设计题（四）：使用 1602 实现一个可调电子时钟

一、题目：

【设计要求】

合理使用 STC 单片机内的定时器资源，并使用 1602 实现数字钟的功能，显示方式 xx：xx：xx（时：分：秒）

基本部分：

能在 1602 上以 xx：xx：xx 的形式显示时间，符合真实工作情况（40 分）

提高部分：

（1）能通过一个按键将 1602 切换到显示年月日，显示格式 xx/xx/xx（年/月/日）（20 分），

（2）通过按键可以调整时、分、秒(30 分）

发挥部分：

完善电子钟的功能（10 分）

注：

（1）设计的电子钟，使用最少的按键，按照电子表，最多使用 3 个按键。

（2）时钟工作时，其进位应该与真实的电子钟相同。

（3）显示时间和显示年月日之间的进位关系符合实际。

二、设计思路

1.顶层模块设计

模式类型	实现功能
0（基本部分）	正常显示时间（时：分：秒）
1（提高部分）	正常显示日期（年.月.日）
2（提高部分）	更改时间——小时
3（提高部分）	更改时间——分钟
4（提高部分）	更改时间——秒
5（发挥部分）	更改日期——年
6（发挥部分）	更改日期——月
7（发挥部分）	更改日期——日

顶层模块设计部分以上面的表格为准。

由于按键越少越好，仅使用两个按键：一个按键 SW17 控制**模式切换**，另一个按键 SW18 控制**更改时间或日期。**

2.一些其他问题的思考

（1）关于平年闰年

这是一个设计细节问题，在日期显示部分，应当充分考虑到平年闰年二月天数不同的问题。

原本考虑使用两个数组，元素为平年和闰年不同月份的天数，但是这么做会带来更多的空间资源消耗，于是优化了算法，仅使用一个数组表示各个月份的天数，把闰年二月这种特殊情况单独拿出来考虑。

（2）关于数组第一个元素

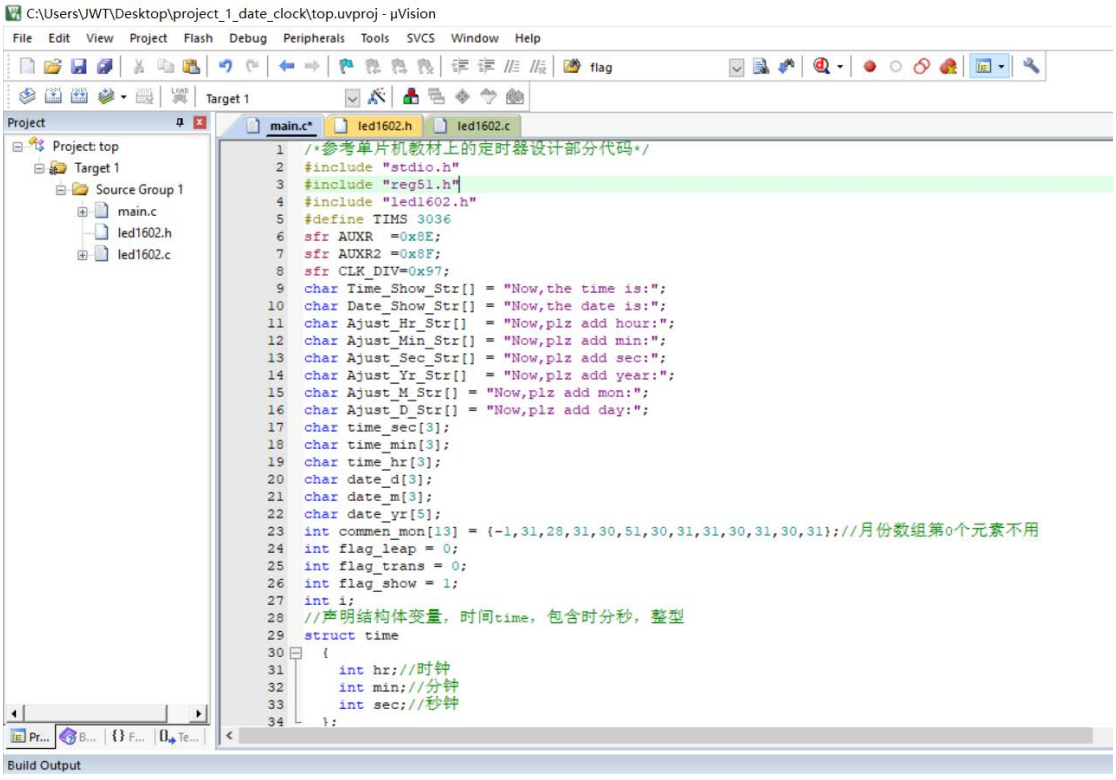
这同样也是我在设计中遇到的一个问题，就是 commom_mon[mydate.m]这种写法存在一个小问题，数组元素是从 0 开始的，而并不存在 0 月的情况，针对于这个细节问题，我在开辟数组时，使用了 13 个元素，第 0 个元素赋值为-1，正常情况下第 0 个元素是不可能用到的，如果使用到了，在 LED1602 上会显示月份为-1，这时候也能及时发现程序设计的错误。

（3）关于调节时间的过程中电子钟是否停表的问题

停表和不停表两种情况我都设计过，最终根据实际电子手表上的使用情况来看，调节时间是不停表的，也就是说时间依然是在走的，于是在本设计案例中我采用了调节时间不停表的模式。

停表的设计我在代码中已经做了注释。

三、具体实现



/*参考单片机教材上的定时器设计部分代码*/

#include "stdio.h"

#include "reg51.h"

```
#include "led1602.h"

#define TIMS 3036

sfr AUXR    =0x8E;

sfr AUXR2 =0x8F;

sfr CLK_DIV=0x97;

char Time_Show_Str[] = "Now,the time is:";

char Date_Show_Str[] = "Now,the date is:";

char Ajust_Hr_Str[]    = "Now,plz add hour:";

char Ajust_Min_Str[] = "Now,plz add min:";

char Ajust_Sec_Str[] = "Now,plz add sec:";

char Ajust_Yr_Str[]    = "Now,plz add year:";

char Ajust_M_Str[] = "Now,plz add mon:";

char Ajust_D_Str[] = "Now,plz add day:";

char time_sec[3];

char time_min[3];

char time_hr[3];

char date_d[3];

char date_m[3];

char date_yr[5];

int commen_mon[13] = {-1,31,28,31,30,51,30,31,31,30,31,30,31};//月份数组第 0
个元素不用

int flag_leap = 0;

int flag_trans = 0;

int flag_show = 1;
```

```
int i;
//声明结构体变量，时间 time，包含时分秒，整型
struct time
    {
        int hr;//时钟
        int min;//分钟
        int sec;//秒钟
    };
//声明结构体变量，日期 date，包含年月日，整型
struct date
  {
        int yr;//年
        int m;//月
        int d;//日
  };
struct time mytime;//定义全局变量 mytime
struct date mydate;//定义全局变量 mydate
void TDinit();//时间和日期初始化操作，用户可以自定义
int is_leap(int year);//判断平年和闰年
void show_time();//显示时间
void show_date();//显示日期
void trans() interrupt 0//切换不同的功能
{
    flag_trans++;
```

```
        lcdwritecmd(0x01);//清屏命令
        if(flag_trans == 8)
        {
            flag_trans = 0;
        }
}

void add() interrupt 2//加 1
{
        if(flag_trans == 2)
        {
            mytime.hr++;//小时+1
        }
        else if(flag_trans == 3)
        {
            mytime.min++;//分钟+1
        }
        else if(flag_trans == 4)
        {
            mytime.sec++;//秒+1
        }
        else if(flag_trans == 5)
        {
            mydate.yr++;//年+1
```

```
    }
    else if(flag_trans == 6)
    {
        mydate.m++;//月+1
    }
    else if(flag_trans == 7)
    {
        mydate.d++;//日+1
    }
}

void timer_0() interrupt 1//定时器 0 中断程序
{
    flag_show = 1;
/*****注释掉的部分是模拟调节时间时，时间暂停，不走，这种情况只在少数电子表中出现*****/
//  if(flag_trans != 2 && flag_trans != 3 && flag_trans != 4)
//  {
//      mytime.sec++;
//  }
    mytime.sec++;//调节时间时，时间继续走，不停表
    if(mytime.sec == 60)
    {
        mytime.sec = 0;
```

```
        mytime.min++;
    }
    if(mytime.min == 60)
    {
        mytime.min = 0;
        mytime.hr++;
    }
    if(mytime.hr == 24)
    {
        mytime.hr = 0;
        flag_leap = is_leap(mydate.yr);
      if( (mydate.d < commen_mon[mydate.m]) || (mydate.d ==
commen_mon[mydate.m] && mydate.m == 2 && flag_leap == 1) )
      {
          mydate.d++;
      }
        else
        {
          mydate.d = 1;
          if(mydate.m<12)
          {
            mydate.m++;
          }
          else
```

```
            {
              mydate.m = 1;
              mydate.yr++;
              }
        }
    }
}

void main()
{
    /*将 P0 口和 P1 口定义为准双向，弱上拉状态*/
    P0M0=0;
    P0M1=0;
   P2M0=0;
    P2M1=0;
    for(i=0;i<10000;i++);//空操作，用于延迟，避免显示闪烁
    /*初始化*/
    lcdwait();
    lcdinit();
    TDinit();
    CLK_DIV=0x03;
    TL0=TIMS;
    TH0=TIMS>>8;
    AUXR&=0x7F;
```

```
AUXR2|=0x01;
TMOD=0x00;
TR0 = 1;
ET0 = 1;//允许定时器 0 中断
IT0 = 1;//INT0 下降沿触发
EX0 = 1;//允许外部中断 0 中断
IT1 = 1;//INT1 下降沿触发
EX1 = 1;//允许外部中断 1 中断
EA = 1;//CPU 允许响应中断，开中断
while(1)
{
    if(flag_trans == 0 ||flag_trans == 2 || flag_trans == 3 || flag_trans == 4)
    {
        if(flag_trans == 0)
        {
            lcdshowstr(0,0,Time_Show_Str);
        }
        else if(flag_trans == 2)
        {
            lcdshowstr(0,0,Ajust_Hr_Str);
        }
        else if(flag_trans == 3)
        {
            lcdshowstr(0,0,Ajust_Min_Str);
```

```
        }
        else if(flag_trans == 4)
        {
              lcdshowstr(0,0,Ajust_Sec_Str);
        }

        if(flag_show == 1)
      {
            flag_show = 0;
              show_time();
      }
}

else if(flag_trans == 1 || flag_trans == 5 || flag_trans == 6 || flag_trans == 7)
{
        if(flag_trans == 1)
        {
              lcdshowstr(0,0,Date_Show_Str);
        }
        else if(flag_trans == 5)
        {
              lcdshowstr(0,0,Ajust_Yr_Str);
        }
        else if(flag_trans == 6)
```

```
                {
                    lcdshowstr(0,0,Ajust_M_Str);
                }
                else if(flag_trans == 7)
                {
                    lcdshowstr(0,0,Ajust_D_Str);
                }
                if(flag_show == 1)
                {
                    flag_show = 0;
                    show_date();
                }
            }
        }
}
/*时间日期初始化函数*/
void TDinit()
{
    mytime.hr = 23;
    mytime.min = 59;
    mytime.sec = 55;
    mydate.yr = 2020;
    mydate.m = 12;
    mydate.d = 31;
```

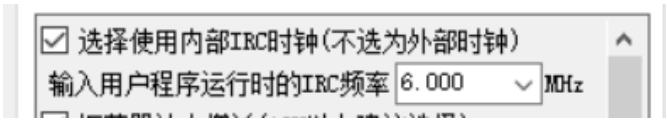
```
}
/*判断平年和闰年*/
int is_leap(int year)
{
    return ((year%4==0&&year%100!=0)||(year%400==0)) ? 1:0;
}
/*显示时间函数*/
void show_time()
{
    /****数字型转化为字符型****/
    if(mytime.sec<10)
    {
        time_sec[0] = '0';
        sprintf(time_sec + 1,"%d",mytime.sec);
    }
    else
    {
        sprintf(time_sec,"%d",mytime.sec);
    }
    if(mytime.min<10)
    {
        time_min[0] = '0';
        sprintf(time_min + 1,"%d",mytime.min);
    }
```

```
        else
        {
            sprintf(time_min,"%d",mytime.min);
        }
        if(mytime.hr<10)
        {
            time_hr[0] = '0';
            sprintf(time_hr + 1,"%d",mytime.hr);
        }
    else
        {
            sprintf(time_hr,"%d",mytime.hr);
        }
        lcdshowstr(0,1,time_hr);//显示小时
        lcdshowstr(2,1,":");
      lcdshowstr(3,1,time_min);//显示分钟
        lcdshowstr(5,1,":");
        lcdshowstr(6,1,time_sec); //显示秒
}
/*显示日期函数*/
void show_date()
{
        if(mydate.d<10)
      {
```

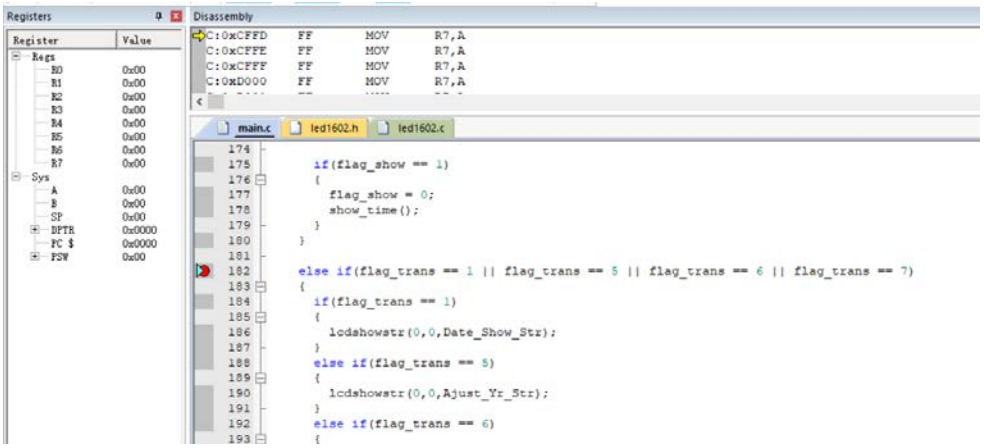
```
        date_d[0]='0';
        sprintf(date_d + 1,"%d",mydate.d);
    }
    else
    {
       sprintf(date_d,"%d",mydate.d);
    }
    if(mydate.m<10)
    {
    date_m[0] = '0';
        sprintf(date_m + 1,"%d",mydate.m);
    }
    else
    {
        sprintf(date_m,"%d",mydate.m);
    }
    sprintf(date_yr,"%d",mydate.yr);
    lcdshowstr(0,1,date_yr);
    lcdshowstr(4,1,".");
    lcdshowstr(5,1,date_m);
    lcdshowstr(7,1,".");
    lcdshowstr(8,1,date_d);
}
```

四、调试与测试

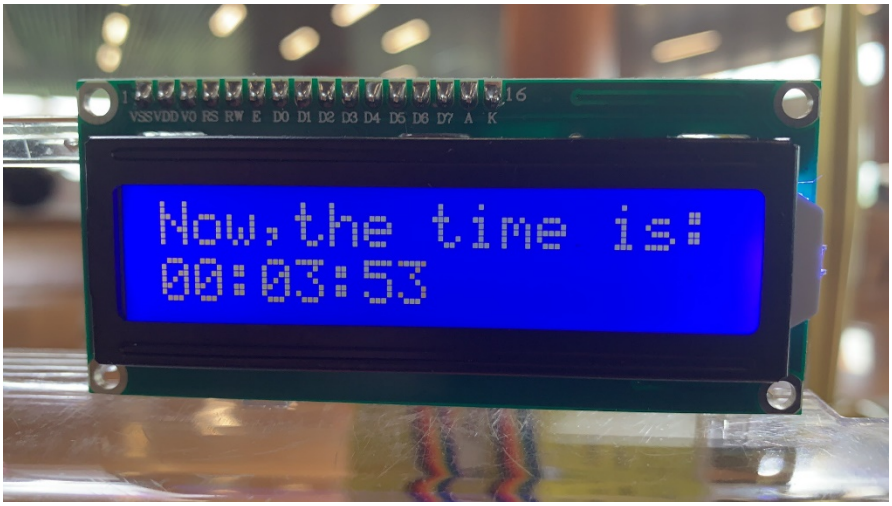
连接好单片机，设定正确的 IRC 频率，开始硬件调试



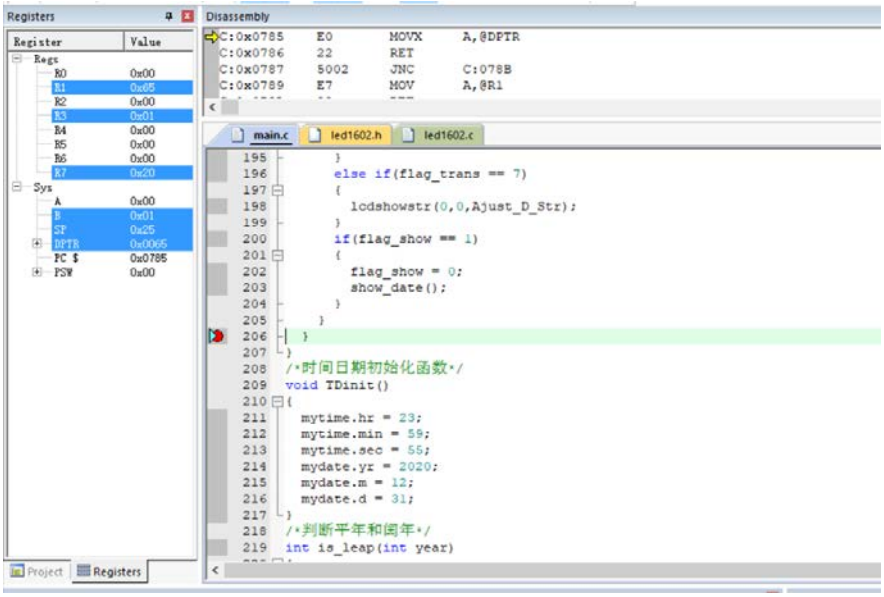
在 182 行处，设置断点



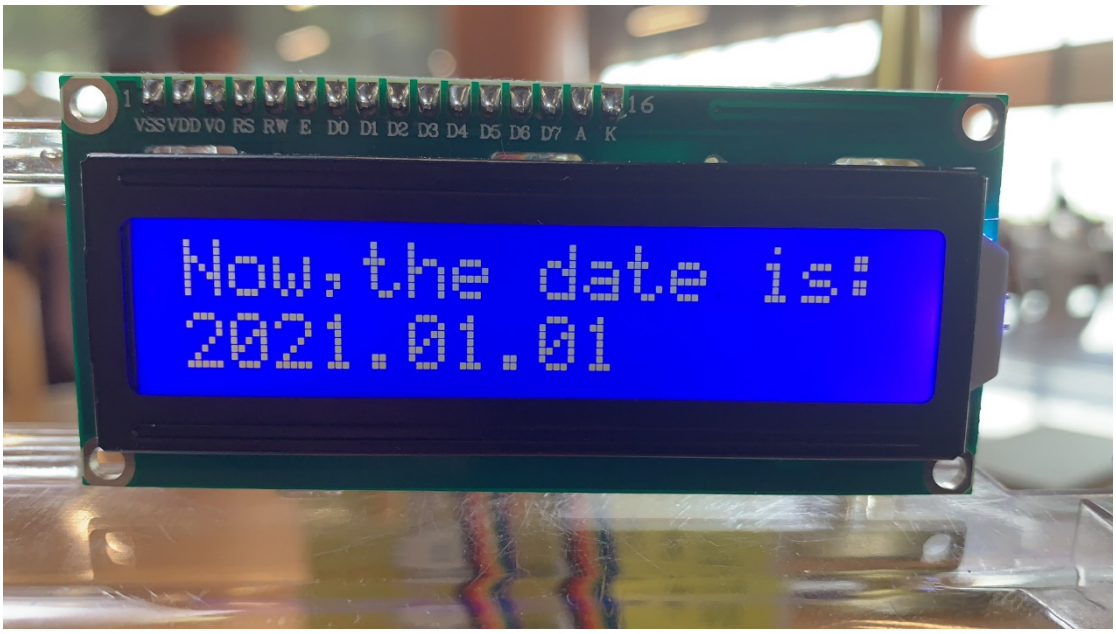
按下 F5，此时可以正常显示时间



然后在 206 行处设置断点



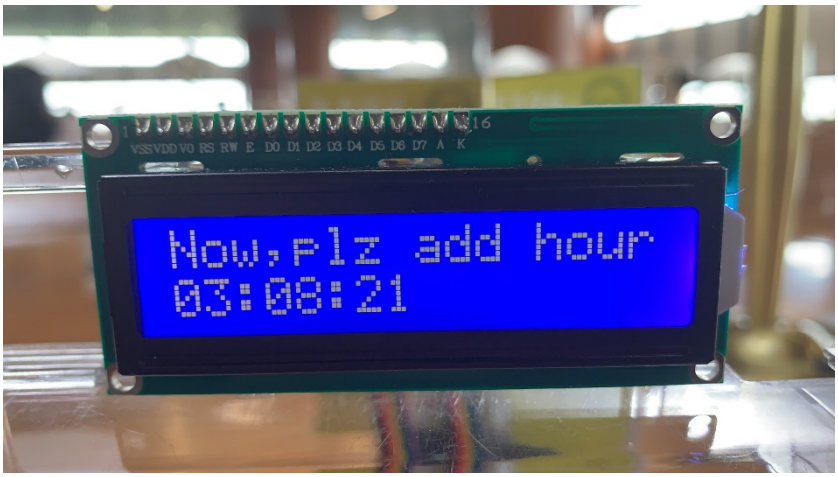
按下 F5，并且按下模式切换按键 SW17



可见，此时可以正常显示日期。
接下来，我们开始测试其他模式的功能。
模块：调节时间——小时（下面是视频和照片）

调节时间——小时.mp4

（双击左边图标即可播放）



调节时间——分钟（下面是视频和照片）

调节时间——分钟.mp4

（双击左边图标即可播放）

调节时间——秒（不停表模式）（下面是视频和照片）

调节时间——秒（不停表）.mp4

（双击左边图标即可播放）

注：停表模式展示放在最后（因为需要重新编译）

调节日期——年（下面是视频和照片）

调节日期——年.mp4

（双击左边图标即可播放）

调节日期——月（下面是视频和照片）

调节日期——月.mp4

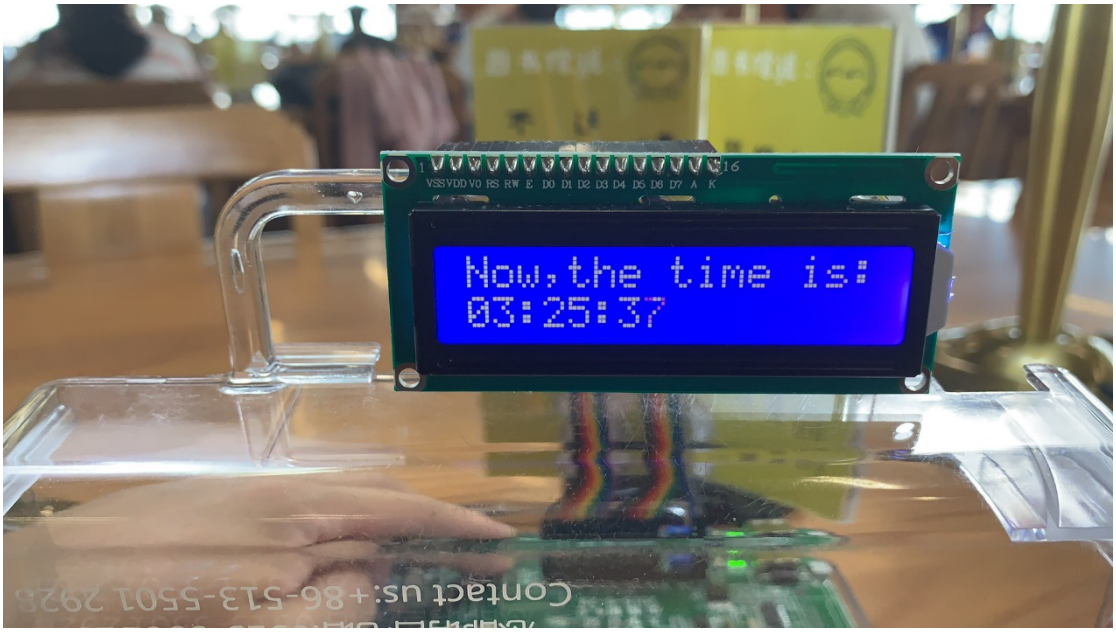
（双击左边图标即可播放）

调节日期——日（下面是视频和照片）

调节日期——日.mp4

（双击左边图标即可播放）

再按下一次模式切换键 SW17，回到最初显示时间的状态。



至此，调试工作已经基本结束，但是我单独再测试一下闰年二月份的特殊情况和调节时间不停表的两种情况，并录制视频如下：

闰年二月.mp4 （双击左边图标即可播放）

平年二月.mp4 （双击左边图标即可播放）

调节时间——时分秒（停表）.mp4

（双击左边图标即可播放）

五、总结与反思

这次作业做得相对比较顺利，由于自己参照了何老师课上以及课本中的很多设计思想，改变了之前不管三七二十一直接打代码的习惯，在设计之前画好了程序流程图，并且先完成了整体的顶层模块设计。

在一开始就要确定好自己的设计要实现哪些功能，以及要利用哪些按键进行切换功能等。这些都是要在具体编程前要做好的事情，不应该等到编程的过程中一边编程，一遍思考设计。

还有就是很多细节方面的问题，比如停表与不停表，闰年等，自己设计的东西要尽量做到和业界主流的实现方式相同，并且要考虑到资源消耗的问题，不要一股脑的随心所欲使用内存空间。

后期代码我也进行了许多优化，例如：将时间和日期定义为两个结构体类型，这样整体上来看更有规整，把显示时间和显示日期这两个函数进行模块化，独立出来，这样也更符合 C 语言程序设计模块化的规范要求。这些设计技巧和设计思想是自己当初学习 C 语言程序设计时所欠缺的，在单片机原理及应用这门课上，才一步一步建立起这样的思想，并且学习了很多编程的技巧。

当然，代码中还有许多不完善的地方，例如使用了较多的 if 语句判断等等，希望自己可以在之后的时间坚持学习，提高并完善自己的代码。