

《单片机原理及应用》第三次作业

【设计目标】

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。

【设计思路】

1. 设置定时器寄存器生成 1Hz 的时钟。

2. 设置全局变量数组存储时间信息和年月日信息，同时设置全局布尔变量作控制显示时间和年月日的标志。

3. 考虑到年月日的进位规则的特殊性，设置全局变量数组存储不同月份的天数。

4. 按键 SW24 循环切换显示模式，切换显示时间，日期，并进入设置时间日期模式。

5. 外部中断 0 控制循环切换设置的位。

6. 外部中断 1 设置时间或日期，每进入一次，设置的位数值加一。

【设计代码】

头文件部分

Lcd1602.h

```
#ifndef _1602_ //条件编译命令，如果没有定义_1602_
```

```
#define _1602_ //定义_1602_
```

```
#include "reg51.h" //包含 reg51.h 头文件
```

```
#include "intrins.h" //包含 intrins.h 头文件
```

```
sbit LCD1602_RS=P2^5; //定义 LCD1602_RS 为 P2.5 引脚
```

```
sbit LCD1602_RW=P2^6; //定义 LCD1602_RW 为 P2.6 引脚
```

```
sbit LCD1602_E =P2^7; //定义 LCD1602_E 为 P2.7 引脚
```

```
sfr LCD1602_DB=0x80; //定义 LCD1602_DB 为 P0 端口
```

```
sfr P0M1=0x93; //定义 P0 端口 P0M1 寄存器地址 0x93
```

```
sfr P0M0=0x94; //定义 P0 端口 P0M0 寄存器地址 0x94
```

```
sfr P2M1=0x95; //定义 P2 端口 P2M1 寄存器地址 0x95
```

```
sfr P2M0=0x96; //定义 P2 端口 P2M0 寄存器地址 0x96
```

```
void lcdwait(); //定义子函数 lcdwait 类型
```

```
void lcdwritecmd(unsigned char cmd); //定义子函数 lcdwritecmd 类型
```

```
void lcdwritedata(unsigned char dat); //定义子函数 lcdwritedata 类型
```

```
void lcdinit(); //定义子函数 lcdinit 类型
```

```
void lcdsetcursor(unsigned char x, unsigned char y); //定义子函数 lcdsetcursor 类型
```

```
void lcdshowstr(unsigned char x, unsigned char y, //定义子函数 lcdshowstr 类型
unsigned char *str);
```

```
#endif //条件预编译命令结束
```

```
Lcd1602.c
```

```
#include "led1602.h" //包含 led1602.h 头文件
```

```
void lcdwait() //声明 lcdwait 函数，用于读取 BF 标志
```

```
{
    LCD1602_DB=0xFF; //读取前，先将 P0 端口设置为“1”
    _nop(); //空操作延迟
    _nop(); //空操作延迟
    _nop(); //空操作延迟
    _nop(); //空操作延迟
    LCD1602_RS=0; //将 LCD1602 的 RS 信号拉低
    LCD1602_RW=1; //将 LCD1602 的 RW 信号拉高
    LCD1602_E=1; //将 LCD1602 的 E 信号拉高
    while(LCD1602_DB & 0x80); //等待标志 BF 为低，表示 LCD1602 空闲
    LCD1602_E=0; //将 LCD1602 的 E 信号拉低
}
```

```
void lcdwritecmd(unsigned char cmd) //声明 lcdwritecmd 函数，写命令到 1602
```

```
{
    lcdwait(); //调用 lcdwait 函数
    _nop(); //空操作延迟
    _nop(); //空操作延迟
    _nop(); //空操作延迟
    _nop(); //空操作延迟
    LCD1602_RS=0; //将 LCD1602 的 RS 信号拉低
    LCD1602_RW=0; //将 LCD1602 的 RW 信号拉低
    LCD1602_DB=cmd; //将命令控制码 cmd 放到 P0 端口
    LCD1602_E=1; //将 LCD1602 的 E 信号拉高
    _nop(); //空操作延迟
    _nop(); //空操作延迟
    _nop(); //空操作延迟
    _nop(); //空操作延迟
    LCD1602_E=0; //将 LCD1602 的 E 信号拉低
}
```

```
void lcdwritedata(unsigned char dat) //声明 lcdwritedata 函数，写数据到 1602
```

```
{
    lcdwait(); //调用 lcdwait 函数
    _nop(); //空操作延迟
    _nop(); //空操作延迟
    _nop(); //空操作延迟
    _nop(); //空操作延迟
}
```

```

    LCD1602_RS=1; //将 LCD1602 的 RS 信号拉高
    LCD1602_RW=0; //将 LCD1602 的 RW 信号拉低
    LCD1602_DB=dat; //将数据码 cmd 放到 P0 端口
    LCD1602_E=1; //将 LCD1602 的 E 信号拉高
    _nop(); //空操作延迟
    _nop(); //空操作延迟
    _nop(); //空操作延迟
    _nop(); //空操作延迟
    LCD1602_E=0; //将 LCD1602 的 E 信号拉低
}
void lcdinit() //声明 lcdinit 子函数，用来初始化 1602
{
    lcdwritcmd(0x38); //发命令 0x38, 2 行模式, 5*8 点阵, 8 位宽度
    lcdwritcmd(0x0c); //发命令 0x0c, 打开显示, 关闭光标
    lcdwritcmd(0x06); //发命令 0x06, 文字不移动, 地址自动加 1
    lcdwritcmd(0x01); //发命令 0x01, 清屏
}
//声明 lcdsetcursor 函数, 设置显示 RAM 的地址, x 和 y 表示在 1602 的列和行参数
void lcdsetcursor(unsigned char x, unsigned char y)
{
    unsigned char address; //声明无符号 char 类型变量 address
    if(y==0) //如果第一行
        address=0x00+x; //存储器地址以 0x00 开始
    else //如果是第二行
        address=0x40+x; //存储器地址以 0x40 开始
    lcdwritcmd(address|0x80); //写存储器地址命令
}
void lcdshowstr(unsigned char x, unsigned char y, unsigned char *str) //在液晶上指定的 x 和
y 位置, 显示字符
{
    lcdsetcursor(x,y); //设置显示 RAM 的地址
    while((*str)!='\0') //如果不是字符串的结尾, 则继续
    {
        lcdwritdata(*str); //发写数据命令, 在 LCD 上显示数据
        str++; //指针加 1, 指向下一个地址
    }
}

```

主程序部分

```

#include "reg51.h"
#include "stdio.h"
#include "led1602.h"
#define TIMS 3036 //定时器/计数器的计数初值

```

```

sfr AUXR=0x8E;                //声明 AUXR 寄存器的地址为 0x8E
sfr CLK_DIV=0x97;              //声明 CLK_DIV 寄存器的地址为 0x97

int time[3]={0,0,0};           //声明存储显示时间的数组
int date[3]={0,1,1};           //声明存储显示日期的数组
int month1[12]={31,28,31,30,31,30,31,31,30,31,30,31}; //声明存储闰年和非闰年的各
月份天数的数组
int month2[12]={31,29,31,30,31,30,31,31,30,31,30,31};
int mode=0;                    //声明控制显示模式的标志
int setbit=0;                  //声明控制设置时间哪一位的标志
int *mon;                      //声明 int 型指针

void delay(int i)              //声明延时函数
{
    while(--i);
}

void CLK_Init(void)            //声明时钟初始化函数
{
    CLK_DIV=0x03;              //设置 CLK_DIV 寄存器, 将主时钟 8 分频
    后作为 SYSclk
    TLO=TIMS;                  //设置定时器计数初值
    TH0=TIMS>>8;
    AUXR&=0x7F;               //AUXR 最高位置 0, STScIk/12 作为
    定时器时钟
    TMOD=0x00;                //定时器 0 工作模式为 16 为自动重加载
    TR0=1;                    //使能定时器/计数器 0
    ET0=1;                    //使能定时器/计数器 0 中断
}

char* timetran(int array[])    //声明产生要显示时间内容的函数
{
    char ti[8];               //产生 xx: xx: xx 的字符串

    ti[0]=array[0]/10+'0';
    ti[1]=array[0]%10+'0';
    ti[2]=': ';
    ti[3]=array[1]/10+'0';
    ti[4]=array[1]%10+'0';
    ti[5]=': ';
    ti[6]=array[2]/10+'0';

```

```

        ti[7]=array[2]%10+'0';
        return ti;
    }

```

```

char* datetran(int array[]) //声明产生要显示日期内容的函数
{
    //产生 xx/xx/xx 的字符串
    char ti[8];
    ti[0]=array[0]/10+'0';
    ti[1]=array[0]%10+'0';
    ti[2]='/';
    ti[3]=array[1]/10+'0';
    ti[4]=array[1]%10+'0';
    ti[5]='/';
    ti[6]=array[2]/10+'0';
    ti[7]=array[2]%10+'0';
    return ti;
}

```

```

void timer_0() interrupt 1 //声明定时器/计数器 0 中断服务函数
{
    if(date[0]%4 == 0) //判断是否为闰年，设置月份天数

    {
        mon=month2;
    }else mon=month1;
    time[2]++; //秒数加 1
    if(time[2]==60) //设置秒对分的进位
    {
        time[2]=0;
        time[1]++;
        if(time[1]==60) //设置分对时的进位
        {
            time[1]=0;
            time[0]++;
            if(time[0]==24) //设置时对日的进位
            {
                time[0]=0;
                date[2]++;
                if(date[2]==mon[date[1]]+1) //设置日对月份的进位
                {
                    date[2]=1;
                    date[1]++;
                    if(date[1]==13) //设置月份对年份的进位

```

```

    {
        date[1]=1;
        date[0]++;
        if(date[0]==100) date[0]=0; //年份溢出是归零
    }
}

void shift() interrupt 0 //声明外部中断控制设置时间的位数
{
    switch(mode) //判断是否处于时间设置模式
    {
        case 2:
            setbit++;break; //控制位数的标志位加 1
        default:break;
    }
    if(setbit>=6)setbit=0;
}

void setttime() interrupt 2 //声明外部中断设置时间
{
    if(date[0]%4 == 0) //判断是否为闰年，设置月份天数

    {
        mon=month2;
    }else mon=month1;
    switch(mode) //判断是否处于时间设置模式
    {
        case 2:
            switch(setbit)
            {
                case 0:
                    time[2]++;
                    if(time[2]>=60) time[2]=0;
                    break;
                case 1:
                    time[1]++;
                    if(time[1]>=60) time[1]=0;
                    break;
                case 2:
                    time[0]++;

```

```

        if(time[0]>=24) time[0]=0;
        break;
    case 3:
        date[2]++;
        if(date[2]>=mon[date[1]]+1) date[2]=1;
        break;
    case 4:
        date[1]++;
        if(date[1]>=13) date[1]=1;
        break;
    case 5:
        date[0]++;
        if(date[0]>=100) date[0]=0;
        break;
    default:break;
    }
    break;
default:break;
}
}

void main()
{
    char *timedis,*datedis;           //声明存储显示时间和日期的字符串
    int i;
    unsigned char c1_new,c1_old=1;
    EX0=1;                             //使能外部中断 0
    IT0=1;                             //设置外部中断 0 为低电平触发
    EX1=1;                             //使能外部中断 1
    IT1=1;                             //设置外部中断 1 为低电平触发
    EA=1;                             //使能 CPU 中断
    CLK_Init();                       //时钟初始化
    P0M0=0;                           //通过 P0M0 和 P0M1 寄存器，将 P0 口
    P0M1=0;                           //定义为准双向，弱上拉
    P2M0=0;                           //通过 P2M0 和 P2M1 寄存器，将 P2 口
    P2M1=0;                           //定义为准双向，弱上拉
    lcdwait();                       //等待 1602 字符 LCD 稳定
    lcdinit();                       //初始化 1602 字符 LCD 稳定
    while(1){
        P0=0xF0;                     //将 P0.0~P0.3 拉低，在读 P0.4~P0.7 前，发 F
        delay(60);
        c1_new=P0&0xF0;
        if(c1_new!=c1_old)

```

```

{
    c1_old=c1_new;
    if(c1_new!=0xF0)
    {
        P0=0xFE;
        delay(60);
        c1_new=P0;
        if(c1_new==0xEE)                //如果扫描到按了一次键，切换一次显示

模式
        {
            mode++;
        }
    }
}
if(mode==3)
{
    mode=0;
    setbit=0;
}
switch(mode)
{
    case 0:                            //显示时间
        timedis=timetran(time);
        lcdsetcursor(0,0); //设置显示 RAM 的地址
        for(i=0;i<8;i++)              //显示八位字符
        {
            lcdwritedata(timedis[i]);
        }
        break;
    case 1:                            //显示日期
        datedis=datetran(date);
        lcdsetcursor(0,0); //设置显示 RAM 的地址
        for(i=0;i<8;i++)              //显示八位字符
        {
            lcdwritedata(datedis[i]);
        }
        break;
    case 2:                            //设置模式，设置位会有闪烁
        switch(setbit)                //setbit0~2 时，设置的是时间；3~5 设置的是日期
        {
            case 0:                    //设置秒
                timedis=timetran(time);
                lcdsetcursor(0,0); //设置显示 RAM 的地址
                for(i=0;i<8;i++)

```



```

    {
        lcdwritedata(timedis[i]);
    }
    delay(5000);
    lcdsetcursor(0,0); //设置显示 RAM 的地址
    for(i=0;i<8;i++)
    {
        if(i==6 || i==7) lcdwritedata(' ');
        else lcdwritedata(timedis[i]);
    }
    delay(5000);
    break;
case 1: //设置分
    timedis=timetran(time);
    lcdsetcursor(0,0); //设置显示 RAM 的地址
    for(i=0;i<8;i++)
    {
        lcdwritedata(timedis[i]);
    }
    delay(5000);
    lcdsetcursor(0,0); //设置显示 RAM 的地址
    for(i=0;i<8;i++)
    {
        if(i==3 || i==4) lcdwritedata(' ');
        else lcdwritedata(timedis[i]);
    }
    delay(5000);
    break;
case 2: //设置时
    timedis=timetran(time);
    lcdsetcursor(0,0); //设置显示 RAM 的地址
    for(i=0;i<8;i++)
    {
        lcdwritedata(timedis[i]);
    }
    delay(5000);
    lcdsetcursor(0,0); //设置显示 RAM 的地址
    for(i=0;i<8;i++)
    {
        if(i==1 || i==0) lcdwritedata(' ');
        else lcdwritedata(timedis[i]);
    }
    delay(5000);
    break;

```

```

case 3:                                     //设置日
    datedis=datetran(date);
    lcdsetcursor(0,0); //设置显示 RAM 的地址
    for(i=0;i<8;i++)
    {
        lcdwritedata(datedis[i]);
    }
    delay(5000);
    lcdsetcursor(0,0); //设置显示 RAM 的地址
    for(i=0;i<8;i++)
    {
        if(i==6 || i==7) lcdwritedata(' ');
        else lcdwritedata(datedis[i]);
    }
    delay(5000);
    break;

case 4:                                     //设置月
    datedis=datetran(date);
    lcdsetcursor(0,0); //设置显示 RAM 的地址
    for(i=0;i<8;i++)
    {
        lcdwritedata(datedis[i]);
    }
    delay(5000);
    lcdsetcursor(0,0); //设置显示 RAM 的地址
    for(i=0;i<8;i++)
    {
        if(i==3 || i==4) lcdwritedata(' ');
        else lcdwritedata(datedis[i]);
    }
    delay(5000);
    break;

case 5:                                     //设置年
    datedis=datetran(date);
    lcdsetcursor(0,0); //设置显示 RAM 的地址
    for(i=0;i<8;i++)
    {
        lcdwritedata(datedis[i]);
    }
    delay(5000);
    lcdsetcursor(0,0); //设置显示 RAM 的地址
    for(i=0;i<8;i++)
    {
        if(i==1 || i==0) lcdwritedata(' ');

```

```

        else lcdwritedata(datedis[i]);
    }
    delay(5000);
    break;
default:break;
}
break;
default:break;
}
}
}

```

【实验结果】

下载程序后，1602 从 00: 00: 00 开始计时



按下 SW24,1602 显示年月日 00/01/01



再按一次 SW24, 进入设置模式 (由于设置位有闪烁效果, 存在不显示的状态)



按下 SW17, 触发外部中断 0, 设置位移至分



按下 SW18, 触发外部中断 1, 分的数值加 1



再按一次 SW24，切换至显示时间模式。实际当中，可以按照顺序随意切换显示模式，在设置时间日期时，也可循环切换设置的位，同时也可以随意设置时间和日期，达到电子表的效果。

