



Panchip Microelectronics Co., Ltd.

PAN3029 MAPM 应用参考文档

当前版本: 1.2

发布日期: 2023.12

上海磐启微电子有限公司

地址: 上海张江高科技园区盛夏路 666 号 D 栋 302 室

联系电话: 021-50802371

网址: <http://www.panchip.com>

文档说明

由于版本升级或存在其他原因，本文档内容会不定期进行更新。除非另有约定，本文档内容仅作为使用指导，本文档中的所有陈述、信息和建议不构成任何明示或暗示的担保。

商标

磐启是磐启微电子有限公司的商标。本文档中提及的其他名称是其各自所有者的商标/注册商标。

免责声明

本文档中描述的全部或部分产品、服务或特性可能不在您的购买或使用范围之内。除非合同另有约定，磐启微电子有限公司对本文档内容不做任何明示或暗示的声明或保证。

修订历史

版本	修订时间	描述
V1.0	2023.06	初始版本
V1.1	2023.09	更新 SDK 例程和部分接口函数
V1.2	2023.12	更新 SF,BW 变化，休眠时间自适应功能

目录

1 功能介绍	1
2 MAPM 帧结构	1
3 SDK 接口函数	3
3.1 rf_set_mapm_on	3
3.2 rf_set_mapm_off	3
3.3 void rf_set_mapm_para	4
3.4 PAN3029_calculate_mapm_preambletime	5
3.5 PAN3029_set_mapm_addr	5
4 软件设计验证	6
4.1 验证内容:	6
4.2 软件设计流程	6
4.3 验证步骤	7
5 系统演示板	18
5.1 模组 RX,TX 设置	18
5.2 按键功能	18
6 注意事项	19
6.1 地址配置	19
6.2 Field 中最后一个 Group 其 ADDR 位置功能选择	19
6.3 同步字配置	19
6.4 Preamble 数量配置	20
6.5 休眠时间	20

1 功能介绍

在 ChirpIOT 星状组网使用场景中，为了保证节点端设备能收到网关发送的数据，网关端会使用超长 Preamble 进行发送，如 Preamble 占用时间 1s。在这种情况下，所有节点端设备在收到 Preamble 后都会一直接收直到 Payload 阶段显示节点网络地址，此时，节点端设备的软件可以比较收到的网络地址和节点设备自己地址，如不同，则停止接收，降低功耗，如满足地址接收条件则继续接收。这种方案存在几个问题：

- 1) 不满足地址接收条件的节点端设备，直到 payload 阶段才能停止接收，浪费大量功耗
- 2) 满足地址接收条件的节点端设备也需要收满超长的 preamble 才能进入 payload 接收阶段，也较浪费功耗

为解决以上两个功耗问题，在芯片 PAN3029 中增加新的帧结构模式 Multiple Address Preamble Mode (MAPM)。该新帧结构模组的思路是在 Preamble 阶段插入类似后面同步字 (Sync Word) 的地址，这样节点端在 Preamble 阶段就可以根据地址来提前判断是否满足地址接收条件，已解决 1) 问题。另外，为了解决第 2) 问题，在地址中增加可变计数器，表明后续还有多少 Preamble 长度，满足地址接收条件的节点端设备可提前进入低功耗休眠状态，后根据该计数器判断需要提前多久在进入接收状态，保证数据正常接收，以便进一步节省满足地址接收条件的节点端设备功耗。

2 MAPM 帧结构

原始的帧结构大致如下图 1 所示。



图 1 原始帧结构

如下图 2 所示。MAPM 方案将 preamble 分为 F_n 个 Field， F_n 为 Field 的数量，每一个 Field 的结构相同，内容可不相同，同时为了保证 payload 数据能正确接收，在 sync word 前至少保证 P_n 个 preamble， P_n 需要大于等于 8。

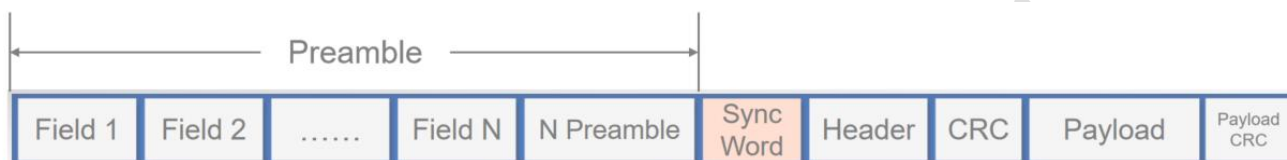


图 2 MAPM 帧结构

如下图 3 所示。在一个 Field 内部，分为若干个 Group。Group 的 G_n 个数最少 1 个，最多 4 个， G_n 可以由寄存器配置。注意：TX 和 RX 的 G_n 配置相同且事先确定。

一个 Group 分为 P_g 个未经调制的 chirp 加两个带调制的 ADDR，第一个 Group 中， P_g 的值最小为 8，其他 Group 中 P_g 的最小值位 0，也就是让所有的 ADDR 都连在一起，也可在其中插入 preamble，其最大数量为 255 个(8bits)，ADDR 为 1 个字节的数据，可以由寄存器自由配置。

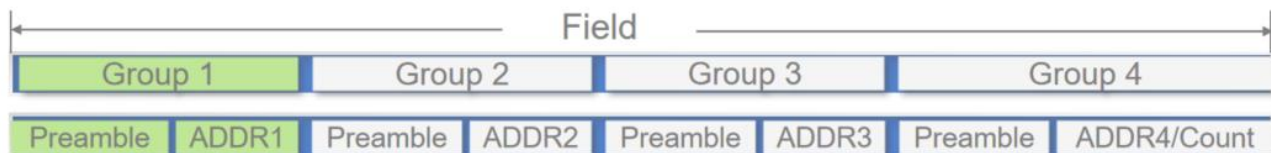


图 3 Field 帧结构

Group1 的 ADDR 固定地作为标识符，其中 ADDR1 用于 MAPM 硬件同步，ADDR2~ADDR4 用于软件应用。注意：TX 和 RX 的地址配置需事先约定好，其中 ADDR1 必须相同 MAPM 功能才能生效。

Group 的数量 G_n ，

为 0 时只有 1 个 Group；

为 1 时有 2 个 Group；

为 2 时有 3 个 Group；

为 3 时有 4 个 Group;

不管有几个 Group, 最后一个 Group 的 ADDR, 存在两种模式:

- 1) 地址模式(ADDR), 可用 1 字节寄存器配置
- 2) 计数器模式(COUNT), 根据当前 Field 数量倒计时, 由 RTL 自动来填入, 该值标识剩下的 Field 数量。

3 SDK接口函数

3.1 rf_set_mapm_on

句法

```
void rf_set_mapm_on(void)
{
    PAN3029_mapm_en();
    PAN3029_set_mapm_mask(MAPM_ON);
}
```

目的

使能 mapm 模式, 开 mapm 中断允许

3.2 rf_set_mapm_off

句法

```
void rf_set_mapm_off(void)
{
    PAN3029_mapm_dis();
    PAN3029_set_mapm_mask(MAPM_OFF);
}
```

目的

关 mapm 模式，关 mapm 中断允许

3.3 void rf_set_mapm_para

句法

```
void rf_set_mapm_para(stc_mapm_cfg_t *p_mapm_cfg)
{
    PAN3029_set_mapm_field_num(p_mapm_cfg->fn);
    PAN3029_set_mapm_field_num_mux(p_mapm_cfg->fnm);
    PAN3029_set_mapm_group_fun_sel(p_mapm_cfg->gfs);
    PAN3029_set_mapm_group_num(p_mapm_cfg->gn);
    PAN3029_set_mapm_firgroup_preamble_num(p_mapm_cfg->pgl);
    PAN3029_set_mapm_group_preamble_num(p_mapm_cfg->pgn);
    PAN3029_set_mapm_neces_preamble_num(p_mapm_cfg->pn);
}
```

目的

设置 mapm 模式相关参数

参数

fn {0x01~0xe0}: mapm preamble 中 Field 数量{field1, field2,.....fieldfn}

Fnm{0x00~0x03}: mapm preamble 中每个 field 重复次数

00 --> 重复 1 次{field1, field2,.....fieldfn}

01 --> 重复 2 次{field1, field1, field2, field2.....fieldfn, fieldfn}

10 --> 重复 4 次

11 --> 重复 8 次

gfs {0x00~0x01}: Field 中最后一个 Group，其 ADDR 位置功能选择

0: 普通地址{addr1, addr2, addr3, addr4}

1: Field 个数计数器{addr1, addr2, addr3, fn}

gn {0x00~0x03}: 一个 Field 中 Group 个数

00--> 1 个 group

01--> 2 个 group

02--> 3 个 group

03--> 4 个 group

pgl {0x08~0xff}: 每个 Field 中第一个 Group 内 Preamble 数量默认 0x08

Pgn{0x00~0xff}: 每个 Field 中其他 Group 内 Preamble 数量,默认 0x08

Pn{0x00~0xffff}: 所有的 Field 都发送完毕后 syncword 前 chirp 的数量,默认 0x08

3.4 PAN3029_calculate_mapm_preambletime

句法

```
uint32_t PAN3029_calculate_mapm_preambletime(void)
```

目的

计算当前收到的 preamble 剩余时长(ms)

参数

无

返回值

当前收到的 preamble 剩余时长

3.5 PAN3029_set_mapm_addr

句法

```
uint32_t PAN3029_set_mapm_addr(uint8_t addr_no, uint8_t addr)
```

目的

配置 mapm Group 中地址

参数

addr_no 想要配置的地址序号

0: ADDR1

1: ADDR2

2: ADDR3

3: ADDR4

addr ADDR 想配置的值

返回值

FAIL 操作失败

OK 操作成功

注：地址配置范围见 6.1 地址配置

4 软件设计验证

注：

1. 只有 TX,RX mapm 地址参数 ADDR1 相同时，RX 才能进入 mapm 中断，所以 TX,RX 代码要配置一样的 ADDR1，mapm 功能才生效。
2. 所有验证基于 HC32F460EVB 底板，详细操作方式见第 5 章节系统演示板。

4.1 验证内容：

本次验证使用 3 个模组进行，一个发送 TX,两个接收 RX。验证代码共用同一工程，通过底板拨码开关选择 TX/RX 功能；通过代码中宏 ADDR_MATCH 来选择 RX 与 TX ADDR2 是否相同，从而判断是否接收 TX 发送数据。两个 RX，一个开启宏，一个关闭宏进行对比。

4.2 软件设计流程

- 1、芯片初始化，上电默认 SF7,BW500K；
- 2、配置 mapm 初始化；
- 3、TX 进入发送模式，RX1,RX2 3029 进入低功耗模式,其中 RX1 ADDR2 与 TX 相同，RX2 ADDR2 与 TX 不同；
- 4、RX 3029 每隔当前前参数下 preamble 时长一半唤醒检测接收数据。

4.3 验证步骤

- 1、TX 按键周期性发送数据；
- 2、RX1, RX2 每隔当前参数下 preamble 时长一半周期性唤醒检测接收数据；
- 3、使用电流表统计两个 RX 功耗情况。
- 4、串口调试工具监视 RX 接收数据

4.3.1.1 SDK 示例

参考代码：

```
#define ADDR_MATCH 1 //地址匹配
/*初始化不同 SF mapm 参数*/
const stc_mapm_cfg_t mapm_cfg[] =
{
    {
        .mapm_addr= {0x31, 0x32, 0x23, 0x22},    //SF5 BW_500K
        .fn = 32,
        .fnm = 0,
        .gfs = FIELD_COUNTER,
        .gn = 2,
        .pg1 = 10,
        .pgn = 120,
        .pn = 8
    },
    {
        .mapm_addr= {0x73, 0x62, 0x51, 0x41},    //SF6 BW_500K
        .fn = 64,
        .fnm = 0,
        .gfs = FIELD_COUNTER,
```

```
.gn = 2,
.pg1 = 10,
.pgn = 120,
.pn = 8
},
{
.mapm_addr= {0x73, 0x62, 0x54, 0x81},    //SF7 BW_500K
.fn = 64,
.fnm = 0,
.gfs = FIELD_COUNTER,
.gn = 2,
.pg1 = 10,
.pgn = 120,
.pn = 8
},
{
.mapm_addr= {0x73, 0x62, 0x54, 0x81},    //SF8 BW_500K
.fn = 32,
.fnm = 0,
.gfs = FIELD_COUNTER,
.gn = 2,
.pg1 = 10,
.pgn = 120,
.pn = 8
},
{
.mapm_addr= {0x73, 0x62, 0x54, 0x81},    //SF9 BW_500K
.fn = 16,
.fnm = 0,
```

```
.gfs = FIELD_COUNTER,
.gn = 2,
.pg1 = 10,
.pgn = 120,
.pn = 8
},
{
.mapm_addr= {0x73, 0x62, 0x54, 0x81},    //SF10 BW_500K
.fn = 8,
.fnm = 0,
.gfs = FIELD_COUNTER,
.gn = 2,
.pg1 = 10,
.pgn = 120,
.pn = 8
},
{
.mapm_addr= {0x73, 0x62, 0x54, 0x81},    //SF11 BW_500K
.fn = 6,
.fnm = 0,
.gfs = FIELD_COUNTER,
.gn = 2,
.pg1 = 10,
.pgn = 120,
.pn = 8
},
{
.mapm_addr= {0x73, 0x62, 0x54, 0x81},    //SF12 BW_500K
.fn = 6,
```

```
.fnm = 0,
.gfs = FIELD_COUNTER,
.gn = 2,
.pg1 = 10,
.pgn = 120,
.pn = 8
},
};/*3029 初始化配置*/
ret = rf_init(); //3029 初始化
if(ret != OK)
{
    printf(" RF Init Fail");
    while(1);
}
rf_set_default_para(); //配置默认射频参数
//使用当前 demo 初始化的射频参数
now_bw = TEST_BW;
now_sf = TEST_SF;
PAN3029_set_bw(now_bw);
PAN3029_set_sf(now_sf);
PAN3029_set_code_rate(TEST_CR);
PAN3029_set_freq(TEST_FREQ);
memcpy((uint8_t*)&cur_mapm_cfg,(uint8_t*)&mapm_cfg[now_sf-SF_5],
sizeof(stc_mapm_cfg_t));//根据 SF 获取当前 mapm 参数，第一个 mapm 参数是 SF5 的
rf_set_mapm_on(); //使能 mapm 模式
rf_set_mapm_para(&cur_mapm_cfg); //配置 mapm 参数
for(uint8_t i=0; i<cur_mapm_cfg.gn+1; i++)
{
    PAN3029_set_mapm_addr(i, cur_mapm_cfg.mapm_addr[i]);
}
```

```
    }

    one_chirp_time = get_chirp_time(now_bw,now_sf);//计算一个包长时间 us
    preamble_time=PAN3029_calculate_mapm_preambletime(&cur_mapm_cfg,
one_chirp_time);//计算 preamble 时长

    rx_timeout_val = preamble_time + calculate_payload_time(tx_len) + 200;//计算完整包
长时间，在此基础上加 200ms 作为接收超时判断时间

    sleep_timer_val = preamble_time/2;

    if(app_tx_mode)
    {
        DDL_Printf("tx\r\n");
    }
    else
    {
        DDL_Printf("rx\r\n");
#endif ADDR_MATCH
//修改 ADDR1,软件用来区分是否接收 TX 发送数据

    if(now_sf == SF_5)
    {
        cur_mapm_cfg.mapm_addr[1] = 0x21;
        PAN3029_set_mapm_addr(1, cur_mapm_cfg.mapm_addr[1]);
    }
    else
    {
        cur_mapm_cfg.mapm_addr[1] = 0x61;
        PAN3029_set_mapm_addr(1, cur_mapm_cfg.mapm_addr[1]);
    }
#endif

    rf_set_cad(CAD_DETECT_THRESHOLD_10, CAD_DETECT_NUMBER_3);
    rf_set_dcde_mode(DCDC_ON);
```

```
        rf_enter_single_timeout_rx(rx_timeout_val);
    }
    while(1){
        rf_irq_process();
        key_scan();
        key_event_process();
        process_rf_events();
    }

    /*MAPM 接收数据后休眠处理*/
    flag = rf_get_rcv_flag();
    if(flag == RADIO_FLAG_MAPM)
    {
        rf_set_rcv_flag(RADIO_FLAG_IDLE);
        if(RxDoneParams.mpam_rcv_index == cur_mapm_cfg.gn + 1)//接收到一个完整
Field
        {
            uint8_t i;
            if(cur_mapm_cfg.gfs == FIELD_COUNTER)
            {
                addr_num = cur_mapm_cfg.gn;
            }
            else
            {
                addr_num = cur_mapm_cfg.gn + 1;
            }
            for(i=0; i<addr_num; i++)
            {
                if(cur_mapm_cfg.mapm_addr[i] != RxDoneParams.mpam_rcv_buf[i])
```

//addr 不匹配，不需要接收

```

        {
            mapm_target = 0;
            break;
        }
    }
    if(i == addr_num)
    {
        mapm_target = 1; //接收地址匹配，需要接收数据
    }
    if(cur_mapm_cfg.gfs == FIELD_COUNTER)//计数器模式
    {
        cur_mapm_cfg.fn = RxDoneParams.mpam_recv_buf[cur_mapm_cfg.gn];
        mapm_preamgle_time=
        PAN3029_calculate_mapm_preambletime(&cur_mapm_cfg, one_chirp_time);
        one_field_chirp = cur_mapm_cfg.pg1+2 + (cur_mapm_cfg.pgn+2) *
        cur_mapm_cfg.gn;
        if(mapm_target)
        {
            mapm sleeptime = mapm_preamgle_time
one_field_chirp*one_chirp_time/1000; //计算可以休眠时间
            if(mapm sleeptime > CODE_RUNTIME)
            {
                rf_sleep();
                mapm_sleep_flag = 1;
                mapm_sleep_begtime = SysTick_GetTick();
            }
            else //休眠时间太短不休眠
            {
                DDL_Printf("sleep time too short\r\n");
            }
        }
    }

```



```

    }

    }

    else

    {

        memset(RxDoneParams.mpam_recv_buf, 0,
RxDoneParams.mpam_recv_index);

        RxDoneParams.mpam_recv_index = 0;

        mapm_sleeptime = mapm_preamble_time +
PAYLOAD_LEN*one_chirp_time/1000;

        rf_sleep();

        mapm_sleep_flag = 1;

        mapm_sleep_begtime = SysTick_GetTick();

    }

    }

    }

    }

    /*mapm 休眠唤醒处理*/

    if(mapm_sleep_flag == 1)

    {

        if((SysTick_GetTick()-mapm_sleep_begtime) >
(mapm_sleeptime-CODE_RUNTIME)) //达到休眠时间后唤醒

        {

            rf_sleep_wakeup();

            mapm_sleep_flag = 0;

            if(mapm_target)

            {

                mapm_target = 0;

            }

            else

            {

```

```

recv_flag = 0;

}

if(rf_enter_continuous_rx() != OK)

{

    DDL_Printf("set rx FAIL\r\n");

}

}

}

```

示例代码配置了 mapm 初始化，TX 发送数据后，RX 接收后判断是否需要接收数据，然后进行休眠唤醒处理。

发送模组发送数据包(默认参数 SF7 BW500K 数据包 preamble 的持续时间约 4200ms,payload 的持续时间约 40ms)，用电流表抓取两个 RX 接收模组电流波形，观察结果。

4.3.1.2 验证结果

电流表抓取 RX1 结果如图 2-1 所示：

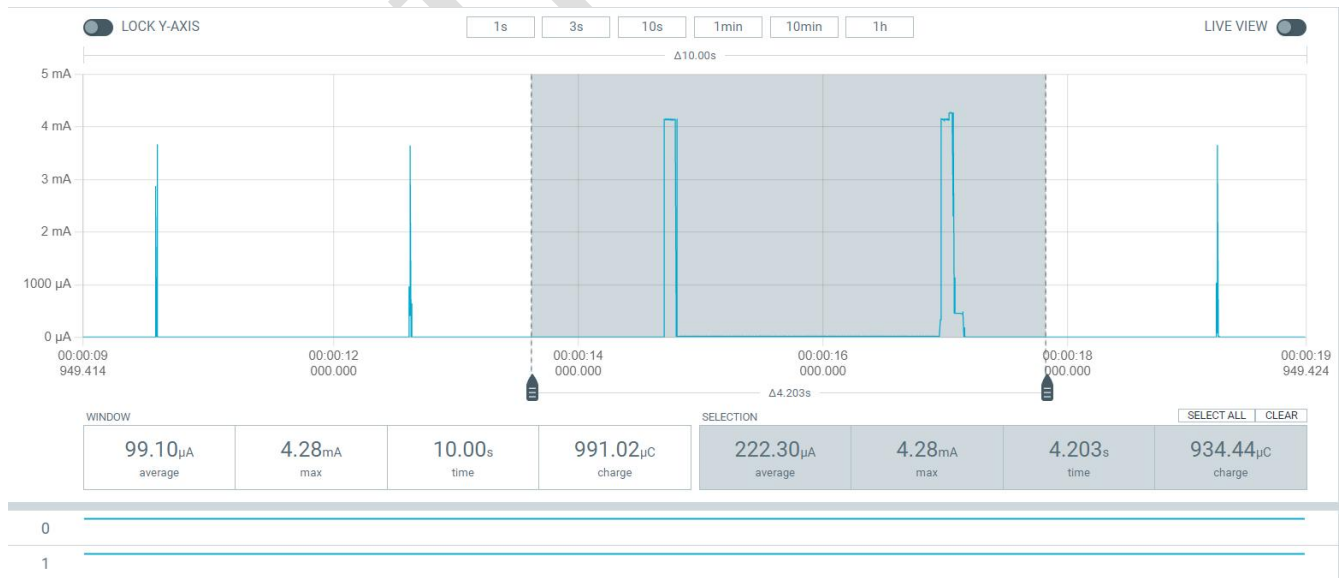


图 2-1 RX1 电流表功耗抓取结果

根据结果显示，当发送模组发送数据包时，RX1 在唤醒后检测到 TX 数据，判断地址匹配后，计算出剩余 preamble 时长，休眠该时间后，唤醒进行 payload 数据接收。

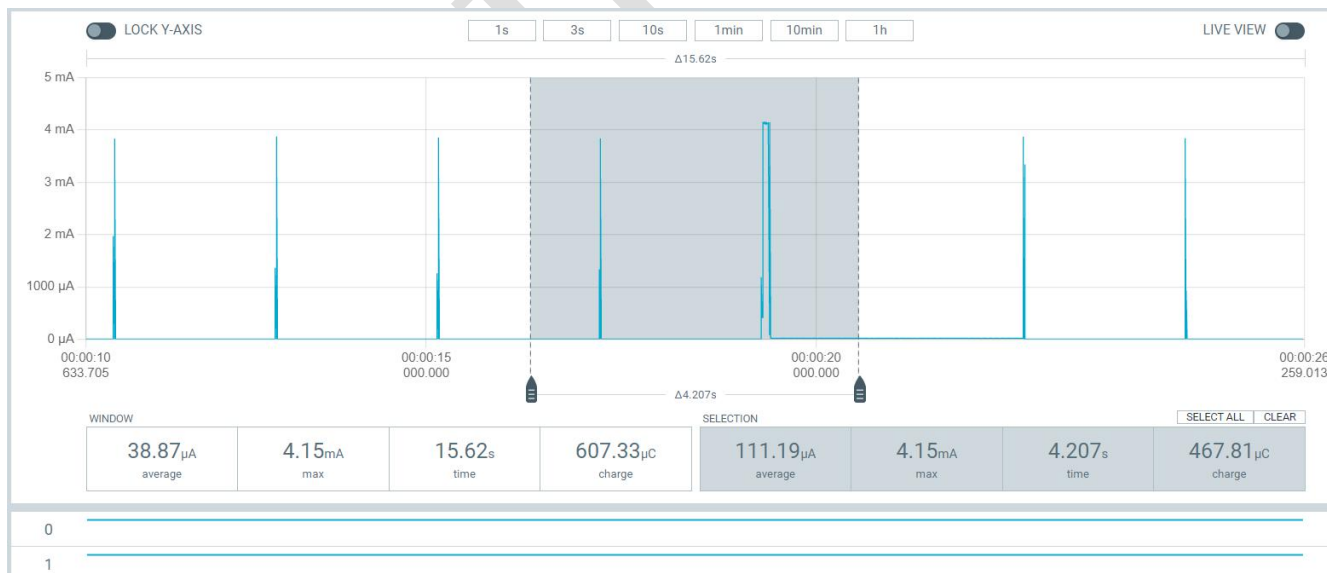
串口打印 RX1 结果如图 2-2 所示：

```
[10:24:39.014]收←◆rx
[10:27:23.067]收←◆RF RX:1
[10:27:23.085]收←◆mapm addr[0] =73
mapm addr[1] =62
mapm addr[2] =3e
mapm addr[
[10:27:23.090]收←◆3] =73
mapm addr[4] =62
mapm addr[5] =01
mapm addr num =6
Rx : SNR: 9.369039 ,RSSI: -11
0x00 0x01 0x02 0x03 0x04 0x05 0x06 0x07 0x08 0x09 0x0a 0x0b 0x0c 0x0d 0x0e 0x0f 0x10 0x11 0x12 0x1
[10:27:23.107]收←◆3 0x14 0x15 0x16 0x17 0x18 0x19 0x1a 0x1b 0x1c 0x1d 0x1e 0x1f 0x20 0x21 0x22 0x23 0x24 0x25 0x26 0x27 0x28 0x29 0x2a 0x2b 0x2c
0x2d 0x2e 0x2f 0x30 0x31 0x32 0x33 0x34 0x35 0x36 0x37 0x38 0x39
[10:27:23.124]收←◆0x3a 0x3b 0x3c 0x3d 0x3e 0x3f 0x40 0x41 0x42 0x43 0x44 0x45 0x46 0x47 0x48 0x49 0x4a 0x4b 0x4c 0x4d 0x4e 0x4f 0x50 0x51 0x52
0x53 0x54 0x55 0x56 0x57 0x58 0x59
[10:27:23.138]收←◆0x5a 0x5b 0x5c 0x5d 0x5e 0x5f 0x60 0x61 0x62 0x63
```

图 2-2 RX1 串口抓取结果

根据结果显示，只收到了 preamble 中 6 个 mapm addr,而 TX 实际发送了 96 个 mapm addr，其余 90 个 mapm addr 时间 RX1 进入了低功耗休眠，然后在 payload 前唤醒接收了 100 字节长度 payload。

电流表抓取 RX2 结果如图 2-3 所示：



根据结果显示，当发送模组发送数据包时，RX2 在唤醒后检测到 TX 数据，判断地址不匹配后，计算出剩余 preamble 和 payload 时长，休眠该时间后，唤醒重新进行接收检测。

串口打印 RX2 结果如图 2-4 所示：



图 2-4 RX2 串口抓取结果

根据结果显示，RX2 判断地址不匹配后直接休眠不再进行数据接收，所以未打印任何接收数信息。

5 系统演示板

演示系统板基于 HC32F460EVB 的底板功能描述如图 5-1 所示

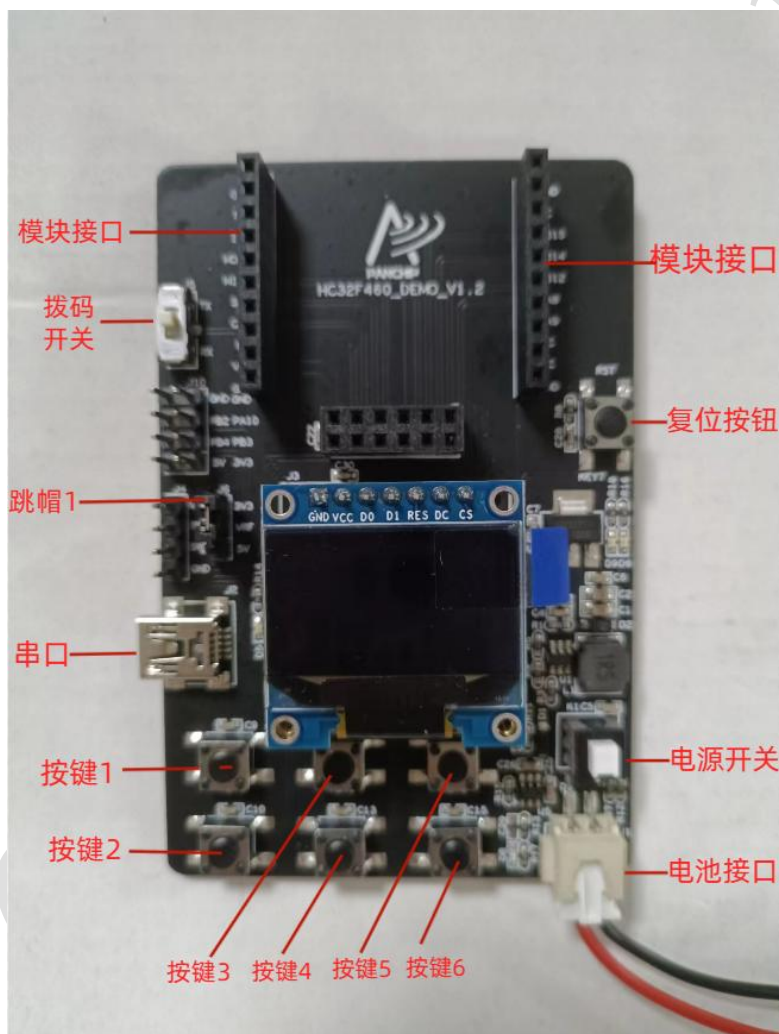


图 5-1 HC32 底板

5.1 模组 RX,TX 设置

拨码开关：收发模式切换。往上拨到 TX 为 TX 模式，往下拨到 RX 为 RX 模式。

5.2 按键功能

按键 3：TX 按下发送数据

按键 5: 切换 SF, 范围 5~12, 步径 1

按键 6: 切换 BW, 62.5K/125K/250K/500K

6 注意事项

6.1 地址配置

1. 由于调制 Chirp 的特点, 所有 ADDR 配置范围 0x11~0xFF, 不支持小于 0x11 和不支持 0xX0 值, 如果软件收到这些值时, 软件操作芯片复位。同时需要注意 Group1 中的 ADDR1、Group 中的 ADDR2~4 以及正常帧结构中的 SyncWord 这三者都不相同。

2. SF7~SF12, ADDR 支持 8bit 配置, SF5 仅支持 0b00xx00xx (0x11, 0x12, 0x13, 0x21, 0x22, 0x23, 0x31, 0x32, 0x33), SF6 支持 0b0xxx0xxx, 这里仍然要求满足不支持小于 0x11 和不支持 0xX0 值。

3. 地址配置配合参数 gn 使用, 根据 gn 数量从 addr1 开始连续配置相应数量的 addr。

4. 每个 ADDR 为无符号 8 位数据, 配置时高 4 位和低 4 位尽量不要一样。

6.2 Field 中最后一个 Group 其 ADDR 位置功能选择

(1) 参数 gfs = 0x00, field 中最后一个 group 配置为普通 Addr 功能

(2) 参数 gfs = 0x01, field 中最后一个 group 配置为计数器, 注意 fn 和 addr 的配置, 避免出现 fn 不断自减出现计数器与 ADDR1 或同步字 syncword 相等的情况误触发 mapm 中断或错误同步的情况。

6.3 同步字配置

SF7~SF12, 同步字 syncword 支持 8bit 配置, SF5 仅支持 0b00xx00xx (0x11, 0x12, 0x13, 0x21, 0x22, 0x23, 0x31, 0x32, 0x33), SF6 支持 0b0xxx0xxx, 这里仍然要求满足不支

持小于 0x11 和不支持 0xX0 值。

6.4 Preamble 数量配置

- (1) .每个 Field 中第一个 Group 内 Preamble (pg1) 的数量最小为 8。
- (2) .所有的 Field 都发送完毕后 syncword 前 Preamble(pn)的数量最小为 8。

6.5 休眠时间

根据参数算出后续 preamble 接收时间，进入休眠时休眠时间需注意减去休眠，休眠唤醒及其他代码运行时间，尽量稍微提前唤醒，避免唤醒错过 syncword，无法接收 payload 数据。