通信、信工、安全专业单片机实验（三）

【实验要求】 在七段数码管上实现电子钟

说明：实验（三）的 2 道题目设计源代码分别在文件夹 exp3.1、exp3.2 中。

【实验内容】

（1）在七段数码管上实现电子钟（80 分）

【实验结果】

（1）保存设计工程，以及设计源代码，源代码每行给出注释。

（2）实验报告，包含设计思路，设计过程等，遇到的问题，以能说明本次实验的内容即可。

（3）实验报告和设计工程保存在一个目录下，文件夹命名规则（学号+班级+名字）

1．设计思路

参照书本第 11 章的例子 11-1。

2．调试过程及遇到问题的解决办法

1) 关于硬件在线调试，无法跳入中断服务程序的问题

是主程序的中断的设置有问题，修改之后，就可以跳入了。

2) 关于显示与预期不符的问题

因为定时器 0 用来计数，计数频率的计算公式为:计数频率=[(IRC 时钟频率/clk_div 的分频系数)/12] /(65536-[RL_TH0,RL_TL0])，定时器 1 用来作为波特率时钟，[RL_TH1,RL_TL1]=65536-SYSclk/(串口 1 的波特率 x4)，现在串口 1 的波特率需要为 115200，综合考虑之下，选择了 IRC 时钟频率=11.0592MHz，CLK_DIV=0x03;//分频系数为 8，SYSclk=主时钟频率/8，这样使得[RL_TH0,RL_TL0]=0，则 13 进制计数器的计数频率=1.75Hz，[RL_TH1,RL_TL1] =65536-3=65533。

3．实验结果：成功实现所有功能。主时钟频率 **main_clock=6MHz**

4．源代码及注释

```
#include "STC15F2K60S2.H"//主时钟频率 main_clock=6MHz
#include "spi.h"//包含自定义文件
//t_display 数组保存着 0~9 的段码
unsigned char code t_display[10]={0x3F,0x06,0x5B,0x4F,
                                  0x66,0x6D,0x7D,0x07,
```

```
                                        0x7F,0x6F};
//t_displayPoint 数组保存着 0~9 含小数点的段码
unsigned char code t_displayPoint[10]={0xBF,0x86,0xDB,0xCF,
                                        0xE6,0xED,0xFD,0x87,
                                        0xFF,0xEF};
//T-COM 数组保存着管选码的反码，在一个时刻只有一个管选信号为低，其余为高
unsigned char code T_COM[8]={0x01,0x02,0x04,0x08,0x10,0x20,0x40,0x80};
//bit flag=0;
unsigned int hour=0;
unsigned int minute=0;
unsigned int second=0;
unsigned char i=0;
void SPI_SendByte(unsigned char dat)//定义 SPI 数据发送函数
{
    SPSTAT=SPIF+WCOL;//写 1 清零 SPATAT 寄存器内容
    SPDAT=dat;//dat 写入 SPDAT SPI 数据寄存器
    while((SPSTAT & SPIF)==0);//判断发送是否完成
    SPSTAT=SPIF+WCOL; //写 1 清零 SPATAT 寄存器内容
}
//定义用于写 7 段数码管的子函数 seg7scan，index1 参数控制管选，index2 控制段码
void seg7scan(unsigned char index1,unsigned char index2)
{
    SPI_SendByte(~T_COM[index1]);//向 74HC595(U5)写入管选信号
    SPI_SendByte(t_display[index2]); //向 74HC595(U6)写入管选信号
    HC595_RCLK=1;//通过 P5.4 端口向两片 74HC595 发送数据锁存
    HC595_RCLK=0;//上升沿有效
}
//定义用于写 7 段数码管的子函数 seg7scanPoint，显示的数字均含小数点，index1 参数控制
管选，index2 控制段码
void seg7scanPoint(unsigned char index1,unsigned char index2)
{
    SPI_SendByte(~T_COM[index1]);
    SPI_SendByte(t_displayPoint[index2]);
    HC595_RCLK=1;
    HC595_RCLK=0;
}
void timer_0() interrupt 1//在定时器 0 中断服务程序中轮流导通 6 个数码管，并显示对应数
码管的数字
{
                        if(i==0)//取秒数的个位
                          seg7scan(i,hour/10);
                        else if(i==1) //取秒数的十位
                          seg7scanPoint(i,hour-(hour/10)*10);
                        else if(i==2) //取分钟数的个位
```

```
                seg7scan(i,minute/10);
            else if(i==3) //取分钟数的十位
                seg7scanPoint(i,minute-(minute/10)*10);
            else if(i==4) //取小时数的个位
                seg7scan(i,second/10);
            else if(i==5) //取小时数的十位
                    seg7scan(i,second-(second/10)*10);
            else ;
        i++;
        i=i%6;
}
void timer_1() interrupt 3//声明定时器 1 中断服务程序，使得总秒数加 1，并且按照正确的
时分秒显示。
{
    P46=!P46;//P4.6 引脚取反
    second++;//秒数自增 1
    if(second==60)//秒数计满 60
    {
        second=0;//秒数清 0
        minute++;//分钟数自增 1
    }
    if(minute==60)//分钟数计满 60
    {
        minute=0;//分钟数清 0
        hour++;//小时数自增 1
    }
    if(hour==24)//小时数计满 24
    {
        hour=0;//小时数清 0
    }
}
void main()
{
    SPCTL=(SSIG<<7)+(SPEN<<6)+(DORD<<5)+(MSTR<<4)
          +(CPOL<<3)+(CPHA<<2)+SPEED_4;//给寄存器 SPCTL 赋值（配置 SPI 模块）
    CLK_DIV=0x03;//主时钟 8 分频作为 SYSclk 频率
    TL0=TIMS;
    TH0=TIMS>>8; //装入计数初值
    TL1=TIMS1;
    TH1=TIMS1>8; //装入计数初值
    AUXR&=0x3F;//定时器 0 和 1 是 12 分频
    AUXR1=0X08;//将 SPI 接口信号切换到第 3 组引脚上
    TMOD=0x00;//定时器 0/1，16 位重加载定时器模式
    TR0=1;//启动定时器 0
```

```
    TR1=1; //启动定时器 1
    ET0=1;//允许定时器 0 溢出中断
    ET1=1; //允许定时器 1 溢出中断
    EA=1;//CPU 允许响应中断请求

    while(1)//无限循环，等待中断
    {

    }
}
//最高位置 0，SYSclk/12 作定时器 0 时钟，次高位置 1，SYSclk 不分频，作定时器 1 时钟
```

（2）可以调整时分秒（20 分）

1．设计思路

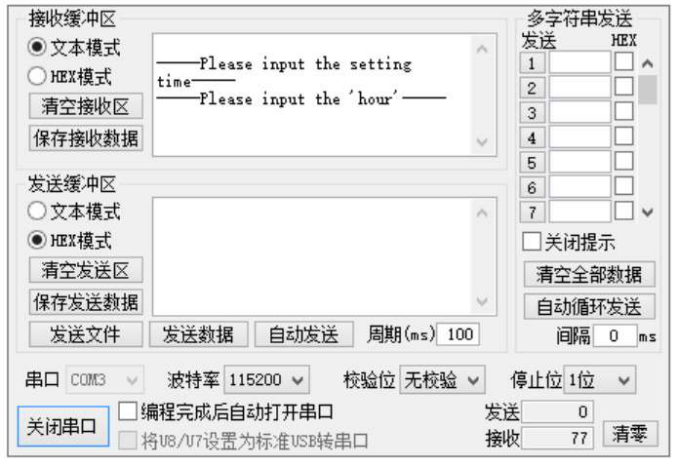
参照书本第 11 章的例子 11-1 和书本第 10 章的例子 10-1。

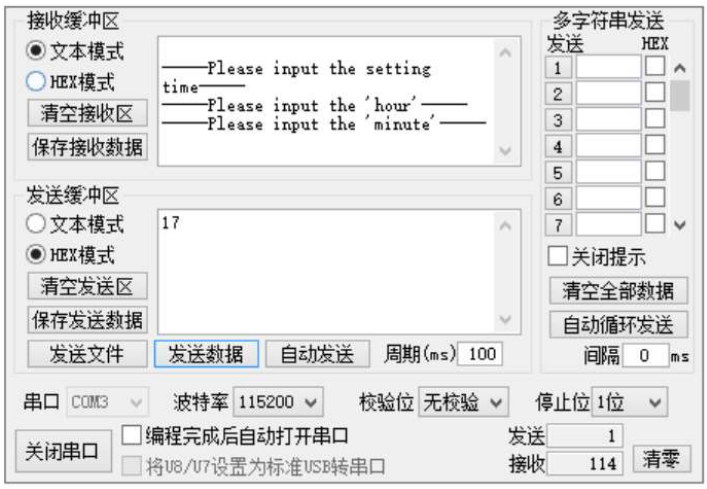
2．调试过程及遇到问题的解决办法

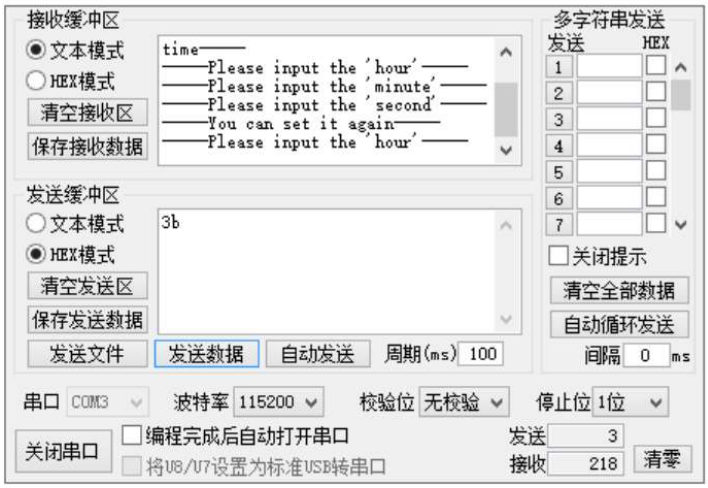
1) 关于串口的接收数据的限制问题

之前都是在发送缓冲区发字符信息（文本模式），开始的时候不知道该怎么发数值信息，后来想到可以在串口助手的发送缓冲区把“文本模式”改成“HEX 模式”，可以发送 16 进制数值。

3．**实验结果**：成功实现通过串口来调整时分秒。在本程序中，主时钟频率为 **11.0592MHz**，会依次出现提示“Please input the setting time”和“Please input the 'hour'”，这时需要在串口助手中，选择“HEX 模式”，以 16 进制方式输入十进制值为 0~23 的数字，设定时间中的“**时**”。然后，接收缓冲区会出现提示“Please input the ' minute '”，这时需要在串口助手中，选择“HEX 模式”，以 16 进制方式输入十进制值为 0~59 的数字，设定时间中的“**分**”。最后，接收缓冲区会出现提示“Please input the ' second '”，这时需要在串口助手中，选择“HEX 模式”，以 16 进制方式输入十进制值为 0~59 的数字，设定时间中的“**秒**”。此时，接收缓冲区会进行新一轮的循环，依次提示此时可以输入设定时间中的“**时**”、“**分**”、“**秒**”，可以进行再次的时间调整。







4．源代码及注释

```
#include "STC15F2K60S2.H"
#include "spi.h"//main_clock=11.0592MHz
#define FOSC    11059200L//声明单片机的工作频率
#define BAUD 115200//声明串口 1 的波特率
sfr TH2=0xD6;
sfr TL2=0xD7;

bit busy=0; //声明 bit 型变量
//声明数组型变量，保存待打印的信息
xdata char begin[]={"\r\n-----Please input the setting time-----"};
xdata char hourTip[]={"\r\n-----Please input the 'hour'-----"};
xdata char minuteTip[]={"\r\n-----Please input the 'minute'-----"};
xdata char secondTip[]={"\r\n-----Please input the 'second'-----"};
xdata char again[]={"\r\n-----You can set it again-----"};
//t_display 数组保存着 0~9 的段码
unsigned char code t_display[10]={0x3F,0x06,0x5B,0x4F,
                                  0x66,0x6D,0x7D,0x07,
```

```
                                        0x7F,0x6F};
//t_displayPoint 数组保存着 0~9 含小数点的段码
unsigned char code t_displayPoint[10]={0xBF,0x86,0xDB,0xCF,
                                        0xE6,0xED,0xFD,0x87,
                                        0xFF,0xEF};
//T-COM 数组保存着管选码的反码，在一个时刻只有一个管选信号为低，其余为高
unsigned char code T_COM[8]={0x01,0x02,0x04,0x08,0x10,0x20,0x40,0x80};

unsigned int hour=0;
unsigned int minute=0;
unsigned int second=0;
unsigned char i=0;

void SendData(unsigned char dat) //声明 SendData 函数
{
    while(busy); //判断是否发送完，没有则等待
    SBUF=dat; //否则，将数据 dat 写入 SBUF 寄存器
    busy=1; //将 busy 标志置 1
}
void SendString(char *s) //声明 SendString 函数
{
    while(*s!='\0') //判断是否是字符串的结尾
        SendData(*s++);//如果没有结束，调用 SendData 发送数据
}
void uart1() interrupt 4//声明串口 1 中断服务程序 uart1
{
    if(RI) //通过 RI 标志，判断是否接收到数据
        RI=0; //如果 RI 为 1，则软件清零 RI
    if(TI) //通过 TI 标志，判断是否发送完数据
        TI=0; //如果 TI 为 1，则软件清零 TI
      busy=0; //将 busy 标志清零
}

void SPI_SendByte(unsigned char dat) //定义 SPI 数据发送函数
{
    SPSTAT=SPIF+WCOL; //写 1 清零 SPATAT 寄存器内容
    SPDAT=dat; ;//dat 写入 SPDAT SPI 数据寄存器
    while((SPSTAT & SPIF)==0); //判断发送是否完成
    SPSTAT=SPIF+WCOL; //写 1 清零 SPATAT 寄存器内容
}
//定义用于写 7 段数码管的子函数 seg7scan，index1 参数控制管选，index2 控制段码
void seg7scan(unsigned char index1,unsigned char index2)
{
```

```
    SPI_SendByte(~T_COM[index1]); //向 74HC595(U5)写入管选信号
    SPI_SendByte(t_display[index2]); //向 74HC595(U6)写入管选信号
    HC595_RCLK=1; //通过 P5.4 端口向两片 74HC595 发送数据锁存
    HC595_RCLK=0; //上升沿有效
}
//定义用于写 7 段数码管的子函数 seg7scanPoint，显示的数字均含小数点，index1 参数控制管选，index2 控制段码
void seg7scanPoint(unsigned char index1,unsigned char index2)
{
    SPI_SendByte(~T_COM[index1]);
    SPI_SendByte(t_displayPoint[index2]);
    HC595_RCLK=1;
    HC595_RCLK=0;
}
//在定时器 0 中断服务程序中轮流导通 6 个数码管，并显示对应数码管的数字
void timer_0() interrupt 1
{
                    if(i==0) //取秒数的个位
                       seg7scan(i,hour/10);
                    else if(i==1) //取秒数的十位
                       seg7scanPoint(i,hour-(hour/10)*10);
                    else if(i==2) //取分钟数的个位
                       seg7scan(i,minute/10);
                    else if(i==3) //取分钟数的十位
                       seg7scanPoint(i,minute-(minute/10)*10);
                    else if(i==4) //取小时数的个位
                       seg7scan(i,second/10);
                    else if(i==5) //取小时数的十位
                          seg7scan(i,second-(second/10)*10);
                    else ;
            i++;
                 i=i%6;
}
    void timer_1() interrupt 3//声明定时器 1 中断服务程序，使得总秒数加 1，并且按照正确的时分秒显示。
    {
        P46=!P46;//P4.6 引脚取反
        second++;//秒数自增 1
        if(second==60)//秒数计满 60
        {
            second=0;//秒数清 0
```

```
            minute++;//分钟数自增 1
        }
        if(minute==60)//分钟数计满 60
        {
            minute=0;//分钟数清 0
            hour++;//小时数自增 1
        }
        if(hour==24)//小时数计满 24
        {
            hour=0;//小时数清 0
        }
    }
void main()
{
    unsigned char c;
    unsigned char j=0;//j=0\1\2 分别代表当下应该调整“时”\“分”\“秒”
    SPCTL=(SSIG<<7)+(SPEN<<6)+(DORD<<5)+(MSTR<<4)
            +(CPOL<<3)+(CPHA<<2)+SPEED_4;//给寄存器 SPCTL 赋值
    CLK_DIV=0x03;//主时钟 8 分频作为 SYSclk 频率

    SCON=0x50;//UART
    TL0=TIMS;//TIMER
    TH0=TIMS>>8; //装入计数初值
    TL1=TIMS1;
    TH1=TIMS1>8; //装入计数初值
    TL2=(65536-((FOSC/8/4)/BAUD));
    TH2=(65536-((FOSC/8/4)/BAUD))>>8; //装入计数初值

    AUXR=0x15;//允许定时器 2，不分频，选择定时器 2 作为波特率发生器
    TMOD=0x00; //设置 GATE=0，定时器 0 模式 0（16 位自动重加载模式）

    AUXR1=0X08; //将 SPI 接口信号切换到第 3 组引脚上

    TR0=1; //启动定时器 0
    TR1=1; //启动定时器 1
    ET0=1;//允许定时器 0 溢出中断
    ET1=1; //允许定时器 1 溢出中断
    ES=1; //允许串口 1 中断
    EA=1;//CPU 允许响应中断请求
```

```
        SendString(&begin);
        SendString(&hourTip);//发送提示信息
        while(1)
        {
            if(RI==1)
             {
                c=SBUF;
                if(j==0)//当 j=0
                {
                    hour=c;//将接受到的数值给 hour 变量，设置小时数
                    j++;
                    SendString(&minuteTip);
                }
                else if(j==1) //当 j=1
                {
                    minute=c; //将接受到的数值给 minute 变量，设置分钟数
                    j++;
                    SendString(&secondTip);
                }
                else if(j==2) //当 j=2
                {
                    second=c; //将接受到的数值给 second 变量，设置秒数
                    j=0;
                    SendString(&again);//输出提示信息，提示可以再次设置调整“时分秒”
                    SendString(&hourTip);
                }
                else ;
            }

        }
}
```