，并拉高 RST 使芯片处于工作状态。

5. void spi_init(void)

   SPI 初始化

6. void spi_cs_enable(void)

   SPI CS 使能，实现 CS 的拉低功能。

7. void spi_cs_disable(void)

   SPI CS 不使能，实现 CS 的拉高功能。

8. void spi_write_byte(u8_t byte)

   SPI 写一个字节，实现 SPI 时序写一个字节。

9. u8_t spi_read_byte(void)

   SPI 读一个字节，实现 SPI 时序读一个字节。

1.2.2 um2010_hal 移植

UM2010 的移植仅需要移植 um2010_hal 文件，UM2010_hal 文件介绍了 3 种 SPI 的移植方法，分别为四线硬件 SPI、四线软件模拟 SPI，三线软件模拟 SPI，可根据实际应用选择合适的 SPI 进行移植，示例代码中，通过 UM2010_hal.c 文件的 UM2010_HAL_SPI_MODE 定义区分 SPI 移植方法其中 0 代表四线硬件 SPI，1 代表四线软件 SPI，2 代表三线软件 SPI。下面分别对 UM2010_hal 的三种移植过程进行介绍：

1.2.2.1 四线硬件 SPI

四线硬件 SPI 移植需要实现以下功能：

1. 硬件相关宏定义

```
#define UM2010_HAL_DELAY_US(us)          /* 延迟函数 us */
#define UM2010_HAL_SDN_HIGH              /* 拉高 SDN 电平 */
```

```
#define UM2010_HAL_SDN_LOW                  /*  拉低 SDN 电平  */
#define UM2010_HAL_RST_HIGH                 /*  拉高 RST 电平  */
#define UM2010_HAL_RST_LOW                  /*  拉低 RST 电平  */
```

2. 硬件初始化

```
void um2010_hal_init(void)
{
    /* SDN 的 GPIO 初始化，设置为输出、低电平  */
    /* RST 的 GPIO 初始化，设置为输出、高电平  */
    /* nIRQ 的 GPIO 初始化，设置为输入下拉，单边上升沿，中断调用 um2010_hal_irq */

    /*  使能 SDN 并等待芯片上电稳定  */
    um2010_hal_set_sdn(LEVEL_LOW);          /*  使能 SDN */
    um2010_hal_set_rst(LEVEL_HIGH);         /*  正常工作  */
    um2010_hal_delay_us(5000);              /*  等待芯片稳定  */
}
```

3. SPI 初始化

```
void spi_init(void)
{
    /*  硬件 SPI 的初始化，SPI 采用 Mode1（CPOL=0，CPHA=1）*/
}
```

4. SPI 的 CS 使能

```
void spi_cs_enable(void)
{
    /* SPI 的 CS 使能  */
}
```

5. SPI 的 CS 不使能

```
void spi_cs_disable(void)
{
    /* SPI 的 CS 不使能 */
}
```

6. SPI 写一个字节

```
void spi_write_byte(u8_t byte)
{
    /* SPI 写一个字节 */
}
```

7. SPI 读一个字节

```
u8_t spi_read_byte(void)
{
    /* SPI 读一个字节 */
}
```

1.2.2.2 四线软件模拟 SPI

四线软件模拟 SPI 移植需要实现以下功能:

1. 硬件相关宏定义

```
#define UM2010_HAL_DELAY_US(us)          /* 延迟函数 us */
#define UM2010_HAL_SDN_HIGH              /* 拉高 SDN 电平 */
#define UM2010_HAL_SDN_LOW               /* 拉低 SDN 电平 */
#define UM2010_HAL_RST_HIGH              /* 拉高 RST 电平 */
#define UM2010_HAL_RST_LOW               /* 拉低 RST 电平 */
```

2. 硬件初始化

```
void um2010_hal_init(void)
{
```

```
    /* SDN 的 GPIO 初始化，设置为输出、低电平 */
    /* RST 的 GPIO 初始化，设置为输出、高电平 */
    /* nIRQ 的 GPIO 初始化，设置为输入下拉，单边上升沿，中断调用 um2010_hal_irq */

    /* 使能 SDN 并等待芯片上电稳定 */
    um2010_hal_set_sdn(LEVEL_LOW);          /* 使能 SDN */
    um2010_hal_set_rst(LEVEL_HIGH);         /* 正常工作 */
    um2010_hal_delay_us(5000);              /* 等待芯片稳定 */
}
```

3. SPI 相关宏定义

```
#define UM2010_HAL_SPI_CS_ENABLE        /* 拉低 CS 电平 */
#define UM2010_HAL_SPI_CS_DISABLE       /* 拉高 CS 电平 */
#define UM2010_HAL_SPI_CLK_LOW          /* 拉低 CLK 电平 */
#define UM2010_HAL_SPI_CLK_HIGH         /* 拉高 CLK 电平 */
#define UM2010_HAL_SPI_MOSI_LOW         /* 拉低 MOSI 电平 */
#define UM2010_HAL_SPI_MOSI_HIGH        /* 拉高 MOSI 电平 */
#define UM2010_HAL_SPI_MISO_GET         /* 获取 MISO 电平 */
#define UM2010_HAL_SPI_DELAY            /* SPI 延迟，用于降低 SPI 速率 */
```

4. SPI 初始化

```
void spi_init(void)
{
    /* CS 初始化，GPIO 设为输出，使能上拉，并输出高电平 */
    /* CLK 初始化，GPIO 设为输出，使能下拉，并输出低电平 */
    /* MOSI 初始化，GPIO 设为输出，并输出高电平 */
    /* MISO 初始化，GPIO 设为输入，并使能输入 */
}
```

1.2.2.3 三线软件模拟 SPI

三线软件模拟 SPI 移植需要实现以下功能:

1. 硬件相关宏定义

```
#define UM2010_HAL_DELAY_US(us)          /* 延迟函数 us */
#define UM2010_HAL_SDN_HIGH              /* 拉高 SDN 电平 */
#define UM2010_HAL_SDN_LOW               /* 拉低 SDN 电平 */
#define UM2010_HAL_RST_HIGH              /* 拉高 RST 电平 */
#define UM2010_HAL_RST_LOW               /* 拉低 RST 电平 */
```

2. 硬件初始化

```
void um2010_hal_init(void)
{
    /* SDN 的 GPIO 初始化，设置为输出、低电平 */
    /* RST 的 GPIO 初始化，设置为输出、高电平 */
    /* nIRQ 的 GPIO 初始化，设置为输入下拉，单边上升沿，中断调用 um2010_hal_irq */

    /* 使能 SDN 并等待芯片上电稳定 */
    um2010_hal_set_sdn(LEVEL_LOW);       /* 使能 SDN */
    um2010_hal_set_rst(LEVEL_HIGH);      /* 正常工作 */
    um2010_hal_delay_us(5000);           /* 等待芯片稳定 */
}
```

3. SPI 相关宏定义

```
#define UM2010_HAL_SPI_CS_ENABLE         /* 拉低 CS 电平 */
#define UM2010_HAL_SPI_CS_DISABLE        /* 拉高 CS 电平 */
#define UM2010_HAL_SPI_CLK_LOW           /* 拉低 CLK 电平 */
#define UM2010_HAL_SPI_CLK_HIGH          /* 拉高 CLK 电平 */
```



```
#define UM2010_HAL_SPI_MOSI_INPUT          /*  设置 MOSI 为输入  */
#define UM2010_HAL_SPI_MOSI_OUTPUT         /*  设置 MOSI 为输出  */
#define UM2010_HAL_SPI_MOSI_LOW            /*  拉低 MOSI 电平  */
#define UM2010_HAL_SPI_MOSI_HIGH           /*  拉高 MOSI 电平  */
#define UM2010_HAL_SPI_MOSI_GET            /*  获取 MOSI 电平  */
#define UM2010_HAL_SPI_DELAY               /* SPI 延迟，用于降低 SPI 速率  */
```

4. SPI 初始化

```
void spi_init(void)
{
    /* CS 初始化，GPIO 设为输出，使能上拉，并输出高电平  */
    /* CLK 初始化，GPIO 设为输出，使能下拉，并输出低电平  */
    /* MOSI 初始化，GPIO 设为输出，并输出高电平  */

    /*  使能三线 SPI */
    spi_cs_enable();
    spi_write_byte(UM2010_REG57&UM2010_REG_WRITE);
    spi_write_byte(UM2010_REG57_SPI_3W);
    spi_cs_disable();
}
```

1.3 TX 流程

UM2010 发射数据流程:

- radio_init 初始化
- radio_send_data 发射数据

详细流程图如下:

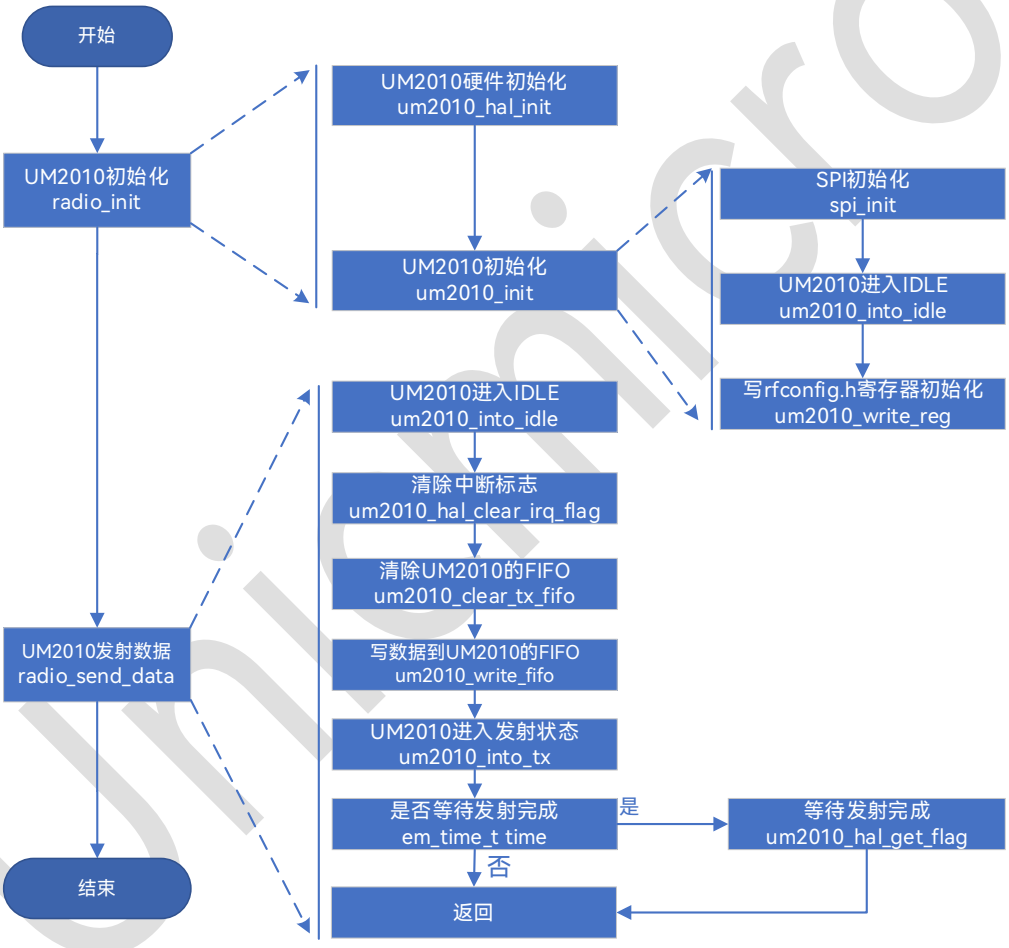

图 1-2: 发射数据流程图

1.4 RX 流程

UM2010 接收数据流程:

- radio_init 初始化

- um2010_into_rx 进入接收
- radio_recv_data 接收数据

详细接收流程图如下:

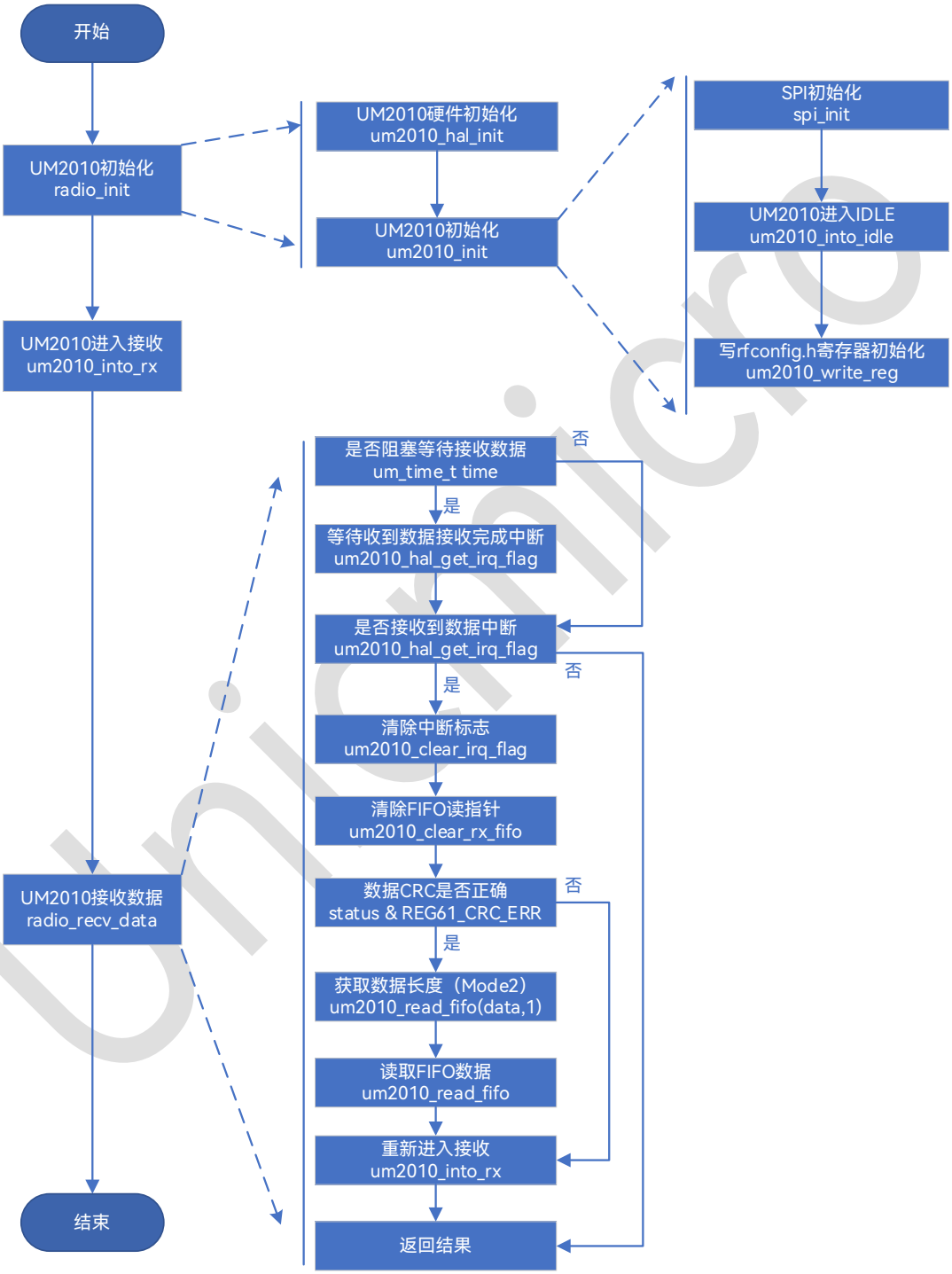

图 1-3：接收数据流程图

2 Radio 接口定义

#include “radio.h”

2.1 radio_init

函数功能	radio_init
函数描述	Radio 初始化
函数定义	em_ret_t radio_init(void)
输入参数	None
输出参数	None
函数返回	em_ret_t 返回初始化结果： ➢ RET_OK：返回成功 ➢ RET_ERR：返回失败

2.2 radio_send_data

函数功能	radio_send_data
函数描述	radio 发射一包数据
函数定义	em_ret_t radio_send_data(u8_t *dat,u8_t len,em_time_t time)
输入参数	u8_t *dat：发射的数据 u8_t len：发射数据长度（长度不能超过 FIFO 长度，如超过 FIFO 长度需要使用 FIFO Empty 配合边发射边写入 FIFO） em_time_t time：发射是否阻塞 ➢ TIME_WAITTING_FOREVER：阻塞方式 ➢ TIME_WAITTING_NO：非阻塞方式
输出参数	None
函数返回	em_ret_t 返回初始化结果： ➢ RET_OK：返回成功 ➢ RET_ERR：返回失败

2.3 radio_recv_data

函数功能	radio_recv_data
函数描述	radio 接收一包数据
函数定义	em_ret_t radio_recv_data(u8_t *dat,u8_t *len,em_time_t time)
输入参数	em_time_t time：是否阻塞接收数据 ➢ TIME_WAITTING_FOREVER：阻塞方式 ➢ TIME_WAITTING_NO：非阻塞方式
输出参数	u8_t *dat：接收到的数据 u8_t *len：接收到数据的长度
函数返回	em_ret_t 返回初始化结果： ➢ RET_OK：返回成功 ➢ RET_ERR：返回失败

3 UM2010 接口定义

#include “um2010.h”

3.1 um2010_init

函数功能	um2010_init
函数描述	初始化 UM2010，对 UM2010 进行寄存器配置
函数定义	em_ret_t um2010_init(void)
输入参数	None
输出参数	None
函数返回	em_ret_t 返回初始化结果： ➢ RET_OK：返回成功 ➢ RET_ERR：返回失败

3.2 um2010_write_reg

函数功能	um2010_write_reg
函数描述	写单个寄存器
函数定义	void um2010_write_reg(u8_t addr,u8_t value)
输入参数	u8_t addr：寄存器地址 u8_t value：寄存器值
输出参数	None
函数返回	None

3.3 um2010_read_reg

函数功能	um2010_read_reg
函数描述	读寄存器
函数定义	u8_t um2010_read_reg(u8_t addr)
输入参数	u8_t addr：寄存器地址
输出参数	None
函数返回	u8_t 返回寄存器值

3.4 um2010_clear_tx_fifo

函数功能	um2010_clear_tx_fifo
函数描述	清空 FIFO 写指针
函数定义	void um2010_clear_tx_fifo(void)
输入参数	None
输出参数	None
函数返回	None

3.5 um2010_write_fifo

函数功能	um2010_write_fifo
函数描述	写 FIFO 数据
函数定义	void um2010_write_fifo(u8_t *dat,u16_t len)
输入参数	u8_t *dat：写 FIFO 的数据 u16_t len：写 FIFO 的长度
输出参数	None
函数返回	None

3.6 um2010_clear_rx_fifo

函数功能	um2010_clear_rx_fifo
函数描述	清除 FIFO 读指针
函数定义	void um2010_clear_rx_fifo(void)
输入参数	None
输出参数	None
函数返回	None

3.7 um2010_read_fifo

函数功能	um2010_read_fifo
函数描述	读 FIFO 数据
函数定义	void um2010_read_fifo(u8_t *dat,u16_t len)
输入参数	u16_t len：读 FIFO 的长度
输出参数	u8_t *dat：读 FIFO 的数据
函数返回	None

3.8 um2010_fifo_merge

函数功能	um2010_fifo_merge
函数描述	RX_FIFO 和 TX_FIFO 共享使用，在 RX 状态或 TX 状态作为 256 字节 FIFO 使用
函数定义	void um2010_fifo_merge(void)
输入参数	None
输出参数	None
函数返回	None

3.9 um2010_fifo_notmerge

函数功能	um2010_fifo_notmerge
函数描述	RX_FIFO 和 TX_FIFO 独立使用，分别为 128 个字节
函数定义	void um2010_fifo_notmerge(void)
输入参数	None
输出参数	None
函数返回	None

3.10 um2010_set_fifo_full_thres

函数功能	um2010_set_fifo_full_thres
函数描述	FIFO 满门限，芯片接收数据写入 FIFO 时，FIFO 剩余的数据空间低于 value 个字节时产生 fifo_flag 中断
函数定义	void um2010_set_fifo_full_thres(u8_t value)
输入参数	u8_t value：FIFO 剩余的数据空间
输出参数	None
函数返回	None

3.11 um2010_set_fifo_empty_thres

函数功能	um2010_set_fifo_empty_thres
函数描述	FIFO 空门限，芯片发射数据时，FIFO 剩余的数据低于 value 个字节时产生 fifo_flag 中断
函数定义	void um2010_set_fifo_empty_thres(u8_t value)
输入参数	u8_t value：FIFO 剩余的数据空间
输出参数	None
函数返回	None

3.12 um2010_into_idle

函数功能	um2010_into_idle
函数描述	进入 IDLE 模式
函数定义	void um2010_into_idle(void)
输入参数	None
输出参数	None
函数返回	None

3.13 um2010_into_tx

函数功能	um2010_into_tx
函数描述	进入 TX 模式
函数定义	void um2010_into_tx(void)
输入参数	None
输出参数	None
函数返回	None

3.14 um2010_into_rx

函数功能	um2010_into_rx
函数描述	进入 RX 模式
函数定义	void um2010_into_rx(void)
输入参数	None
输出参数	None
函数返回	None

3.15 um2010_into_standby

函数功能	um2010_into_sleep
函数描述	进入 Standby 模式
函数定义	void um2010_into_standby(void)
输入参数	None
输出参数	None
函数返回	None

3.16 um2010_into_fson

函数功能	um2010_into_fson
函数描述	进入 FSON 模式
函数定义	void um2010_into_fs(void)
输入参数	None
输出参数	None
函数返回	None

3.17 um2010_tx_carrier

函数功能	um2010_tx_carrier
函数描述	发射载波
函数定义	void um2010_tx_carrier (void)
输入参数	None
输出参数	None
函数返回	None

3.18 um2010_tx_modulate

函数功能	um2010_tx_modulate
函数描述	发射调制信号
函数定义	void um2010_tx_modulate (void)
输入参数	None
输出参数	None
函数返回	None

3.19 um2010_tx_direct_gpio

函数功能	um2010_tx_direct_gpio
函数描述	直通发射，数据通过 GPIO 输入（GPIO2 不支持）
函数定义	void um2010_tx_direct_gpio(em_gpio_t gpio)
输入参数	em_gpio_t gpio：GPIO 引脚 ➢ NIRQ ➢ GPIO0 ➢ GPIO1
输出参数	None
函数返回	None

3.20 um2010_rx_direct_gpio

函数功能	um2010_rx_direct_gpio
函数描述	直通接收，数据从 GPIO 输出
函数定义	void um2010_rx_direct_gpio(em_gpio_t gpio)
输入参数	em_gpio_t gpio：GPIO 引脚 ➢ NIRQ ➢ GPIO0 ➢ GPIO1 ➢ GPIO2
输出参数	None
函数返回	None

3.21 um2010_set_gpio_sel

函数功能	um2010_set_gpio_sel
函数描述	设置芯片 NIRQ、GPIO0、GPIO1、GPIO2 引脚输出信号
函数定义	void um2010_set_gpio_sel(em_gpio_t gpio,em_gpio_sel sel)
输入参数	em_gpio_t gpio：GPIO 引脚 ➢ NIRQ

	➢ GPIO0 ➢ GPIO1 ➢ GPIO2 em_gpio_sel sel：输出信号 ➢ GPIO_SEL_NIRQ：中断信号（包括数据包完成中断 pkt_int，接收前导码中断 preamble_int，接收同步字中断 syncword_int） ➢ GPIO_SEL_PKT：包完成中断，包含接收完成和发射完成 ➢ GPIO_SEL_PREAMBLE：接收 Preamble 有效中断 ➢ GPIO_SEL_SYNCWORD：接收 Syncword 有效中断 ➢ GPIO_SEL_FIFO：RX FIFO 快满中断，TX FIFO 快空中断 ➢ GPIO_SEL_RX_DATA：解调器输出的串行数据 ➢ GPIO_SEL_TX_DATA：进入发射调制的串行数据 ➢ GPIO_SEL_TR_SW：发射使能标志 ➢ GPIO_SEL_NTR_SW：反向发射使能标志 ➢ GPIO_SEL_HIGH：高电平 ➢ GPIO_SEL_LOW：低电平
输出参数	None
函数返回	None

3.22 um2010_set_gpio_brclk_sel

函数功能	um2010_set_gpio_brclk_sel
函数描述	nIRQ，GPIO0、GPIO1 引脚输出时钟
函数定义	void um2010_set_gpio_brclk_sel(em_gpio_t gpio,em_brclk_t brclk)
输入参数	em_gpio_t gpio：GPIO 引脚 ➢ NIRQ ➢ GPIO0 ➢ GPIO1 em_brclk_t brclk:时钟信号选择 ➢ BRCLK_RX_CLK：rx_clk 接收同步时钟 ➢ BRCLK_XTAL_CLK：xtal_clk 晶振时钟 ➢ BRCLK_XTAL_CLK_DIV8：xtal_clk/8 ➢ BRCLK_XTAL_CLK_DIV16：xtal_clk/16 ➢ BRCLK_FS_CLK：频综时钟 ➢ BRCLK_TX_CLK：tx_clk 发射同步时钟 ➢ BRCLK_ADC_CLK：ADC_clk

输出参数	None
函数返回	None

3.23 um2010_set_packet_mode

函数功能	um2010_set_packet_mode
函数描述	设置包模式
函数定义	void um2010_set_packet_mode(em_packet_t mode)
输入参数	em_packet_mode_t mode：数据包模式 ➢ PACKET_MODE_0：模式 0 ➢ PACKET_MODE_1：模式 1 ➢ PACKET_MODE_2：模式 2 ➢ PACKET_MODE_3：模式 3
输出参数	None
函数返回	None

3.24 um2010_set_preamble

函数功能	um2010_set_preamble
函数描述	设置 preamble 使能及字节长度
函数定义	void um2010_set_preamble(BOOL enable,u8_t value,u8_t len)
输入参数	BOOL enable：使能选择 ➢ TRUE：使能 ➢ FALSE：不使能 u8_t value：Preamble 值 u16_t len：Preamble 长度
输出参数	None
函数返回	None

3.25 um2010_set_preamble_match_len

函数功能	um2010_set_preamble_match_len
函数描述	设置 RX 时，preamble match 的字节长度
函数定义	void um2010_set_preamble_match_len(u8_t len)
输入参数	u8_t len：匹配长度
输出参数	None
函数返回	None

3.26 um2010_set_preamble_irq

函数功能	um2010_set_preamble_irq
函数描述	设置 Preamble 接收匹配中断使能
函数定义	void um2010_set_preamble_irq(BOOL enable)
输入参数	BOOL enable ➢ TRUE：使能 ➢ FALSE：不使能
输出参数	None
函数返回	None

3.27 um2010_set_syncword_enable

函数功能	um2010_set_syncword_enable
函数描述	设置同步字使能
函数定义	void um2010_set_syncword_enable(BOOL enable)
输入参数	BOOL enable：使能选择 ➢ TRUE：使能 ➢ FALSE：不使能
输出参数	None
函数返回	None

3.28 um2010_set_syncword_manchester_enable

函数功能	um2010_set_syncword_manchester_enable
函数描述	设置同步字 bit 顺序
函数定义	void um2010_set_syncword_manchester_enable(BOOL enable)
输入参数	BOOL enable：使能选择 ➢ TRUE：使能 ➢ FALSE：不使能
输出参数	None
函数返回	None

3.29 um2010_set_syncword_bit_order

函数功能	um2010_set_syncword_bit_order
函数描述	设置同步字的 bit 顺序
函数定义	void um2010_set_syncword_bit_order(em_order_t order)
输入参数	em_order_t order ➢ LSB ➢ MSB
输出参数	None
函数返回	None

3.30 um2010_set_syncword_value

函数功能	um2010_set_syncword_value
函数描述	设置同步字及长度
函数定义	void um2010_set_syncword_value(u8_t *sync,u8_t len)
输入参数	u8_t *sync：同步字值 u8_t len：同步字长度
输出参数	None
函数返回	None

3.31 um2010_set_syncword_irq

函数功能	um2010_set_syncword_irq
函数描述	设置接收同步字中断使能
函数定义	void um2010_set_syncword_irq(BOOL enable)
输入参数	BOOL enable：使能选择 ➢ TRUE：使能 ➢ FALSE：不使能
输出参数	None
函数返回	None

3.32 um2010_set_payload_length

函数功能	um2010_set_payload_length
函数描述	设置模式 1 和模式 3 的 Payload 长度
函数定义	void um2010_set_payload_length(u16_t len)
输入参数	u16_t len：Payload 长度
输出参数	None
函数返回	None

3.33 um2010_set_payload_order

函数功能	um2010_set_payload_order
函数描述	设置 Payload 的 bit 顺序
函数定义	void um2010_set_payload_order(em_order_t order)
输入参数	em_order_t order： ➢ LSB ➢ MSB
输出参数	None
函数返回	None

3.34 um2010_set_crc_enable

函数功能	um2010_set_crc_enable
函数描述	设置 CRC 使能
函数定义	void um2010_set_crc_enable(BOOL enable)
输入参数	BOOL enable：使能选择 ➢ TRUE：使能 ➢ FALSE：不使能
输出参数	None
函数返回	None

3.35 um2010_set_crc_bit_order

函数功能	um2010_set_crc_bit_order
函数描述	设置 CRC Bit 顺序
函数定义	void um2010_set_crc_bit_order(em_order_t order)
输入参数	em_order_t order： ➢ LSB ➢ MSB
输出参数	None
函数返回	None

3.36 um2010_set_crc_invert

函数功能	um2010_set_crc_invert
函数描述	设置 CRC Bit 取反使能
函数定义	void um2010_set_crc_invert(BOOL enable)
输入参数	BOOL enable：使能选择 ➢ TRUE：使能 ➢ FALSE：不使能
输出参数	None
函数返回	None

3.37 um2010_set_crc_manchester_enable

函数功能	um2010_set_crc_manchester_enable
函数描述	设置 CRC 曼彻斯特使能
函数定义	void um2010_set_crc_manchester_enable(BOOL enable)
输入参数	BOOL enable：使能选择 ➢ TRUE：使能 ➢ FALSE：不使能
输出参数	None
函数返回	None

3.38 um2010_set_crc_len

函数功能	um2010_set_crc_len
函数描述	设置 CRC 长度
函数定义	void um2010_set_crc_len(em_crc_len_t crc_len)
输入参数	em_crc_len_t crc_len：CRC 长度 ➢ CRC_LEN_8 ➢ CRC_LEN_16 ➢ CRC_LEN_24 ➢ CRC_LEN_32
输出参数	None
函数返回	None

3.39 um2010_set_crc_poly

函数功能	um2010_set_crc_poly
函数描述	设置 CRC 多项式
函数定义	void um2010_set_crc_poly(u32_t poly)
输入参数	u32_t poly：CRC 多项式
输出参数	None
函数返回	None

3.40 um2010_set_crc_init_value

函数功能	um2010_set_crc_init_value
函数描述	设置 CRC 初始值
函数定义	void um2010_set_crc_init_value(u32_t crc_init)
输入参数	u32_t crc_init：CRC 初始值
输出参数	None
函数返回	None

3.41 um2010_set_freq

函数功能	um2010_set_freq
函数描述	设置频点、通道、步进
函数定义	void um2010_set_freq(float f0,u8_t chn,float step)
输入参数	float f0：频点，单位 MHz u8_t chn：通道 float step：步进，单位 MHz
输出参数	None
函数返回	None

3.42 um2010_set_f0

函数功能	um2010_set_f0
函数描述	设置频点
函数定义	void um2010_set_f0(float f0)
输入参数	float f0：设置频点，单位 MHz
输出参数	None
函数返回	None

3.43 um2010_set_chn

函数功能	um2010_set_chn
函数描述	设置通道
函数定义	void um2010_set_chn(u8_t chn)
输入参数	u8_t chn：通道
输出参数	None
函数返回	None

3.44 um2010_set_step

函数功能	um2010_set_step
函数描述	设置步进
函数定义	void um2010_set_step(float step)
输入参数	float step：步进，单位 MHz
输出参数	None
函数返回	None

3.45 um2010_set_ref_freq

函数功能	um2010_set_ref_freq
函数描述	设置参考晶振频率
函数定义	void um2010_set_ref_freq(float ref_freq)
输入参数	float ref_freq：晶振频率，单位 MHz
输出参数	None
函数返回	None

3.46 um2010_get_rssi

函数功能	um2010_get_rssi
函数描述	获取 RSSI 值
函数定义	float um2010_get_rssi(void)
输入参数	None
输出参数	None
函数返回	返回 RSSI 值

3.47 um2010_get_status

函数功能	um2010_get_status
函数描述	获取芯片的状态，REG61
函数定义	u8_t um2010_get_status(void)
输入参数	None
输出参数	None
函数返回	返回 Reg61 寄存器的状态

4 UM2010 硬件接口

#include “um2010_hal.h”

4.1 um2010_hal_get_irq_flag

函数功能	um2010_hal_get_irq_flag
函数描述	获取中断标志
函数定义	u8_t um2010_hal_get_irq_flag(void)
输入参数	None
输出参数	None
函数返回	u8_t 返回中断标志 ➢ 0：无中断 ➢ 1：有中断产生

4.2 um2010_hal_clear_irq_flag

函数功能	um2010_hal_clear_irq_flag
函数描述	清除中断标志
函数定义	void um2010_hal_clear_irq_flag(void)
输入参数	None
输出参数	None
函数返回	None

4.3 um2010_hal_delay_us

函数功能	um2010_hal_delay_us
函数描述	延迟函数，单位 us
函数定义	void um2010_hal_delay_us(u16_t us)
输入参数	u16_t us：延迟 us
输出参数	None
函数返回	None

4.4 um2010_hal_set_sdn

函数功能	um2010_hal_set_sdn
函数描述	UM2010 SDN 控制电平，拉高关断芯片
函数定义	void um2010_hal_set_sdn(em_level_t level)
输入参数	em_level_t level，SDN 输出电平： ➢ LEVEL_LOW：低电平 ➢ LEVEL_HIGH：高电平
输出参数	None
函数返回	None

4.5 um2010_hal_set_rst

函数功能	um2010_hal_set_rst
函数描述	UM2010 RST 控制电平，拉低使芯片处于复位状态，拉高使芯片处于工作状态
函数定义	void um2010_hal_set_rst(em_level_t level)
输入参数	em_level_t level，RST 输出电平： ➢ LEVEL_LOW：低电平 ➢ LEVEL_HIGH：高电平
输出参数	None
函数返回	None

4.6 um2010_hal_init

函数功能	um2010_hal_init
函数描述	UM2010 硬件初始化
函数定义	void um2010_hal_init(void)
输入参数	None
输出参数	None
函数返回	None

4.7 spi_init

函数功能	spi_init
函数描述	SPI 初始化
函数定义	void spi_init(void)
输入参数	None
输出参数	None
函数返回	None

4.8 spi_cs_enable

函数功能	spi_cs_enable
函数描述	SPI CS 使能
函数定义	void spi_cs_enable(void)
输入参数	None
输出参数	None
函数返回	None

4.9 spi_cs_disable

函数功能	spi_cs_disable
函数描述	SPI CS 不使能
函数定义	void spi_cs_disable(void)
输入参数	None
输出参数	None
函数返回	None

4.10 spi_write_byte

函数功能	spi_write_byte
函数描述	SPI 写一个字节数据
函数定义	void spi_write_byte(u8_t byte)
输入参数	u8_t byte：写入的数据
输出参数	None
函数返回	None

4.11 spi_read_byte

函数功能	spi_read_byte
函数描述	SPI 读一个字节数据
函数定义	u8_t spi_read_byte(void)
输入参数	None
输出参数	None
函数返回	SPI 读回的数据

5 版本修订

版本	日期	描述
V1.0	2025.12.15	初始版