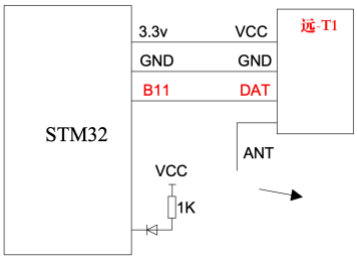
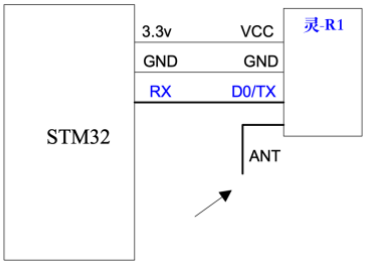
蜂鸟远-T1 和灵-R1 无线模块调试后记

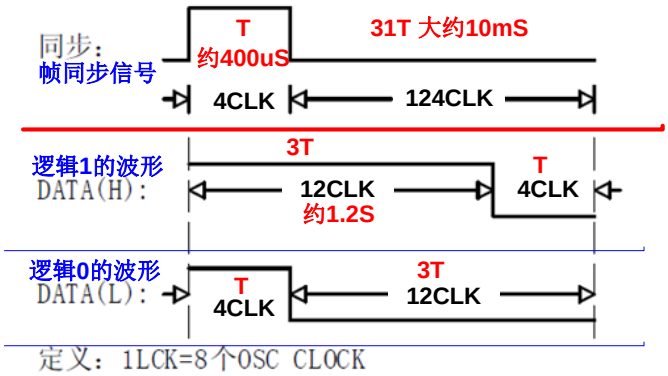
测调	西安.老雷子	2020 年 6 月 1 日
软件平台	WINDOWS	Keil uVision,STM32 ST-LINK
硬件平台	STM32S108C8B6	通用 32 开发板调试
发射端：	蜂鸟远-T1	输入需要用 MCU 进行编码，利于指定编码
接收端：	蜂鸟灵-R1	输出有五种模式，其中模式 5 就是串口直接输出编码的
编码模式	EV1527	软件模拟

一、 系统调试硬件结构说明

发射端硬件示意图	接收端硬件示意图
	

二、 如何对位进行编码

先简单了解一下编码中如何把二进制体现出来，如下图，模块是采用调幅模式，用发射和不发射的时长和间隔来体现是逻辑 1 还是逻辑 0。如下，看懂这个逻辑关系也就注意到了后面发射的时候怎么编 1 的码，怎么编 0 的码了。



建议不考虑多少个clock问题，后面延时函数里面去测
定就可以，但时长比例1:3要注意

上面的对于 1 位的编码按照从高位到低位顺序连接起来，再在开始加上一个同步码就完成了一个字节的传输，实现起来是很方便的。

发射一个字节流程如下：

1、 发射同步：逻辑 1 持续 1T(400uS 左右)然后逻辑 0 持续 31T(10mS~15mS 左右)

2、 顺序发射 bit7、bit6、bit5、bit4、bit3、bit2、bit1、bit0（逻辑 0 和 1 按照上面规则）

三、 如何对一个发射码进行编码？

对于一个发射码而言，按照 EV1527 的编码规范，发送/接收的码总共应该是 3 个字节，而这 3 个字节，并不是将上面单一字节内容直接拼接完成的。

例如编码“80A7E4”

字节	0x80		0xA7		0xE4	
顺序	Z1	Z2	Z3	Z4	Z5	Z6
Bit 数	**4**	**4**	**4**	**4**	**4**	**4**
值	**8**	**0**	**A**	**7**	**E**	**4**

上面 4 个字节,每个字节 8 位。但需要注意的是**蓝色部分（Z1~Z5）**是遥控器对码时候用的内部编码（对于远 R1 自学习过程而言），总共 2^20 个也就是编码规范中常说的百万编码，以此确保不会串码；**黄色部分(Z6)**只有半个字节，但这 4 位才对应实际按键编码，这个四位键盘编码在标准用法中只有四个值（1,2,4,8），也可以扩展直接用 1~15（0x01~0x0F）。

如果要完成 80A7E40B 代码的发射，流程如下：

1、 准备一个缓冲区 unsigned char Ask_send_buf[12]

2、 Ask_send_buf[0]=0x80；Ask_send_buf[1]=0xA7；Ask_send_buf[2]=0Xe4；

3、 Ask_send_buf[2]的低 4 位清零并给 Ask_send_buf[2]加上键盘码（1~15）；

4、 发射 Ask_send_buf[0]

5、 发射 Ask_send_buf[1]

6、 发射 Ask_send_buf[2]

7、 发射同步码（这个过程在编码规范里面是在前面的，但放在这个位置通讯更稳）

8、 暂停发射做一个发射间歇（一般有 15 毫秒合适）

上述实际上，按照通讯编码规约，完成一个发射码的过程必须要先后发送 2 次正确数据才符合 1527 的编码规范，因此，需要将上述过程中 4~8 步骤重复多次以确保准确接收。

四、 灵 R1 模块收码说明

对于一个发射码而言，上面完成了编码过程，对于接收端而言，首先必须经过对码学习过程才能完成接收和输出，

1、对码，在灵 R1 上将 K/O 端口和 GND 连续短暂短路两次，看到 LED 闪烁就进入对码状态，此刻开始发射端发码就能完成对码。数秒无操作则关灯退出，对码完成的码进入芯片保存掉电不会丢失。

2、设置输出模式：对码状态中，继续将 K/O 短暂接地，LED 连闪次数有变化，连续闪灯次数就是当前输出工作模式编号。

3、关于收到码

每次收到正确编码后 R1 的工作灯会闪烁一次，同时，在不同输出模式中将进行对应输出，工作模式 5 中需要提醒大家一下，作为串口输出模式，原本接收到的是发射端的 3 个字节，但输出为了保证串口数据传输可靠，数据前后均增加了一些信息。

继续以上面输出编码“80A7E4”为例可以看到接收端串口输出实际上是下面的数据

序	1	2	3	4	5	6	7	8	9	10	11	12	13
码	L	C	:	8	0	A	7	E	4	0	B	\r	\n

可以看到，发出的编码只有绿色部分的“80A7E4”,实际接收后，为确保数据准确前面增加了蓝色的数据头，后面增加了黄色的校验码，最终形成“LC:80A7E40B”的编码。

五、 调试注意事项

1、 硬件连接逻辑确认

（前面给出的连接关系别搞错线，注意所选用的模块工作电压不要出现烧片即可。）

2、 如何确认发射端已开始发码？

无线模块开始发射以后，由于无线 433 信号不依赖专用设备比较难以判断是否已经发射，建议可采用其他无线设备检查是否已经开始发射，可以用对讲机将频率调整为433MHz，发射端发码时候会有明显噪音出现。（凡带有自学习功能的遥控器或者遥控模块在未成功学习之前即使收到信号也没反应的）

3、 如何确认发射码正确？

接收端用灵 R1 模块，在确认发射端发射时进入学习状态进行学习，调整灵 R1 的输出模式为第 5 模式，然后通过 R1 模块自带串口输出功能将接受到的发射码发送到上位机查看。（需要注意模块接收的代码会自动加上前置字符串“LC:”,还会有后面的通讯校验字节）

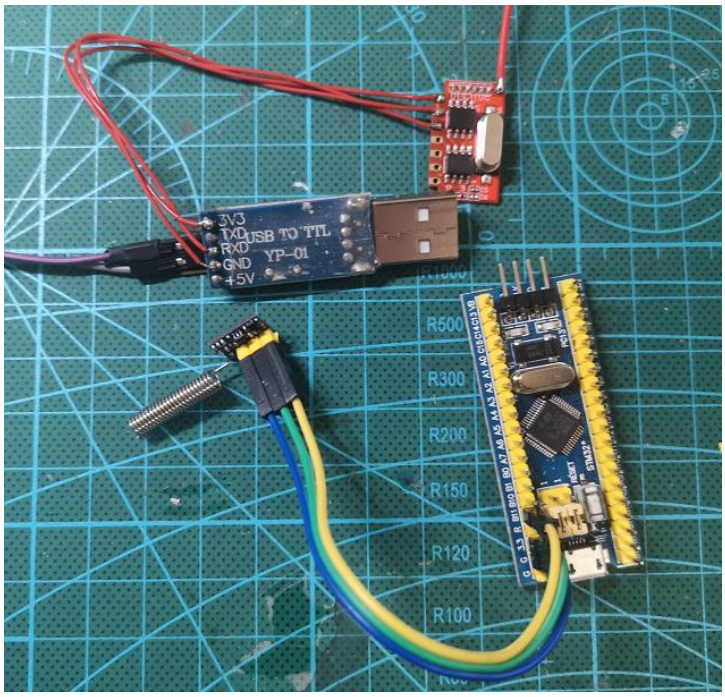
4、 如何确认 MCU 的机器周期时钟

不同的 MCU 和不同的时钟配置，指令周期都可能不同，采用代码延时（计时器更方便大家可自己移植）时，为了保证第一个时序图中一个 T 的时间基本准确在 400uS，最好采用示波器来矫正正确的循环次数。

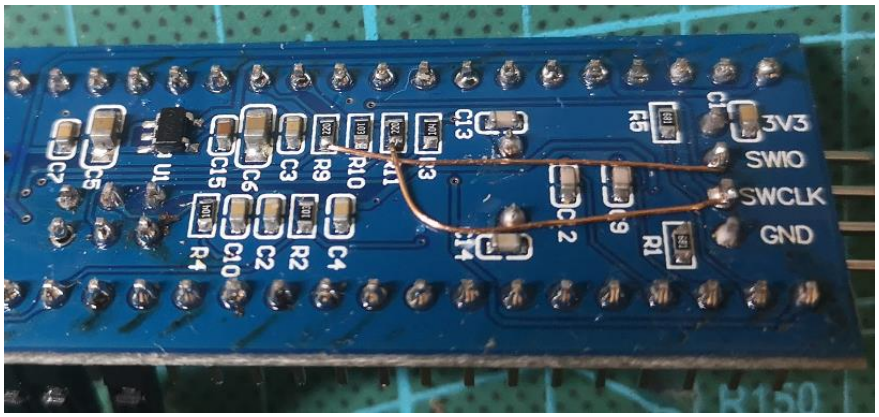
示波器不方便的时候，先搞个 1 分钟或者几分钟的延时通过秒表来反向推算一下 400uS 到底需要多少个等待周期；

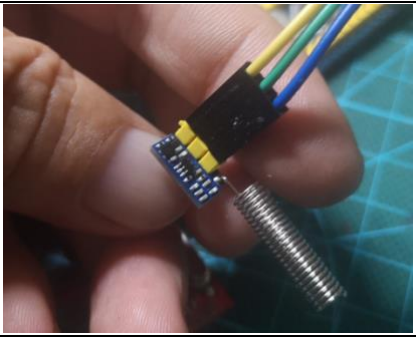
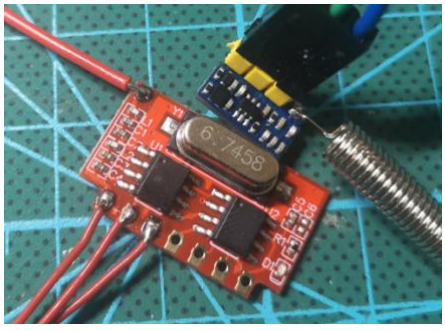
六、 硬件照片

顺手把调试硬件照片发上来，看看这么简单的连接就搞定了。



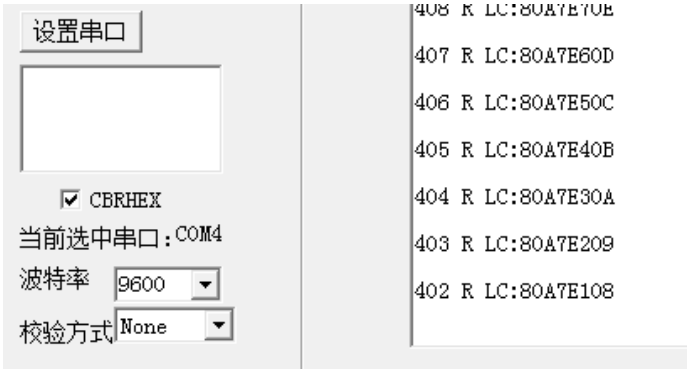
看看我用的调试板背面，手动飞了两根线，USB 接口就直接可以 SWD 方式下载调试这个 STM32 的开发板了，比较方便。



发射模块远-T1 的靓照	远-T1 和灵-R1
	

七、 上位机接收到的代码

调试过程中，灵 R1 模块除了完成与串口通讯模块的三根线连接之外，需要按照手册进行对码学习过程和输出模式设置过程。标准 9600 的串口通讯，使用还是比较简单的。



八、 调试完成的源代码

```
//==========================================================================
//文件名称：main.c
//功能概要： CPU 对远 T1 模块进行编码输出，按照 1527 编码规范,发送指定编码完成 433 发射端测试
//版权所有： 老雷子
//版权更新：2020-06-01
//调试方式：st-Link v2 swd 8MHz STM32F103C8T6  KEIL 4
//==========================================================================


//==========================================================================
//编码格式 nn nn nX tt                                    例如 80 A7 E2 0C （实际上是 80A7E20C）
//其中  nn nn n 是通讯对码要对的码                          例如上面的 80A7E
//       X       是键盘码，规范使用应该是 1,2,4,8 四个按键 例如 2 ,第二个按键 （对应 bit）
//       tt      通讯校验码，发射芯片收码后加入的码        例如 0C
//
//==========================================================================

//头文件
#include "stm32f10x.h"
#include "GPIOLIKE51.h"


#define ASK PBout(11)
```

```
#define LEDPOP PCout(13)


unsigned char Ask_send_buf[12]={0};
unsigned char table[]={"5EA7E8ED"};
unsigned char stm8s_id[12]={"5EA7E8ED"};//没用

int LedStat=1;

//函数声明
void GPIO_Configuration(void);

//============================================================================
//文件名称：Delay
//功能概要：延时
//参数说明：nCount：延时长短
//函数返回：无
//============================================================================

//pop  延时函数  没用这个延时
void Delay(uint32_t nCount)
{uint32_t i,j;
for(; nCount != 0; nCount--);;
}
//============================================================================

//pop  延时 1 毫秒
void Delay1ms(int mscount)
{int i,j;
      for (i=0;i<mscount;i++)
      for (j=0;j<8000;j++) //pop  示波器测试 1S
           ;
   //for(; nCount != 0; nCount--);;
}
//============================================================================

//POP  通讯编码中的长延时，3T 长度，按照 1527 编码规范要求，应该是 1.2 毫秒左右
void rf_delay_long()
{//实测 1.2ms
      //POP unsigned char i, j;
      int i;
      /*for(i=0;i<167;i++)
      {
            for(j=0;j<21;j++)
```

```
                ;
    }*/
    for(i=0;i<9600;i++)     //pop  示波器测试 1.2mS
        ;
}


//=============================================================================
//POP  通讯编码中的短延时，1T 长度，按照 1527 要求，应该是 400 纳秒左右
void rf_delay_short()
{    //实测 396~404us
    //POP unsigned char i, j;
    int i;
    /*for(i=0;i<55;i++)
    {
        for(j=0;j<21;j++)
            ;
    }*/
    for(i=0;i<3200;i++)       //pop  示波器测试 400uS
        ;
}

unsigned long gRfDelay = 0;//POP DEMO 里面自带，无用
//=============================================================================
//POP  发送编码 1，编码 1 的波形要求:高电平持续 3T+低电平持续 1T
void send_one()
{
    ASK=1;
    rf_delay_long();
    ASK=0;
    rf_delay_short();
}

//=============================================================================
//POP  发送编码 0，编码 1 的波形要求:高电平持续 1T+低电平持续 3T
void send_zero()
{
    ASK=1;
    rf_delay_short();
    ASK=0;
    rf_delay_long();
}

//=============================================================================
```

```
//POP 发送一个字节
void send_byte(unsigned char da)
{
    unsigned char i;
    for(i=8;i>0;i--)
    {
        if(da & 0x80)
        {
            send_one();
        }
        else
        {
            send_zero();
        }
        da = da<<1;
    }
}

//=============================================================================
//POP 发送一次编码
void ask_send(unsigned char datt[], unsigned char len)
{
     unsigned char i;
     for(i=0;i<len;i++)
     {
          /*pop 加，本意是为了同步码，但和 1527，2240，2260 编码表的实现方式都不一
样才正确，放弃这个以后正确
          ASK=1;
          rf_delay_short();
          ASK=0;
          rf_delay_long();
          //pop 上
          */
          send_byte(datt[i]);
     }
     //DEMO 带这一行，应该是增加的尾码，实际上起到同步码作用
     //按照通讯规范而言应该在每个字节之前的，也可在后面
     send_one();
     Delay1ms(15);//结合上面一行，形成了帧同步信号，注意 124CLK 直接用了 mS 延时
}

//=============================================================================
void Ask_process()
{
```

```
    unsigned char i;
    unsigned char key_value=0;
      LEDPOP=LedStat;LedStat=!LedStat;

      /////key_value=key_scan();
        //pop  此处的 key_value 规范用值应该是 1,2,4,8 四个数字分别对应数据四个 BIT
        //pop  扩展 key_value 可以用 1~15 这 15 个码
          key_value=0x02;

    while(1)
    {
      if(key_value != 0)//pop  原来的 DEMO 检测按键获得按键码做输出
      {
        for(i=0; i<3; i++)
        {//这个过程没用
          Ask_send_buf[i]=stm8s_id[1+i];
        }
              Ask_send_buf[0]=0x80;
              Ask_send_buf[1]=0xA7;
              Ask_send_buf[2]=0xE0;
              //Ask_send_buf[3]=0x33;

        Ask_send_buf[2] = (Ask_send_buf[2]&0xf0)| key_value;
        for(i=0; i<3; i++)
        {
          ///////Uart_Sendbyte(Ask_send_buf[i]);//demo  带有串口数据上行
        }
              //连续发 4 次确保通讯 OK,解码出一个
        ask_send(Ask_send_buf, 3);
        ask_send(Ask_send_buf, 3);
        ask_send(Ask_send_buf, 3);
        ask_send(Ask_send_buf, 3);
        Delay1ms(1000);//通讯间隙
              key_value=key_value+1;
              if (key_value>15) key_value=1;

      }
          LEDPOP=LedStat;LedStat=!LedStat;

    }
}

//==========================================================================
//main()
```

```
//==============================================================================

int main(void)
{int i,j,k;
        GPIO_Configuration();
          LEDPOP=0;
        Delay1ms(500);
          LEDPOP=1;
        Delay1ms(500);
          LEDPOP=0;
        Delay1ms(500);
          LEDPOP=1;
        Delay1ms(500);

          {
              Ask_process();

    }
}

//==============================================================================
//文件名称：GPIO_Configuration
//功能概要：GPIO 初始化
//参数说明：无
//函数返回：无
//==============================================================================
void GPIO_Configuration(void)
{
  GPIO_InitTypeDef GPIO_InitStructure;

  RCC_APB2PeriphClockCmd( RCC_APB2Periph_GPIOC , ENABLE);
  RCC_APB2PeriphClockCmd( RCC_APB2Periph_GPIOB , ENABLE);
//==============================================================================
//LED -> PC13
//==============================================================================

    //pop Pc13 用于 led 输出
  GPIO_InitStructure.GPIO_Pin = GPIO_Pin_13;
  GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
  GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP;
  GPIO_Init(GPIOC, &GPIO_InitStructure);
//----------------
    //pop PB11 用于写码输出
  GPIO_InitStructure.GPIO_Pin = GPIO_Pin_11|GPIO_Pin_10;
```

```
    GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP;////// GPIO_Mode_IPU;;
    GPIO_Init(GPIOB, &GPIO_InitStructure);

}


//=====end=============================================================================
```

天猫 APP 扫一扫

进入蜂鸟天猫旗舰店

微信扫一扫

关注蜂鸟微信公众号下载资料