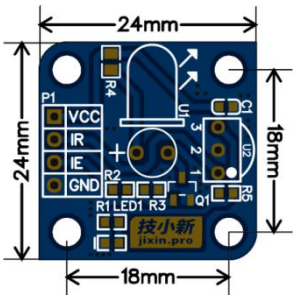
红外收发模块 IR333C + IRM-3638T 学习手册

1、模块介绍

本模块将红外发射器与红外接收器集成到一起，红外发射器是 IR333C，红外接收器是 IRM-3638T。既可以实现收发自测，也可以使用两个模块实现远距离红外通信。工作电压宽，体积小，通信距离长。



- 工作电压宽：2.7V~5.5V
- 收/发红外波长：940nm
- 收/发载波频率：38kHz
- 收发模式：自发自收 / 双模块远程通信

➢ 模块接口引脚功能表：

Symbol（符号）	Type（类型）	Description（描述）
VCC	电源	电源电压
IR	信号输出（对模块而言）	IRM-3638T 电平信号输出端
IE	信号输入（对模块而言）	IR333C 红外发射控制端
GND	电源	地

注：

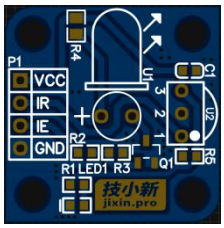
- VCC、GND：本模块的供电电压：2.7V~5.5V
- IR：IRM-3638T 的信号输出端（IRM-3638T 的 1 脚），根据接收到的红外载波，输出电平信号
- IE：IR333C 红外发射控制端，由微控制器控制 IR333C 发射红外载波

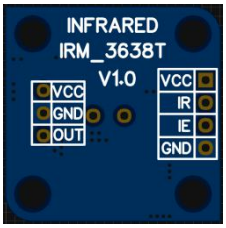
➢ LED 指示灯说明：

- LED1：电源指示灯。灯亮表明模块供电正常。

➢ 丝印说明：

- 元器件附近的“字符+数字”表示元器件编号
- 焊盘位置附近的“字符串”表示接口的引脚功能





2、模块用途

用途一：当只有一个模块时，可以配合微控制器实现自发自收，学习红外编解码。

用途二：当只有一个模块时，可以配合微控制器实现红外解码（如：解码电视遥控器），然后可以将本模块作为遥控器使用（就可以遥控电视啦！）。

用途三：当有两个模块时，可以配合微控制器进行远程通信，实现遥控功能。（无线通信）

3、硬件设计

3.1、红外光简介

红外线是太阳光线中众多不可见光线中的一种，又称为红外热辐射。太阳光谱中，红光的外侧存在着看不见的光线，这就是红外线。也可以用来传输信息。太阳光谱上红外线的波长大于可见光线，波长为 0.75～1000μm。红外线可分为三部分，近红外线：波长为 0.75～1.50μm 之间；中红外线：波长为 1.50～4.0μm 之间；远红外线：波长为 4.0～l000μm 之间。

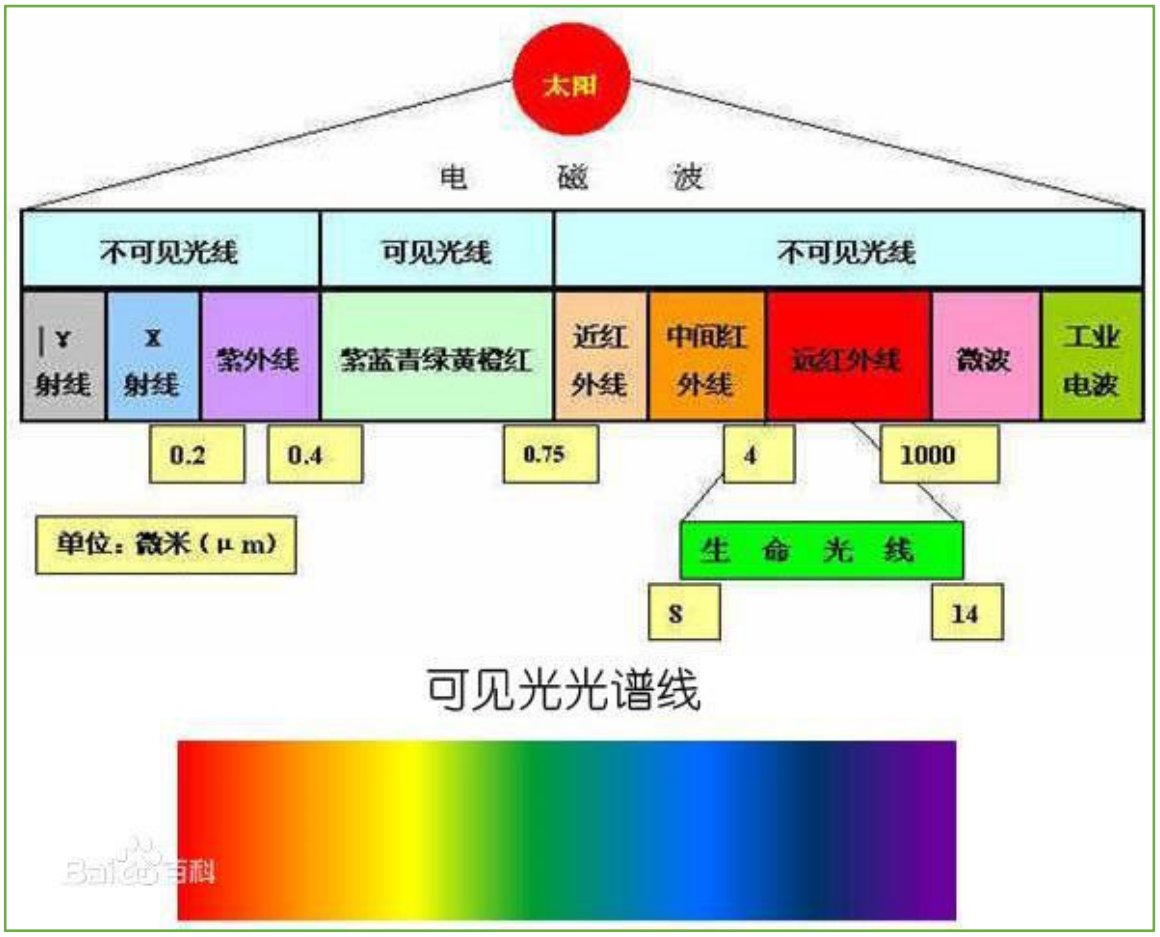


图 1：太阳光光谱图

3.2、红外收发系统简介

如果想让红外收发系统（发射器 + 接收器）能够正常收发信号，必须满足下面两个条件：

- ①：红外发射器的红外发射波长必须和红外接收器的接收波长一致或相近。(器件硬件属性)
- ②：为了在信号传输过程中免受其他红外信号的干扰，通常是发射方调制信号，接收方解调信号。

  - 发射端调制信号：由外部的控制信号（一般由微控制器实现），控制红外发射器以特定的频率发射红外光；发射出去的红外光以特定频率的载波形式存在于空间中，这就是调制。红外发射器所发射的特定频率的载波必须和红外接收器的接收载波频率一致或相近。
    注：外部的控制信号一般由微控制器实现。也就是说，调制的过程由微控制器实现。

  - 接收端解调信号：红外接收器将会滤除其他杂波，只接收该特定频率的载波信号，并将其还原成电平信号（一连串的高/低电平信号，就组成了一连串的脉冲信号），这就是解调。
    注：解调是由红外接收器的硬件完成。而将脉冲信号译码成具体数据的过程，由微控制器实现。

3.3、红外发射器 IR333C

➢ Peak wavelength（峰值波长） λp = 940nm（IR333C 发射的红外光的最强波长为 940nm）

Peak Wavelength	λp	I_F=20mA	--	940	--	nm
Spectral Bandwidth	Δλ	I_F=20mA	--	45	--	nm

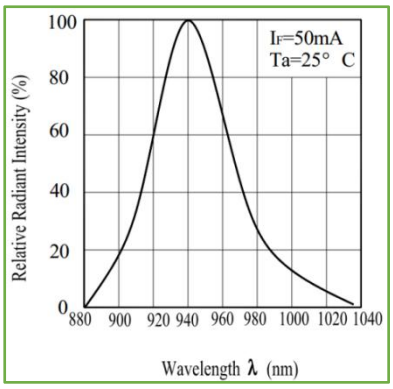


图 2：IR333C 发射的红外波长强度

注：

◆ 红外发射器和红外接收器，这两者的波长参数必须一致或接近。

■ 红外发射器 IR333C 发出的红外波长为 940nm 附近。

■ 红外接收器 IRM-3638T 的接收波长为 940nm。

（见图 5：红外接收器 IRM-3638T 的电气特性）。

➢ 红外发射器的电路设计

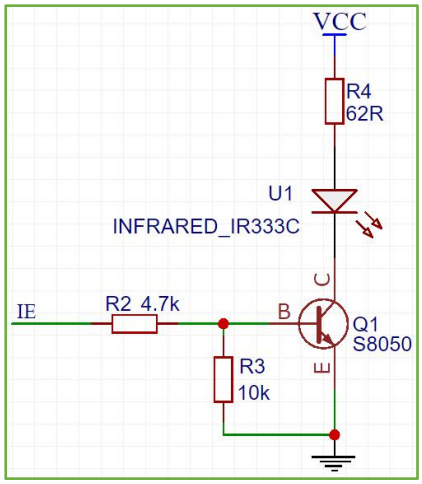


图 3：IR333C 红外发射原理图

◆ 当 IE 为高电平时，IR333C 发射红外光

◆ 当 IE 为低电平时，IR333C 不发射红外光。

注：

◆ 红外发射器 IR333C 只能由 IE 管脚控制发射或不发射红外光。

◆ 如果想让红外发射器 IR333C 发出红外接收器 IRM-3638T 能够接收的红外载波，需要微控制器控制 IE 端来实现。

3.4、红外接收器 IRM-3638T

◆ 工作电压：2.7V~5.5V

◆ IRM-3638T 是适用于红外控制系统的小型化芯片。接收器集成了管脚二极管和前置放大器，环氧基树脂的封装提高红外滤波性能。解调输出的信号可以由微控制器译码。

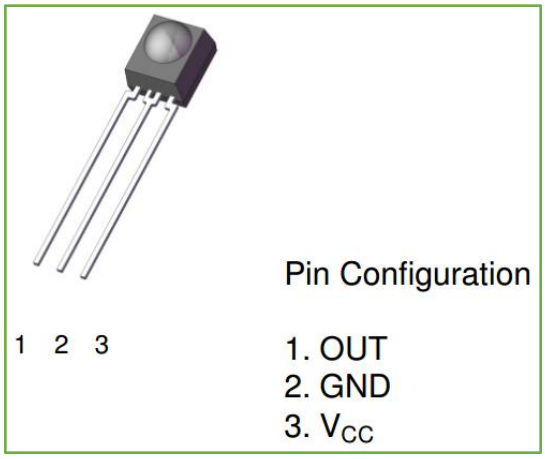


图 4：IRM-3638T 管脚说明

注：

◆ 当 IRM-3638T 没有接收到红外载波信号时，OUT 管脚输出高电平。（外部弱上拉）

◆ 当 IRM-3638T 接收到红外载波信号时，OUT 管脚输出低电平。

◆ 关于红外载波信号，详见 3.3 节

Parameter	Symbol	Min.	Typ.	Max.	Unit	Condition
Current consumption	Icc	---	1.0	1.2	mA	No input signal
Peak wavelength	λ_p	---	940	---	nm	
Reception range	L_0	14	---	---	m	At the ray axis
	L_{45}	6	---	---		

图 5：红外接收器 IRM-3638T 的电气特性

Model No.	Carrier Frequency
IRM-3636T	36 kHz
IRM-3638T	38 kHz
IRM-3640T	40 kHz
IRM-3656T	56 kHz

图 3-5：红外接收器 IRM-3638T 接收的载波频率

注：对于 IRM-3638T，接收的载波频率为 38kHz。那对于红外发射器 IR333C，需发射频率为 38KHz 的载波。

- 根据 IRM-3638T 的接收载波频率，计算 IR333C 红外光的发射方式：（这就是载波调制）
  - 载波周期：1s / 38KHz（频率）≈ 26.3us（周期）(注：不要求是准确的 38kHz，但必须和 38kHz 相近)
  - 26.3us（一个周期）= 8.77us（IE 为高，发射红外光）+ 17.53us（IE 为低，不发射红外光）
    - 注：一个载波周期中，发射红外光 8.77us + 不发射红外光 17.53us，这个周期就是**载波发射周期**。
    - 注：**载波发射周期**中，发射红外光的占空比一般为 1/3。
  - 26.3us（一个周期）= 26.3us（IE 为低，不发射红外光）
    - 注：整个周期内，都不发射红外光，这个周期为**载波不发射周期**。

- 红外发射器+红外接收器的电路设计

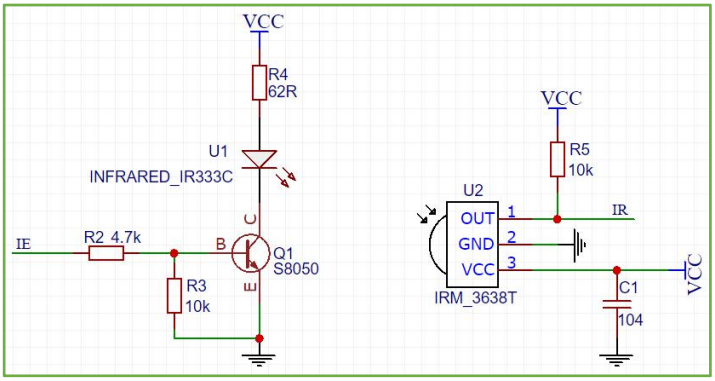


图 3-4：发射与接收电路

注：

- 如果 IR333C 发射 x 个**载波发射周期**(频率：38kHz、周期：26.3us)，IRM-3638T 接收到 x 个持续 26.3us 的红外载波，OUT 管脚将输出“x*26.3us”的低电平。（如：0.56ms ≈ 21 * 26.3us ）
- 如果 IRM-3638T 没有接收到**载波发射周期**，OUT 管脚输出高电平。(10k 电阻上拉)

4、软件设计（微控制器为：STM32F103VET6）

4.1、红外编解码协议

- 红外遥控包括两种方式：
  - PWM（脉冲宽度调制）：以红外载波的占空比表示‘0’和‘1’
    - 一般情况下，发射红外载波的时间固定，通过改变不发射载波的时间来改变占空比。
  - PPM（脉冲位置调制）：以发射载波的位置表示‘0’和‘1’
    - 从发射载波到不发射载波为‘0’，从不发射载波到发射载波为‘1’
    - 其发射载波和不发射载波的时间相同，都为 0.68ms，也就是每位的时间是固定的。
  - 注：这里只对 PWM（脉冲宽度调制）进行说明。

- PWM（脉冲宽度调制）：这种形式的编码的代表为“NEC”
  - 常用的是 NEC_upd6121 协议（电视遥控器等）：
    - ‘0’：载波发射 0.56ms + 不发射 0.56ms
    - ‘1’：载波发射 0.56ms + 不发射 1.68ms
    - 引导码：载波发射 9ms + 不发射 4.5ms
    - 结束码‘0’：载波发射 0.56ms + 不发射 0.56ms

- 详解 NEC_upd6121 协议
  - 对于红外发射器 IR333C
    - 发送协议数据‘0’= 发射载波 0.56ms + 不发射载波 0.56ms
      - 前部分的发射载波 0.56ms，不是指将 IE 端抬高 0.56ms 来让 IR333C 发射 0.56ms 的红外光，而是控制 IE 端发射时长为 0.56ms 的载波（ 0.56ms ≈ 26.3us * 21 ）。
      - 后部分的不发射载波 0.56ms，指控制 IE 端不发射载波 0.56ms（ 0.56ms ≈ 26.3us * 21 ）。
    - 发送协议数据‘1’= 发射载波 0.56ms + 不发射载波 1.68ms
      - 前部分的发射载波 0.56ms，不是指将 IE 端抬高 0.56ms 来让 IR333C 发射 0.56ms 的红外光，而是控制 IE 端发射时长为 0.56ms 的载波（ 0.56ms ≈ 26.3us * 21 ）。
      - 后部分的不发射载波 1.68ms，指控制 IE 端不发射载波 1.68ms（ 1.686ms ≈ 26.3us * 64 ）
    - 发送协议数据‘0’和‘1’的区别：
      - 前部分的发射载波时长是相同的。（发射载波 0.56ms）
      - 后部分的不发射载波的时长是不同的。（不发射载波 0.56ms / 1.68ms）
      - ‘0’和‘1’发射红外载波的占空比不同，由此来区别‘0’和‘1’。即为：PWM（脉冲宽度调制）
  - 对于红外接收器 IRM-3638T，OUT 管脚（模块接口引脚 IR）的电平输出情况：
    - IRM-3638T 接收到 38kHz 的红外载波时，OUT 管脚（模块接口引脚 IR）输出低，否则输出高
    - 接收到协议数据‘0’= 0.56ms 低电平 + 0.56ms 高电平
    - 接收到协议数据‘1’= 0.56ms 低电平 + 1.68ms 高电平

- 数据帧的格式：
  - 数据是以帧为单位，每一帧由 8 位数据组成
  - 顺序：低位在前，由低到高发送 8 位数据
  - NEC 协议中，有以下 4 个数据帧：
    16 位用户码低字节(8 位)、16 位用户码高字节(8 位)、8 位数据码、8 位数据码的反码

◆ 引导码
- IR333C 发射引导码 = 载波发射 9ms + 载波不发射 4.5ms
  - 载波发射 9ms：由 342 个载波发射周期组成（9ms ≈ 26.3us * 342）
  - 载波不发射 4.5ms：由 171 个载波不发射周期组成（4.5ms ≈ 171 * 26.3us）
- RM-3638T 接收引导码 = 低电平 9ms + 高电平 4.5ms

◆ 结束码‘0’（结束位）
- IR333C 发射停止码（协议数据‘0’） = 载波发射 0.56ms + 载波不发射 0.56ms
- RM-3638T 接收停止码 = 低电平 0.56ms + 高电平 0.56ms

◆ NEC 编解码的格式顺序：
- ①：引导码（载波发射 9ms + 载波不发射 4.5ms）
- ②：低 8 位用户码
- ③：高 8 位用户码
- ④：8 位数据码
- ⑤：8 位数据的反码
- ⑤：结束码“0”

4.2、程序编写

注：硬件平台为 STM32F103VET6 最小系统板

4.2.1、红外发射程序编写

➢ 首先将微控制器的红外发射控制管脚初始化为输出低

```
// 初始化红外发射管脚：Infrared_IE = PC1
//-------------------------------------------------------------------------------------------
void Infrared_IE_Init_JX(void)
{
    GPIO_InitTypeDef GPIO_InitStruct;                       // 定义一个GPIO_InitTypeDef类型的变量

    RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOC, ENABLE);   // 允许GPIOC时钟

    GPIO_InitStruct.GPIO_Pin = GPIO_Pin_1;                  // GPIO_Pin_1
    GPIO_InitStruct.GPIO_Mode = GPIO_Mode_Out_PP;           // 通用推挽输出
    GPIO_InitStruct.GPIO_Speed = GPIO_Speed_50MHz;          // 50MHz速度
    GPIO_Init(GPIOC, &GPIO_InitStruct);

    GPIO_ResetBits(GPIOC, GPIO_Pin_1);                      //PC1拉低
}
//-------------------------------------------------------------------------------------------

```

➢ 编写载波调制时的延时函数

```
// 载波调制时的延时函数
//=============================================================

// 调用SysTick定时器的延时函数
// 优点：跟程序无关，延时时钟比较准确
// 缺点：不能完全精准到8.77us、17.53us
//-------------------------------------
#define     delay8_77us()   delay_us(9)
#define     delay17_53us()  delay_us(17)
//-------------------------------------


/*
// 调用无用指令的延时函数
// 优点：能达到比较准确的8.77us、17.53us
// 缺点：跟程序有关，程序修改后的优化，可能导致延时不准
// 建议：在程序架构、函数确定后，再用示波器等工具微调计数值
//-------------------------------------------------
volatile void delay8_77us(void)     // 延时8.77us
{
    u8 i=68;        // 不完全准确
    while(i--);     // 需用示波器检测调整
}

volatile void delay17_53us(void)    // 延时17.53us
{
    u8 i=136;       // 不完全准确
    while(i--);     // 需用示波器检测调整
}
//-------------------------------------------------
*/

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

➢ 编写数据协议“0”、“1”的调制函数

```
// NEC协议数据"0"= 载波发射0.56ms + 载波不发射0.56ms
//-------------------------------------------------
void NEC_IE_Send_zero(void)
{
    u8 i;

    // 载波发射0.56ms ≈ 26.3us * 21
    //-------------------------------
    for(i=0;i<22;i++)
    {
        //26.3us（载波发射周期）
        //-------------------------------------
        PC_out(1)=1;    // IE抬高，发射红外光
        delay8_77us();  // 延时8.77us

        PC_out(1)=0;    // IE拉低，不发射红外光
        delay17_53us(); // 延时17.53us
        //-------------------------------------
    }


    // 载波不发射0.56ms ≈ 26.3us * 21
    //-------------------------------
    for(i=0;i<21;i++)
    {
        //26.3us（载波不发射周期）
        //-------------------------------------
        PC_out(1)=0;    // IE拉低，不发射红外光
        delay8_77us();  // 延时8.77us

        PC_out(1)=0;    // IE拉低，不发射红外光
        delay17_53us(); // 延时17.53us
        //-------------------------------------
    }
}
//-------------------------------------------------
```

```
// NEC协议数据"1" = 载波发射0.56ms + 载波不发射1.68ms
//-------------------------------------------------
void NEC_IE_Send_one(void)
{
    u8 i;

    // 载波发射0.56ms ≈ 26.3us * 21
    //-------------------------------
    for(i=0;i<22;i++)
    {
        //26.3us（载波发射周期）
        //-------------------------------------
        PC_out(1)=1;    // IE抬高，发射红外光
        delay8_77us();  // 延时8.77us

        PC_out(1)=0;  // IE拉低，不发射红外光
        delay17_53us(); // 延时17.53us
        //-------------------------------------
    }

    // 载波不发射1.68ms ≈ 26.3us * 64
    //-------------------------------
    for(i=0;i<64;i++)
    {
        //26.3us（载波不发射周期）
        //-------------------------------------
        PC_out(1)=0;    // IE拉低，不发射红外光
        delay8_77us();  // 延时8.77us

        PC_out(1)=0;    // IE拉低，不发射红外光
        delay17_53us(); // 延时17.53us
        //-------------------------------------
    }
}
//-------------------------------------------------
```

➢ 编写发送一帧数据(8 位)的函数

```
// 将一帧数据调制为NEC协议规定的红外载波发射出去
// 一帧数据格式：低位在前，由低到高发送8位数据
// 这帧数据可以是：②16位用户码低字节(8位) / ③16位用户码高字节(8位) / ④8位数据码 / ⑤8位数据码的反码
//-----------------------------------------------------------------------------------------------
void NEC_IE_One_Data(u8 IE_One_Data)
{
    u8 i;

    for(i=0;i<8;i++)
    {
        if( IE_One_Data & 0x01 )
            NEC_IE_Send_one();
        else
            NEC_IE_Send_zero();

        IE_One_Data>>=1;
    }
}
//-----------------------------------
```

➢ 编写发送 NEC 协议规定的信息格式

```
//发送NEC编码格式的信息
//-----------------------------------------------------------------------------------------------------------
// NEC 编码格式: ①引导码 + ②16位用户码低字节(8位) + ③16位用户码高字节(8位) + ④8位数据码 + ⑤8位数据码的反码 + ⑥结束码‘0’
//-----------------------------------------------------------------------------------------------------------
void NEC_IE_Message(u16 user_code_16bit, u8 data_code_8bit)
{
    u16 i;

    u8 T_user_code_high = user_code_16bit>>8;

    u8 T_user_code_low = user_code_16bit;

    // ①引导码: 载波发射9ms，不发射4.5ms
    //-------------------------------------------------------------
    // 载波发射 9ms ≈ 26.3us * 342
    for(i=0;i<342;i++)
    {
        PC_out(1)=1;     // IE拉高，发射红外光
        delay8_77us();   // 延时8.77us

        PC_out(1)=0;;    // IE拉低，不发射红外光
        delay17_53us(); // 延时17.53us
    }

    // 载波不发射 4.5ms ≈ 26.3us * 171
    for(i=0;i<171;i++)
    {
        PC_out(1)=0;     // IE拉低，不发射红外光
        delay8_77us();   // 延时8.77us

        PC_out(1)=0;     // IE拉低，不发射红外光
        delay17_53us(); // 延时17.53us
    }
    //-------------------------------------------------------------


    //-------------------------------------------------------------
    // ②16位用户码低字节 : 8位数据
    NEC_IE_One_Data(T_user_code_low);

    // ③16位用户码高字节: 8位数据
    NEC_IE_One_Data(T_user_code_high);

    // ④8位数据码 : 8位数据
    NEC_IE_One_Data(data_code_8bit);

    // ⑤8位数据码的反码: 8位数据
    NEC_IE_One_Data(~data_code_8bit);

    //-------------------------------------------------------------


    // ⑥结束码‘0’
    //--------------
    NEC_IE_Send_zero();
    //--------------
}
```

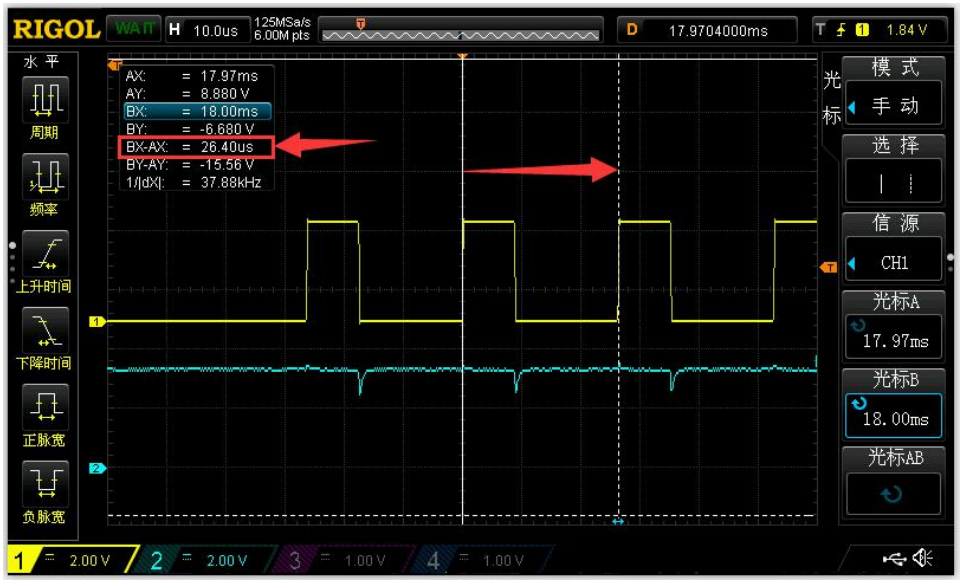
➢ 用示波器测得红外收发波形

注：通道 1(黄色)：控制红外发射器 IR333C 发射红外载波的控制端，即 MCU 的 PC1 端口

通道 2(蓝色)：红外接收器 IRM-3638T 的输出信号(接收到红外载波，解调后输出)

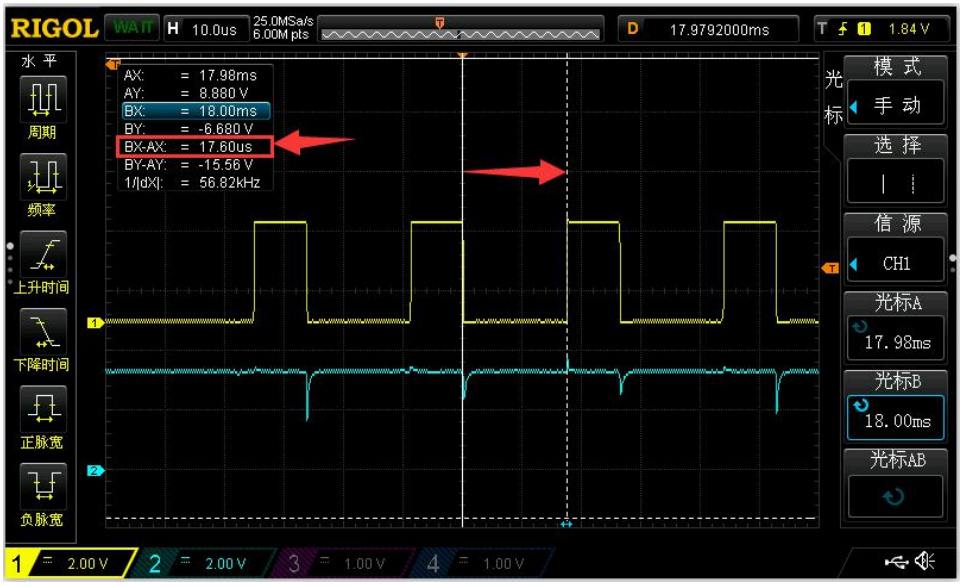
❖ 1、Infrared 红外发射接收波形概览

❖ 2.0、一个载波发射周期：26.3us= 8.77us(IE 为高，发射红外光) + 17.53us(IE 为低，不发射红外光)



■ 2.1、8.77us（IE 为高，发射红外光）

■ 2.2、17.53us（IE 为低，不发射红外光）



❖ 3.0、NEC 协议码“0”：长 1.12ms

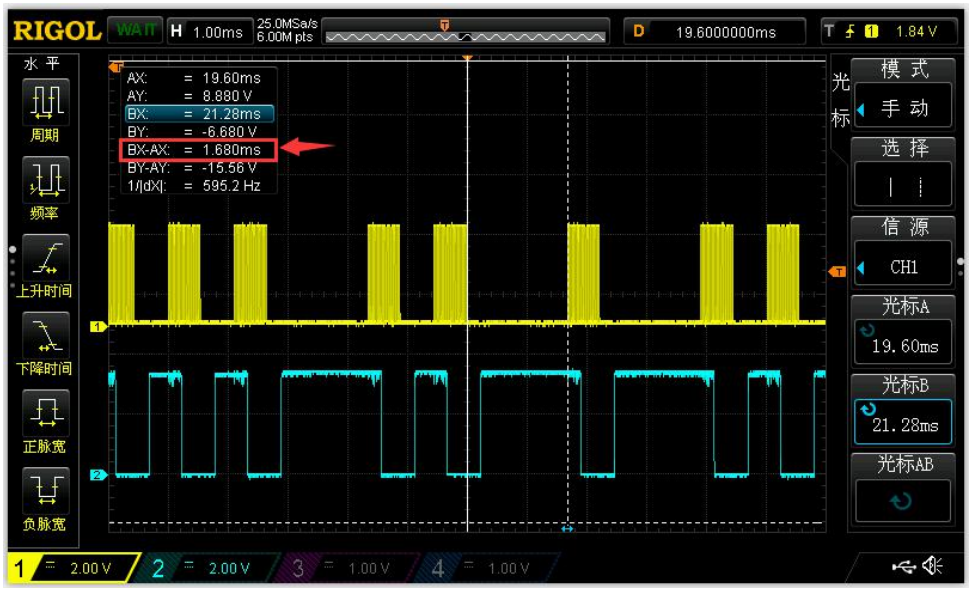
■ 3.1、NEC 协议码“0”的前部分：0.56ms 的载波发射

■ 3.2、NEC 协议码“0”的后部分：0.56ms 的载波不发射

❖ 4.0、NEC 协议码“1”：长 2.24ms

■ 4.1、NEC 协议码“1”的前部分：0.56ms 的载波发射

■ 4.2、NEC 协议码“0”的后部分：1.68ms 的载波不发射



❖ 5.0、NEC 引导码：长 13.5ms

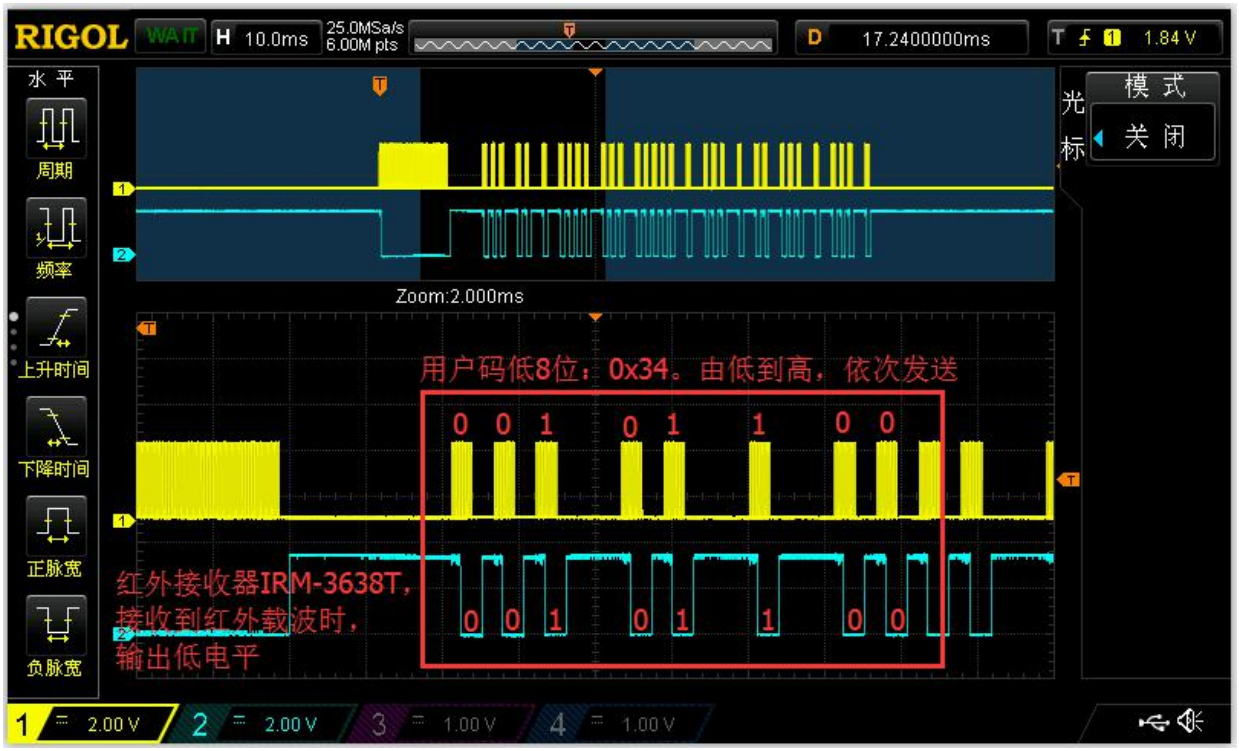
■ 5.1、NEC 引导码的前部分：9ms 的载波发射

■ 5.2、NEC 引导码的后部分：4.5 的载波不发射

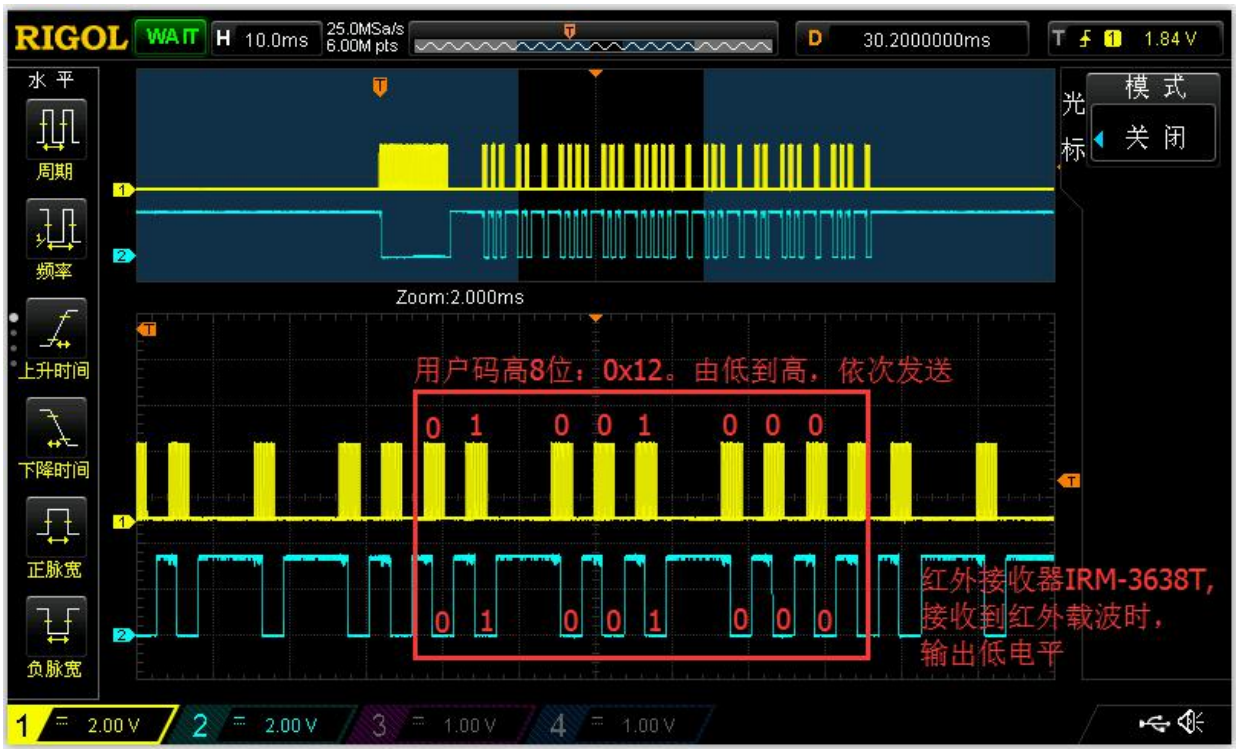
❖ 发送：用户码 0x1234

```
NEC_IE_Message(user_code_16bit,data_code_8bit); // 红外发送：用户码 + 数据码
```

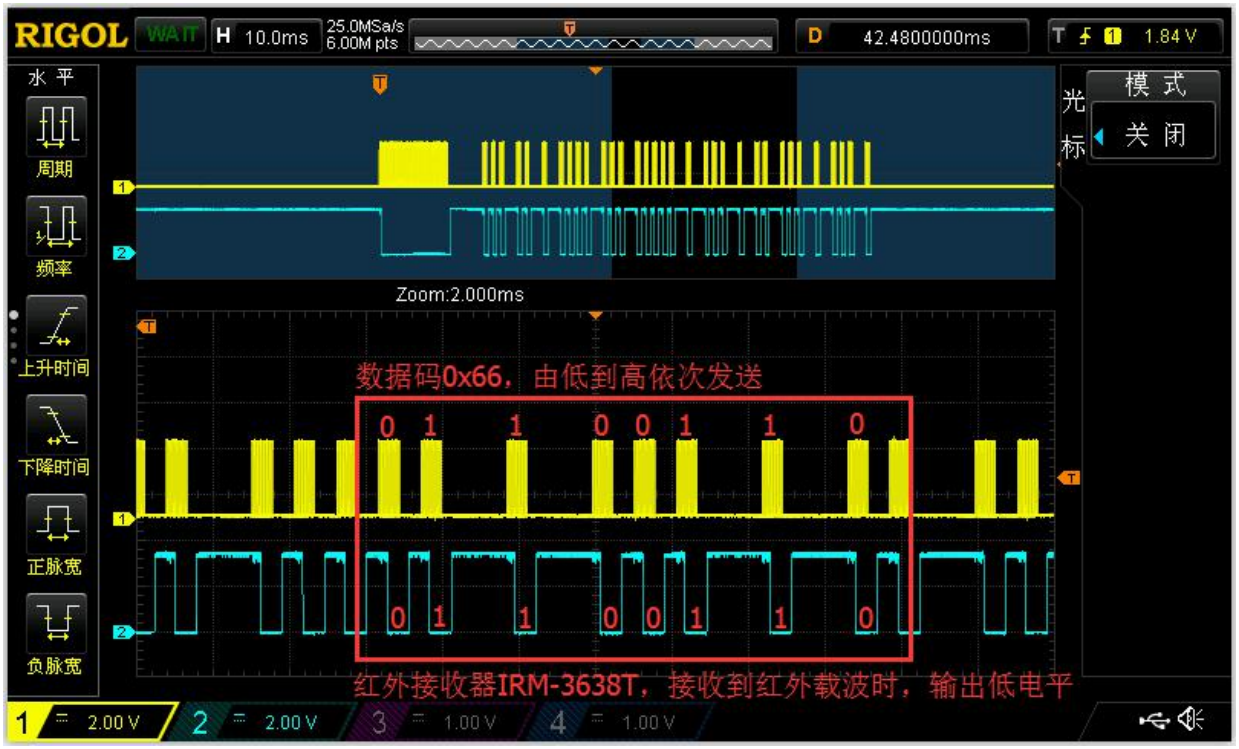
■ 用户码低 8 位_0x34

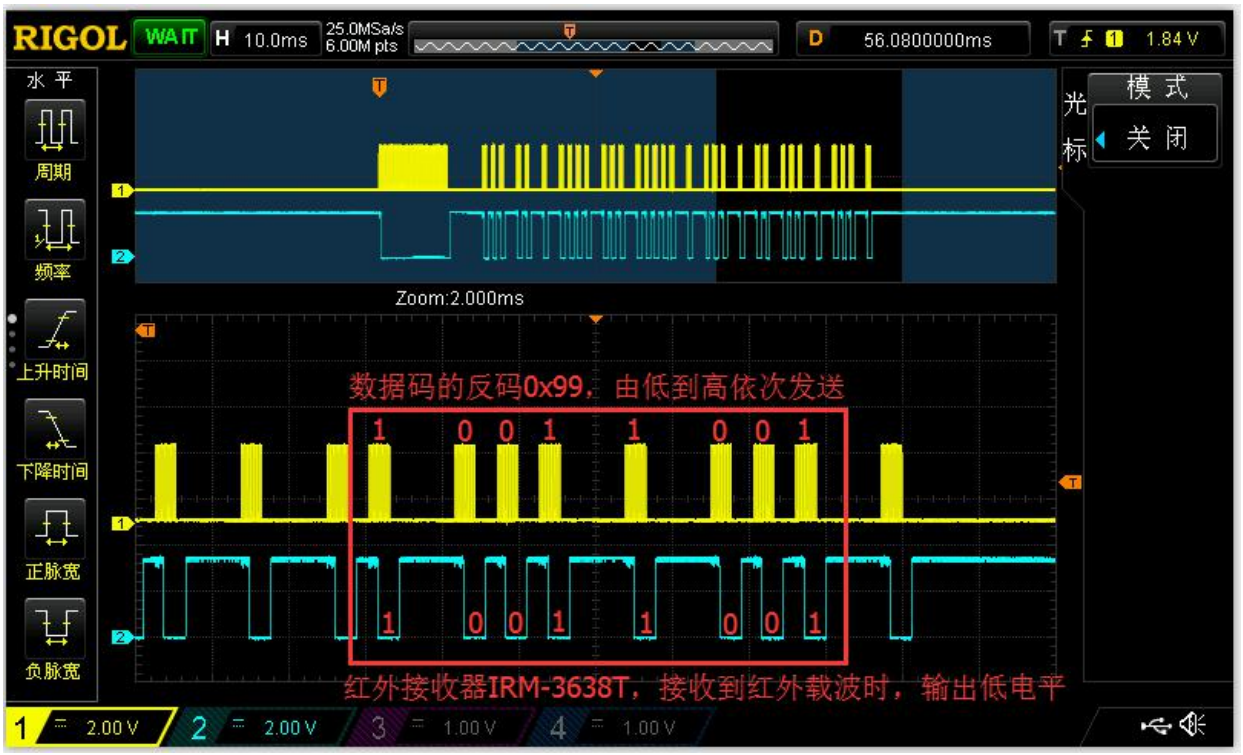


■ 用户码高 8 位_0x12



❖ 发送：数据码 0x66 + 数据码的反码 0x99





❖ 发送一位停止位：协议“0”

4.2.2、红外接收程序编写

由上述红外发射的叙述和波形可知，决定数据为“0”或“1”的是 0.56ms 载波发射后面跟着的“小尾巴”：

■ 小尾巴为：0.56ms 载波不发射，此位数据即为“0”

■ 小尾巴为：1.68ms 载波不发射，此位数据即为“1”

那么只需检测小尾巴的长度/整个数据位的长度，就可以判断出此数据位为“0”或“1”。

注：此解码方式较简单，但不是特别严谨。

更严谨的解码方式为：红外接收器输出的所有高/低电平，都应检测是否在规定的长度范围内。

笔者选择较简单的解码方式：只检测整个数据位的长度，来判断数据位为“0”或“1”

➢ 首先将红外接收管脚 PC0 初始化为上拉输入，并开启 PC0 端口的下降沿中断

```
// 初始化红外发射管脚：Infrared_IR = PC0
//------------------------------------------------------------------------------
void Infrared_IR_Init_JX(void)
{
    GPIO_InitTypeDef GPIO_InitStruct;
    EXTI_InitTypeDef EXTI_InitStruct;
    NVIC_InitTypeDef NVIC_InitStruct;

    /* 配置PC0引脚 */
    //--------------------------------------------------------------------------
    RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOC, ENABLE);        // 使能PC端口时钟
    GPIO_InitStruct.GPIO_Mode = GPIO_Mode_IPU;                   // 上拉输入模式
    GPIO_InitStruct.GPIO_Pin = GPIO_Pin_0;                       // PC0
    GPIO_Init(GPIOC, &GPIO_InitStruct);
    //--------------------------------------------------------------------------

    /* 配置PC0引脚中断模式 */
    //--------------------------------------------------------------------------
    RCC_APB2PeriphClockCmd(RCC_APB2Periph_AFIO,ENABLE);          // 使能映射时钟

    GPIO_EXTILineConfig(GPIO_PortSourceGPIOC, GPIO_PinSource0); // 将中断线0映射到PC0线上

    EXTI_InitStruct.EXTI_Mode = EXTI_Mode_Interrupt;             // 选择为中断
    EXTI_InitStruct.EXTI_Trigger = EXTI_Trigger_Falling;         // 下降沿中断
    EXTI_InitStruct.EXTI_LineCmd = ENABLE;
    EXTI_InitStruct.EXTI_Line = EXTI_Line0;                      // 选择外部中断线0
    EXTI_Init(&EXTI_InitStruct);

     /* 配置NVIC */
    NVIC_InitStruct.NVIC_IRQChannelPreemptionPriority = 1;       // 抢占优先级
    NVIC_InitStruct.NVIC_IRQChannelSubPriority = 2;              // 响应优先级
    NVIC_InitStruct.NVIC_IRQChannelCmd = ENABLE;
    NVIC_InitStruct.NVIC_IRQChannel = EXTI0_IRQn;                // 中断通道：外部中断线0
    NVIC_Init(&NVIC_InitStruct);
}
```

➢ 开启 TIM2 中断，编写 TIM2 中断函数

TIM2 的计数周期为 1us，溢出值为 20000：

■ 记录红外接收管脚 PC0 的相邻的 2 个下降沿信号的时长

■ 限定相邻两下降沿之间的最大时长，不能超过 20ms

```
//-------------------------------------------------------------------------------------------------
// TIMx_IT_Update_time = (TIMx_Reload_Num + 1) * ( TIMx_Frequency_Divide + 1 ) / TIMx_input_CLK
//   TIMx溢出时间          重装载值                    分频系数                TIMx的输入时钟
//-------------------------------------------------------------------------------------------------

// 20ms中断一次
//-------------------------------------------------------------------------------------------------
#define TIM2_Frequency_Divide   71      // TIM2时钟预分频值
#define TIM2_Reload_Num         19999   // 自动重装载寄存器的值

// 分频系数=71 => 每计数一次的周期 = 1s/(72MHz/(71+1)) = 1us

// 向上技术模式下：重装载值=19999 => 计数20000次，用时20ms（装填重装载值时，耗一个时钟周期）
//-------------------------------------------------------------------------------------------------
```

```
// 初始化TIM2:
// 根据"timer.h"文件中的宏定义，设置TIM2的溢出时间
//-------------------------------------------------------------------------------------------------
void TIM2_Time_Init(void)
{
    TIM_TimeBaseInitTypeDef  TIM_TimeBaseStructure;
    NVIC_InitTypeDef NVIC_InitStructure;

    RCC_APB1PeriphClockCmd(RCC_APB1Periph_TIM2, ENABLE);        // 使能TIM2的时钟

    //定时器TIM2初始化
    TIM_TimeBaseStructure.TIM_Period = TIM2_Reload_Num ;        // 设置下一个更新事件后，装入自动重装载寄存器的值
    TIM_TimeBaseStructure.TIM_Prescaler = TIM2_Frequency_Divide;// 设置TIM2时钟预分频值
    TIM_TimeBaseStructure.TIM_ClockDivision = TIM_CKD_DIV1;     // 设置时钟分割:TDTS = Tck_tim
    TIM_TimeBaseStructure.TIM_CounterMode = TIM_CounterMode_Up; // TIM向上计数模式
    TIM_TimeBaseInit(TIM2, &TIM_TimeBaseStructure);             // 根据参数初始化TIM2的计数方式

    TIM_ITConfig(TIM2,TIM_IT_Update,ENABLE );                   // 允许TIM2更新中断


    //TIM2的中断NVIC设置
    NVIC_InitStructure.NVIC_IRQChannel = TIM2_IRQn;             // TIM3中断
    NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority = 0;   // 抢占优先级0级
    NVIC_InitStructure.NVIC_IRQChannelSubPriority = 3;          // 子优先级3级
    NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;             // 使能TIM2_IRQn通道
    NVIC_Init(&NVIC_InitStructure);                             // 初始化NVIC寄存器


    TIM_Cmd(TIM2, ENABLE);                                      // 使能TIM2
}
//-------------------------------------------------------------------------------------------------

```

```
// TIM2中断：20ms更新中断一次
//-------------------------------------------------------------------------------------------------
void TIM2_IRQHandler(void)                                      //TIM2中断
{
    if (TIM_GetITStatus(TIM2, TIM_IT_Update) != RESET)          //判断是否为TIM2的更新中断
    {
        TIM_ClearITPendingBit(TIM2, TIM_IT_Update );            //清除TIM2更新中断标志


        // 当接收到一个下降沿后，下一个下降沿须在20ms内被接收到，否则此次红外接收认为是出错的
        //-------------------------------------------------------------------------
        Current_bit_CNT = 0;     // 将当前红外接收的位数清0



        TIM2_IT_Update_Cnt ++ ;

        if( TIM2_IT_Update_Cnt >= 25 )       // 蓝灯闪烁速率：1s
        {
            TIM2_IT_Update_Cnt = 0 ;

            PD_out(12) = !PD_out(12);        // 蓝灯闪烁
        }
    }
}
//-------------------------------------------------------------------------------------------------
```

➢ 编写 PC0 下降沿中断函数

此中断函数的作用：

■ 将每个下降沿的 TIM2 计数存入 Each_bit_duration[x]中：记录两个下降沿之间的时长

```
125 //红外接收管脚PC0的下降沿中断函数
126 //-------------------------------------------------------------------------------------------------------
127 void EXTI0_IRQHandler(void)
128 {
129     if(EXTI_GetITStatus(EXTI_Line0) != RESET)   // 判断是否为中断线0的中断
130     {
131         EXTI_ClearITPendingBit(EXTI_Line0);     // 清除中断标志位
132 
133 
134         Each_bit_duration[Current_bit_CNT] = TIM2->CNT; // 将此下降沿的TIM2计数存入Each_bit_duration[x]中
135 
136         Current_bit_CNT ++ ;    // 将当前红外接收的位数+1
137 
138 
139         // 1、方便下一个下降沿的计时
140         // 2、等待下一个下降沿20ms
141         //-------------------------------------------------------------------
142         TIM2->CNT=0;            // 红外接收管脚接收到一个下降沿后，将TIM2计数器清0
143     }
144 }
145 //-------------------------------------------------------------------------------------------------------
```

➢ 解码 NEC 编码格式的信息

■ 协议码的时长根据需要，可自行限定判断区间

■ TIM2 的计数周期为 1us，溢出值为 20000

```
269 // 解码NEC编码格式的信息
270 // 注：协议码的时长根据需要，可自行限定判断区间
271 // 注：TIM2的计数周期为1us，溢出值为20000
272 //-----------------------------------------------------------------------------------
273 u8 NEC_IR_decode_message(void)
274 {
275     u8 NEC_decode_cnt;
276 
277     // 协议引导码的标准长度为13.5ms
278     // 判断引导码的长度是否在(12ms,15ms)区间内
279     //-----------------------------------------------------
280     if( Each_bit_duration[1]<12000 || Each_bit_duration[1]>15000 )
281     { return F_decode_error; }  // 不在规定的区间内，返回解码出错
282 
283 
284     // 解码16位用户码
285     //-----------------------------------------------------
286     for(NEC_decode_cnt=2; NEC_decode_cnt<18; NEC_decode_cnt++)
287     {
288         // 协议数据"1"的标准长度为2.24ms
289         // 协议数据位的长度在(2ms,2.5ms)区间内 => 协议数据"1"
290         //-------------------------------------------------------------------------------
291         if( Each_bit_duration[NEC_decode_cnt]>2000 && Each_bit_duration[NEC_decode_cnt]<2500 )
292         {
293             Receive_user_code_16bit >>= 1;
294 
295             Receive_user_code_16bit |= 0x8000;
296         }
297 
298         // 协议数据"0"的标准长度为1.12ms
299         // 协议数据位的长度在(1ms,1.25ms)区间内 => 协议数据"0"
300         //-------------------------------------------------------------------------------------------
301         else if( Each_bit_duration[NEC_decode_cnt]>1000 && Each_bit_duration[NEC_decode_cnt]<1250 )
302         {
303             Receive_user_code_16bit >>= 1;
304 
305             Receive_user_code_16bit &= ~0x8000;
306         }
307 
308         // 协议数据位不在限定范围内，出错
309         //-----------------------------
310         else { return F_decode_error; }
311     }
312 
```

```
311
312     // 解码8位数据码
313     //-------------------------------------------------------
314     for(NEC_decode_cnt=18; NEC_decode_cnt<26; NEC_decode_cnt++)
315     {
316         // 协议数据"1"的标准长度为2.24ms
317         // 协议数据位的长度在(2ms,2.5ms)区间内 => 协议数据"1"
318         //-------------------------------------------------------------------------------------
319         if( Each_bit_duration[NEC_decode_cnt]>2000 && Each_bit_duration[NEC_decode_cnt]<2500 )
320         {
321             Receive_data_code_8bit >>= 1;
322
323             Receive_data_code_8bit |= 0x80;
324         }
325
326         // 协议数据"0"的标准长度为1.12ms
327         // 协议数据位的长度在(1ms,1.25ms)区间内 => 协议数据"0"
328         //-------------------------------------------------------------------------------------
329         else if( Each_bit_duration[NEC_decode_cnt]>1000 && Each_bit_duration[NEC_decode_cnt]<1250 )
330         {
331             Receive_data_code_8bit >>= 1;
332
333             Receive_data_code_8bit &= ~0x80;
334         }
335
336         // 协议数据位不在限定范围内，出错
337         //-----------------------------
338         else { return F_decode_error; }
339     }
340
```

```
340
341     // 解码8位数据码的反码
342     //-------------------------------------------------------
343     for(NEC_decode_cnt=26; NEC_decode_cnt<34; NEC_decode_cnt++)
344     {
345         // 协议数据"1"的标准长度为2.24ms
346         // 协议数据位的长度在(2ms,2.5ms)区间内 => 协议数据"1"
347         //-------------------------------------------------------------------------------------
348         if( Each_bit_duration[NEC_decode_cnt]>2000 && Each_bit_duration[NEC_decode_cnt]<2500 )
349         {
350             Receive_data_code_opposite >>= 1;
351
352             Receive_data_code_opposite |= 0x80;
353         }
354
355         // 协议数据"0"的标准长度为1.12ms
356         // 协议数据位的长度在(1ms,1.25ms)区间内 => 协议数据"0"
357         //-------------------------------------------------------------------------------------
358         else if( Each_bit_duration[NEC_decode_cnt]>1000 && Each_bit_duration[NEC_decode_cnt]<1250 )
359         {
360             Receive_data_code_opposite >>= 1;
361
362             Receive_data_code_opposite &= ~0x80;
363         }
364
365         // 协议数据位不在限定范围内，出错
366         //-----------------------------
367         else { return F_decode_error; }
368     }
369
370     return F_decode_success;    // 解码成功
371 }
```

- 编写按键、串口相关函数（此处略）

- 编写 main.c 文件、main 函数

- 包含头文件、创建全局变量等

```
1  #include "stm32f10x.h"
2  #include "bit_band.h"
3  #include "delay.h"
4  #include "led.h"
5  #include "USART1_Func.h"
6  #include "key.h"
7  #include "timer.h"
8  #include "infrared.h"
9
10
11 u16 TIM2_IT_Update_Cnt = 0 ;        // TIM2的溢出次数，用于LED1的闪烁计时
12
```

- 初始化各硬件配置

```
int main(void)
{
    u16 user_code_16bit = 0x1234;   // 初始化16位用户码

    u8 data_code_8bit = 0x66 ;      // 初始化8位数据码

    u8 F_key_down = 0 ;             // 按键按下标志位


    // 注：程序中使用中断时，NVIC分组设置应尽量位于程序起始处，并且在设置后尽量不要再更改NVIC分组
    //*********************************************************************************************

    NVIC_PriorityGroupConfig(NVIC_PriorityGroup_2);      //NVIC分组2：2位抢占优先级，2位响应优先级

    delay_Init();                // 延时初始化（注：调用延时函数之前，必须先调用delay_Init()将SysTick初始化）

    LED_Init_JX() ;              // 初始化LED硬件接口

    KEY_Init_JX() ;              // 初始化按键的硬件配置

    USART1_Func_Init(115200) ;   // 初始化USART1

    TIM2_Time_Init();            // 初始化定时器2:1ms中断一次

    Infrared_IE_Init_JX();       // 初始化红外发射管脚：Infrared_IE = PC1

    Infrared_IR_Init_JX();       // 初始化红外发射管脚：Infrared_IR = PC0


    NEC_IE_code_message(user_code_16bit,data_code_8bit);     // 上电发送：用户码0x1234 + 数据码0x66

```

- 循环检测是否接收到 NEC 格式的红外信息

  如果接收到，则保存用户码+数据码，并通过串口将接收到的用户码+数据码发送到串口助手

```
    while(1)
    {
        // 判断是否接收到有效的NEC红外信息
        //-------------------------------------------------------------------------
        if(Current_bit_CNT>=34)
        {
            // 解码NEC格式红外信息
            //-------------------------------
            if( ! NEC_IR_decode_message() )
            {
                // 判断数据码和数据码反码是否相反
                //-------------------------------
                if( Receive_data_code_8bit == (u8)(~Receive_data_code_opposite) )
                {
                    // 避免重复保存
                    //-------------
                    if( user_code_16bit!=Receive_user_code_16bit || data_code_8bit!=Receive_data_code_8bit )
                    {
                        user_code_16bit = Receive_user_code_16bit;  // 保存用户码

                        data_code_8bit  = Receive_data_code_8bit ;  // 保存数据码

                        //将接收到的：用户码高字节 / 用户码低字节 / 数据码，通过串口发送
                        //-----------------------------------------------------------
                        USART_SendData( USART1, user_code_16bit>>8 );
                        while ( USART_GetFlagStatus(USART1, USART_FLAG_TC) == RESET );
                        USART_SendData( USART1, user_code_16bit );
                        while ( USART_GetFlagStatus(USART1, USART_FLAG_TC) == RESET );
                        USART_SendData( USART1, data_code_8bit );
                        while ( USART_GetFlagStatus(USART1, USART_FLAG_TC) == RESET );
                        //-----------------------------------------------------------

                        PB_out(5) = 0;  // LED0亮一下，表示将红外码接收
                        delay_ms(200);
                        PB_out(5) = 1;
                    }
                }
            }
        }
        //-------------------------------------------------------------------------
```

- 循环检测 Key_2 是否按下

  如果 Key_2 按下，则将保存的 NEC 红外信息通过红外模块发射出去

  并通过串口将发送出去的用户码+数据码发送到串口助手

```
86
87          // 按键发码
88          //-------------------------------------------------------------------------------------------------
89          KEY_Scan_delay_JX();    // 扫描按键是否按下
90
91          if( S_KEY_down == 0x04 )
92          {
93              if( F_key_down == 0 )
94              {
95                  F_key_down = 1; // 长按只有效一次
96
97                  NEC_IE_code_message(user_code_16bit,data_code_8bit);    // 红外发送：用户码 + 数据码
98
99                  //将接收到的：用户码高字节 / 用户码低字节 / 数据码，通过串口发送
100                 //-----------------------------------------------------------
101                 USART_SendData( USART1, user_code_16bit>>8 );
102                 while ( USART_GetFlagStatus(USART1, USART_FLAG_TC) == RESET );
103                 USART_SendData( USART1, user_code_16bit );
104                 while ( USART_GetFlagStatus(USART1, USART_FLAG_TC) == RESET );
105                 USART_SendData( USART1, data_code_8bit );
106                 while ( USART_GetFlagStatus(USART1, USART_FLAG_TC) == RESET );
107                 //-----------------------------------------------------------
108             }
109         }
110
111         else F_key_down = 0;
112         //-------------------------------------------------------------------------------------------------
113
114     }   /* while(1) */
115
116 }   /* int main(void) */
117
```

➢ 下载测试

注：外部遥控器发送的是 NEC 格式码【0xFF 0x00 0x45】