通信、信工、安全专业单片机开放实验（一）

【实验要求】　　实现对 STC 开发板上四个 LED 的控制（包含两部分内容）

说明：实验（一）的 3 道题目设计源代码分别在文件夹 exp1.1、exp1.2、exp1.3 中。

1）使用外部中断对四个 LED 灯进行计数控制（40 分）。

（1）外部中断 0，　用于控制 LED 的递增计数，计数范围（30 分）

①信工专业（13 进制）；

②通信专业（14 进制）；

（2）外部中断 1，用于对计数清零（10 分）。

使用 word 文档详细进行说明设计过程。包括设计思路、调试过程，遇到问题的解决方法。

1. 设计思路

参照书本 6.15.3 节的例子 6-64, 先看懂例子, 再尝试进行迁移拓展解决本题的问题。

1) 先确定 LED 灯的显示方向，我设定为从高位到低位依次是 LED10\LED9\LED8\LED7。

2) 定好高低位之后，明确 LED 灯对应管脚输出为 0 时 LED 灯亮，为 1 时，LED 灯灭。

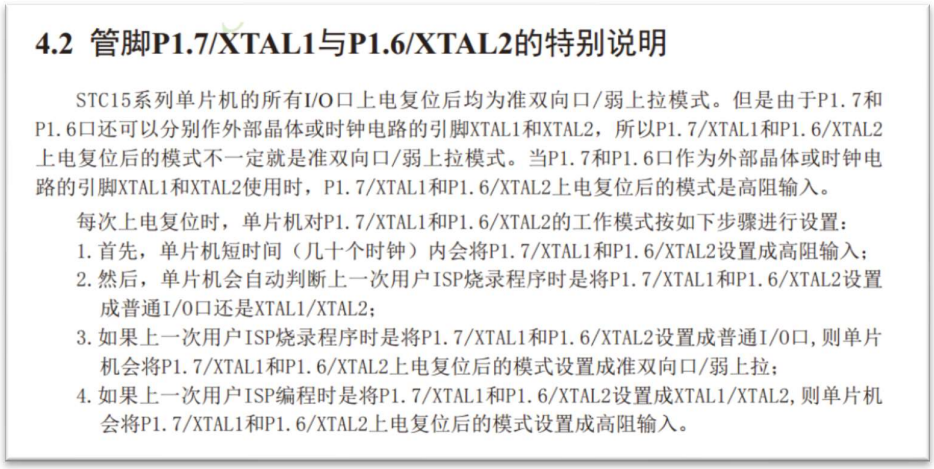
3) 将例子的 4 进制改成本题中的 13 进制，加多几个 if 判断，根据计数值设定四个引脚的高低电平，当计数值达到 13 时归 0。

4) 仿照 INT0 的写法，写 INT1 的中断服务程序以及 INT1 相关寄存器的设置。

2. 调试过程及遇到问题的解决办法

1) 驱动模式问题

开始我没有设置驱动模式，采用的是各个端口的默认模式，下载程序到板子上跑，发现只有 LED9 和 LED10 是能够亮的。后来查看了手册发现：



4.2 管脚P1.7/XTAL1与P1.6/XTAL2的特别说明

STC15系列单片机的所有I/O口上电复位后均为准双向口/弱上拉模式。但是由于P1.7和P1.6口还可以分别作外部晶体或时钟电路的引脚XTAL1和XTAL2，所以P1.7/XTAL1和P1.6/XTAL2上电复位后的模式不一定就是准双向口/弱上拉模式。当P1.7和P1.6口作为外部晶体或时钟电路的引脚XTAL1和XTAL2使用时，P1.7/XTAL1和P1.6/XTAL2上电复位后的模式是高阻输入。

每次上电复位时，单片机对P1.7/XTAL1和P1.6/XTAL2的工作模式按如下步骤进行设置：

1. 首先，单片机短时间（几十个时钟）内会将P1.7/XTAL1和P1.6/XTAL2设置成高阻输入；
2. 然后，单片机会自动判断上一次用户ISP烧录程序时是将P1.7/XTAL1和P1.6/XTAL2设置成普通I/O口还是XTAL1/XTAL2；
3. 如果上一次用户ISP烧录程序时是将P1.7/XTAL1和P1.6/XTAL2设置成普通I/O口，则单片机会将P1.7/XTAL1和P1.6/XTAL2上电复位后的模式设置成准双向口/弱上拉；
4. 如果上一次用户ISP编程时是将P1.7/XTAL1和P1.6/XTAL2设置成XTAL1/XTAL2，则单片机会将P1.7/XTAL1和P1.6/XTAL2上电复位后的模式设置成高阻输入。

于是我决定以后只用到引脚，都在程序里设置它的驱动模式，这样就不用管它的默认驱动模式是什么了。然后我有想直接设置 P1M0=0，P1M1=0，然后我同

学建议我不要这样设置，这样就把别的端口也改变了，也把其默认值也改了，这样使用单片机是不科学的。所以，我询问了同学的驱动模式设置方法，学会了用&=来给寄存器特定位 0，我通过看书上的例子，又学会了用|=来给寄存器特定位置 1。

2) 外部中断 1 的中断号是多少的问题

我开始觉得 INT0 的中断号是 0，那么 INT1 的中断号应该是 1，所以开始写的程序下载到板子上之后，按 SW18 没有反应，后来我查了一下书本前面关于中断部分，看到了 P75 的表 3.44 才知道了 INT1 指外部中断 1，中断号是 2。改了之后，程序就能在板子上正常地跑了。

3. 实验结果：成功实现所有功能。主频率为 11.0592MHz。

4. 源代码及注释：

```
#include "STC15F2K60S2.H"//此头文件已存在于 Keil_v5\C51\INC\STC 路径下，为了方便老师查看源代码的运行情况，特将此头文件放在了提交的作业的压缩包中
//因为这个头文件包含了本芯片的的很多寄存器定义，所以引用此头文件
int i=0;
service_int0() interrupt 0//这个是外部中断 0 的中断服务程序，在程序中根据计数值，确定灯的亮灭情况来实现计数值的改变。
{
    if(i==13) i=0;
    if(i==0)
    {
        P17=1;
        P16=1;
        P47=1;
        P46=1;
  }
    else if(i==1)
    {
        P17=0;
        P16=1;
        P47=1;
        P46=1;
  }
    else if(i==2)
    {
        P17=1;
        P16=0;
        P47=1;
        P46=1;
  }
```

```
  else if(i==3)
  {
        P17=0;
        P16=0;
        P47=1;
        P46=1;
}
  else if(i==4)
  {
        P17=1;
        P16=1;
        P47=0;
        P46=1;
}
  else if(i==5)
  {
        P17=0;
        P16=1;
        P47=0;
        P46=1;
}
  else if(i==6)
  {
        P17=1;
        P16=0;
        P47=0;
        P46=1;
}
  else if(i==7)
  {
        P17=0;
        P16=0;
        P47=0;
        P46=1;
}
  else if(i==8)
  {
        P17=1;
        P16=1;
        P47=1;
        P46=0;
}
  else if(i==9)
  {
```

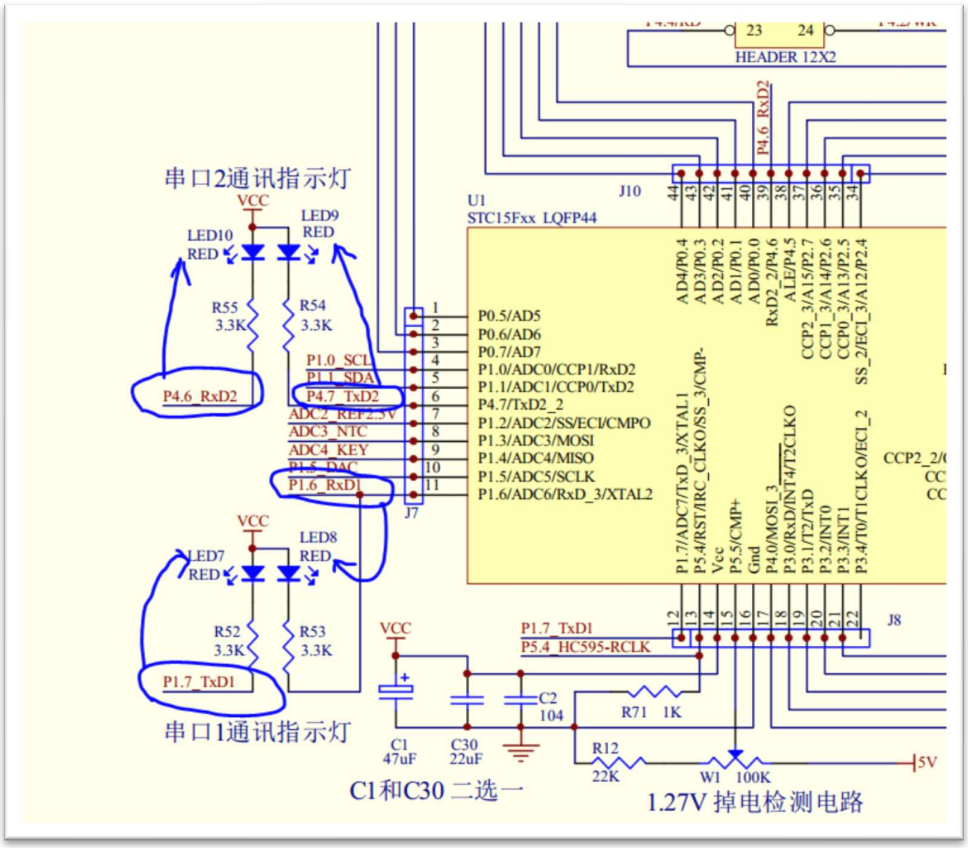
```
            P17=0;
            P16=1;
            P47=1;
            P46=0;
    }
      else if(i==10)
      {
            P17=1;
            P16=0;
            P47=1;
            P46=0;
    }
      else if(i==11)
      {
            P17=0;
            P16=0;
            P47=1;
            P46=0;
    }
      else if(i==12)
      {
            P17=1;
            P16=1;
            P47=0;
            P46=0;
    }
      else ;
      i=i+1;
}
servivce_int1() interrupt 2//这个是外部中断 1 的中断服务程序，让每一个 LED 灯
熄灭，从而对计数清零。
{
      P17=1;
      P16=1;
      P47=1;
      P46=1;
      i=1;
}
void main()
{
      P1M0&=0x3F;
      P1M1&=0x3F;
      P4M0&=0x3F;
```

P4M1&=0x3F;//通过设置 P1M0\P1M1 和 P4M0\P4M1，设置四个引脚的驱动模式为准双向弱上拉。

//查芯片原理图可知 P1.7、P1.6 和 P4.7、P4.6 分别对应 LED7\LED8\LED9\LED10

```
    //INT0=1;
    IT0=1;//只允许 INT0 下降沿触发
    IT1=1; //只允许 INT1 下降沿触发
    EX0=1;//允许外部中断 0 产生中断事件
    EX1=1; //允许外部中断 1 产生中断事件
    EA=1;//CPU 允许中断
  while(1);
}
```

原理图中 LED 灯要对应端口对应关系如下：



2）使用 C 语言，实现通过使用定时器 0 对四个 LED 灯的计数控制（50 分）

（1）定时器 0 中断，控制 LED 灯的递增计数，同上面要求（30 分）

①通信专业定时器溢出间隔为 1s；

②信工专业定时器溢出间隔为 2s。

（2）外部触发中断 INT0，用于计数器清零。（10 分）

（3）外部触发中断 INT1，用于改变计数方向，原来递增，改为递减，原来递减，改为递增（10 分）。

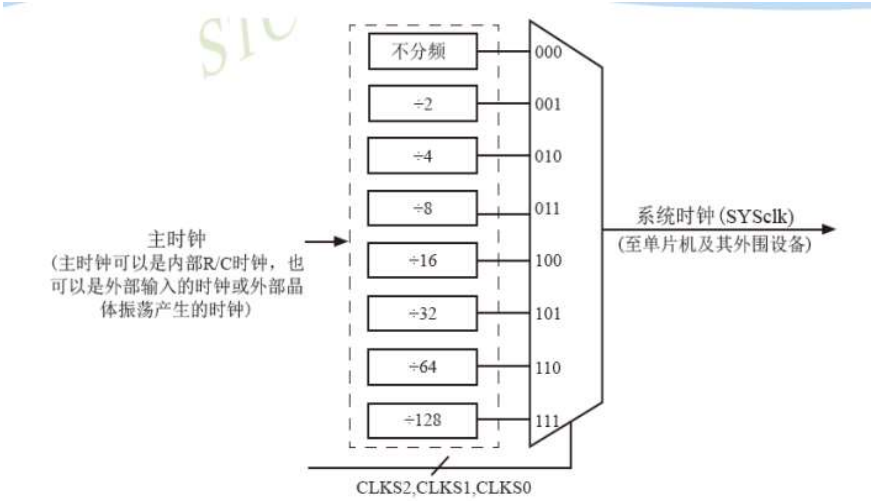
1. 设计思路

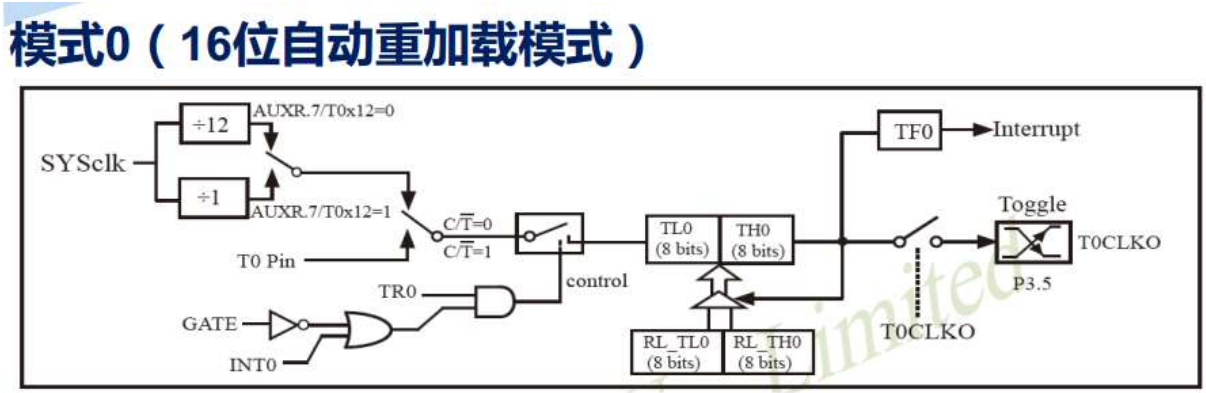
参照书本第 8 章的例子 8-1 和 8-2，以及例子 6-64。参照例子，设置了定时器 0 工作模式为 16 位自动重加载模式。

1) 把判断计数 i 值点灯的代码段写在定时器 0 的中断服务程序里。

2) 设计分频功能的寄存器位以及计数初值

首先明确定时器 0 的内部结构，以及定时器 0 溢出率的计算公式:





我设定主时钟频率为 11.0592MHz，设置 CLK_DIV 的 CLK2:CLK0=101，使得 SYSclk=主是主时钟频率/32，按照上面的结构图，设置 AUXR.7=0 再进行 12 分

频。题目要求溢出间隔为 2s，即 0.5Hz。根据计算公式[(11.0592MHz/32)/12]/(65536-TIMES)=0.5Hz，得到计数初值 TIMES=7936。

3) 首先实现第(1)题的功能

4) 然后再尝试加上(2)(3)题的功能

5) 此外，因为老师说过大段的程序尽量不要放在中断服务程序里，所以在 exp1.2 的文件夹的源文件中，我尝试着把大段代码放在主程序中，在中断服务程序中只改标志和保留少量的程序代码，结果与把大段代码放在中断服务程序里的实际效果一致。以下的源代码是基于把大段代码转移到主程序之前的源代码版本进行注释和解说。

2. 调试过程及遇到问题的解决办法

1) 开始不太清楚要怎么实现外部中断 1 改变计数方向，想把判断点灯的代码块写到主程序的循环里。之后一时间没想出来怎么办，就保留了点灯程序段在 time0 中断程序里，然后想到设置一个标志 flag，加个判断以实现在定时器 0 的中断程序的最后使 i 自增还是自减。

2) 关于 GATE、INT0 和 TR0 的设置问题
根据结构图以及下图，考虑到要避免 INT0 对定时器运行状态的影响，选择设置 GATE=0，TR0=1。这样就避免了 INT0 对定时器的影响。

GATE、INT0和TR0之间的关系

GATE	INT0	TR0	功能
0	0	0	不启动定时器
0	0	1	启动定时器
0	1	0	不启动定时器
0	1	1	启动定时器
1	0	0	不启动定时器
1	0	1	不启动定时器
1	1	0	不启动定时器
1	1	1	启动定时器

3) 我发现 INT1 无法即刻改变计数方向，总是需要先执行完 INT0 的中断服务程序，出现我按了 INT1 本来应该改变计数方向的，但是实际总要先按照原来的方向多计 1 次数，才改变方向。我希望达到即时改变计数方向的效果，于是我尝试将外部中断 1 设置为最高优先级。但是我发现这并没有改变这种现状。我认真思考了定时器 0 的中断服务程序，发现我写的程序总是先按照前一次得到的改变后的 i 值先点灯，再根据 flag 的值去改变计数值。所以即使在执行定时器 0 的中断服务程序中，嵌套的外部中断 1 打断了定时器 0 的中断服务程序，改变了 flag 的值，那也有在下一次执行定时器 0 的中断服务程序时才会生效。所以，我改变了改变计数值的程序段的位置，果然实现了即时改变计数方向。

```
if(flag==0)
    i=i+1;
```

else
i=i-1;

但是我随后发现按了 INT0 无法显示 0 的计数状态了，因为一旦 INT0 的中断服务程序给 i 清零，定时器的中断服务程序就会先给改变 i 的值，再进行显示。所以我加上了 j 变量，在 INT0 的中断服务程序中给 j 值置 1，来使得定时器 0 的中断服务程序跳过开头的改变计数值的语句，先显示 0 值，随后在定时器 0 的中断服务程序末尾又让 j 值恢复正常的 0 值。这些标志位 flag、j 等类似开关一样控制着程序走向不同的分支。

4) 关于代码优化：此外在原程序可以运行正常的基础上，我尝试把定时器 0 的中断服务程序的点灯部分程序移到了主程序里的 while 循环里，程序依然可以正常运行。

3. 实验结果：成功实现所有功能。主时钟频率为 11.0592MHz。

4. 源代码及其注释

```
#include "STC15F2K60S2.H"
#define TIMS 7936//声明定时器 0 的计数初值
//IRC=11.0592MHz
sfr AUXR2=0x8F;//因为编译报错 AUXR2 未定义，所以我手动添加了这个寄存器的定义

int i=0;
int flag=0;
int j;

void timer_0() interrupt 1//定时器 0 实现自动计数，根据标志变量 flag 的值来确定计数值 i 是递增还是递减，再根据计数值 i 来确定 4 盏 LED 灯的亮灭情况。
{
        if(j==0)
        {
        if(flag==0)//由外部中断 1 的服务程序改变 flag，以实现递增或递减计数
            i=i+1;
        else
            i=i-1;
        }
        if(i==13) i=0;
        if(i<0) i=12;
        if(i==0)
        {
            P17=1;
            P16=1;
            P47=1;
            P46=1;
        }
        else if(i==1)
        {
            P17=0;
```

```
        P16=1;
        P47=1;
        P46=1;
    }
    else if(i==2)
    {
        P17=1;
        P16=0;
        P47=1;
        P46=1;
    }
    else if(i==3)
    {
        P17=0;
        P16=0;
        P47=1;
        P46=1;
    }
    else if(i==4)
    {
        P17=1;
        P16=1;
        P47=0;
        P46=1;
    }
    else if(i==5)
    {
        P17=0;
        P16=1;
        P47=0;
        P46=1;
    }
    else if(i==6)
    {
        P17=1;
        P16=0;
        P47=0;
        P46=1;
    }
    else if(i==7)
    {
        P17=0;
        P16=0;
        P47=0;
```

```
            P46=1;
        }
        else if(i==8)
        {
            P17=1;
            P16=1;
            P47=1;
            P46=0;
        }
        else if(i==9)
        {
            P17=0;
            P16=1;
            P47=1;
            P46=0;
        }
        else if(i==10)
        {
            P17=1;
            P16=0;
            P47=1;
            P46=0;
        }
        else if(i==11)
        {
            P17=0;
            P16=0;
            P47=1;
            P46=0;
        }
        else if(i==12)
        {
            P17=1;
            P16=1;
            P47=0;
            P46=0;
        }
        else ;
        j=0;
}

servivce_int0() interrupt 0//外部中断 1 的服务程序实现计数清 0
{
    j=1;//使得定时器 0 的中断服务程序跳过开头的改变计数值的语句，先显示 0 值
```

```
    i=0;
}
servivce_int1() interrupt 2//外部中断 1 的服务程序，通过改变标志变量 flag 来控制计数方向。
{
    flag=!flag;
}
main()
{
    P1M0&=0x3F;
    P1M1&=0x3F;
    P4M0&=0x3F;
    P4M1&=0x3F;//配置引脚驱动模式为准双向

    CLK_DIV=0x05;//对主时钟进行 32 分频
    TL0=TIMS;
    TH0=TIMS>>8;//装入计数初值
    AUXR&=0x7F;//最高位置 0，SYSclk/12 作定时器时钟
    AUXR2&=0xFE;//禁止 P3.5 输出 T0CLK0
    TMOD=0x00;//设置 GATE=0，定时器 0 模式 0（16 位自动重加载模式）

    P17=1;
    P16=1;
    P47=1;
    P46=1;

    TR0=1; //设置 TR0=1，启动定时器 0
    ET0=1;//允许定时器 0 中断

    IT0=1;//只允许 INT0 下降沿触发
    IT1=1; //只允许 INT1 下降沿触发
    EX0=1;//允许外部中断 0
    EX1=1; //允许外部中断 1

    EA=1;//CPU 允许中断
    while(1);
}
```

3）发挥部分（10 分）

基于上面的设计要求，使用 C 语言完成，主要考察学生创新设计的能力。

1. **实验目标**：**会呼吸的交通灯**
2. **详细阐述**：模仿交通灯，红灯和绿灯依次亮灭，会呼吸的黄灯在绿灯准备灭的时候开始呼吸，然后呼吸结束，红灯亮起。计数值采取倒计时，在数码管上显示。开始时为

只有右边的灯（绿灯）亮，当右边的灯准备灭的时候，中间那盏灯（黄灯）进入呼吸模式，然后右边的灯灭，左边的灯（红灯）亮起，经过一定的时间后，右边的灯再次亮起，其他的灯灭，开始新的一轮循环。

3. **设计思路**

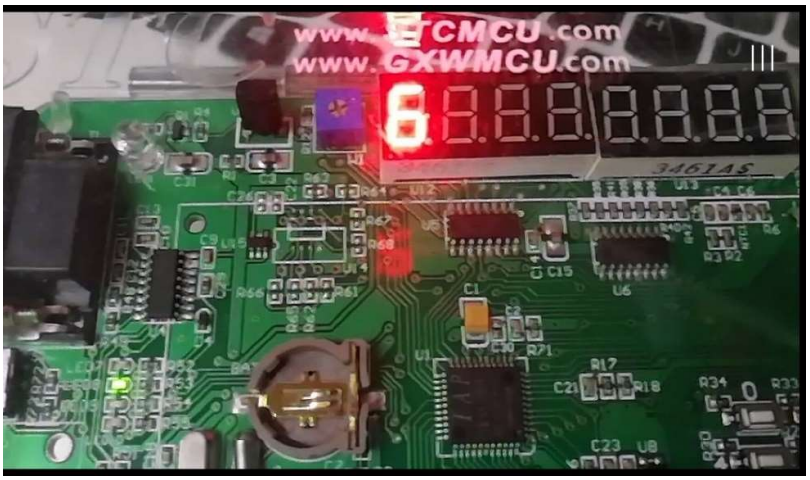
这个程序是我写完了三份实验题中的其他所有的题之后，最后写的程序，因为我想综合一下自己写过的作业代码，有更好的创意。因为老师说过，学了单片机，就应该会写红绿黄交通灯，所以我就想尝试写一个交通灯程序，同时我又想结合自己写的这 3 个实验的代码，所以我就想把黄灯的闪烁改成中间的灯的呼吸，把呼吸灯的代码揉进交通灯程序里，于是就有了现在这个“会呼吸的交通灯”创意想法。

4. **调试过程及遇到问题的解决办法**

开始想这个 idea 的时候，因为不太记得交通灯的运行流程了，所以就按照一个大概的模糊印象以及自己定下的所要实现的效果，作为程序代码的实现目标。

因为之前写的代码奠定的一个比较好的逻辑思考方式习惯，所以在本程序中，除了开始的实验最终效果的确定外，别的没有太多的纠结和调试。

5. **实验结果**：成功实现上述功能。



6. **源代码及注释**

```
#include "STC15F2K60S2.H"
#include "spi.h"
#define TIMS0 62080// 定时器 0 计数初值
#define TIMS1 7936//定时器 1 计数初值
//11,059,200Hz,clk_div=3->8div,TIMER0 0div
//t_display 数组保存着 0~9，A~F 的段码，顺序 h,g,f,e,d,c,b,a
unsigned char code t_display[16]={0x3F,0x06,0x5B,0x4F,
                                  0x66,0x6D,0x7D,0x07,
                                  0x7F,0x6F,0x77,0x7C,
                                  0x39,0x5E,0x79,0x71};
//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};

int uN=0;
```

```
//设置初始的占空比为 0%，其中 uN=0、1、2~9 分别对应占空比为 0%、10%、20%~90%
int nCount=0;// 设置初始的计数值为 0
int dir=0;//设置初始的呼吸方向是由亮变暗
int start=1;//这是一个开关变量，用来控制呼吸方向改变的时刻和条件
unsigned char count=7;//初始化数码管显示的计数值
unsigned char led=0;//初始化灯 P1.6 和 P4.6 的状态选择

    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
    }

void timer_1() interrupt 3//定时器 1 中断服务程序，使得数码管呈现从 6 开始的递减计数
{
    count--;//计数值递减计数
    seg7scan(0,count);//显示计数值
    if(count==0) //当计数值减到 0 时
    {
        count=7;//重新装载计数值为 7
        led=!led;//状态变量 led 取反
    }
}

void timer_0() interrupt 1//定时器 0 中断服务程序改变变量 nCount 的值。
{
    if(count>3)//如果计数值大于 3
        P47=1; //黄灯（中间那盏灯，即 P4.7）保持灭的状态
    if(led==0&&count<=3)//如果当前是绿灯（P1.6）亮，且还有 3 秒就快要灭了
```

```
        {
                if(dir==0)//判断当前的呼吸方向，是由暗变亮（占空比由小变大）还是由亮变暗（占空比由大变小），dir=0 时，占空比由大变小
        {
                P47=( nCount<uN )?0:1;
               //当前计数值小于占空比的时候引脚电平为 0，否则为 1
                nCount++; nCount=nCount%10;//计数值 nCount 的变化范围为 0~9
                if(nCount==0)
                {
                    uN++;//当计数值恢复到 0 时，占空比增加 10%
                    uN=uN%10;//占空比 uN 的变化范围为 0~9
                    start=0;//从第 1 轮 nCount 循环结束起，start 开关变量开启，允许等待符合条件时改变呼吸的方向
                }
      }
        else// dir=1 时，占空比由小变大
        {
                P47=( nCount<uN )?1:0;
               //当前计数值小于占空比的时候引脚电平为 1，否则为 0
                nCount++; nCount=nCount%10;
                if(nCount==0)
                {
                    uN++;
                    uN=uN%10;
                    start=0; //从第 1 轮 nCount 循环结束起，start 开关变量开启，允许等待符合条件时改变呼吸的方向
                 }
      }
        if(start!=1&&uN==0)
        {
            dir=!dir;//改变呼吸的方向
            start=1; //start 开关变量关闭，禁止在第 1 轮 nCount 计数循环期间改变呼吸的方向，因为此期间的 uN=0，但是却不希望呼吸方向改变。
        }
    }
}

void main()
{
      SPCTL=(SSIG<<7)+(SPEN<<6)+(DORD<<5)+(MSTR<<4)
            +(CPOL<<3)+(CPHA<<2)+SPEED_4; //给寄存器 SPCTL 赋值
      P1M0&=0xBF;
      P1M1&=0xBF; //配置 P1.6 引脚驱动模式为准双向
      P4M0&=0x3F;
```

```
    P4M1&=0x3F; //配置 P4.6\P4.7 引脚驱动模式为准双向

    P16=1;
    P47=1;
    P46=1;//3 盏灯的起始状态为灭

    CLK_DIV=0x03;//主时钟 8 分频作为 SYSclk 频率
    TL0=TIMS0;
    TH0=TIMS0>>8; //装入计数初值
    TL1=TIMS1;
    TH1=TIMS1>8; //装入计数初值

    AUXR&=0xBF;//定时器 1 为 12 分频
    AUXR|=0x80; //定时器 0 不分频
    TMOD=0x00;//定时器 0/1 均为 16 位重加载模式
    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
    EA=1;//CPU 允许响应中断请求
    seg7scan(0,count);//显示初始计数值 0
    while(1)
    {
        if(led==0)//如果 led=0，P1.6（LED8）亮，P4.6（LED10）灭
        {
            P16=0;
            P46=1;
        }
        else if(led==1) //如果 led=1，P1.6（LED8）灭，P4.6（LED10）亮
        {
            P16=1;
            P46=0;
        }
        else ;
    }
}
```