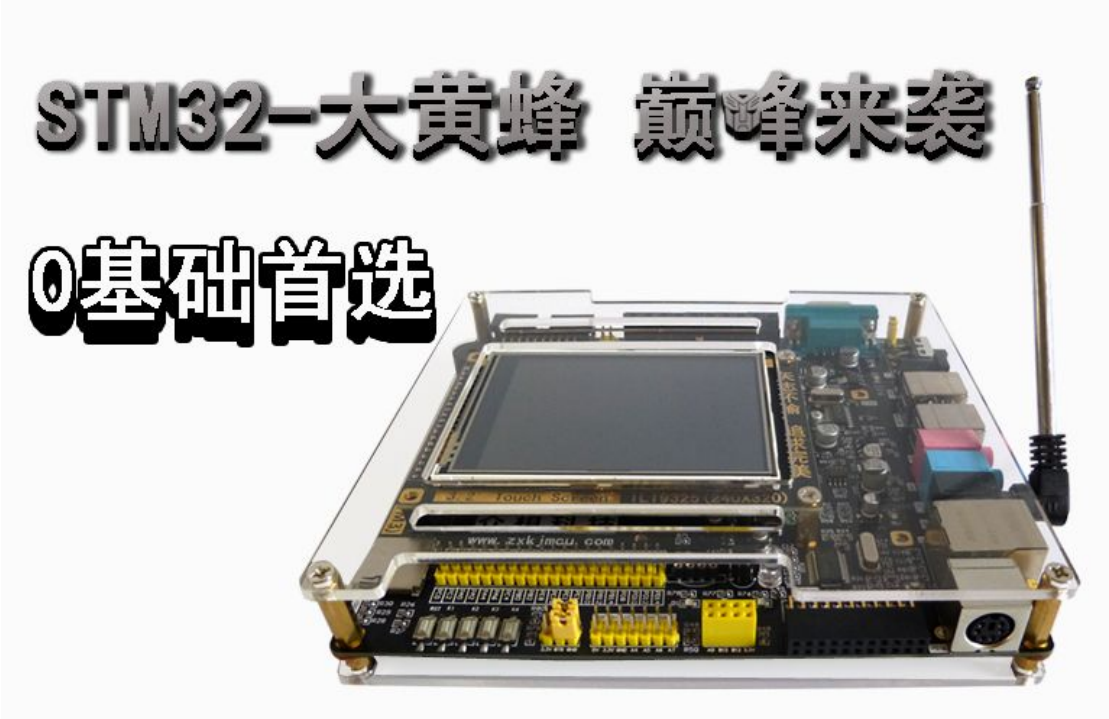
学 ARM 从 STM32 开始

STM32 开发板库函数教程--实战篇



官方网站：http://www.zxkjmcu.com
官方店铺：http://zxkjmcu.taobao.com
官方论坛：http://bbs.zxkjmcu.com
刘洋课堂：http://school.zxkjmcu.com

4.23 STM32 红外线工作原理

4.23.1 概述

人的眼睛能看到的可见光按波长从长到短排列，依次为红、橙、黄、绿、青、蓝、紫。其中红光的波长范围为 0.62～0.76μm；紫光的波长范围为 0.38～0.46μm。比紫光光波长更短的光叫紫外线，比红光波长更长的光叫红外线最广义地来说，传感器是一种能把物理量或化学量转变成便于利用的电信号的器件，红外传感器就是其中的一种。随着现代科学技术的发展，红外线传感器的应用已经非常广泛。

4.23.1.1 红外接收头工作原理

红外接收头一般是接收、放大、解调一体头，一般红外信号经接收头解调后，数据 “0”和“1”的区别通常体现在高低电平的时间长短或信号周期上，单片机解码时，通常将接收头输出脚连接到单片机的外部中断，结合定时器判断外部中断间隔的时间从而获取数据。重点是找到数据“0”与“1”间的波形差别。

3 条腿的红外接收头一般是接收、放大、解调一体化，接收头输出的是解调后的数据信号，单片机里面需要相应的读取程序。具体详细的使用参数和时序请参考官方技术手册。

4.23.1.2 在 STM32 实验系统中红外系统的组成

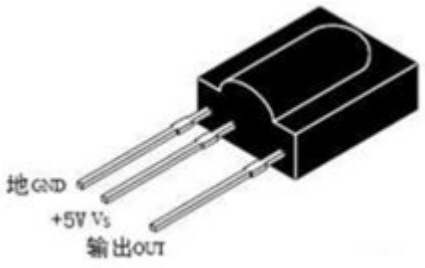
在我们是试验中使用的是红外线遥控器。因为红外线遥控器已经被广泛的使用在各类型的家电产品上，它的出现给使用电器提供了很多的便利。

红外线系统一般由红外发射装置和红外接收设备两大部分组成。红外发射装置又可由键盘电路、红外编码芯片、电源和红外发射电路组成。红外接收设备可由红外接收电路、红外解码芯片、电源和应用电路组成。通常为了使信号更好的被发射端发送出去，经常会将二进制数据信号调制成脉冲信号，通过红外发射管发射。常用的有通过脉冲宽度来实现信号调制的脉宽调制（PWM）和通过脉冲串之间的时间间隔来实现信号调制（PPM)两种方法。

1、常用的红外发光二极管其外形和发光二极管 LED 相似，发出红外光。管压降约 1.4v，工作电流一般小于 20mA。为了适应不同的工作电压，回路中常常串有限流电阻。

一些彩电红外遥控器，其红外发光管的工作脉冲占空比约为 1/3-1/4；一些电器产品红外遥控器，其占空比是 1/10。减小脉冲占空比还可使小功率红外发光二极管的发射距离大大增加。常见的红外发光二极管，其功率分为小功率(1mW-10mW)、中功率（20mW-50mW)和大功率（50mW-100mW 以上)三大类。红外发光二极管由红外辐射效率高的材料（常用砷化镓 GaAs）制成 PN 结，外加正向偏压向 PN 结注入电流激发红外光。光谱功率分布为中心波长 830～950nm，半峰带宽约 40nm 左右。

2、红外接收头的种类很多，如右图所示。引脚定义也不相同，一般都有三个引脚，包括供电脚，接地和信号输出脚。根据发射端调制载波的不同应选用相应解调频率的接收头。具体的选型要参考厂家选型手册。



红外接收头内部放大器的增益很大，很容易引起干扰，因此在接收头的供电脚上须加上滤波电容，一般在 22uf 以上。 有的厂家建议在供电脚和电源之间接入 330 欧电阻，进一步降低电源干扰。

4.23.1.3 红外发光二极管简易测试

高亮度 LED、红外 LED、光电三极管外形是一样的，非常容易搞混，因此需要通过简易测试将它们区分出来。用指针式万用表(1k 挡)黑表笔接阳极、红表笔接阴极(应采用带夹子的表笔)测得正向电阻在 20～40kΩ；黑表笔接阴极、红表笔接阳极测得反向电阻大于 500kΩ以上者是红外发光二极管。透明树脂封装的可用目测法：有圆形浅盘的极是负极。若正向电阻在 200kΩ以上(或指针微动)，反向电阻接近∞者是普通发光二极管。若黑表笔接短脚，红表笔接长脚，遮住光线时电阻大于 200kΩ，有光照射时阻值随光线强弱而变化(光线强时，电阻小)，这是光电三极管。

4.23.1.4 红外遥控常用的载波频率

红外遥控常用的载波频率为 38kHz，这是由发射端所使用的 455kHz 陶振来决定的。在发射端要对晶振进行整数分频，分频系数一般取 12，所以 455kHz÷12≈37.9 kHz≈38kHz。也有一些遥控系统采用 36kHz、40kHz、56kHz 等，一般由发射端晶振的振荡频率来决定。红外遥控的特点是不影响周边环境、不干扰其它电器设备。由于其无法穿透墙壁，故不同房间的家用电器可使用通用的遥控器而不会产生相互干扰；电路调试简单，只要按给定电路连接无误，一般不需任何调试即可投入工作；编解码容易，可进行多路遥控。

4.23.2 数据格式

熟悉数据格式是编程的基础。下面我们着重说明红外发射和接收的数据格式。

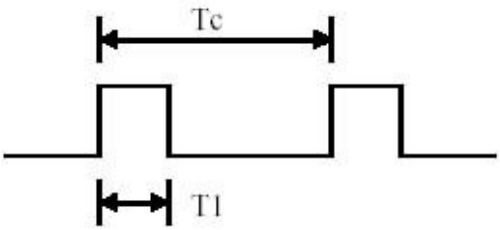
在同一个遥控电路中通常要使用实现不同的遥控功能或区分不同的机器类型，这样就要求信号按一定的编码传送，编码则会由编码芯片或电路完成。对应于编码芯片通常会有相配对的解码芯片或包含解码模块的应用芯片。在实际的产品设计或业余电子制作中，编码芯片并一定能完成我们要求的功能，这时我们就需要了解所使用的编码芯片到底是如何编码的。只有知道编码方式，我们才可以使用单片机或数字电路去定制解码方案。

- 载波波形

使用 455KHz 晶体，经内部分频电路，信号被调制在 37.91KHz，占空比为 3 分之 1。调制频率（晶振使用 455KHz 时）

fCAR = 1/Tc = fOSC/12 ≈ 38KHz （fOSC 是晶振频率）

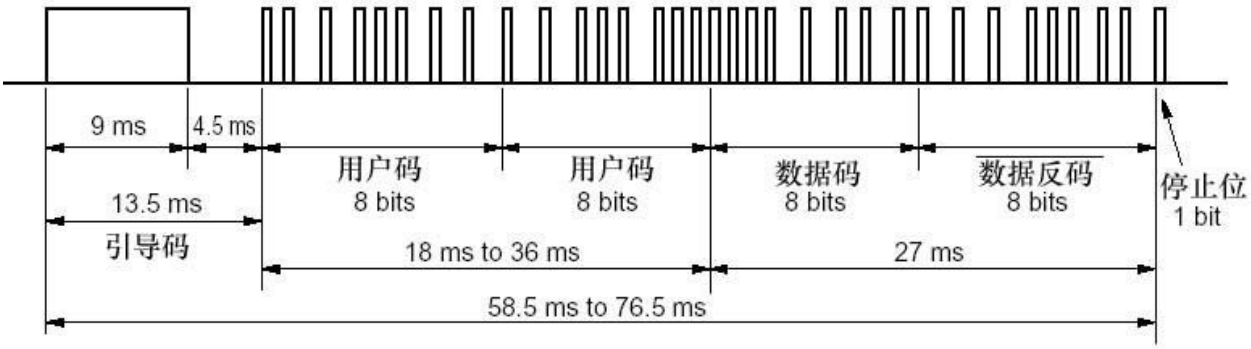
占空比 = T1/Tc = 1/3



- 数据格式

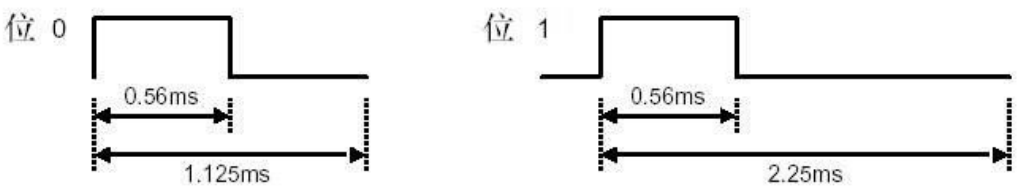
数据格式包括了引导码、用户码、数据码和数据码反码，编码总占 32 位。数据反码是 数据码反相后的编码，编码时可用于对数据的纠错。注意：第二段的用户码也可以在遥控应用电路中被设置成第一段用户码的反码。

使用 455KHz 晶振时各代码所占的时间



位定义

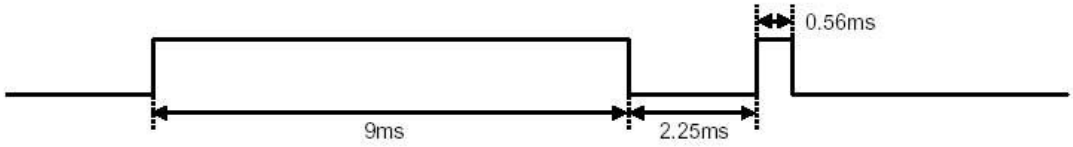
用户码或数据码中的每一个位可以是位‘1’，也可以是位‘0’。区分‘0’和‘1’是利 用脉冲的时间间隔来区分，这种编码方式称为脉冲位置调制方式，英文简写 PPM。



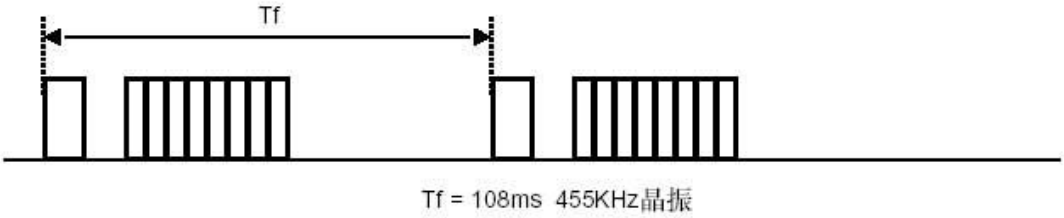
按键输出波形

uPD6121G 按键输出有二种方式：一种是每次按键都输出完整的一帧数据；另一种是按 下相同的按键后每发送完整的一帧数据后，再发送重复码，再到按键被松开。

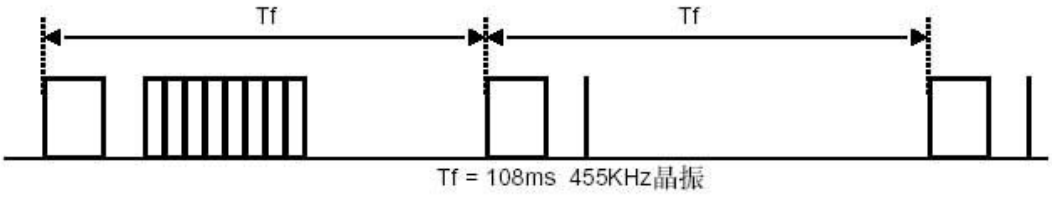
重复码



单一按键波形



连续按键波形



4.23.3 实验目的

通过我们选用的红外遥控器发射键盘值数据码，大黄蜂实验板上集成了 VS838 一体接收头接收遥控器发来的键盘值编码，经过 CPU 处理后送至

USB-RS232 串口输出至计算机显示。

4.23.4 硬件设计

利用实验板上集成的 VS838 红外接收电路，通过程序设计把接收到的红外线键盘编码打印输出到计算机显示。硬件设计见图 4.23.1　红外线发送原理图；图 4.23.2 红外线接收原理图。

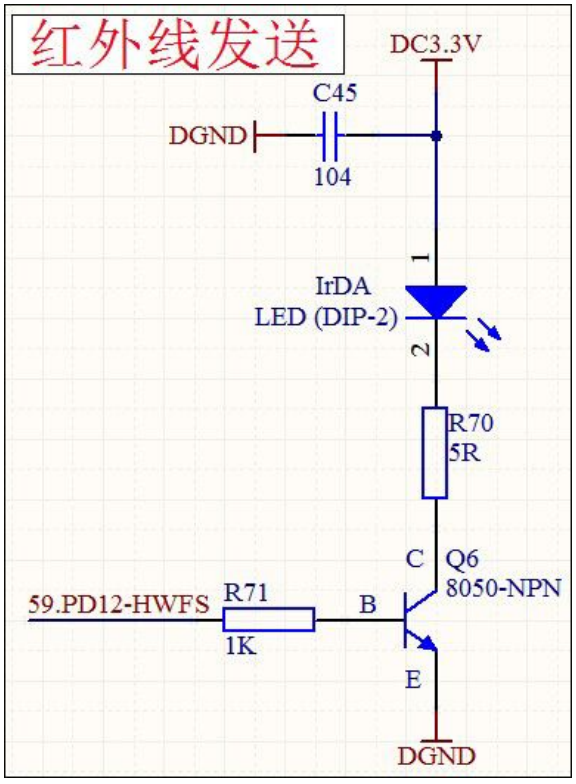


图 4.23.1　红外线发送原理图

红外线发送控制是 CPU 的第 59 管脚直接控制，控制三极管 Q6 的通断频率来使红外发光二极管（IrDA）发光。

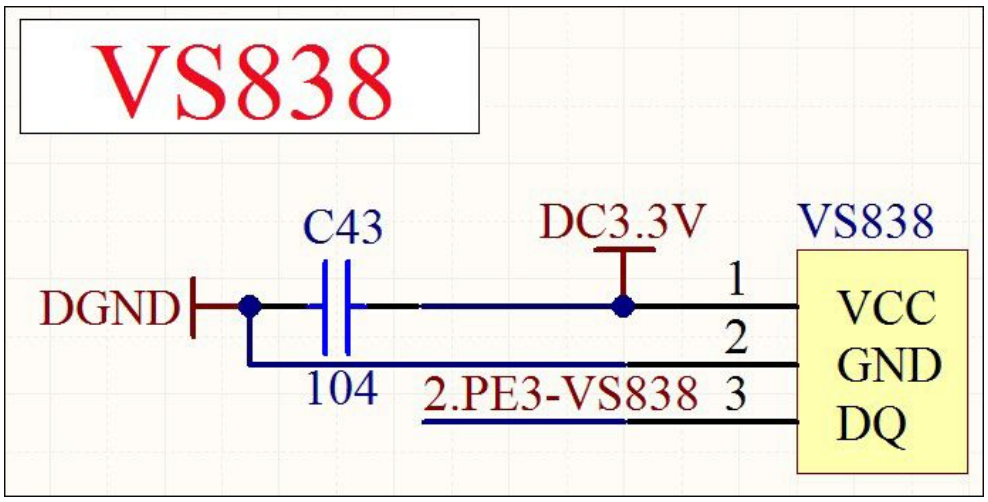


图 4.23.2 红外线接收原理图

红外线接收控制与 CPU 的第 2 管脚相连接，接收到的红外编码直接发送到 CPU,CPU 通过程序解码。

4.23.5 软件设计

4.23.5.1 软件设计说明

当红外线遥控器的按键按下时，VS838 接收到红外线波形，在它的输出端就会产生一个低电平信号，CPU 就可以检测到这个信号，解码后打印输出到 USB-RS232 串口。

在这节程序设计中，用到了外部中断函数；prinif 重定向打印输出函数； USART 串口通讯函数；定时器函数。

4.23.5.2 STM32 库函数文件

```
stm32f10x_gpio.c
stm32f10x_rcc.c
Misc.c  // 中断控制字（优先级设置）库函数
stm32f10x_exti.c // 外部中断库处理函数
stm32f10x_tim.c  // 定时器库处理函数
stm32f10x_usart.c  // 串口通讯函数
```

本节实验及以后的实验我们都是用到库文件，其中 stm32f10x_gpio.h

头文件包含了 GPIO 端口的定义。stm32f10x_rcc.h 头文件包含了系统时钟配置函数以及相关的外设时钟使能函数，所以我们要把这两个头文件对应的 stm32f10x_gpio.c 和 stm32f10x_rcc.c 加到工程中；Misc.c 库函数主要包含了中断优先级的设置，stm32f10x_exti.c 库函数主要包含了外部中断设置参数，tm32f10x_tim.c 库函数主要包含定时器设置，tm32f10x_usart.c 库函数主要包含串行通讯设置，这些函数也要添加到函数库中。以上库文件包含了本次实验所有要用到的函数使用功能。

4.23.5.3 自定义头文件

```
pbdata.h
pbdata.c
```

我们已经创建了两个公共的文件，这两个文件主要存放我们自定义的公共函数和全局变量，以方便以后每个功能模块之间传递参数。

4.23.5.4 pbdata.h 文件里的内容是

```
#ifndef _pbdata_H
#define _pbdata_H

#include "stm32f10x.h"
#include "misc.h"
#include "stm32f10x_exti.h"
#include "stm32f10x_tim.h"
#include "stm32f10x_usart.h"
#include "sttdio.h"
#include "hw_js.h"

extern u8 dt;       //定义全局变量
extern u32 hw_jsm;  //定义全局变量
extern u8  hw_jsbz; //定义全局变量

void RCC_HSE_Configuration(void); //定义函数
void delay(u32 nCount);
void delay_us(u32 nus);
void delay_ms(u16 nms);
```

```
#endif
```

语句 #ifndef、#endif 是为了防止 pbdata.h 文件被多个文件调用时出现错误提示。如果不加这两条语句，当两个文件同时调用 pbdata 文件时，会提示重复调用错误。

4.23.5.5 pbdata.c 文件里的内容是

pbdata.c 文件的开头先定义全局变量“u32 hw_jsm=0；u8 hw_jsbz=0”在整个程序中要调用他们。下面是 pbdata.c 文件详细内容，在文件开始还是引用“pbdata.h”文件。

```
    #include "pbdata.h"

    u8  dt=0;       //定义全局变量
    u32 hw_jsm=0;   //定义全局变量
    u8  hw_jsbz=0; //定义全局变量


    void RCC_HSE_Configuration(void) //HSE 作为 PLL 时钟，PLL 作为 SYSCLK
    {
       RCC_DeInit(); /*将外设 RCC 寄存器重设为缺省值 */
       RCC_HSEConfig(RCC_HSE_ON);    /*设置外部高速晶振（HSE） HSE 晶振打开
(ON)*/

       if(RCC_WaitForHSEStartUp() == SUCCESS) {  /*等待 HSE 起振,  SUCCESS：HSE
晶振稳定且就绪*/

       RCC_HCLKConfig(RCC_SYSCLK_Div1);/*设置 AHB 时钟(HCLK)RCC_SYSCLK_Div1——
—AHB 时钟 = 系统时*/
       RCC_PCLK2Config(RCC_HCLK_Div1); /*设置高速 AHB 时钟(PCLK2)RCC_HCLK_Div1
———APB2 时钟 = HCLK*/
       RCC_PCLK1Config(RCC_HCLK_Div2); /*设置低速 AHB 时钟(PCLK1)RCC_HCLK_Div2
———APB1 时钟 = HCLK / 2*/

       RCC_PLLConfig(RCC_PLLSource_HSE_Div1, RCC_PLLMul_9);/*设置 PLL 时钟源及
倍频系数*/
       RCC_PLLCmd(ENABLE);    /*使能 PLL */
       while(RCC_GetFlagStatus(RCC_FLAG_PLLRDY) == RESET) ; /*检查指定的 RCC
标志位(PLL 准备好标志)设置与否*/
```

```
    RCC_SYSCLKConfig(RCC_SYSCLKSource_PLLCLK);  /*设置系统时钟(SYSCLK) */
    while(RCC_GetSYSCLKSource() != 0x08);      /*0x08：PLL 作为系统时钟 */

  }
}

void delay(u32 nCount)
{
   for(;nCount!=0;nCount--);
}
/*******************************************************************************
* 名    称：delay_us(u32 nus)
* 功    能：微秒延时函数
* 入口参数：u32  nus
* 出口参数：无
* 说    明：
* 调用方法：无
*******************************************************************************/
void delay_us(u32 nus)
{
    u32 temp;
    SysTick->LOAD = 9*nus;
    SysTick->VAL=0X00;//清空计数器
    SysTick->CTRL=0X01;//使能，减到零是无动作，采用外部时钟源
    do
    {
     temp=SysTick->CTRL;//读取当前倒计数值
    }while((temp&0x01)&&(!(temp&(1<<16))));//等待时间到达

    SysTick->CTRL=0x00; //关闭计数器
    SysTick->VAL =0X00; //清空计数器
}

/*******************************************************************************
* 名    称：delay_ms(u16 nms)
* 功    能：毫秒延时函数
* 入口参数：u16 nms
* 出口参数：无
* 说    明：
* 调用方法：无
*******************************************************************************/
void delay_ms(u16 nms)
{
```

```
    //注意 delay_ms 函数输入范围是 1-1863
    //所以最大延时为 1.8 秒

    u32 temp;
    SysTick->LOAD = 9000*nms;
    SysTick->VAL=0X00;//清空计数器
    SysTick->CTRL=0X01;//使能，减到零是无动作，采用外部时钟源
    do
    {
     temp=SysTick->CTRL;//读取当前倒计数值
    }while((temp&0x01)&&(!(temp&(1<<16))));//等待时间到达
    SysTick->CTRL=0x00; //关闭计数器
    SysTick->VAL =0X00; //清空计数器
}
```

4.23.6 STM32 系统时钟配置 SystemInit()

我们总在强调，每个工程都必须在开始时配置并启动 STM32 系统时钟，这次也不例外。

4.23.7 GPIO 引脚时钟使能

```
    SystemInit();//72m
    RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOA,ENABLE);
    RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOE,ENABLE);
    RCC_APB1PeriphClockCmd(RCC_APB2Periph_USART1,ENABLE);//设置串口 1 时钟使
能
    RCC_APB2PeriphClockCmd(RCC_APB2Periph_AFIO,ENABLE);//功能复用 IO 时钟使能
```

本节实验用到了 PE 端口，所以要把 PE 端口的时钟打开；串口 1 时钟源是通过 APB2 预分频器得到的，串口 1 时钟初始化；因为要与外部芯片通讯，所以要打开功能复用时钟。

4.23.8 GPIO 管脚电平控制函数

在主程序中采用 while(1)循环语句，等待外部中断的到来，当红外接收中断来了以后，在主程序中判断 PE3 是否为低电平，如果是说明有红外遥控编码输入，就打印输出到屏幕。

```
    while(1)
```

```
    {
        if(hw_jsbz==1) //如果标志等于 1，就顺序执行
        {
            hw_jsbz=0; //先清标志
            printf("红外接收码 %0.8X\r\n",hw_jsm);//打印输入接收到的编码

            hw_jsm=0;  //清编码存储器
        }
        delay_ms(1000);
    }
}
```

4.23.9 stm32f10x_it.c 文件里的内容是

在中断处理 stm32f10x_it.c 文件里中仅串口 1 子函数非空，进入中断处理函数后，只有串口 1 有参数输出。

```
#include "stm32f10x_it.h"
#include "stm32f10x_exti.h"
#include "stm32f10x_rcc.h"
#include "misc.h"
#include "pbdata.h"

void NMI_Handler(void)
{
}

void USART1_IRQHandler(void)
{
    if(USART_GetITStatus(USART1,USART_IT_RXNE)!=RESET)
    {
        USART_SendData(USART1,USART_ReceiveData(USART1));
        while(USART_GetFlagStatus(USART1,USART_FLAG_TXE)==RESET);
    }
}
/*******************************************************************************
* 名    称：void EXTI3_IRQHandler(void)
* 功    能：EXTI3 中断处理程序
* 入口参数：无
* 出口参数：无
* 说    明：
* 调用方法：无
*******************************************************************************/
void EXTI3_IRQHandler(void)
```

```
{
    u8 Tim=0,Ok=0,Data,Num=0;

    while(1)
    {
        if(GPIO_ReadInputDataBit(GPIOE,GPIO_Pin_3)==1)
        {
            Tim=HW_jssj();//获得此次高电平时间

            if(Tim>=250) break;//不是有用的信号

            if(Tim>=200 && Tim<250)
            {
              Ok=1;//收到起始信号
            }
            else if(Tim>=60 && Tim<90)
            {
              Data=1;//收到数据 1
            }
            else if(Tim>=10 && Tim<50)
            {
              Data=0;//收到数据 0
            }

            if(Ok==1)
            {
              hw_jsm<<=1;    //接收码寄存器向左移动一位
              hw_jsm+=Data; //把接收到的编码存入接收码寄存器的最低位

              if(Num>=32)    //判断是否接收了 32 位
              {
                  hw_jsbz=1;//接收完 32 位后置接收完标志
                  break;
              }
            }

            Num++;
        }
    }

    EXTI_ClearITPendingBit(EXTI_Line3);//清中断
}
```

4.23.10 hw_hs.h 文件里的内容是

函数 hw_hs.h 在这里是为了红外程序自定义的功能函数，hw_hs.h 的内容如下：

```
#ifndef _HW_JS_H
#define _HW_JS_H
#include "pbdata.h"

void EXTI_Configuration_HW(void);
u8 HW_jssj(void);

#endif
```

4.23.11 hw_hs.c 文件里的内容是

自定义函数 hw_hs.c 的内容如下：

```
#include "pbdata.h"

void EXTI_Configuration_HW(void)
{
   EXTI_InitTypeDef EXTI_InitStructure;

   EXTI_ClearITPendingBit(EXTI_Line3);
   GPIO_EXTILineConfig(GPIO_PortSourceGPIOE, GPIO_PinSource3);

   EXTI_InitStructure.EXTI_Line=EXTI_Line3;
   EXTI_InitStructure.EXTI_Mode=EXTI_Mode_Interrupt;
   EXTI_InitStructure.EXTI_Trigger=EXTI_Trigger_Falling;
   EXTI_InitStructure.EXTI_LineCmd=ENABLE;

   EXTI_Init(&EXTI_InitStructure);
}

u8 HW_jssj(void)
{
   u8 t=0;

   while(GPIO_ReadInputDataBit(GPIOE,GPIO_Pin_3)==1)//高电平
   {
      t++;
      delay_us(20);

      if(t>=250) return t;//超时溢出
   }
```

```
    return t;
}
```

4.23.12 main.c 文件里的内容是

```
#include "pbdata.h"

void RCC_Configuration(void);
void GPIO_Configuration(void);
void NVIC_Configuration(void);
void USART_Configuration(void);

int fputc(int ch,FILE *f)
{
    USART_SendData(USART1,(u8)ch);
    while(USART_GetFlagStatus(USART1,USART_FLAG_TXE)==RESET);
    return ch;
}

int main(void)
{
    RCC_Configuration(); //系统时钟初始化
    GPIO_Configuration();//端口初始化
    USART_Configuration();
    NVIC_Configuration();
    EXTI_Configuration_HW();

    while(1)
    {
            if(hw_jsbz==1)
        {
            hw_jsbz=0;
            printf("红外接收码 %0.8X\r\n",hw_jsm);

            hw_jsm=0;
        }

        delay_ms(1000);
    }
}

void RCC_Configuration(void)
{
```

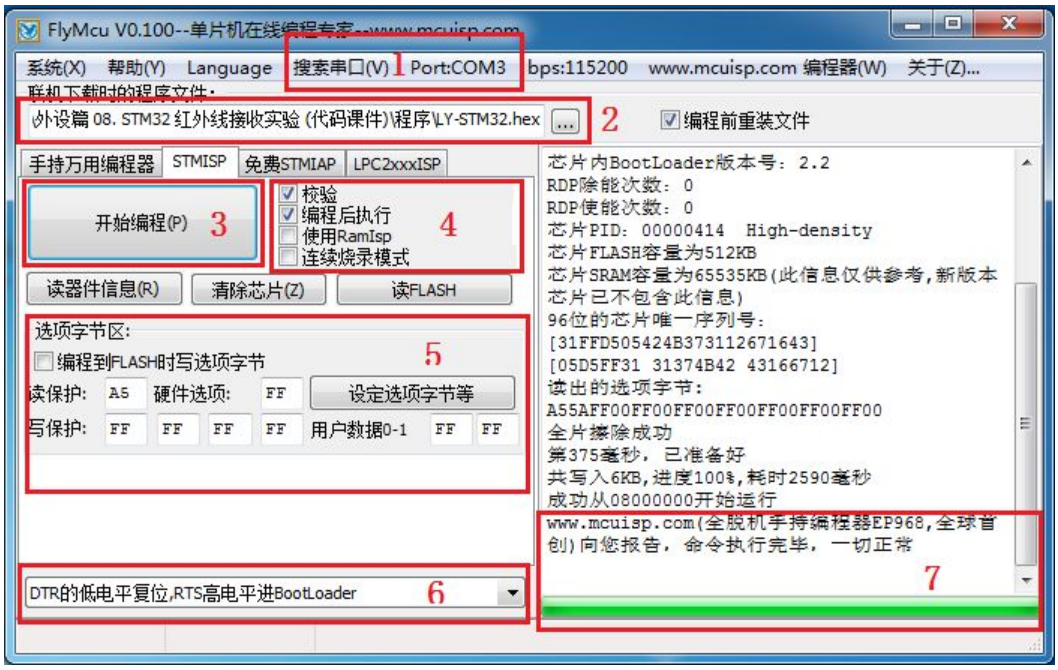
```
    SystemInit();//72m
   RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOA,ENABLE);
   RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOE,ENABLE);
   RCC_APB2PeriphClockCmd(RCC_APB2Periph_USART1,ENABLE);
   RCC_APB2PeriphClockCmd(RCC_APB2Periph_AFIO,ENABLE);
}

void GPIO_Configuration(void)
{
    GPIO_InitTypeDef GPIO_InitStructure;
   //LED
   GPIO_InitStructure.GPIO_Pin=GPIO_Pin_9;//TX
   GPIO_InitStructure.GPIO_Speed=GPIO_Speed_50MHz;
   GPIO_InitStructure.GPIO_Mode=GPIO_Mode_AF_PP;
   GPIO_Init(GPIOA,&GPIO_InitStructure);

   GPIO_InitStructure.GPIO_Pin=GPIO_Pin_10;//RX
   GPIO_InitStructure.GPIO_Mode=GPIO_Mode_IN_FLOATING;
   GPIO_Init(GPIOA,&GPIO_InitStructure);

   GPIO_InitStructure.GPIO_Pin=GPIO_Pin_3;//红外接收
   GPIO_InitStructure.GPIO_Mode=GPIO_Mode_IPU;
   GPIO_Init(GPIOE,&GPIO_InitStructure);
}

void NVIC_Configuration(void)
{
      NVIC_InitTypeDef NVIC_InitStructure;

   NVIC_PriorityGroupConfig(NVIC_PriorityGroup_1);

   NVIC_InitStructure.NVIC_IRQChannel = USART1_IRQn;
   NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority = 0;
   NVIC_InitStructure.NVIC_IRQChannelSubPriority = 1;
   NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;
   NVIC_Init(&NVIC_InitStructure);

   NVIC_InitStructure.NVIC_IRQChannel = EXTI3_IRQn;
   NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority = 0;
   NVIC_InitStructure.NVIC_IRQChannelSubPriority = 1;
   NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;
   NVIC_Init(&NVIC_InitStructure);
}
```

```
    void USART_Configuration(void)
    {
        USART_InitTypeDef  USART_InitStructure;

       USART_InitStructure.USART_BaudRate=9600;
       USART_InitStructure.USART_WordLength=USART_WordLength_8b;
       USART_InitStructure.USART_StopBits=USART_StopBits_1;
       USART_InitStructure.USART_Parity=USART_Parity_No;
       USART_InitStructure.USART_HardwareFlowControl=USART_HardwareFlowContr
ol_None;
       USART_InitStructure.USART_Mode=USART_Mode_Rx|USART_Mode_Tx;

       USART_Init(USART1,&USART_InitStructure);
       USART_ITConfig(USART1,USART_IT_RXNE,ENABLE);
       USART_Cmd(USART1,ENABLE);
       USART_ClearFlag(USART1,USART_FLAG_TC);
    }
```

4.23.13 程序下载

在这一章节中要掌握红外发射和接收的工作时序，了解常用的红外遥控器功能和原理。

请根据下图所指向的 7 个重点区域配置。其中（1）号区域根据自己机器的实际情况选择，我的机器虚拟出来的串口号是 COM3。（2）号区域请自己选择程序所在的文件夹。（7）号区域当程序下载完后，进度条会到达最右边，并且提示一切正常。（4、5、6）号区域一定要按照上图显示的设置。当都设置好以后就可以直接点击（3）号区域的开始编程按钮下传程序了。



本节实验的源代码在光盘中：（LY-STM32 光盘资料\1.课程\2,外设篇\外设篇 08. STM32 红外线接收实验\程序）

4.23.14 实验效果图

使用众想科技多功能监控软件，可以观察到大黄蜂实验板接收到的红外遥控器发射过来的键值编码，不同的编码对应不同的键值编码。

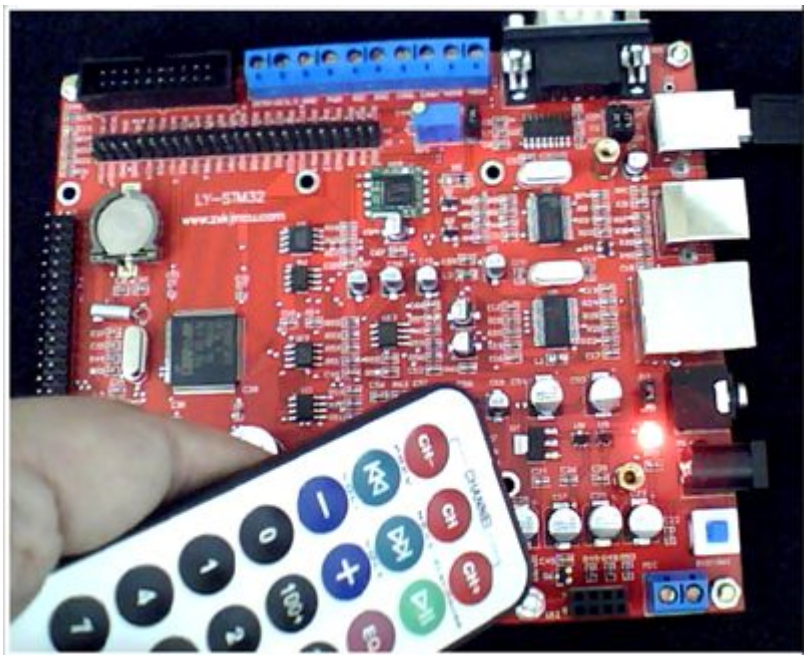


图 4.23.14.1 红外线接收实验效果图一

图 4.23.14.2 红外线接收实验效果图二