西安交通大學

XI'AN JIAOTONG UNIVERSITY

电子系统设计

实验报告

班级：xxxx

姓名：xxxx

学号：xxxxxxxxx

流水灯设计

一、实验目的

1.了解单片机的结构及工作原理。
2.掌握按键检测的原理并能实际应用
3.掌握数码管的显示方法。

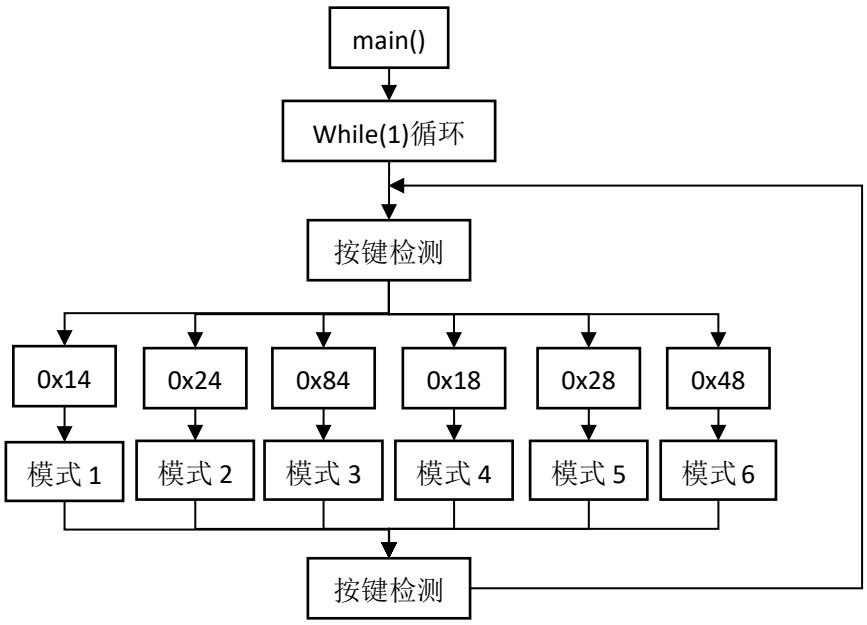
二、实验要求

1.按键控制不低于 6 种模式流水灯显示。
2.需包含“循环左移，循环右移、数字递增、按键改变数字”等四种模式。

三、系统方案设计

1.总体方案设计

1）系统总体框图



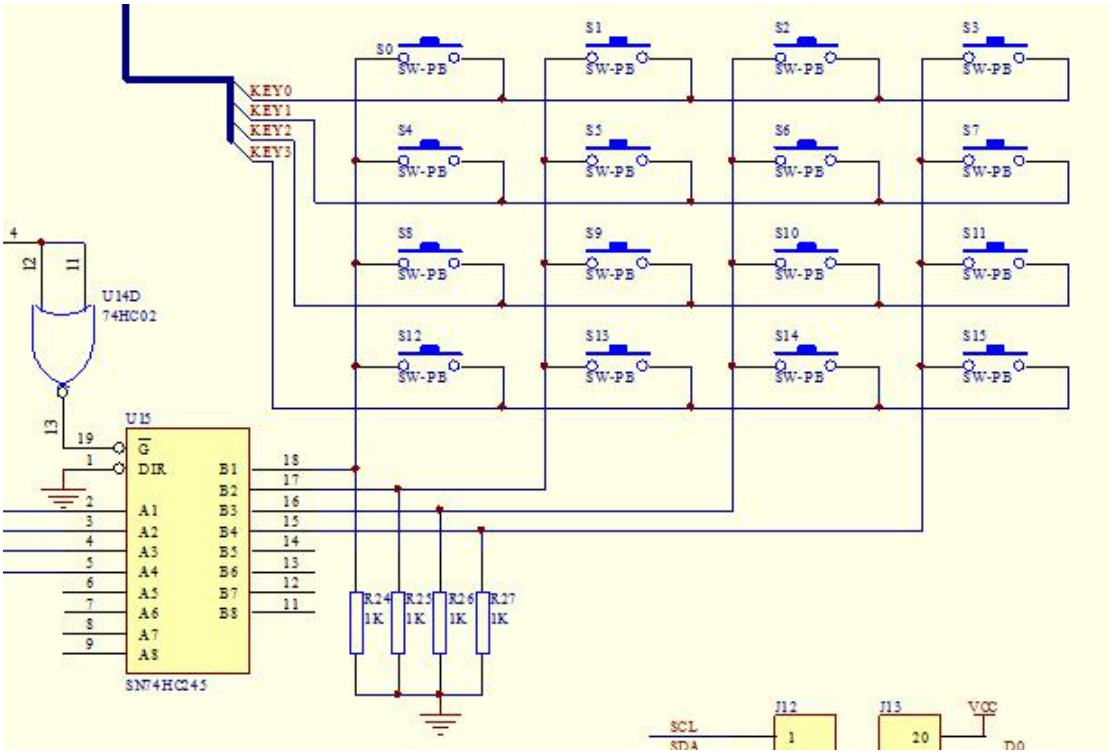
2）系统设计思路

将六种流水灯模式分别作为六个子函数，在主函数中进行调用。在主函数中利用 while(1)语句实现实时按键检测。另外，利用 switch 语句实现按键控制，当没有按键按下或者有无效按键按下时，继续循环检测；当有效按键按下时，则执行该按键对应的 case 语句，以此实现不同的流水灯显示模式。

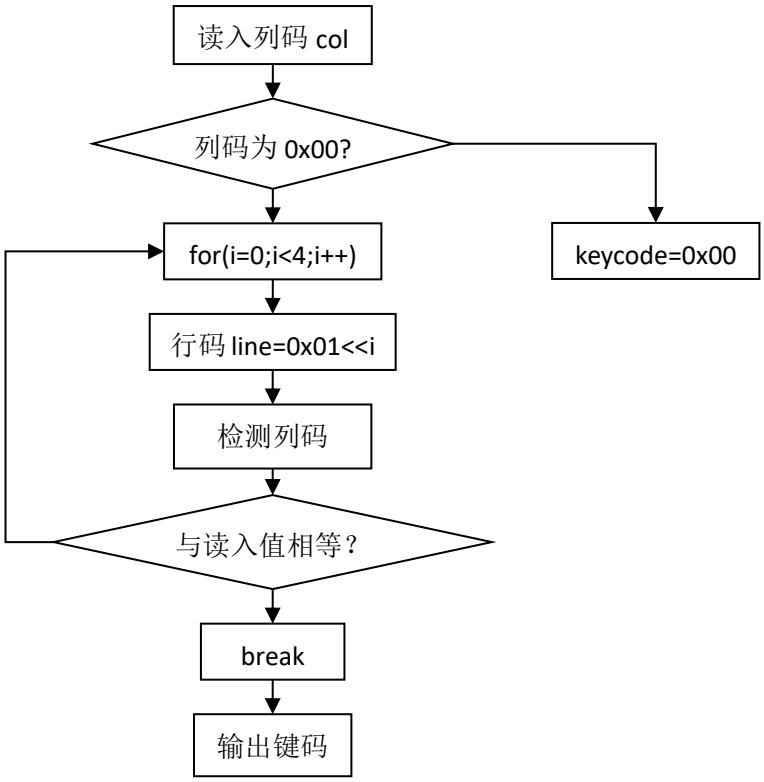
2. 各模块方案设计

1）按键检测模块

按键检测模块的电路图如下图所示，其中每列均下拉至地，故当无按键按下时，列码应为 0x00。当有按键按下时，为检测按下的是哪一个按键，可先给第一行赋值为高电平，再检测列码，若列码不为 0x00，则说明第一行有按键按下。此时，只需根据列码判断按下的是哪一列按键。从第一行到第四行重复这个过程，即可检测按键的具体位置。



程序流程图如下：



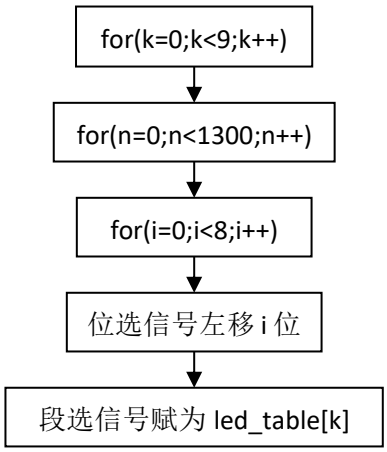
2）流水灯显示模块

本设计中六种流水灯显示模块分别为：循环左移、循环右移、数字递增、左右移动闪烁、亮灭间隔闪烁、三次快速闪烁。下面分别就每种模式的显示原理进行分析。

循环左移：将七段数码表存储在数组 led_table 中，通过循环控制位选信号从右向左移动。位选信号移动到某一个数码管处时，同时接通段选信号并延时一段时间，以获得较长的显示时长。这样，即可实现数字从右向左的移动。此外，循环中还加入了数字设置功能，即通过检测按键来改变显示的数字。

循环右移：循环右移与循环左移的原理相同，只需把位选信号移动的方向改为从左向右即可。

数字递增：采用动态扫描来实现八个数码管同时显示不同的数，利用循环来实现数字递增，原理图如下。其中外层的 k 循环用于控制数字递增，第二层的 n 循环用于延时，内层的 i 循环用于控制位选信号，实现动态扫描。



左右移动闪烁：左右移动闪烁模式为，开始数字“8”在 0x01,0x04,0x10,0x40 位进行短暂的显示，接着这四位熄灭，数字“8”在 0x02,0x08,0x20,0x80 位上显示，循环进行上述过程。此模式与循环左移相似，只需对位选信号进行调整即可，用到了动态扫描的原理。

亮灭间隔闪烁：亮灭间隔闪烁模式为，数字在 0x01,0x04,0x10,0x40 位上进行亮、灭间隔的显示。此模式与左右移动闪烁相似，在控制数字熄灭时，只需将位选信号或段选信号置零，再进行适当延时即可。

三次快速闪烁：此种闪烁模式为，同一数字在 0x01,0x02,0x04,0x08 位上连续快速闪烁三次，然后熄灭。与前几种闪烁模式相似，在此不再赘述。

四、资源使用情况

资源使用情况为：

data=17.0

xdata=0

code=1285

说明占用的内部 RAM 为 17 字节，外部 RAM 为 0 字节，占用的程序存储器空间为 1285 字节。

附录

完整代码：

```
#include <REG51.H>                  //特殊功能寄存器
#include <absacc.h>
#include <stdio.h>

#define uchar unsigned char

Unsigned char code led_table[]=
{0x3f,0x06,0x5b,0x4f,0x66,0x6d,0x7d,0x07,0x7f,0x6f,0x00,0x08,0x40,0x79};
//七段数码表
void delay(unsigned char p);
unsigned char getkeycode();
void leftcircle();
void rightcircle();
void numadd();
void move();
void twinkle1();
void twinkle2();

/*********************************主函数*********************************/
void main (void)
{
  while (1)
  {
   XBYTE[0x9000]= 0x00;
   switch(getkeycode())
   {
      case 0x14:leftcircle();break;   //第三行第 1 列
      case 0x24:rightcircle();break;  //第三行第 2 列
      case 0x44:break;                //第三行第 3 列
      case 0x84:numadd();break;       //第三行第 4 列

      case 0x18:move();break;         //第四行第 1 列
      case 0x28:twinkle1();break;     //第四行第 2 列
      case 0x48:twinkle2();break;     //第四行第 3 列
      case 0x88:break;                //第四行第 4 列
      default: break;
    }
  }
}

/*********************************延时函数*********************************/
```

```
void delay(unsigned char  p)                //延时函数
{
    unsigned char  i,j;
    for(;p>0;p--)
        for(i=181;i>0;i--)
            for(j=181;j>0;j--);
}

/*******************************按 键 扫 描 *******************************/

 unsigned char getkeycode()                 //键盘扫描函数，返回获得键码
{
    unsigned char line=0x00;                //行码
    unsigned char col=0x00;                 //列码
    unsigned char scancode=0x01;            //行扫描码
    unsigned char keycode;                  //键号

    XBYTE[0x8000]=0xff;
    col=XBYTE[0x8000]&0x0f;                 //从列端口读入四位列码
    if (col==0x00)
        keycode=0x00;
    else
    {
        while((scancode&0x0f)!=0)           //取 scancode 的低四位，没变为全 0，
循环
        {
            line=scancode;                  //行号
            XBYTE[0x8000]=scancode;         //给行赋扫描码，第一行为 0x01
            if((XBYTE[0x8000]&0x0f)==col)   //检测按键所在的行跳出循环
                break;
            scancode=scancode<<1;           //行扫描码左移一位，转下一行
        }
        col=col<<4;                         //把列码移到高四位
        keycode=col|line;
    }
    return keycode;
}

/*******************************六种流水灯模式*******************************/
void leftcircle(void)               //循环左移
{
   int i,n=0;
   int k=0;
   while(1)
```

```
    {
       XBYTE[0x9000]= 0x00;
       for(i=0;i<8;i++)
       {
        switch(getkeycode())
        {
           case 0x11:k=1;break;     //第一行第 1 列
           case 0x21:k=2;break;     //第一行第 2 列
           case 0x41:k=3;break;     //第一行第 3 列
           case 0x81:k=4;;break;    //第一行第 4 列

           case 0x12:k=5;break;     //第二行第 1 列
           case 0x22:k=6;break;     //第二行第 2 列
           case 0x42:k=7;break;     //第二行第 3 列
           case 0x82:k=8;break;     //第二行第 4 列

           default: break;
        }
         XBYTE[0x8000]=  0x01<<i;
         XBYTE[0x9000]= led_table[k];
         delay(5);
       }
       if(getkeycode()&0x0c)
           break;
       }
}

void rightcircle(void)              //循环右移
{
   int i,n=0;
   int k=0;
   while(1)
   {
//    XBYTE[0x9000]= 0x00;
       for(i=0;i<8;i++)
       {
        switch(getkeycode())
        {
           case 0x11:k=1;break;      //第一行第 1 列
           case 0x21:k=2;break;      //第一行第 2 列
           case 0x41:k=3;break;      //第一行第 3 列
           case 0x81:k=4;;break;     //第一行第 4 列

           case 0x12:k=5;break;      //第二行第 1 列
```

```
          case 0x22:k=6;break;      //第二行第 2 列
          case 0x42:k=7;break;      //第二行第 3 列
          case 0x82:k=8;break;      //第二行第 4 列

          default: break;
       }
        XBYTE[0x8000]=  0x80>>i;
        XBYTE[0x9000]= led_table[k];
        delay(5);
      }
      if(getkeycode()&0x0c)
          break;
    }
}

void numadd(void)                   //数字递增
{
   while(1)
   {
   int i,n;
   int k;
   for(k=0;k<9;k++)
           for(n=0;n<1300;n++)
                    for (i=0;i<8;i++)
                    {
                         XBYTE[0x8000]= 0x01<<i;
                         XBYTE[0x9000]= led_table[k];
                    }
    if(getkeycode()&0x0c)
        break;

    }
}


void move(void)                     //左右移动模式
{
   while(1)
   {
   int i,n;
       for(n=0;n<1300;n++)
            for (i=0;i<4;i++)
            {
                 XBYTE[0x8000]= 0x01<<2*i;
```

```
                XBYTE[0x9000]= led_table[8];
            }
        for(n=0;n<1300;n++)
            for (i=0;i<4;i++)
            {
                XBYTE[0x8000]= 0x02<<2*i;
                XBYTE[0x9000]= led_table[8];
            }
    if(getkeycode()&0x0c)
        break;
    }
}

void twinkle1(void)                  //闪烁模式1
{
   int k=0;
   while(1)
   {
   int i,n;
   switch(getkeycode()){
      case 0x11:k=1;break;           //第一行第1列
      case 0x21:k=2;break;           //第一行第2列
      case 0x41:k=3;break;           //第一行第3列
      case 0x81:k=4;;break;          //第一行第4列

      case 0x12:k=5;break;           //第二行第1列
      case 0x22:k=6;break;           //第二行第2列
      case 0x42:k=7;break;           //第二行第3列
      case 0x82:k=8;break;           //第二行第4列

      default: break;
       }
       for(n=0;n<1800;n++)
            for (i=0;i<4;i++)
            {
                XBYTE[0x8000]= 0x01<<2*i;
                XBYTE[0x9000]= led_table[k];
            }
        for(n=0;n<1800;n++)
            for (i=0;i<4;i++)
            {
                XBYTE[0x8000]= 0x00;
                XBYTE[0x9000]= led_table[k];
            }
```

```
    if(getkeycode()&0x0c)
        break;

    }
}

void twinkle2(void)                    //闪烁模式2
{
   int k=0;
   while(1)
   {
   int i,j,n;
   switch(getkeycode()){
      case 0x11:k=1;break;             //第一行第1列
      case 0x21:k=2;break;             //第一行第2列
      case 0x41:k=3;break;             //第一行第3列
      case 0x81:k=4;;break;            //第一行第4列

      case 0x12:k=5;break;             //第二行第1列
      case 0x22:k=6;break;             //第二行第2列
      case 0x42:k=7;break;             //第二行第3列
      case 0x82:k=8;break;             //第二行第4列

      default: break;
       }
       for(j=0;j<3;j++)
       {
       for(n=0;n<500;n++)
            for (i=0;i<4;i++)
            {
                 XBYTE[0x8000]= 0x01<<i;
                 XBYTE[0x9000]= led_table[k];
            }
       for(n=0;n<500;n++)
            for (i=0;i<4;i++)
                 XBYTE[0x8000]= 0x00;
       }
       delay(15);
    if(getkeycode()&0x0c)
        break;

    }
}
```