第12章 STC单片机串行异步收发器原理及实现

何宾

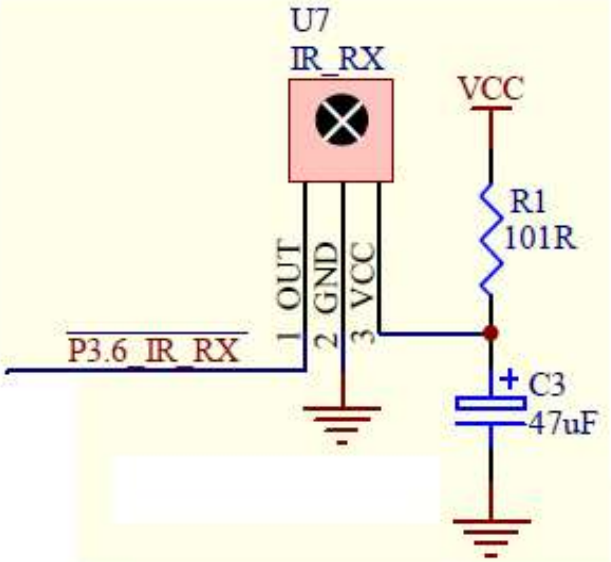
2018.03

红外通信实现
--红外收发器的电路原理

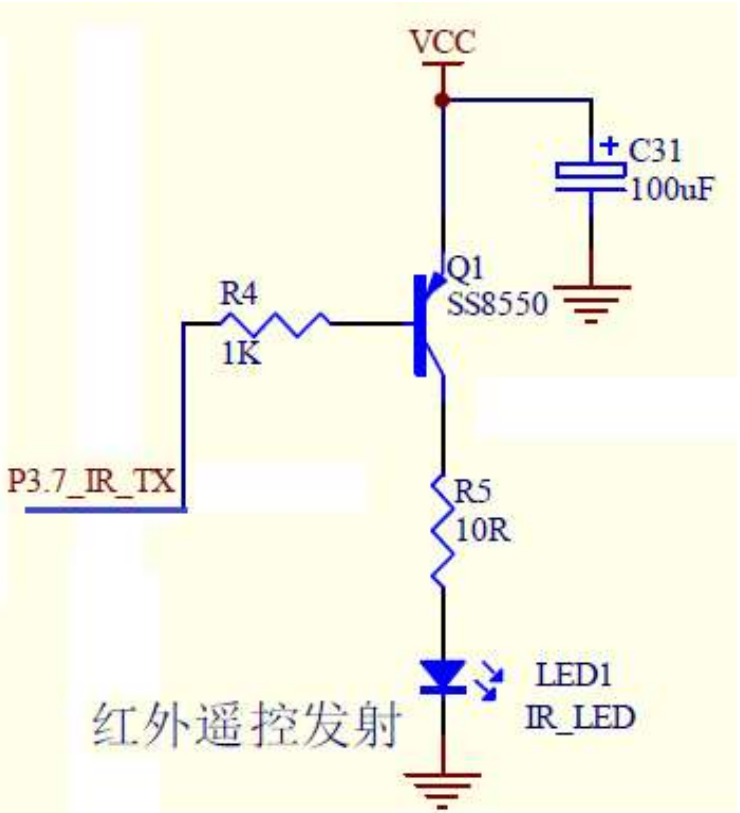
在STC提供的学习板上集成了一个红外发生器和红外接器

- 红外发射器和我们所见过的发光二极管外形很像，但是红外发射器发出的光是不可见的红外光。

红外通信实现
--红外收发器的电路原理



红外遥控接收



红外遥控发射

红外通信实现
--红外收发器的电路原理

在STC学习板中，在IAP15W4K58S4芯片的P3.7引脚处，连接了红外遥控发射器。

- 其本质就是由一个PNP的晶体管构成的放大电路，用于发射红外线的二极管通过R5与PNP晶体管的发射极连接。
- 此外，用于接收红外线通信信息的红外接收器连接到单片机的P3.6引脚处。该接收器将红外光携带的信息，转换成电信号，通过P3.6引脚传给单片机进行处理。

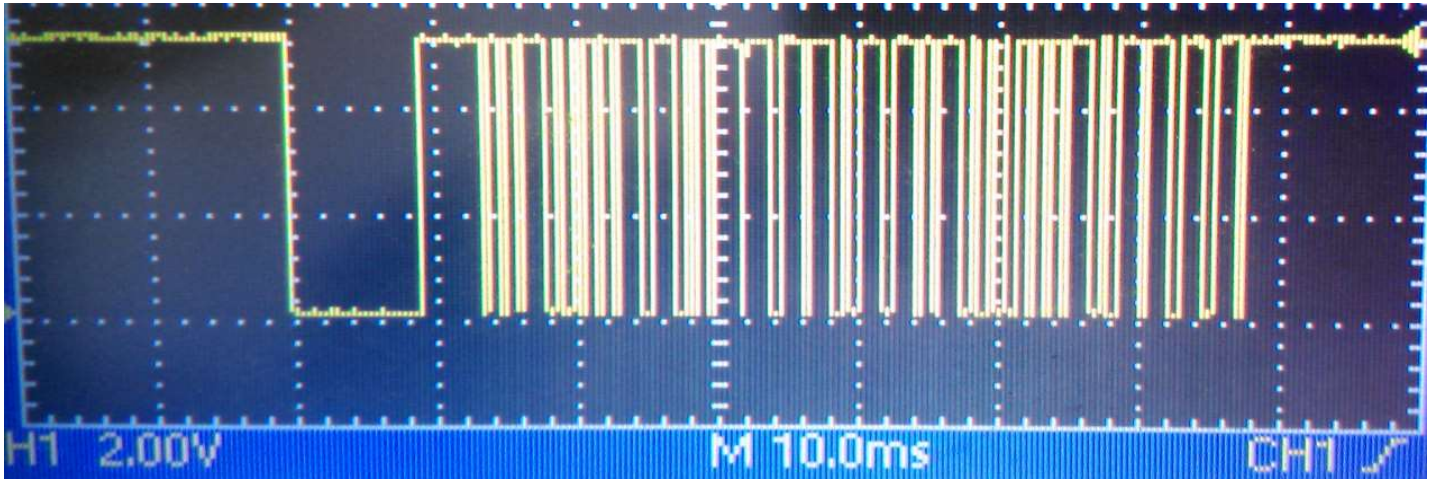
红外通信实现
--红外通信波形捕获

通过实验捕获红外通信所传输的信息的步骤包括：

- 给STC学习板上电。
- 打开示波器。将示波器的一个探头插到STC学习板J9插座标记为P36的插孔中。
- 找一个用于控制家里电视用的遥控器。

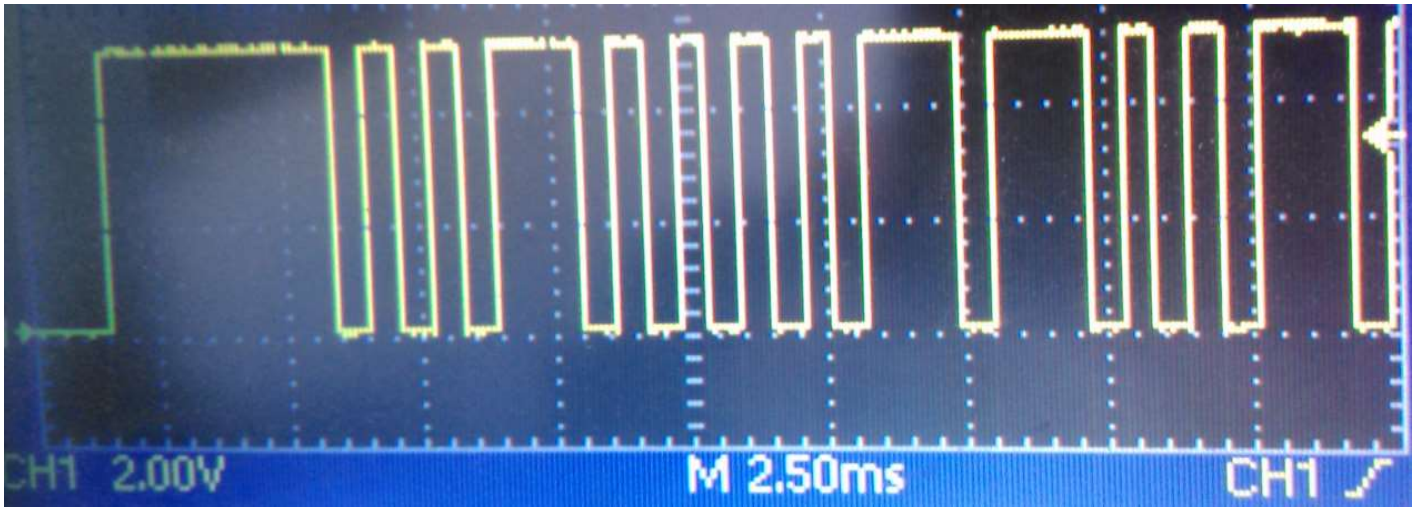
红外通信实现
--红外通信波形捕获

- 将遥控器对准红外接收器，并按下红外遥控器的按键。同时，让示波器捕获波形。



红外通信实现
--红外通信波形捕获

- 将波形前部进行放大，如图所示。



红外通信实现
--红外通信协议

本节将介绍红外通信协议，包括：

- 在红外发射器一侧，为了使红外线在无线传输的过程中避免受到其他红外信号的干扰，通常是将逻辑0和逻辑1调制在某一特定频率的载波上，然后经过红外发光二极管发射出去。
- 在红外接收器一侧，接收到这个被调制后的信号。在本设计中，将通过单片机对接收到的红外信号进行解调，即:去掉载波信号，恢复出原始的二进制脉冲码。

红外通信实现
--红外通信波形捕获

红外通信调制方法

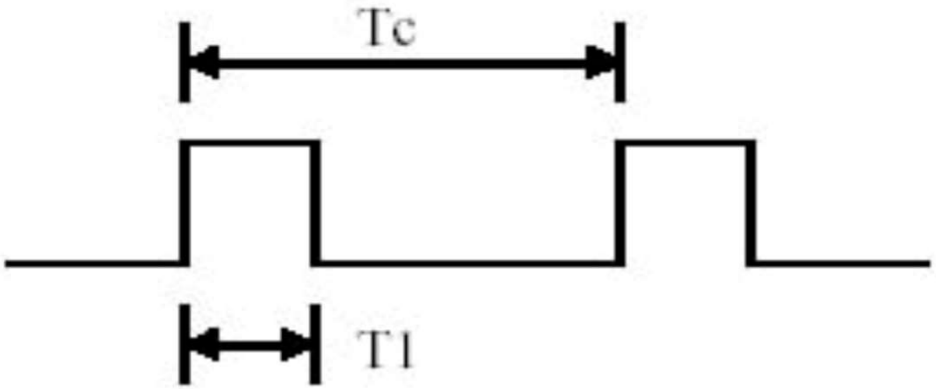
- 脉冲宽度调制（Pluse Width Modulation，PWM）
- 脉冲位置调制（Pulse Position Modulation, PPM）

红外遥控中使用的基带通信协议的类型很多，大概有几十种，常用的就有ITT协议、NEC协议、Sharp协议、Philips协议等。

红外通信协议
--红外发射数据

载波波形

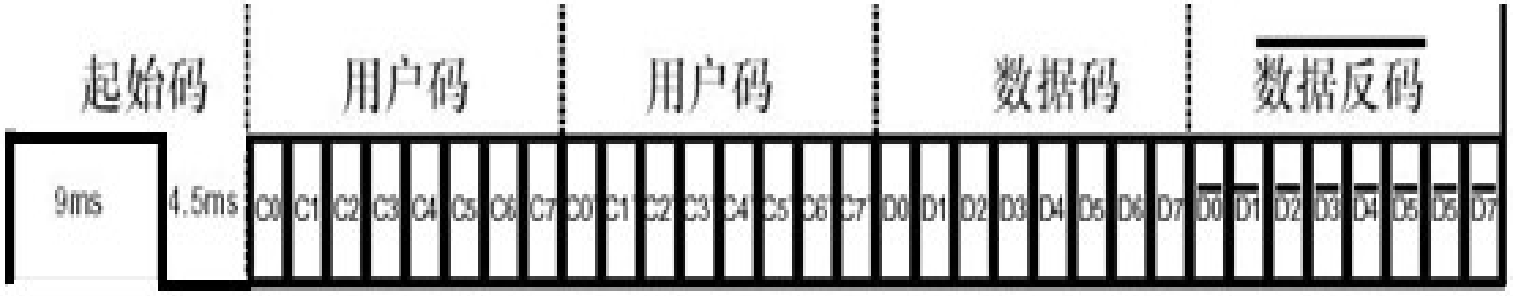
- 可以使用455KHz晶体，经内部分频电路的12分频，将信号调制在37.91kHz，占空比为1/3。



红外通信协议
--红外发射数据

数据格式

- 数据格式包括了起始码、用户码、数据码和数据码反码。
  - 数据反码是对数据码取反后的编码，编码时可用于对数据的纠错。
- 在该数据格式中，编码（包括16位用户码、8位数据码和8位数据反码）长度总共32位。



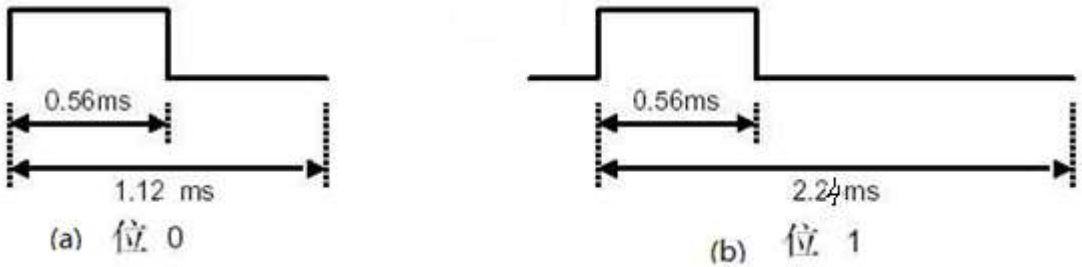
红外通信协议
--红外发射数据

- 红外发射的波形中最前面的是起始码。
  - 起始码的前半部分为高电平，时长大约为9ms；
  - 后半部分为空闲低电平，时长大约为4.5ms。

红外通信协议
--红外发射数据

位定义

- 用户码或数据码中的每一个二进制位或者是1，或者是0。通过脉冲的时间间隔来区分它们。因此，这种编码方式称为PPM调制方式。
  - 对于逻辑0来说，前面的高电平周期为0.56ms，后面的低电平周期大约为0.56ms的空闲时刻。
  - 对于逻辑1来说，前面的高电平周期为0.56ms，后面的低电平周期大约为1.68ms的空闲时刻。



红外通信协议
--红外发射数据

根据对位的定义，得到：

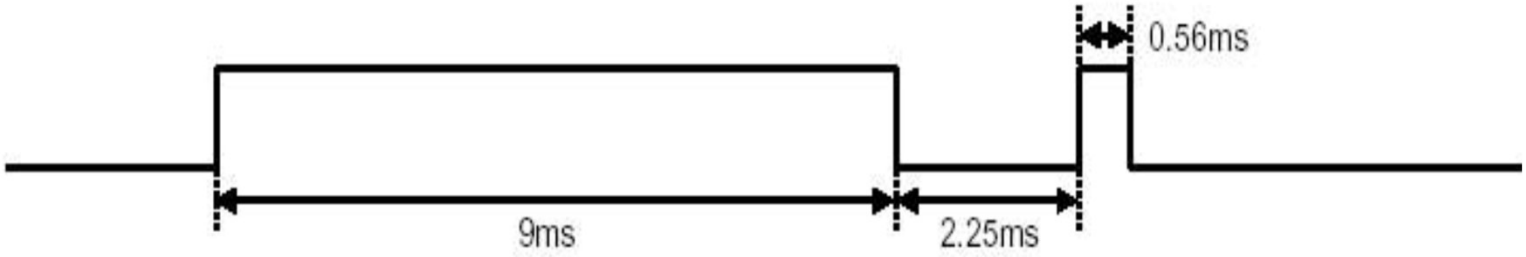
- 16位地址码的最短持续时间为：1.12×16=18ms ； 最长持续时间为：2.24ms×16=36ms。
- 8位数据码和8位数据反码的总时间恒定为：

（1.12ms+2.24ms）×8=27ms。

因此，所有32位码的持续时间在45~63ms。

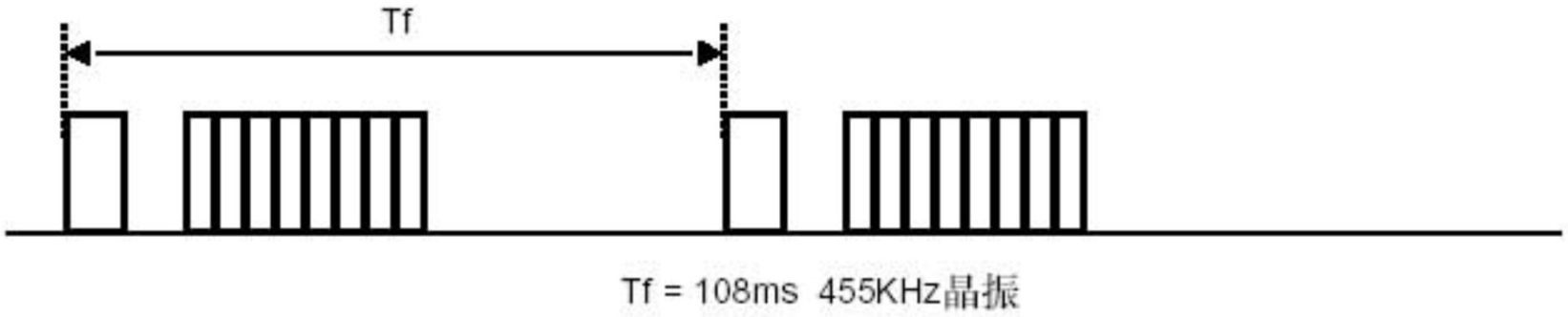
红外通信协议
--按键输出波形

重复码



红外通信协议
--按键输出波形

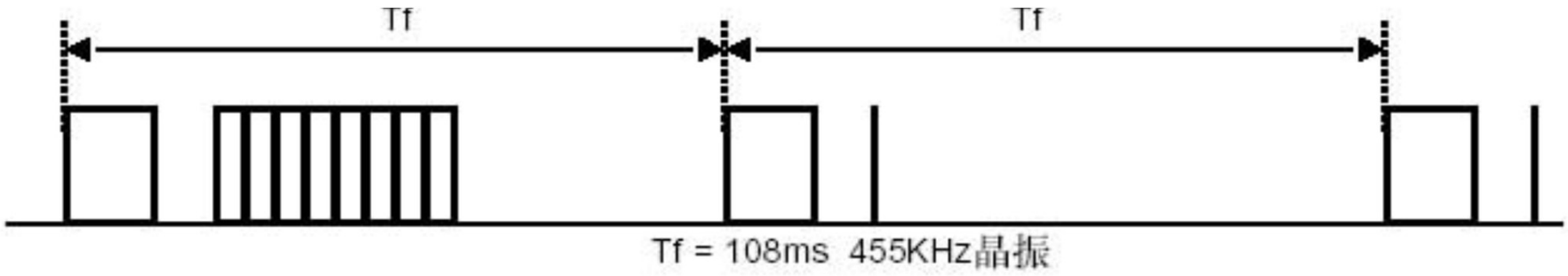
单一按键波形



Tf = 108ms 455KHz晶振

红外通信协议
--按键输出波形

连续按键波形



红外通信协议
--红外接收数据

- 红外接收头将38K载波信号过虑，接收到的波形刚好与发射波形相反
  - 前导码以低电平开始，持续9ms；然后维持4.5ms的高电平。

红外通信协议
--红外接收数据

解码的关键是如何识别0和1

- 从位的定义可以知道，在接收时，0和1均以 0.56ms的低电平开始，不同的是高电平的宽度不同，
  - 对于0来说，持续0.56ms；
  - 对于1来说，持续1.68ms。
  - 所以，必须根据接收信号的高电平时间长度来区分0和1。
- 如果从0.56ms低电平过后，只持续0.56ms的高电平，则为0；如果持续1.68ms的高电平，则为1。

注：根据码的格式，应该等待起始码结束后才能读码。

红外通信实现
--红外检测原理

检测红外传输信息的目的是为了获得四个字节的数据，包括：

- 二个字节的用户码；
- 一个字节的数据码；
- 一个字节的数据反码。

红外通信实现
--红外检测原理

从硬件设计可以知道，红外接收端接到了P3.6引脚，该引脚也是INT2中断触发引脚的位置。

- 在INT_CLKO（AUXR2）寄存器中的EX2就是外部中断允许位。
  - 当该位为1时，允许中断；
  - 否则，禁止中断。
  - 中断2必须采用下降沿触发的方式。

红外通信实现
--红外检测原理

该该设计中，码型特征很明显。

- 对于单一按键来说，只要区分引导码，逻辑0和逻辑1。它们的持续时间有很大的不同。
  - 可以考虑使用定时器通过判断时间的边界来区分它们。
  - 在该设计中，使用定时器/计数器0的模式1（自动16位重加载模式）。
- 红外传输信息的检测在中断2服务程序中实现。
  - 在程序中，一个关键的地方就是设置判决条件。在该设计中，使用的定时0作为判决计数条件。

红外通信实现
--红外检测原理

- 当P3.6为0时，表示低电平，启动定时器0一直计数，以此获得低电平的持续时间。
  - 计数器0的时钟是Sysclk/12，系统时钟频率在烧写程序到STC单片机的时候设置为6.000MHz，这是考虑到了计数器的范围是16位，计数范围是0~65535。在每个时钟沿时，计数器0加1。计算机公式为：

    时间长度=(12× Sysclk)/计数值[TH0*256+TL0]

注：在设计中，考虑时钟的误差，将时间长度设置在一个合理的范围内。

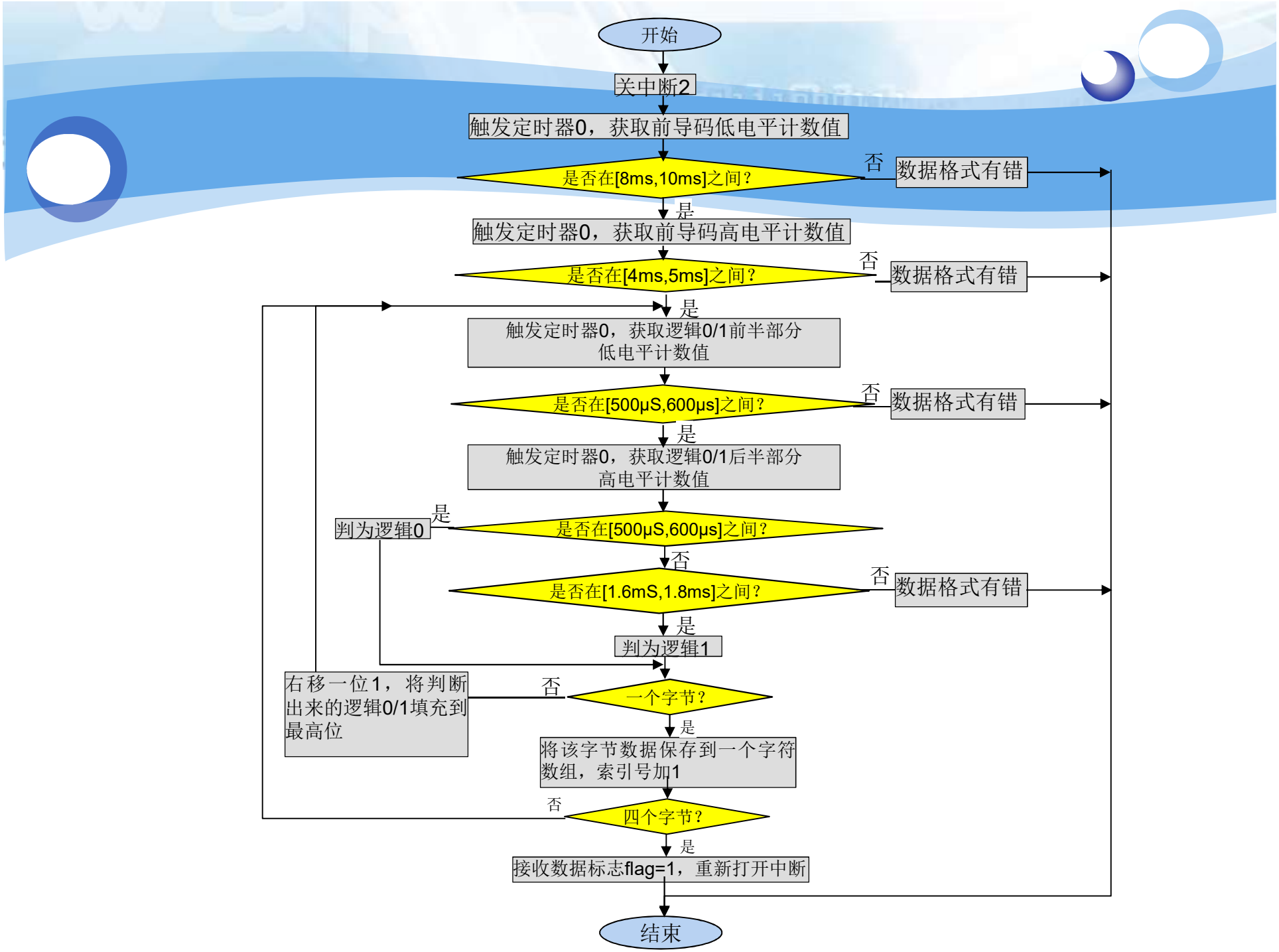
红外通信实现
--红外检测原理

- 当P3.6为1时，表示高电平，启动定时器0一直计数，以此获得高电平的持续时间。
  - 计数器0的时钟是Sysclk/12，系统时钟频率在烧写程序到STC单片机的时候设置为6.000MHz，这是考虑到了计数器的范围是16位，计数范围是0~65535。在每个时钟沿时，计数器0加1。计算机公式为：

    时间长度=(12× Sysclk)/计数值[TH0*256+TL0]

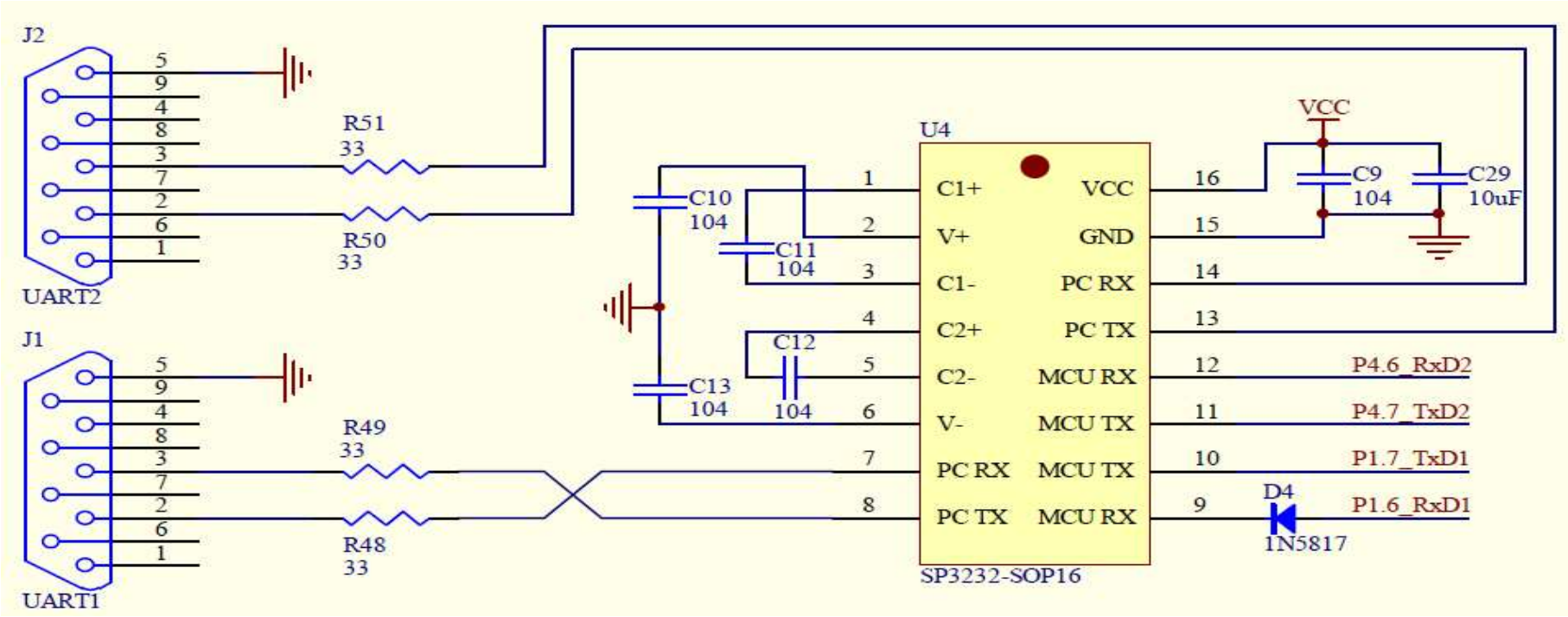
注：在设计中，考虑时钟的误差，将时间长度设置在一个合理的范围内。

综合上述，最终的门限是通过确定时间长度范围后，通过上面公式得到计数值的范围来作为实际的判断条件。


开始
关中断2
触发定时器0，获取前导码低电平计数值
是否在[8ms,10ms]之间？
否
数据格式有错
是
触发定时器0，获取前导码高电平计数值
是否在[4ms,5ms]之间？
否
数据格式有错
是
触发定时器0，获取逻辑0/1前半部分
低电平计数值
是否在[500μS,600μs]之间？
否
数据格式有错
是
触发定时器0，获取逻辑0/1后半部分
高电平计数值
是否在[500μS,600μs]之间？
是
判为逻辑0
否
是否在[1.6mS,1.8ms]之间？
否
数据格式有错
是
判为逻辑1
一个字节？
否
右移一位1，将判断出来的逻辑0/1填充到最高位
是
将该字节数据保存到一个字符数组，索引号加1
四个字节？
否
是
接收数据标志flag=1，重新打开中断
结束

红外通信实现
--串口通信原理

在该设计中使用了串口2作为STC单片机和PC机/笔记本电脑通信的接口



红外通信实现
--串口通信原理

在该设计中，使用了标识为J2的串口

- 串口2接收和发送信号分别连接到STC单片机IAP15W4K32S4单片机的P4.6/RxD2和P4.7/TxD2引脚上。
- 经过SP3232芯片转换成RS-232电平标准，连接到9针的UART母头连接器上。

红外通信实现
--串口通信原理实现

【例】 STC学习板上红外遥控数据通过串口2显示在主机上C语言描述的例子。

```
#include "reg51.h"
#include "intrins.h"
#define FOSC  6000000L          //声明单片机主时钟频率6MHz，为了后面计算
#define BAUD 115200             //声明串口通信的波特率115200，为了后面计算
#define S2RI 0x01               //定义S2RI的值
#define S2TI 0x02               //定义S2TI的值
sfr AUXR  =0x8E;                //声明AUXR寄存器的地址0x8E
sfr AUXR1 =0xA2;                //声明AUXR1寄存器的地址0xA2
```

红外通信实现
--串口通信原理实现

```
sfr AUXR2 =0x8F;          //声明AUXR2寄存器的地址0x8F
sfr TH2   =0xD6;          //声明TH2寄存器的地址0xD6
sfr TL2   =0xD7;          //声明TL2寄存器的地址0xD7
sfr S2CON =0x9A;          //声明S2CON寄存器的地址0x9A
sfr S2BUF =0x9B;          //声明S2BUF寄存器的地址0x9B
sfr P3M1  =0xB1;          //声明P3M1寄存器的地址0xB1
sfr P3M0  =0xB2;          //声明P3M0寄存器的地址0xB2
sfr P_SW2 =0xBA;          //声明P_SW2寄存器的地址0xBA
sfr IE2   =0xAF;          //声明IE2寄存器的地址0xAF
sbit P36  =P3^6;          //声明P3.6引脚为P36
bit busy=0;               //声明busy变量为bit
```

红外通信实现
--串口通信原理实现

```
unsigned char irdata[4]={0,0,0,0};          //声明数组irdata
                                            //保存红外解码的4字节数据
bit flag=0;                                 //声明flag变量为bit
void SendData(unsigned char dat)            //声明串口2发送数据函数SendData
{           while(busy);                    //如果busy为1，表示忙，则一直等待
            S2BUF=dat;                     //往串口2数据缓冲寄存器S2BUF写数据
            busy=1; }                       //置busy为1
void SendString(char *s)                    //声明串口2发送字符串函数SendString
{           while(*s!= '\0' )              //判断字符串是否结束
            SendData(*s++);                //如果没有结束，则调用SendData函数
}
```

红外通信实现
--串口通信原理实现

```
unsigned int high_level_time()                    //声明检测红外发送数据的
 {                                                 //高电平持续时间函数
        TL0=0;                                     //置定时器0初值低8位寄存器TL0为0
        TH0=0;                                     //置定时器0初值高8位寄存器TH0为0
        TR0=1;                                     //启动定时器0开始计数
        while(P36==1)                        //如果读取P3.6的输入为1，一直继续，否则退出
        {
          if(TH0>=0xEE)                       //如果计数时间太长，系统有问题，退出循环
            break;
          }
        TR0=0;                               //如果读取P3.6的输入为0，则停止定时器0计数
        return(TH0*256+TL0);                       //返回计数计数器0的计数值
}
```

红外通信实现
--串口通信原理实现

```
unsigned int low_level_time()                    //声明检测红外发送数据的
 {                                               //低电平持续时间函数
        TL0=0;                                   //置定时器0初值低8位寄存器TL0为0
        TH0=0;                                   //置定时器0初值高8位寄存器TH0为0
        TR0=1;                                   //启动定时器0开始计数
        while(P36==0)                  //如果读取P3.6的输入为0，一直继续，否则退出
        {
          if(TH0>=0xEE)                  //如果计数时间太长，系统有问题，退出循环
           break;
        }
        TR0=0;                           //如果读取P3.6的输入为1，则停止定时器0计数
        return(TH0*256+TL0);             //返回计数计数器0的计数值
}
```

红外通信实现
--串口通信原理实现

```
void int2() interrupt 10                    //声明外部中断2的服务程序
{
    unsigned char i,j;                      //定义无符号字符变量i，j
    unsigned int count=0;                   //定义无符号整型变量count
    unsigned char dat=0;                    //定义无符号字符变量dat
    AUXR2&=0x00;                            //关闭外部中断2，即：禁止中断
   count=low_level_time();                  //读取低电平的计数值
   if(count<4000 || count>5000)             //如果不在给定范围内，退出中断服务程序
    {
        return;
    }
```

红外通信实现
--串口通信原理实现

```
count=high_level_time();                    //读取高电平的计数值
    if(count<2000 || count>2500)            //如果不在给定范围内，退出中断服务程序
    return;
    //下面开始处理32位数据
    for(i=0;i<4;i++)                        //四个字节的循环处理
     { P36=1;                               //读取P3.6引脚前，需要置P3.6为高
        dat=0;                              //dat赋值为0
          for(j=0;j<8;j++)                  //8个比特的循环处理
```

红外通信实现
--串口通信原理实现

```
if(count<250 || count>300)          //如果不在给定范围内，则退出中断服务程序
    return;
count=high_level_time();            //读取高电平的计数值，即：逻辑位高后半部分
if(count>250 && count<300)          //如果在逻辑“0”的范围内，填充0
dat>>=1;
else if(count>800 && count<1000) //如果在逻辑“1”的范围内
 {
    dat>>=1;                        //右移一位
    dat|=0x80; }                    //高位用“1”填充
    else return;                    //否则，不在给定逻辑位高后半部分范围，退出
 }
```

红外通信实现
--串口通信原理实现

```
        {
            count=low_level_time();    //读取低电平的计数值
            irdata[i]=dat;
        }                              //将8位数据保存在irdata数组当前索引号
    flag=1;                            //当四字节填满后，将flag置1，表示有数据
    AUXR2|=0x10;                       //开中断
}
```

红外通信实现
--串口通信原理实现

```
void uart2() interrupt 8                    //声明串口2中断服务程序
{
    if(S2CON & S2RI)                        //如果S2CON的S2RI为1，表示接收到数据
        S2CON&=~S2RI;                       //将S2CON寄存器的S2RI标志清零
    if(S2CON & S2TI)                        //如果S2CON的S2TI为1，表示发送完数据
    {
        S2CON&=~S2TI;                       //将S2CON寄存器的S2TI标志清零
        busy=0;                             //将busy标志清零
     }
}
```

红外通信实现
--串口通信原理实现

```
void main()                                  //定义主程序
{    unsigned char k;                        //定义无符号的字符变量k
     P36=1;                                  //设置P3.6引脚为高
     P3M1=0x00;                              //将P3端口设置为准双向弱上拉
     P3M0=0x00;                              //通过P3M1和P3M0寄存器，设置P3端口模式
     TMOD=0x00;                              //设置定时器0的工作模式
     S2CON=0x50;                             //设置串口2方式0，使能串口2接收
     AUXR=0x14;                              //定时器2/12，启动定时器2，定时器模式
     P_SW2|=0x01;                            //串口2切换到P4.6/RxD2_2和P4.7/TxD2_2
     TL2=(65536-((FOSC/4)/BAUD));  //计数初值低8位给定时器2TL2寄存器
   TH2=(65536-((FOSC/4)/BAUD))>>8;//计数初值高8位给定时器2TH2寄存器
```

红外通信实现
--串口通信原理实现

```
IE2|=0x01;                                  //允许串口2中断
AUXR2|=0x10;                                //允许外部中断2
EA=1;                                       //CPU允许响应中断请求
SendString("\r\n--begin-----\r\n");         //打印信息
while(1)                                    //无限循环
{
    if(flag==1)                             //如果flag为1，表示接收到四字节解码信息
    {
        flag=0;                             //将flag位置0
        SendString("\r\n--Received IR data is-----\r\n");  //打印信息
```

红外通信实现
--串口通信原理实现

```
            for(k=0;k<4;k++)              //循环四次
            SendData(irdata[k]);          //打印4字节，一共32位的信息
            SendString(“\r\n”);           //打印回车换行符号
            AUXR2|=0x10;                  //使能外部中断2
        }
    }
}
```

红外通信实现
--串口通信原理实现

下面说明该代码的设计原理和验证方法，步骤包括：

- 准备一根UART-USB的串口电缆，将STC学习板上标记为J2的串口插座和电脑主机进行连接。
- 打开STC-ISP软件，在该界面内，选择硬件选项。将“输入用户程序运行时的IRC频率”设置为6.000MHz。
- 在STC-ISP软件右侧串口中，选择串口助手标签。在该标签串口界面下，选择COM7，波特率为115200，一个停止位，无奇偶校验。
- 在接收缓冲区标题栏下，选中HEX模式前面的复选框。

红外通信实现
--串口通信原理实现

- 单击打开串口按钮。
- 单击下载/编程按钮，按前面的方法下载设计到STC单片机。
- 将红外遥控器对准STC红外接收器，按一下按键，出现按键信息。

```
0D 0A 2D 2D 62 65 67 69 6E 2D 2D 2D 2D 2D 0D 0A 0D 0A
2D 2D 52 65 63 65 69 76 65 64 20 49 52 20 64 61 74 61
20 69 73 2D 2D 2D 2D 2D 0D 0A 84 79 14 EB 0D 0A
```

- 第一行是没有按遥控器时，给出的提示信息；
- 根据设计代码，可以看到在最后的回车换行符的前面4个字节的数字“84 79 14 EB”就是解码的红外遥控器的数据。该数据14和EB是取反关系，也就是协议规定的用户和用户反码。