红外控制电子表

信工 1602 班 孟繁阳 2016014507

【作业要求】

使用 STC 单片机上的红外接收器和配套的红外遥控器，实现对 STC 单片机实验箱上的资源进行控制和交互。

(1)STC 单片机能正确接**收到红外遥控器的编码信息，并显示**(不限制显示介质，串口或 1602)

(2) 能**控制 LED**

(3) 能实现更**复杂的显示交互和控制功能**

【设计思路】

①第一问：接收红外编码并显示。这一问的主要重点在于一定要了解**红外数据具体是如何从遥控器发射出来并被接收的**，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。**

在此处设计中，检测红外传输信息的目的是为了获得四个字节（共 32 位）的数据，包括两个字节的用户码、一个字节的**数据码**、一个字节的数据反码。

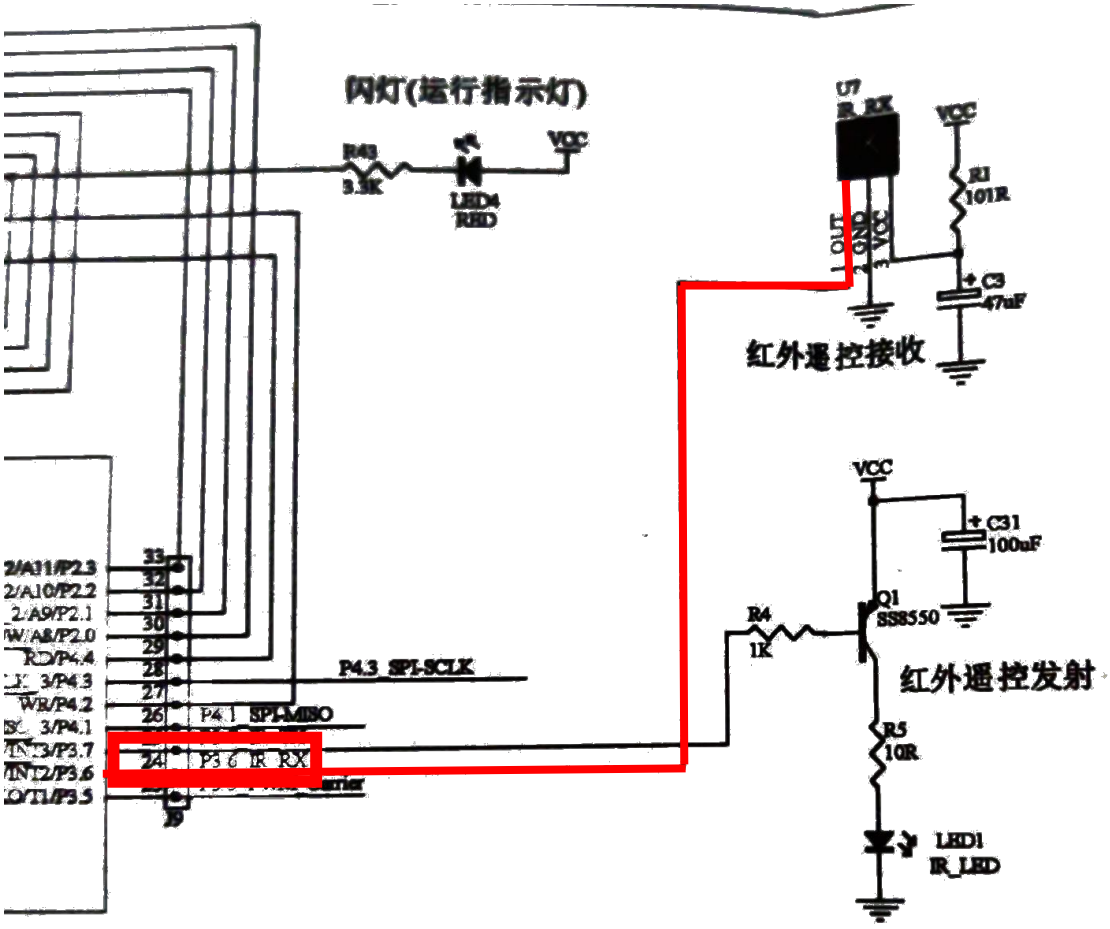


图 4.1 红外接收端引脚原理图

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**的位置，在 INT_CLKO（**AUXR2**）寄存器中的 EX2 位就是外部 INT2 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。**

该设计中，码型特征很明显。对于单一按键来说只要区分起始码逻辑 0 和逻辑 1，它们的持续时间有很大的不同。因此，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，**使用定时器 0 作为判决计数条件。**

（1）**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 1 位，计数范围是 0～65535.在每个时钟沿时，计数器 0 加 1。计算公式为

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

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**

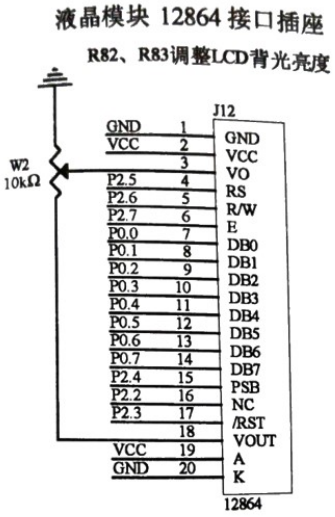
（2）**当 P3.6 为 1 时，表示高电平**，启动定时器 0 一直计数，以此获得高电平的续时间。计数器 0 的时钟是 SYSelk/12，系统时钟频率在写程序到 STC 单片机的时候设置为 **6.000MHz**，这是考虑到了计数器的范围是 16 位，计数范围是 0～65535.在每个时钟沿时，计数器 0 加 1。计算公式为

时间长度 =（12×）/计数值［TH0 * 256 + TL］

类似地，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。

接下来了解了如何接收以后，就可以将**接收到的解码数据存储在相应数组里**然后输出显示在 1602 液晶屏上了。



我们首先应该**初始化相关引脚的模式寄存器。**经过**查询原理图**， 发现 1602 显示模块与 STC51 单片机相连接占**用的引脚是 P0 和 P2 的**，所以要**先初始化**，设置为**准双向弱上拉模式。**初始化之后应该延时一段时间防止后续通讯出现问题。

然后就是**显示**，先运行在 led1602.c 文件中设置好的**读忙子函数**和**初始化子函数**，确定 lcd 能够显示稳定并且初始化之后，再运行**显示子函数**显示学号即可。

图 4.2 液晶模块连接原理图

这里的 1602 操作其实和前几次作业的初始化大同小异，可以说是几乎没有区别。

②第二问：控制 LED 灯。

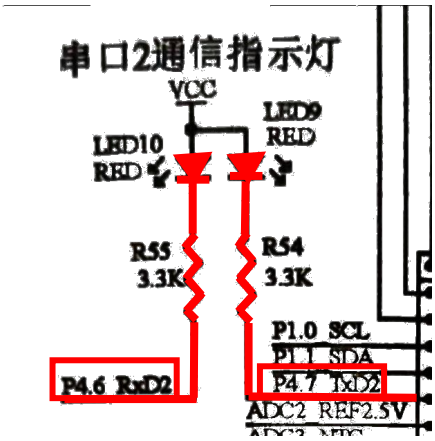


图 4.3 LED 灯对应引脚原理图

通过查询原理图我们可以知道，两个 LED 灯对应的引脚分别是 **P4.6** 和 **P4.7**，那么我们只需要在知道了遥控器传输出的**对应按键的数据码**，然后通过主函数红外中断中，将**收到相应码组与要做的操作对应起来**即可。这里两个 LED 灯用两个按键控制，分别是遥控器上面的“1”和“2”按键，操作也非常简单，每个灯的引脚默认为 1 高电平，灭，当**需要点亮任何一个灯的时候，将相对应的引脚置为 0** 即可。

③第三问：完成更加高级的交互操作——**红外遥控电子表**

这里其实我首先想到的是如何**将上次作业的电子表设计搬移到这次作业中**，成功实现代码的复用来减少编程的工作量。（然而并没有卵用最后工作量还是依然很大）

关于时钟设计，我设置了几个标志位来控制切换：

1、**显示模式切换标志位**：这个标志位主要有两个值，每个值对应的是显示时间或者显示日期，通过按键中断来改变标志位的值实现**显示模式的切换**。

2、**时间调整模式标志位**：这个标志位主要有两个值，一个值表示现在的日期时间正常显示，另外一个值表示进入**调整模式**，停止计数。这样才能实现调整时间的时候停止时间的走动。

3、**调整位标志位**：这个标志位有 6 个值，分别对应时、分、秒和年、月、日。通过按键改变这个标志位来实现**调整不同位的数字**。

4、**分频标志位**：由于单片机分频之后还是很快，所以要加入一个人为标志位，来继续将已经分频完的频率**继续分频**，来达到无限接近实际 1 秒的目的。

标志位设置完成后，还需要的就是设定**时间的进位函数**，这个函数包括有以下几个方面：

时、分、秒之间的进位，日期的大月、小月以及闰年 2 月的天数计算。同时也要包括小的位增加或减少到阈值之后较大位同时进位。

在设定时、分、秒之间的进位关系的时候，为了**最简化代码的难度**，设定**统一以秒为基础，分钟和小时在秒的基础上通过运算显示具体数值**，这样就不用嵌套很多 if 循环了，加快了编程效率。

设定完这些关键函数以后，由于要多次运用到显示日期和显示时间，那么我们不可能**每一次都完完整整的把显示函数写出来**，这样**非常麻烦并且会增加代码的体积**，所以我们不如**将显示函数作为单独的函数**，这样在主函数的书写中只需要**调用即可**。

接着，由于在调整的时候，用户**不可能通过自己按动按键的次数来猜测自己调整到了哪一位**，所以我们应该设定一个函数，来显示目前调整的是哪个变量，从而便于用户的操作。**由于第二行显示的时间是一个不断变化的量，那么显示调整的哪一位就不能同样放在这一行了**，而**应该放在第一行和编码信息一起，只要有足够大的间隔区分开即可**。

至此，三问的大体思路就完成了，接下来我们就要依照着以上的思路进行代码的编写与调试工作了。

【问题详解与代码展示】

①首先，我们应该把肯定会用到的头文件写入：

```
#include "reg51.h"
#include "intrins.h"
#include "led1602.h"
#include "stdio.h"
```

②接着，要定义一些一会一定会用到的变量与函数：

```
#define FOSC 6000000L   //单片机主时钟频率 6MHz
#define BAUD 115200     //串口通信波特率 115200
#define S2RI 0x01       //定义 S2RI 初值
#define S2TI 0x02       //定义 S2TI 初值
#define TIMS 3036       //定义定时器初值
long Second=0;          //总秒数
unsigned int  Adjust=1; //调整标志位
unsigned int  hour;     //小时
unsigned int  min;      //分钟
unsigned int  sec;      //秒
unsigned int  year=2019;//年
unsigned int  month=5;  //月份
unsigned int  date=30;  //天数
unsigned int  Mode=0;   //模式
unsigned int  count=0;  //控制秒的间隔的标志位
unsigned int  Time_flag=0;//显示模式控制位
sfr AUXR  =0x8E;
sfr AUXR1 =0xA2;
sfr AUXR2 =0x8F;
sfr TH2   =0xD6;
sfr TL2   =0xD7;
sfr P3M1  =0xB1;
sfr P3M0  =0xB2;
sfr P4    =0xC0;        //设定各端口物理地址
sbit P46  =P4^6;
sbit P47  =P4^7;        //声明 LED 对应引脚
sbit P36  =P3^6;        //声明 P3.6 引脚
bit busy=0;
```

```
unsigned int irdata[4]={0,0,0,0};  //设定 irdata 为接受红外解码的 4 字节数组
unsigned char str1[20];     //声明解码数据用 str1 显示
unsigned char str2[20];     //声明时间日期用 str2 显示
void SetTime(void);         //声明时间设置函数
void ShowTime(void);        //声明显示时间函数
void ShowDate(void);        //声明显示日期函数
void TimeCarry(void);       //声明时间日期进位函数
bit flag=0;                 //设定标志位用作日期时间显示切换
```

③开始对每个要用到的函数进行详细书写：

```
void timer_1() interrupt 3  //定时器初始化中断
{
    count++;
    if(count==8)            //分频，使其秒数接近于实际 1 秒
    {
        if(!Mode)           //非 0，不在设置状态，时间正常
        Second++;
        count=0;
    }
}

void ShowTime(void)         //显示时间函数
{
    sprintf((char*)str2,"    %02d:%02d:%02d   ",hour,min,sec);
    lcdshowstr(0,1,str2);
}
void ShowDate(void)         //显示日期函数
{
    sprintf((char*)str2,"   %04d-%02d-%02d   ",year,month,date);
    lcdshowstr(0,1,str2);
}

void SetTime(void)          //以秒为基础，设定时间显示
{
    hour=Second/3600;
    min=Second%3600/60;
    sec=Second%3600%60;
}

void TimeCarry(void)        //设定时间进位函数
{
    if(Second>=86400)       //一天进位
    {
```

```
        Second=0;
        date++;
    }
    else if(Second<0)
    {
        Second=86399;
        date--;
    }

    if(month>12)            //一年进位
    {
        month=1;
        year++;
        ShowDate();
    }
    else if(month<1)
    {
        month=12;
        year--;
        ShowDate();
    }


if(month==1|month==3|month==5|month==7|month==8|month==10|month==12)
//大月
    {
        if(date==32)
        {
            date=1;
            month++;
            ShowDate();
        }
        else if(date==0)
        {
            date=31;
            month--;
            ShowDate();
        }
    }
    else if(month==4|month==6|month==9|month==11)//小月
    {
        if(date==31)
        {
            date=1;
```

```
            month++;
            ShowDate();
        }
        else if(date==0)
        {
            date=31;
            month--;
            ShowDate();
        }
    }
    else if(month==2)//二月
    {
        if(year%400==0)
        {
            if(date==30)
            {
                date=1;
                month++;
                ShowDate();
            }
            else if(date==0)
            {
                date=31;
                month--;
                ShowDate();
            }
        }
        else if(year%4==0&year%100!=0)
        {
            if(date==30)
            {
                date=1;
                month++;
                ShowDate();
            }
            else if(date==0)
            {
                date=31;
                month--;
                ShowDate();
            }
        }
        else if(1)
        {
```

```
            if(date==29)
            {
                date=1;
                month++;
                ShowDate();
            }
            else if(date==0)
            {
                date=31;
                month--;
                ShowDate();
            }
        }
    }

}

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 P3.6 输入为 1 一直继续，否则退出
    {
        if(TH0>=0xEE)   //如果计数时间太长退出循环
            break;
    }
    TR0=0;              //如果 P3.6 输入为 0 则停止定时器 0 计数
    return(TH0*256+TL0);//返回计数器 0 的计数值
}
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 P3.6 输入为 0 一直继续，否则退出
    {
        if(TH0>=0xEE)   //如果计数时间太长退出循环
            break;
    }
    TR0=0;              //如果 P3.6 输入为 0 则停止定时器 0 计数
    return(TH0*256+TL0);//返回计数器 0 的计数值
}
```

```
void int2() interrupt 10//声明外部中断 2 的服务程序
{
    unsigned char i,j;
    unsigned int count=0;
    unsigned char dat=0;

    count=low_level_time();//读取低电平的计数值，即起始码的低前半部分

    if(count<3750 || count>5250)//如果不在范围内退出中断
    {
       return;
    }
    count=high_level_time();//读取高电平的计数值，即起始码的高后半部分
    if(count<1750 || count>2750)//如果不在范围内退出中断
    {
     return;
    }
    //下面开始处理 32 位数据
    for(i=0;i<4;i++)     //4 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;
        for(j=0;j<8;j++)//8 个比特的循环处理
          {
            count=low_level_time();//读取低电平的计数值，即逻辑位的低前半部分
            if(count<200 || count>350)  //如果不在范围内退出中断
                return;
           count=high_level_time();//读取高电平的计数值，即逻辑位的高后半部分
            if(count>200 && count<350)  //如果在逻辑 0 的范围内填充 0
                dat>>=1;
            else if(count>700 && count<1100)
                {                       //如果在逻辑 1 的范围内
                    dat>>=1;           //右移 1 位
                    dat|=0x80;         //高位用 1 填充
                }
                else return;           //否则不在给定的逻辑位高后半部分范围，退出


            }
            irdata[i]=dat;          //将 8 位数据保存在 irdata 数组当前索引号
    }
    flag=1;         //四个字节填满后，将 flag 置 1，表示有数据
    AUXR2|=0x10;    //开中断
}
```

④函数声明完毕，按照刚才的设计思路开始进行主程序的书写：

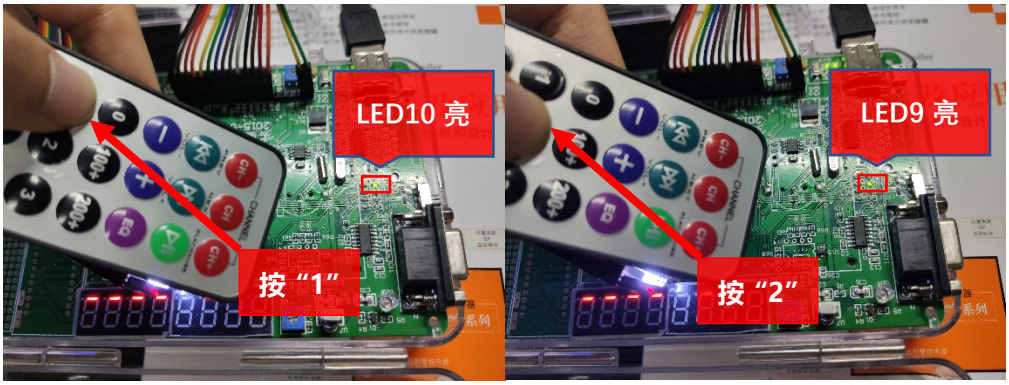
```
void main() //主程序
{
    unsigned int i;  //用作后续延时

    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
    AUXR=0x14;       //定时器 2/12，启动定时器 2，定时器模式
    TL2=(65536-((FOSC/4)/BAUD));    //计数初值低 8 位给定时器 2 的 TL2 寄存器
    TH2=(65536-((FOSC/4)/BAUD))>>8; //计数初值高 8 位给定时器 2 的 TH2 寄存器
    AUXR2|=0x10;     //允许外部中断 2


    TL1=TIMS;
    TH1=TIMS>>8;//初始化定时器 1
    TR1=1;      //启动定时器/计数器 1
    ET1=1;      //使能定时器/计数器 1

    P0M0=0;
    P0M1=0;
    P2M0=0;
    P2M1=0;
    for(i=0;i<10000;i++);
    lcdwait();
    lcdinit();
    EA=1;       //初始化 1602



    while(1)    //无限循环
    {
        TimeCarry();        //包含时间进位函数
        SetTime();          //包含时间设定函数
        if(Time_flag==0)    //设定显示时间标志位
            ShowTime();     //如果标志位为 0 显示时间
        else if(Time_flag==1)
            ShowDate();     //如果标志位为 1 显示日期

        if(flag==1)         //如果收到红外解码数据
        {
            flag=0;         //标志位清 0 来下次接收
```

```
sprintf(str1,"        %x%x%x%x",irdata[0],irdata[1],irdata[2],irdata[
3]);
          lcdshowstr(0,0,str1);   //设定在 1602 上显示接受到的数据
          switch(irdata[2])   //如果红外解码数据【数据码】部分为以下参数
          {
              case 12:        //如果对应遥控器数字 1
                  P46=0;
                  P47=1;
                  break;      //点亮 LED10

              case 24:        //如果对应遥控器数字 2
                  P46=1;
                  P47=0;
                  break;      //点亮 LED9

              case 8:         //如果对应遥控器数字 4
                  Time_flag=0;
                  break;      //显示时间

              case 28:        //如果对应遥控器数字 5
                  Time_flag=1;
                  break;      //显示日期

              case 90:        //如果对应遥控器数字 6
                  if(Adjust)  //调整对应位
                      Mode++; //从秒开始依次到年
                  if(Mode>6)
                      Mode=0; //调整完年恢复正常显示
                  switch(Mode)//设定模式现实告知用户此时正在更改哪一位
                  {
                      case 0: //正常显示模式
                          lcdshowstr(0,0,"Display");
                          break;
                      case 1: //调整秒模式
                          lcdshowstr(0,0,"Second");
                          break;
                      case 2: //调整分模式
                          lcdshowstr(0,0,"Minute");
                          break;
                      case 3: //调整时模式
                          lcdshowstr(0,0,"Hour");
                          break;
                      case 4: //调整日期模式
```

```
                lcdshowstr(0,0,"Day");
                break;
            case 5: //调整月份模式
                lcdshowstr(0,0,"Month");
                break;
            case 6: //调整年份模式
                lcdshowstr(0,0,"Year");
                break;
        }
        break;

    case 21:    //如果对应遥控器的+键
        switch(Mode)    //设定不同位具体每次调整多少
        {
            case 1:     //调整秒，每按一下+1s，调整完之后显示
                Second++;
                Time_flag=0;
                break;
            case 2:     //调整分，每按一下+60s，调整完之后显示
                Second+=60;
                Time_flag=0;
                break;
            case 3:     //调整时，每按一下+3600s，调整完之后显示
                Second+=3600;
                Time_flag=0;
                break;
            case 4:     //调整日期，每按一下+1d，调整完之后显示
                date++;
                Time_flag=1;
                break;
            case 5:
                month++;//调整月份，每按一下+1m，调整完之后显示
                Time_flag=1;
                break;
            case 6:     //调整年份，每按一下+1y，调整完之后显示
                year++;
                Time_flag=1;
                break;
        }
        break;

    case 7:     //如果对应遥控器-键
        switch(Mode)    //设定不同位具体每次调整多少
        {
```

```
                    case 1:     //调整秒，每按一下-1s，调整完之后显示
                        Second--;
                        Time_flag=0;
                        break;
                    case 2:     //调整时，每按一下-60s，调整完之后显示
                        Second-=60;
                        Time_flag=0;
                        break;
                    case 3:     //调整秒，每按一下-3600s，调整完之后显示
                        Second-=3600;
                        Time_flag=0;
                        break;
                    case 4:     //调整日期，每按一下-1d，调整完之后显示
                        date--;
                        Time_flag=1;
                        break;
                    case 5:     //调整月份，每按一下-1m，调整完之后显示
                        month--;
                        Time_flag=1;
                        break;
                    case 6:     //调整年份，每按一下-1y，调整完之后显示
                        year--;
                        Time_flag=1;
                        break;
                }
                break;

            }
            AUXR2|=0x10;    //使能外部中断 2
        }
    }
}
```

【结果展示】




0ffcf37
“1”对应的红外解码


0ff18e7
“2”对应的红外解码


0ff8f77
“4”对应的红外解码
切换到显示时间模式


0ff1ce3
2019-05-30
“5”对应的红外解码
切换到显示日期模式


Second 0ff5aa5
00:01:26
“6”对应的红外解码
切换到显示时间模式
左上角显示调整的位


0ff15ea
00:01:27
此时无论按遥控器上的
“+”或“-”按键都能达
到秒的加减


0ff7f8a
00:01:00


0ff7f8a
00:00:59
当为 1 分 0 秒时调整秒-
1，分钟自动减为 0，秒变
为 59


Minute 0ff5aa5
00:00:59
0ff15ea
00:01:59
切换到分钟位调整，此时无论按遥控器的“+”或“-”均可进行加减


Hour 0ff5aa5
01:00:00
0ff15ea
02:00:00
切换到小时位调整，此时无论按遥控器的“+”或“-”均可进行加减


Day 0ff5aa5
2019-05-29
0ff15ea
2019-05-31
0ff15ea
2019-06-01
切换到日期调整，此时日期天数可加减，并且到阈值月份自动进位


Month 0ff5aa5
2018-12-04
0ff7f85
2018-11-04
0ff15ea
2019-01-04
切换到月份调整，此时月份可加减，并且到阈值年份自动进位

关于更加详尽的的展示可以参考同目录下的视频文件。

【调试过程中遇到的问题】

在程序的调试过程中我遇到了很多的小问题：

首先是**时钟分频标志位 count 的调整**，经过很多次调整之后才让秒数显示接近真实的 1 秒。然后就是在函数编写过程中，**原本**我在每一次更改相应位置之后，都在**相对应的 case 后面添加了 ShowTime() 或者 ShowDate()函数**，但是**这样的话会遇到函数执行顺序先后的问题**，导致日期调整无法调整为 1 号或者 31 号，后来经过我的改进，**运用了现在的显示方法才实现了完美的显示**。

另外**一开始**的时候，我是让**红外解码以十进制显示**的，后来为了让程序显示在 1602 上更加美观，我将**红外解码变成了十六进制显示**，这样**节省了很多空间**可以让我加上很多更加细节的提示，比如提示用户目前调试的是哪一位，这样会更加友好。

【总结与思考】

经过了这次的实验，我收获到了很多，首先就是在面对这种多功能多情况的程序设计时，**脑海中一定要提前有一个程序的大致框架**，并且要把这个框架写出来或者画出一个思维导图，这样有利于后续的编程，不至于想到哪写到哪导致最后的代码非常混乱。

其次就是编程过程中对与多种情况的选择**没有必要运用全部的红外解码来进行情况选择**（即 case 后面没必要写全部的红外解码），因为其实仔细分析一下红外解码，其实**真正最有用的就是数据码**，只要用这个数据就足矣。

这次实验我学到了很多红外遥控的知识，也了解了通信过程中的编码解码，收获颇丰。