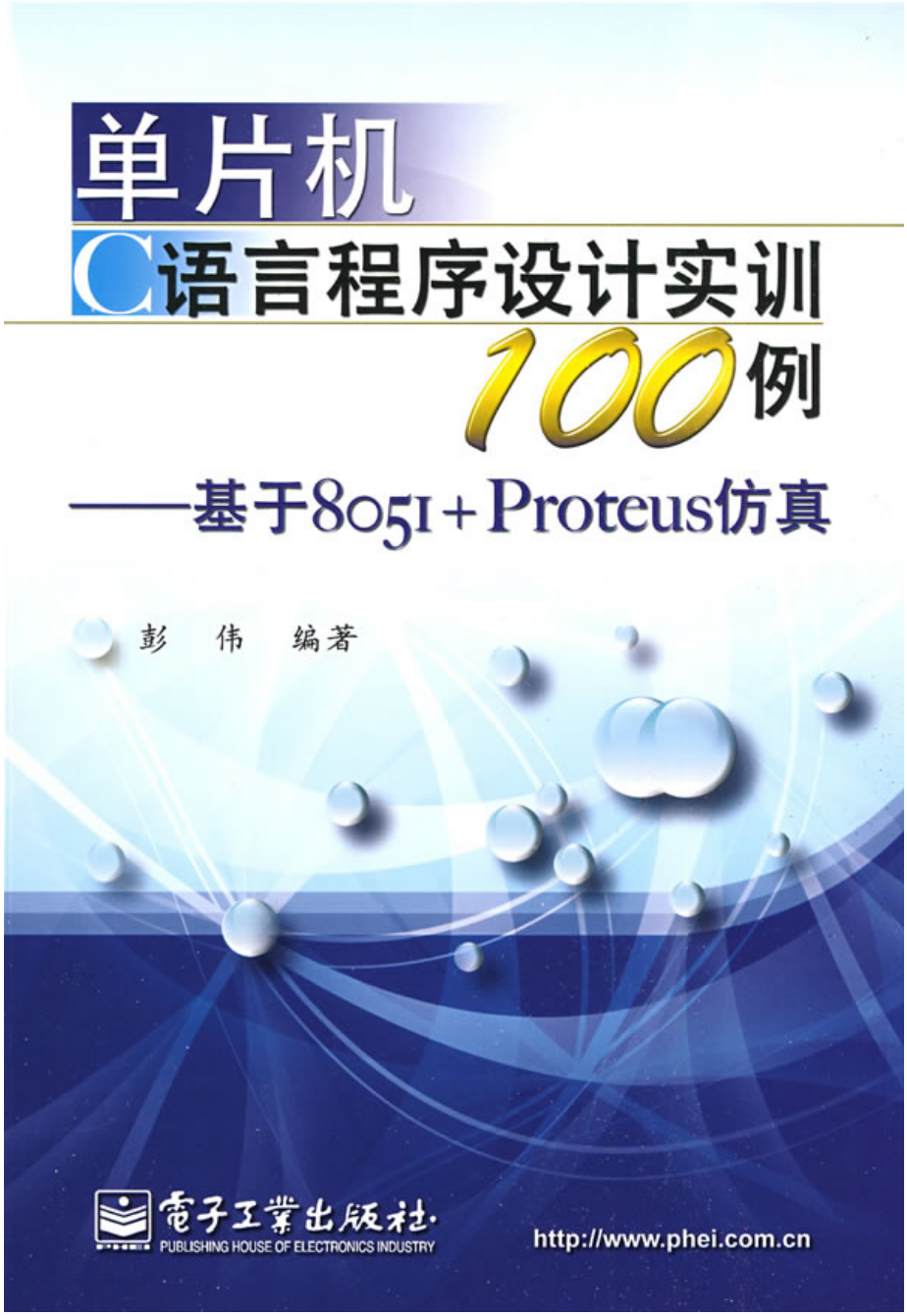


本书简介

本书基于Keil μVision程序设计平台和Proteus硬件仿真平台，精心编写了100余个8051单片机C语言程序设计案例。

全书基础设计类案例贯穿8051单片机最基本的端口编程、定时／计数器应用、中断和串口通信；硬件应用类案例涵盖常用外部存储器扩展、接口扩展、译码、编码、驱动、光机、机电、A／D与D／A转换等内容；综合设计类案例涉及大量消费类电子产品、仪器仪表及智能控制设备的相关技术。

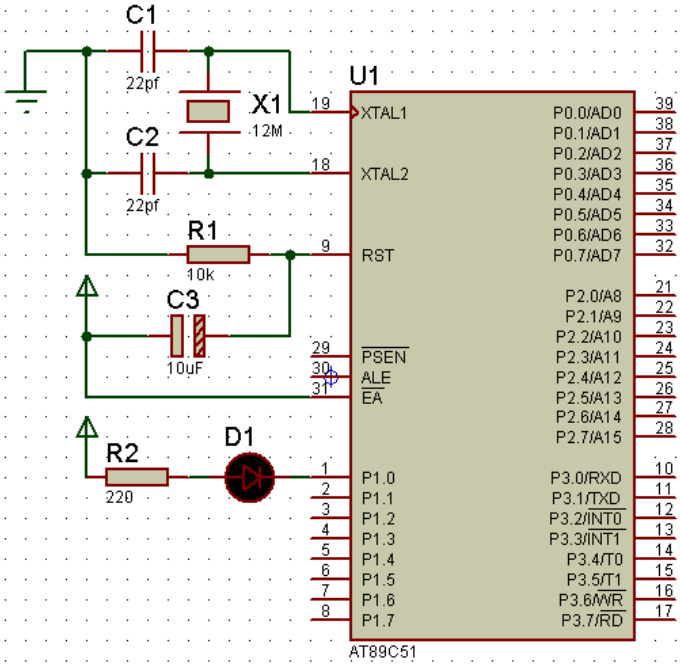
本书可作为大专院校学生学习实践单片机c语言程序设计的教材或参考书，也可作为电子工程技术人员或单片机技术爱好者的参考资料。

目录

《单片机C语言程序设计实训 100 例—基于 8051+Proteus仿真》案例

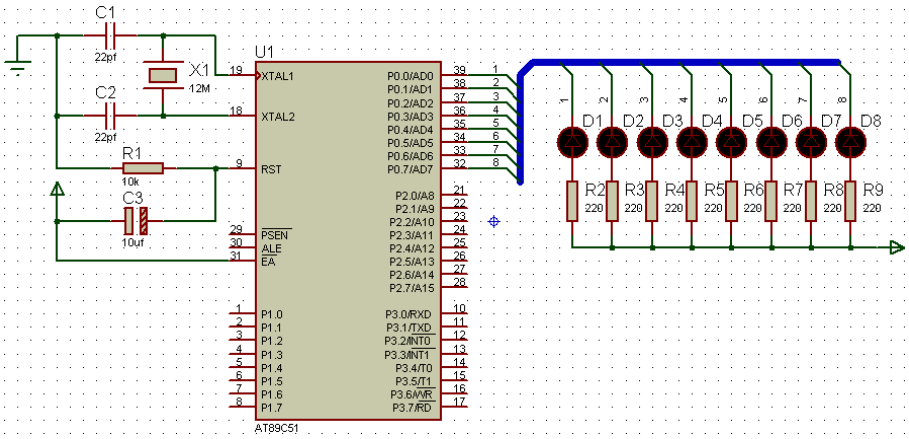
第 01 篇 基础程序设计

01 闪烁的 LED




```
/*    名称：闪烁的 LED
      说明：LED 按设定的时间间隔闪烁
*/
#include<reg51.h>
#define uchar unsigned char
#define uint unsigned int
sbit LED=P1^0;
//延时
void DelayMS(uint x)
{
      uchar i;
      while(x--)
      {
            for(i=0;i<120;i++);
      }
}
//主程序
void main()
{
      while(1)
      {
            LED=~LED;
            DelayMS(150);
      }
}
```

02 从左到右的流水灯

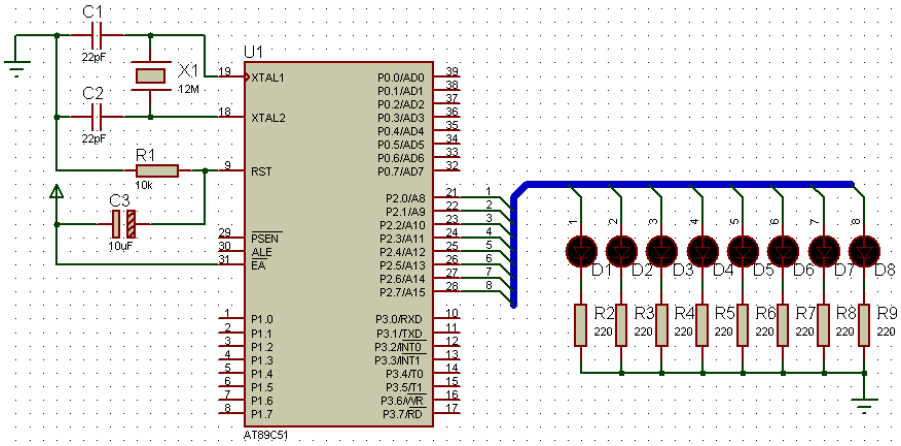



```
/*    名称：从左到右的流水灯
      说明：接在 P0 口的 8 个 LED 从左到右循环依次点亮，产生走马灯效果
*/
#include<reg51.h>
#include<intrins.h>
#define uchar unsigned char
#define uint unsigned int
```

```
//延时
void DelayMS(uint x)
{
    uchar i;
    while(x--)
    {
        for(i=0;i<120;i++);
    }
}
//主程序
void main()
{
    P0=0xfe;
    while(1)
    {
        P0=_crol_(P0,1); //P0 的值向左循环移动
        DelayMS(150);
    }
}
```

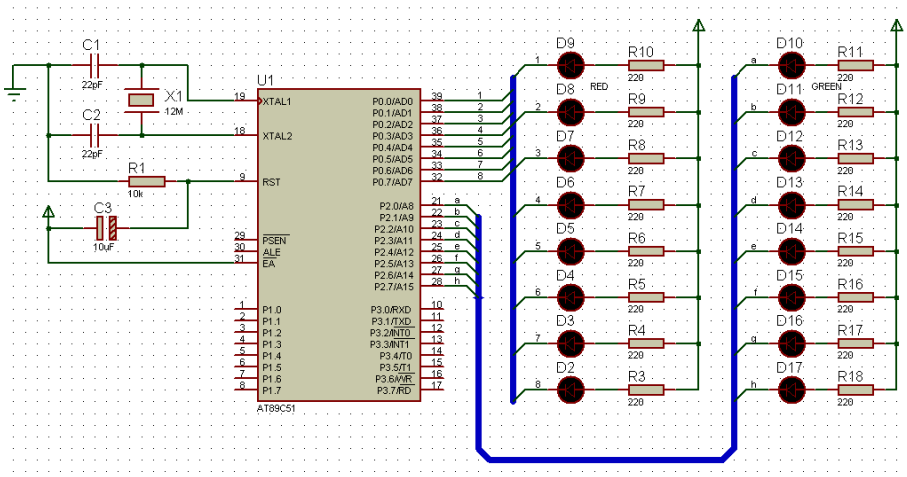
03　8 只 LED 左右来回点亮

```
/*  名称：8 只 LED 左右来回点亮
    说明：程序利用循环移位函数_crol_和_cror_形成来回滚动的效果
*/
#include<reg51.h>
#include<intrins.h>
#define uchar unsigned char
#define uint unsigned int
//延时
void DelayMS(uint x)
{
    uchar i;
    while(x--)
    {
        for(i=0;i<120;i++);
    }
}
//主程序
void main()
{
    uchar i;
    P2=0x01;
    while(1)
    {
```



```
        for(i=0;i<7;i++)
        {
            P2=_crol_(P2,1); //P2 的值向左循环移动
            DelayMS(150);
        }
        for(i=0;i<7;i++)
        {
            P2=_cror_(P2,1); //P2 的值向右循环移动
            DelayMS(150);
        }
    }
}
```

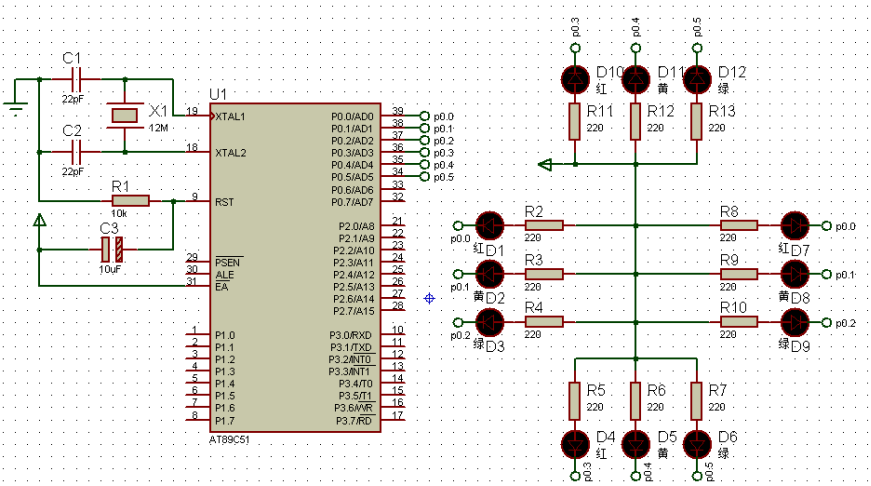
04　花样流水灯




```
/*    名称：花样流水灯
      说明：16 只 LED 分两组
按预设的多种花样变换显示
*/
#include<reg51.h>
#define uchar unsigned char
#define uint unsigned int
uchar code Pattern_P0[]=
{
 0xfc,0xf9,0xf3,0xe7,0xcf,0x9f,0x3f,0x7f,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,
  0xe7,0xdb,0xbd,0x7e,0xbd,0xdb,0xe7,0xff,0xe7,0xc3,0x81,0x00,0x81,0xc3,0xe7,0xff,
  0xaa,0x55,0x18,0xff,0xf0,0x0f,0x00,0xff,0xf8,0xf1,0xe3,0xc7,0x8f,0x1f,0x3f,0x7f,
  0x7f,0x3f,0x1f,0x8f,0xc7,0xe3,0xf1,0xf8,0xff,0x00,0x00,0xff,0xff,0x0f,0xf0,0xff,
  0xfe,0xfd,0xfb,0xf7,0xef,0xdf,0xbf,0x7f,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,
  0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0x7f,0xbf,0xdf,0xef,0xf7,0xfb,0xfd,0xfe,
  0xfe,0xfc,0xf8,0xf0,0xe0,0xc0,0x80,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
  0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x80,0xc0,0xe0,0xf0,0xf8,0xfc,0xfe,
  0x00,0xff,0x00,0xff,0x00,0xff,0x00,0xff
};
uchar code Pattern_P2[]=
{
  0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xfe,0xfc,0xf9,0xf3,0xe7,0xcf,0x9f,0x3f,0xff,
  0xe7,0xdb,0xbd,0x7e,0xbd,0xdb,0xe7,0xff,0xe7,0xc3,0x81,0x00,0x81,0xc3,0xe7,0xff,
  0xaa,0x55,0x18,0xff,0xf0,0x0f,0x00,0xff,0xf8,0xf1,0xe3,0xc7,0x8f,0x1f,0x3f,0x7f,
  0x7f,0x3f,0x1f,0x8f,0xc7,0xe3,0xf1,0xf8,0xff,0x00,0x00,0xff,0xff,0x0f,0xf0,0xff,
  0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xfe,0xfd,0xfb,0xf7,0xef,0xdf,0xbf,0x7f,
  0x7f,0xbf,0xdf,0xef,0xf7,0xfb,0xfd,0xfe,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,
  0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xfe,0xfc,0xf8,0xf0,0xe0,0xc0,0x80,0x00,
  0x00,0x80,0xc0,0xe0,0xf0,0xf8,0xfc,0xfe,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,
```

```
 0x00,0xff,0x00,0xff,0x00,0xff,0x00,0xff
};
//延时
void DelayMS(uint x)
{
    uchar i;
    while(x--)
    {
        for(i=0;i<120;i++);
    }
}
//主程序
void main()
{
    uchar i;
    while(1)
    {   //从数组中读取数据送至 P0 和 P2 口显示
        for(i=0;i<136;i++)
        {
            P0=Pattern_P0[i];
            P2=Pattern_P2[i];
            DelayMS(100);
        }
    }
}
```

05 LED 模拟交通灯



/* 名称：LED 模拟交通灯

说明：东西向绿灯亮若干秒，黄灯闪烁 5 次后红灯亮， 红灯亮后，南北向由红灯变为绿灯，若干秒后南北向黄灯闪烁 5 此后变红灯，东西向变绿灯，如此重复。

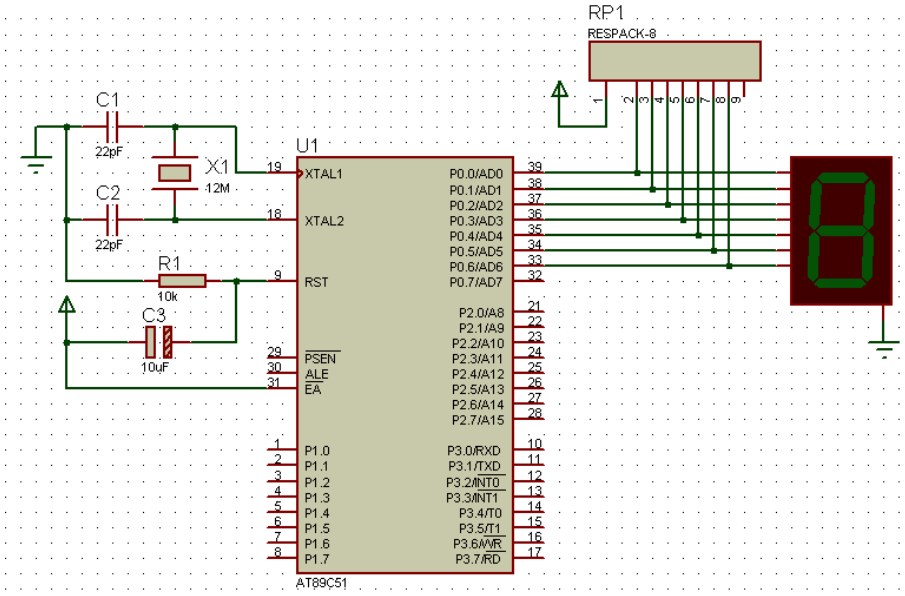
*/

```
#include<reg51.h>
#define uchar unsigned char
#define uint unsigned int
sbit RED_A=P0^0;      //东西向灯
sbit YELLOW_A=P0^1;
sbit GREEN_A=P0^2;
sbit RED_B=P0^3;      //南北向灯
sbit YELLOW_B=P0^4;
sbit GREEN_B=P0^5;
uchar Flash_Count=0,Operation_Type=1; //闪烁次数，操作类型变量
```

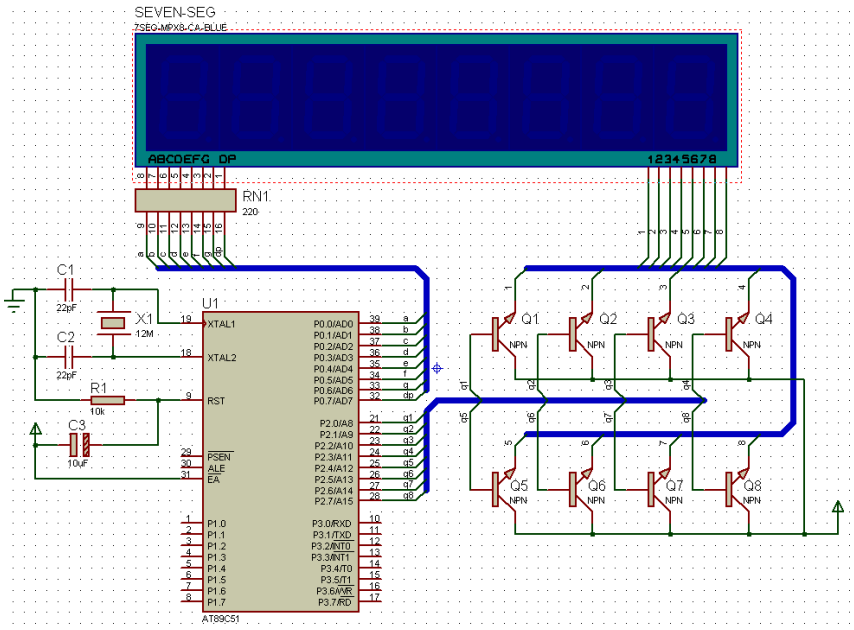
```
//延时
void DelayMS(uint x)
{
    uchar i;
    while(x--) for(i=0;i<120;i++);
}
//交通灯切换
void Traffic_Light()
{
    switch(Operation_Type)
    {
        case 1:  //东西向绿灯与南北向红灯亮
                 RED_A=1;YELLOW_A=1;GREEN_A=0;
                 RED_B=0;YELLOW_B=1;GREEN_B=1;
                 DelayMS(2000);
                 Operation_Type=2;
                 break;
        case 2:  //东西向黄灯闪烁，绿灯关闭
                 DelayMS(300);
                 YELLOW_A=~YELLOW_A;GREEN_A=1;
                 if(++Flash_Count!=10) return; //闪烁 5 次
                 Flash_Count=0;
                 Operation_Type=3;
                 break;
        case 3:  //东西向红灯，南北向绿灯亮
                 RED_A=0;YELLOW_A=1;GREEN_A=1;
                 RED_B=1;YELLOW_B=1;GREEN_B=0;
                 DelayMS(2000);
                 Operation_Type=4;
                 break;
        case 4:  //南北向黄灯闪烁 5 次
                 DelayMS(300);
                 YELLOW_B=~YELLOW_B;GREEN_B=1;
                 if(++Flash_Count!=10) return;
                 Flash_Count=0;
                 Operation_Type=1;
    }
}
//主程序
void main()
{
    while(1) Traffic_Light();
}
```

06　单只数码管循环显示 0~9

```
/*    名称：单只数码管循环显示 0~9
      说明：主程序中的循环语句反复将 0~9 的段码送至 P0 口，使数字 0~9 循环显示
*/
#include<reg51.h>
#include<intrins.h>
#define uchar unsigned char
#define uint unsigned int
uchar code DSY_CODE[]={0xc0,0xf9,0xa4,0xb0,0x99,0x92,0x82,0xf8,0x80,0x90,0xff};
//延时
void DelayMS(uint x)
{
      uchar t;
      while(x--) for(t=0;t<120;t++);
}
//主程序
void main()
{
      uchar i=0;
      P0=0x00;
      while(1)
      {
            P0=~DSY_CODE[i];
            i=(i+1)%10;
            DelayMS(300);
      }
}
```



07　8 只数码管滚动显示单个数字




```
/*    名称：8 只数码管滚动显示单个数字
      说明：数码管从左到右依次滚动显示 0~7，程序通过每次仅循环选通一只数码管
*/
#include<reg51.h>
#include<intrins.h>
#define uchar unsigned char
#define uint unsigned int
uchar code DSY_CODE[]={0xc0,0xf9,0xa4,0xb0,0x99,0x92,0x82,0xf8,0x80,0x90};
//延时
void DelayMS(uint x)
{
      uchar t;
      while(x--) for(t=0;t<120;t++);
```

```
}
//主程序
void main()
{
    uchar i,wei=0x80;
    while(1)
    {
        for(i=0;i<8;i++)
        {
            P2=0xff;      //关闭显示
            wei=_crol_(wei,1);
            P0=DSY_CODE[i];   //发送数字段码
            P2=wei;           //发送位码
            DelayMS(300);
        }
    }
}
```

08　8 只数码管动态显示多个不同字符

电路如上图

```
/*   名称：8 只数码管动态显示多个不同字符
     说明：数码管动态扫描显示 0~7。
*/
#include<reg51.h>
#include<intrins.h>
#define uchar unsigned char
#define uint unsigned int
uchar code DSY_CODE[]={0xc0,0xf9,0xa4,0xb0,0x99,0x92,0x82,0xf8,0x80,0x90};
//延时
void DelayMS(uint x)
{
    uchar t;
    while(x--) for(t=0;t<120;t++);
}
//主程序
void main()
{
    uchar i,wei=0x80;
    while(1)
    {
        for(i=0;i<8;i++)
        {
            P0=0xff;
            P0=DSY_CODE[i];   //发送段码
            wei=_crol_(wei,1);
```

```
            P2=wei;             //发送位码
            DelayMS(2);
        }
    }
}
```

09　8 只数码管闪烁显示数字串

电路如上图

```
/*  名称：8 只数码管闪烁显示数字串
    说明：数码管闪烁显示由 0~7 构成的一串数字
          本例用动态刷新法显示一串数字，在停止刷新时所有数字显示消失。
*/
#include<reg51.h>
#define uchar unsigned char
#define uint unsigned int
//段码表
uchar code DSY_CODE[]={0xc0,0xf9,0xa4,0xb0,0x99,0x92,0x82,0xf8,0x80,0x90};
//位码表
uchar code DSY_IDX[]={0x01,0x02,0x04,0x08,0x10,0x20,0x40,0x80};
//延时
void DelayMS(uint x)
{
    uchar t;
    while(x--) for(t=0;t<120;t++);
}
//主程序
void main()
{
    uchar i,j;
    while(1)
    {
        for(i=0;i<30;i++)
        {
            for(j=0;j<8;j++)
            {
                P0=0xff;
                P0=DSY_CODE[j];   //发送段码
                P2=DSY_IDX[j];    //发送位码
                DelayMS(2);
             }
        }
        P2=0x00;   //关闭所有数码管并延时
        DelayMS(1000);
    }
}
```

10　8 只数码管滚动显示数字串

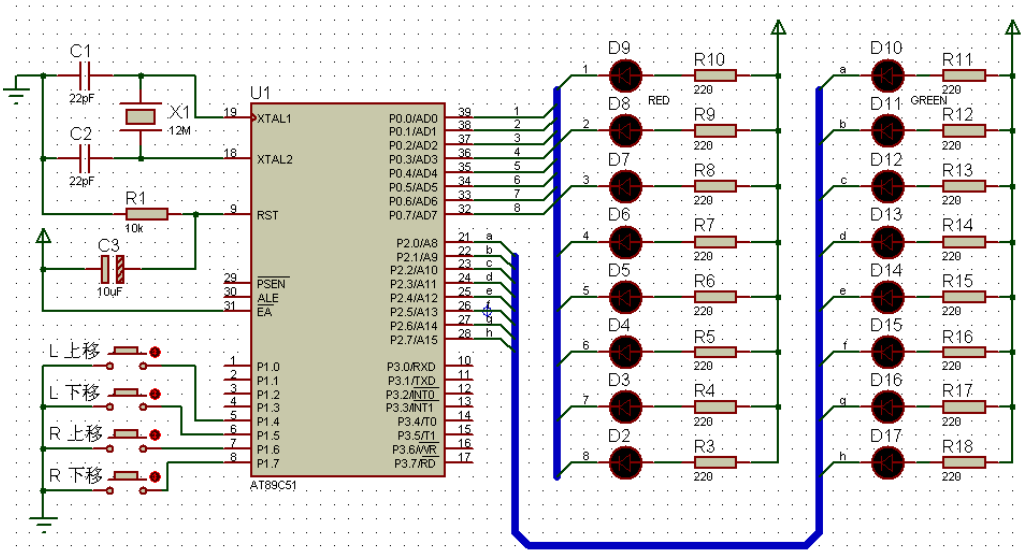
电路如上图

```
/*    名称：8 只数码管滚动显示数字串
      说明：数码管向左滚动显示 3 个字符构成的数字串
*/
#include<reg51.h>
#include<intrins.h>
#define uchar unsigned char
#define uint unsigned int
//段码表
uchar code DSY_CODE[]={0xc0,0xf9,0xa4,0xb0,0x99,0x92,0x82,0xf8,0x80,0x90,0xff};
//下面数组看作环形队列，显示从某个数开始的 8 个数（10 表示黑屏）
uchar Num[]={10,10,10,10,10,10,10,10,2,9,8};
//延时
void DelayMS(uint x)
{
      uchar t;
      while(x--) for(t=0;t<120;t++);
}
//主程序
void main()
{
      uchar i,j,k=0,m=0x80;
      while(1)
      {     //刷新若干次，保持一段时间的稳定显示
            for(i=0;i<15;i++)
            {
                  for(j=0;j<8;j++)
                  {     //发送段码，采用环形取法，从第 k 个开始取第 j 个
                        P0=0xff;
                        P0=DSY_CODE[Num[(k+j)%11]];
                        m=_crol_(m,1);
                        P2=m;       //发送位码
                        DelayMS(2);
                    }
            }
            k=(k+1)%11;  //环形队列首支针 k 递增，Num 下标范围 0~10，故对 11 取余
      }
}
```

11　K1-K4 控制 LED 移位

```
/*    名称：K1-K4 控制 LED 移位
      说明：按下 K1 时，P0 口 LED 上移一位；
```

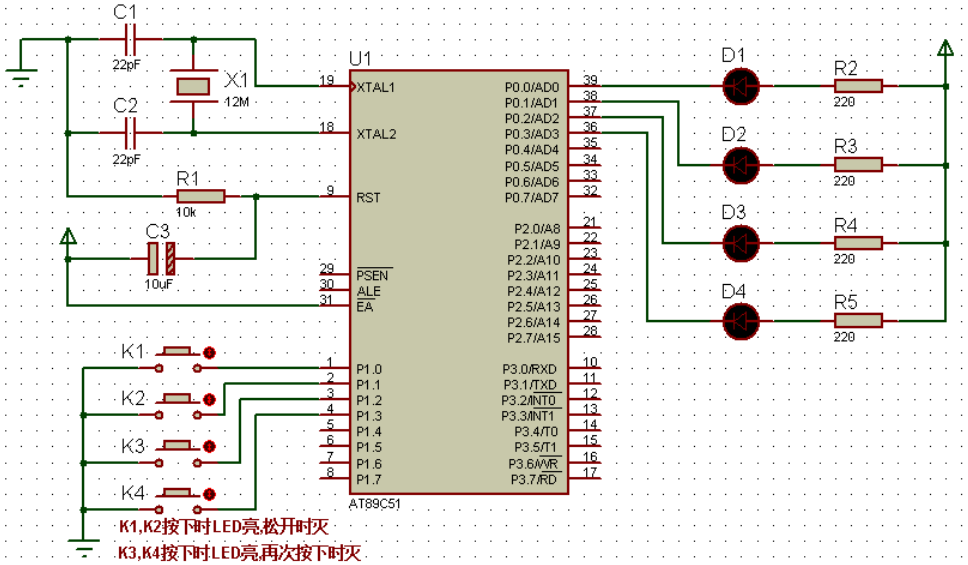
按下 K2 时，P0 口 LED 下移一位；
按下 K3 时，P2 口 LED 上移一位；
按下 K4 时，P2 口 LED 下移一位；
*/




```
#include<reg51.h>
#include<intrins.h>
#define uchar unsigned char
#define uint unsigned int
//延时
void DelayMS(uint x)
{
    uchar i;
    while(x--)
for(i=0;i<120;i++);
}
//根据 P1 口的按键移动 LED
void Move_LED()
{
    if      ((P1&0x10)==0) P0=_cror_(P0,1);   //K1
    else if((P1&0x20)==0) P0=_crol_(P0,1);    //K2
    else if((P1&0x40)==0) P2=_cror_(P2,1);   //K3
    else if((P1&0x80)==0) P2=_crol_(P2,1);    //K4
}
//主程序
void main()
{
    uchar Recent_Key; //最近按键
    P0=0xfe;
    P2=0xfe;
    P1=0xff;
    Recent_Key=0xff;
    while(1)
    {
        if(Recent_Key!=P1)
        {
            Recent_Key=P1;      //保存最近按键
            Move_LED();
            DelayMS(10);
        }
    }
}
```

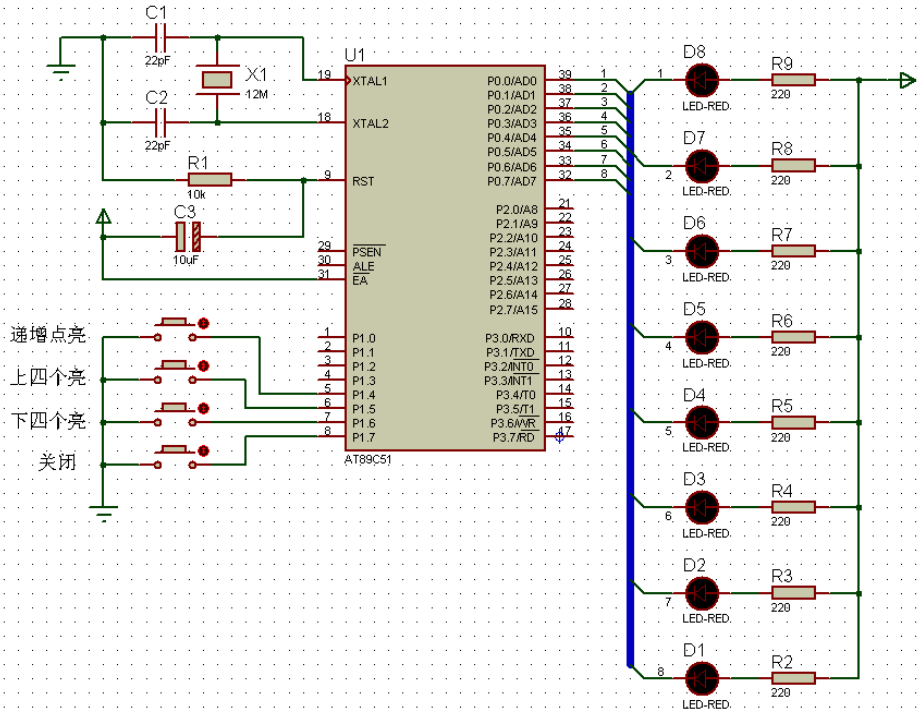
12 K1-K4 按键状态显示

```
/*    名称：K1-K4  按键状态显示
      说明：K1、K2 按下时 LED 点亮，松开时熄灭，
            K3、K4 按下并释放时 LED 点亮，再次按下并释放时熄灭；
*/
#include<reg51.h>
#define uchar unsigned char
#define uint unsigned int
sbit LED1=P0^0;
sbit LED2=P0^1;
sbit LED3=P0^2;
sbit LED4=P0^3;
sbit K1=P1^0;
sbit K2=P1^1;
sbit K3=P1^2;
sbit K4=P1^3;
//延时
void DelayMS(uint x)
{
    uchar i;
    while(x--) for(i=0;i<120;i++);
}
//主程序
void main()
{
    P0=0xff;
    P1=0xff;
    while(1)
    {
        LED1=K1;
        LED2=K2;
        if(K3==0)
        {
            while(K3==0);
            LED3=~LED3;
        }
        if(K4==0)
        {
            while(K4==0);
            LED4=~LED4;
        }
        DelayMS(10);
    }
}
```



13　K1-K4 分组控制 LED

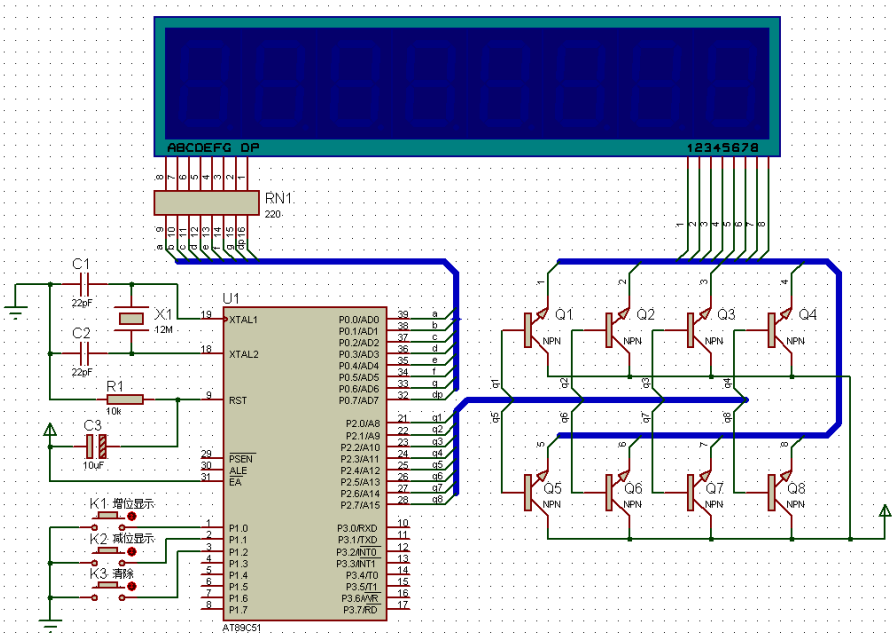
```
/*    名称：K1-K4  分组控制 LED
      说明：每次按下 K1 时递增点亮一只 LED，全亮时再次按下则再次循环开始，
            K2 按下后点亮上面 4 只 LED，K3 按下后点亮下面 4 只 LED，K4 按下后关闭所有 LED
*/
#include<reg51.h>
#define uchar unsigned char
#define uint unsigned int
//延时
void DelayMS(uint x)
{
    uchar i;
    while(x--) for(i=0;i<120;i++);
}
//主程序
void main()
{
    uchar k,t,Key_State;
    P0=0xff;
    P1=0xff;
    while(1)
    {
        t=P1;
        if(t!=0xff)
        {
            DelayMS(10);
            if(t!=P1) continue;
            //取得 4 位按键值，由模式 XXXX1111(X 中有一位为 0，其他均为 1)
            //变为模式 0000XXXX(X 中有一位为 1，其他均为 0)
            Key_State=~t>>4;
            k=0;
            //检查 1 所在位置，累加获取按键号 k
            while(Key_State!=0)
            {
                k++;
                Key_State>>=1;
            }
            //根据按键号 k 进行 4 种处理
            switch(k)
            {
                case 1:    if(P0==0x00) P0=0xff;
                           P0<<=1;
                           DelayMS(200);
                           break;
                case 2:    P0=0xf0;break;
```



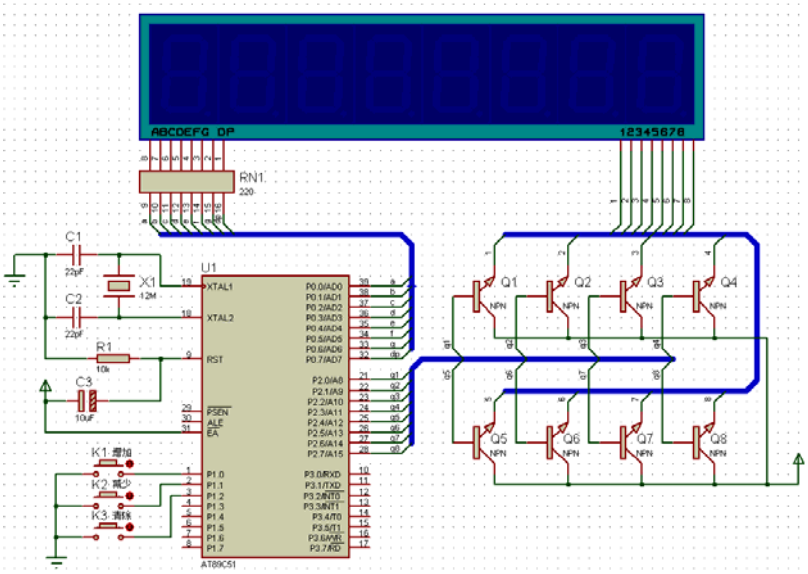
```
                case 3:    P0=0x0f;break;
                case 4:    P0=0xff;
            }
        }
    }
}
```

14 K1-K4 控制数码管移位显示

```
/*    名称：K1-K4 控制数码管移位显示
      说明：按下 K1 时加 1 计数并增加显示位，
            按下 K2 时减 1 计数并减少显示位，
            按下 K3 时清零。
*/
#include<reg51.h>
#define uchar unsigned char
#define uint unsigned int
//段码
uchar code DSY_CODE[]={0xc0,0xf9,0xa4,0xb0,0x99,0x92,0x82,0xf8,0x80,0x90,0xff};
//位码
uchar code DSY_Index[]={0x80,0x40,0x20,0x10,0x08,0x04,0x02,0x01};
//待显示到各数码管的数字缓冲（开始仅在 0 位显示 0，其他黑屏）
uchar Display_Buffer[]={0,10,10,10,10,10,10,10};
//延时
void DelayMS(uint x)
{
    uchar i;
    while(x--) for(i=0;i<120;i++);
}
void Show_Count_ON_DSY()
{
    uchar i;
    for(i=0;i<8;i++)
    {
        P0=0xff;

    P0=DSY_CODE[Display_Buffer[i]];
        P2=DSY_Index[i];
        DelayMS(2);
    }
}
//主程序
void main()
{
    uchar i,Key_NO,Key_Counts=0;
```



```
    P0=0xff;
    P1=0xff;
    P2=0x00;
    while(1)
    {
        Show_Count_ON_DSY();
        P1=0xff;
        Key_NO=P1;
        //P1 口按键状态分别为 K1-0xfe，K2-0xfd,K3-0xfb
        switch(Key_NO)
        {
            case 0xfe:    Key_Counts++;
                          if(Key_Counts>8) Key_Counts=8;
                          Display_Buffer[Key_Counts-1]=Key_Counts;
                          break;
            case 0xfd:    if(Key_Counts>0)Display_Buffer[--Key_Counts]=10;
                          break;
            case 0xfb:    Display_Buffer[0]=0;
                          for(i=1;i<8;i++) Display_Buffer[i]=10;
                          Key_Counts=0;
        }
        //若键未释放则仅刷新显示，不进行键扫描
        while(P1!=0xff) Show_Count_ON_DSY();
    }
}
```



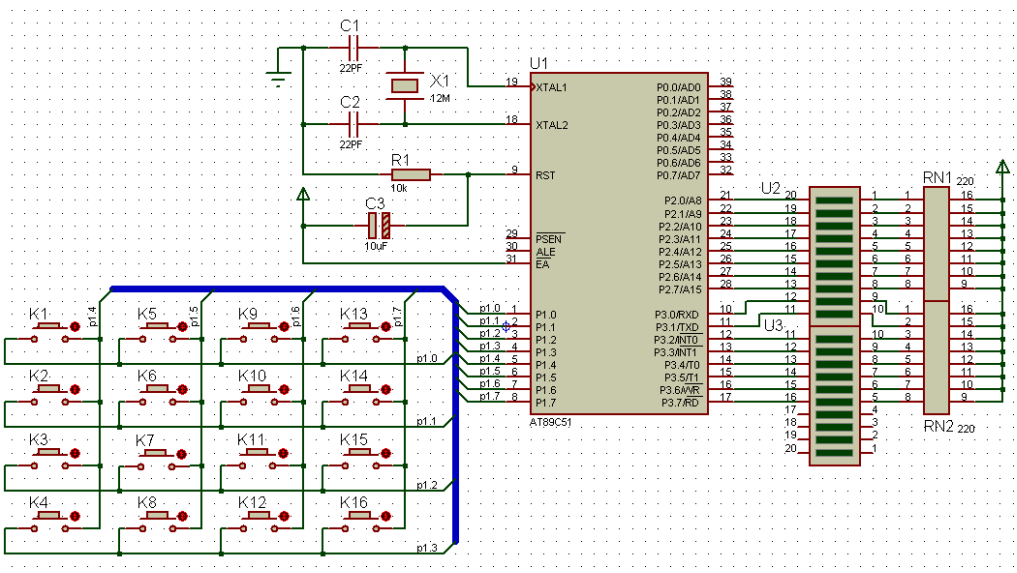
15　K1-K4 控制数码管加减演示

```
/*    名称：K1-K4 控制数码管加减演示
      说明：按下 K1 后加 1 计数，按下 K2 后减 1 计数，按下 K3 后清零。
*/
#include<reg51.h>
#include<intrins.h>
#define uchar unsigned char
#define uint unsigned int
//段码
uchar code DSY_CODE[]={0xc0,0xf9,0xa4,0xb0,0x99,0x92,0x82,0xf8,0x80,0x90,0xff};
//待显示的 3 位缓冲
uchar Num_Buffer[]={0,0,0};
//按键代码，按键计数
uchar Key_Code,Key_Counts=0;
//延时
void DelayMS(uint x)
{
```

```
    uchar i;
    while(x--) for(i=0;i<120;i++);
}
//显示函数
void Show_Counts_ON_DSY()
{
    uchar i,j=0x01;
    Num_Buffer[2]=Key_Counts/100;
    Num_Buffer[1]=Key_Counts/10%10;
    Num_Buffer[0]=Key_Counts%10;
    for(i=0;i<3;i++)
    {
        j=_cror_(j,1);
        P0=0xff;
        P0=DSY_CODE[Num_Buffer[i]];
        P2=j;
        DelayMS(1);
    }
}
//主程序
void main()
{
    uchar i;
    P0=0xff;
    P1=0xff;
    P2=0x00;
    Key_Code=0xff;
    while(1)
    {
        Show_Counts_ON_DSY();
        P1=0xff;
        Key_Code=P1;
        //有键按下时，数码管刷新显示 30 次，该行代码同时起到延时作用
        if(Key_Code!=0xff)
        for(i=0;i<30;i++) Show_Counts_ON_DSY();
        switch(Key_Code)
        {
            case 0xfe:    if(Key_Counts<255) Key_Counts++;
                          break;
            case 0xfd:    if(Key_Counts>0) Key_Counts--;
                          break;
            case 0xfb:    Key_Counts=0;
        }
        Key_Code=0xff;
    }
```

```
}
```

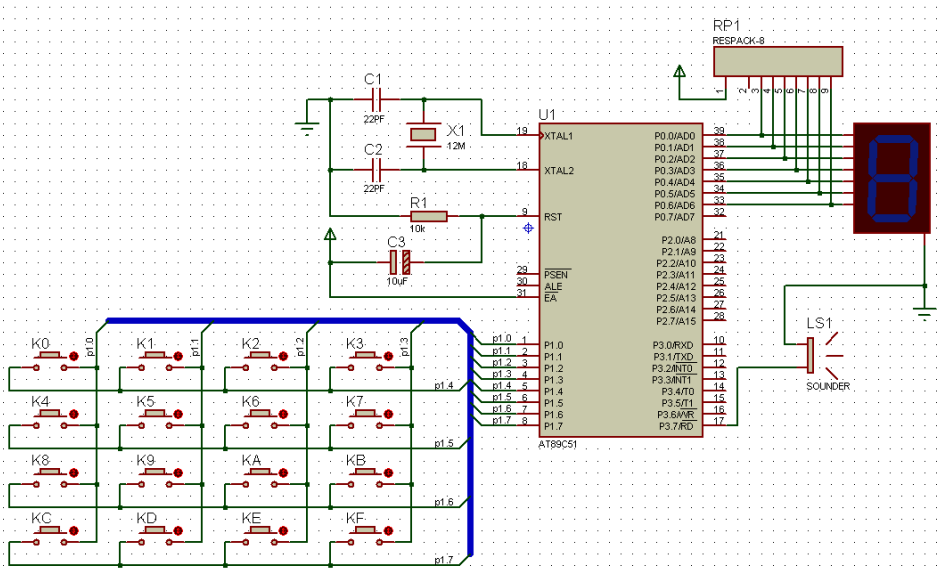
16　4X4 矩阵键盘控制条形 LED 显示




```
/*　名称：4X4 矩阵键盘控制条形 LED 显示
　　说明：运行本例时，按下的按键值越大点亮的 LED 越多。
*/
#include<reg51.h>
#include<intrins.h>
#define uchar unsigned char
#define uint unsigned int
//矩阵键盘按键特征码表
uchar code KeyCodeTable[]={0x11,0x12,0x14,0x18,0x21,
0x22,0x24,0x28,0x41,0x42,0x44,0x48,0x81,0x82,0x84,0x88};
//延时
void DelayMS(uint x)
{
    uchar i;
    while(x--) for(i=0;i<120;i++);
}
//键盘扫描
uchar Keys_Scan()
{
    uchar sCode,kCode,i,k;
    //低 4 位置 0，放入 4 行
    P1=0xf0;
    //若高 4 位出现 0，则有键按下
    if((P1&0xf0)!=0xf0)
    {
        DelayMS(2);
        if((P1&0xf0)!=0xf0)
        {
            sCode=0xfe;             //行扫描码初值
            for(k=0;k<4;k++)  //对 4 行分别进行扫描
            {
                P1=sCode;
                if((P1&0xf0)!=0xf0)
                {
                    kCode=~P1;
                    for(i=0;i<16;i++)  //查表得到按键序号并返回
                        if(kCode==KeyCodeTable[i])
```

```
                    return(i);
                }
                else
                    sCode=_crol_(sCode,1);
            }
        }
    }
    return(-1);
}
//主程序
void main()
{
    uchar i,P2_LED,P3_LED;
    uchar KeyNo=-1;          //按键序号，-1 表示无按键
    while(1)
    {
        KeyNo=Keys_Scan();   //扫描键盘获取按键序号 KeyNo
        if(KeyNo!=-1)
        {
            P2_LED=0xff;
            P3_LED=0xff;
            for(i=0;i<=KeyNo;i++)  //键值越大，点亮的 LED 越多
            {
                if(i<8)
                P3_LED>>=1;
                else
                P2_LED>>=1;
            }
            P3=P3_LED;        //点亮条形 LED
            P2=P2_LED;
        }
    }
}
```

17　数码管显示 4X4 矩阵键盘按键号

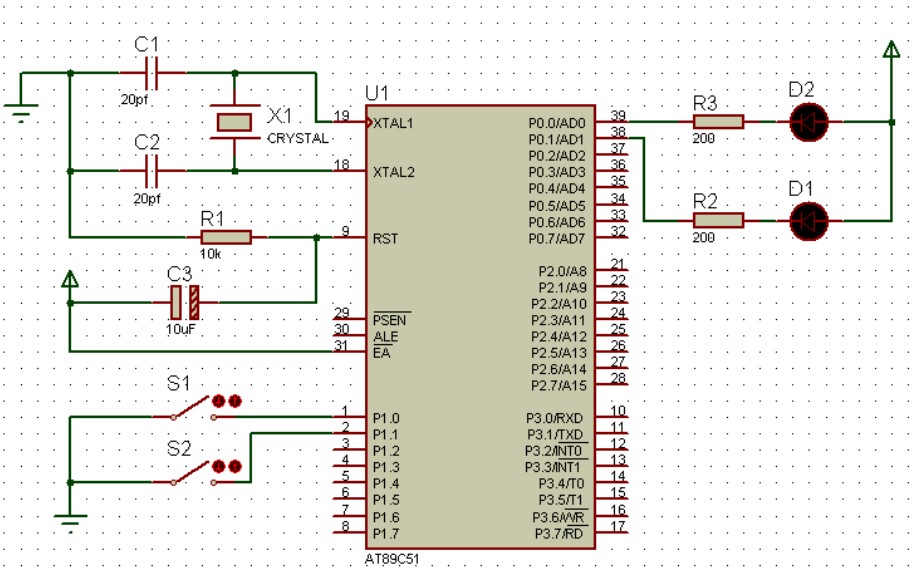



```
/*    名称：数码管显示 4X4 矩阵键盘按键号
      说明：按下任意键时，数码管都会显示其键的序号，扫描程序首先判断按键发生在哪一列，然后根据所发生的行附加不同的值，从而得到按键的序号。
*/
```

```
#include<reg51.h>
#define uchar unsigned char
#define uint unsigned int
//段码
uchar code DSY_CODE[]={0xc0,0xf9,0xa4,0xb0,0x99,0x92,0x82,0xf8,0x80,0x90,
                       0x88,0x83,0xc6,0xa1,0x86,0x8e,0x00};
sbit BEEP=P3^7;
//上次按键和当前按键的序号，该矩阵中序号范围 0~15，16 表示无按键
uchar Pre_KeyNo=16,KeyNo=16;
//延时
void DelayMS(uint x)
{
    uchar i;
    while(x--) for(i=0;i<120;i++);
}
//矩阵键盘扫描
void Keys_Scan()
{
    uchar Tmp;
    P1=0x0f; //高 4 位置 0，放入 4 行
    DelayMS(1);
    Tmp=P1^0x0f;//按键后 0f 变成 0000XXXX，X 中一个为 0，3 个仍为 1，通过异或把 3 个 1 变为 0，唯一的 0 变为 1
    switch(Tmp)   //判断按键发生于 0~3 列的哪一列
    {
        case 1:    KeyNo=0;break;
        case 2:    KeyNo=1;break;
        case 4: KeyNo=2;break;
        case 8:    KeyNo=3;break;
        default:KeyNo=16; //无键按下
    }
    P1=0xf0;     //低 4 位置 0，放入 4 列
    DelayMS(1);
    Tmp=P1>>4^0x0f;//按键后 f0 变成 XXXX0000，X 中有 1 个为 0，三个仍为 1；高 4 位转移到低 4 位并异或得到改变的值
    switch(Tmp)   //对 0~3 行分别附加起始值 0，4，8，12
    {
        case 1:    KeyNo+=0;break;
        case 2:    KeyNo+=4;break;
        case 4:    KeyNo+=8;break;
        case 8:    KeyNo+=12;
    }
}
//蜂鸣器
void Beep()
```

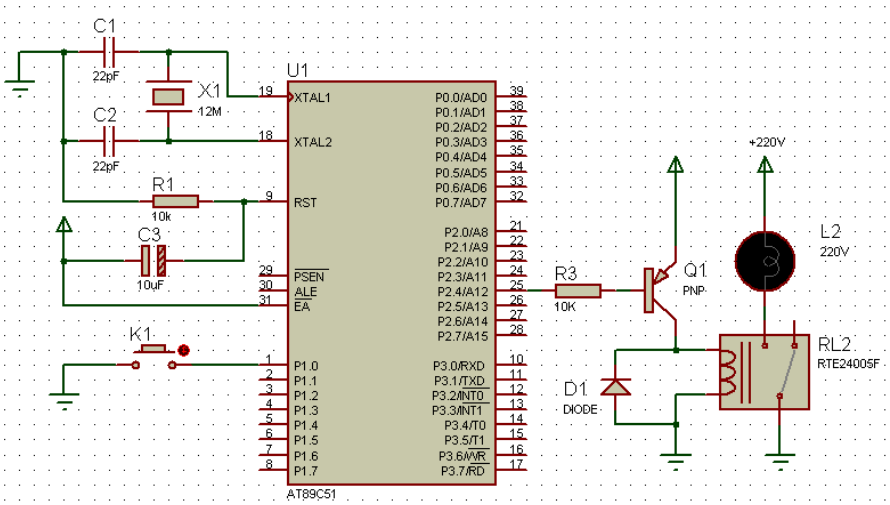
```
{
    uchar i;
    for(i=0;i<100;i++)
    {
        DelayMS(1);
        BEEP=~BEEP;
    }
    BEEP=0;
}
//主程序
void main()
{
    P0=0x00;
    BEEP=0;
    while(1)
    {
        P1=0xf0;
        if(P1!=0xf0) Keys_Scan();   //获取键序号
        if(Pre_KeyNo!=KeyNo)
        {
            P0=~DSY_CODE[KeyNo];
            Beep();
            Pre_KeyNo=KeyNo;
        }
        DelayMS(100);
    }
}
```

18　开关控制 LED



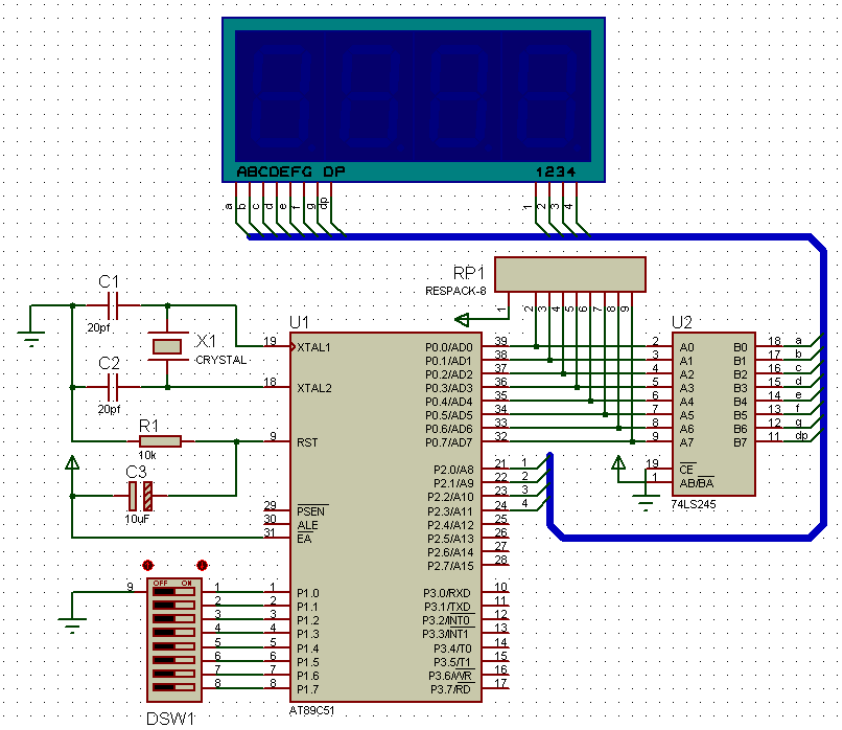

```
/*    名称：开关控制 LED
      说明：开关 S1 和 S2 分别控制 LED1 和 LED2。
*/
#include<reg51.h>
sbit S1=P1^0;
sbit S2=P1^1;
sbit LED1=P0^0;
sbit LED2=P0^1;
//主程序
void main()
{
    while(1)
    {
        LED1=S1;
```

```
        LED2=S2;
    }
}
```



19 继电器控制照明设备

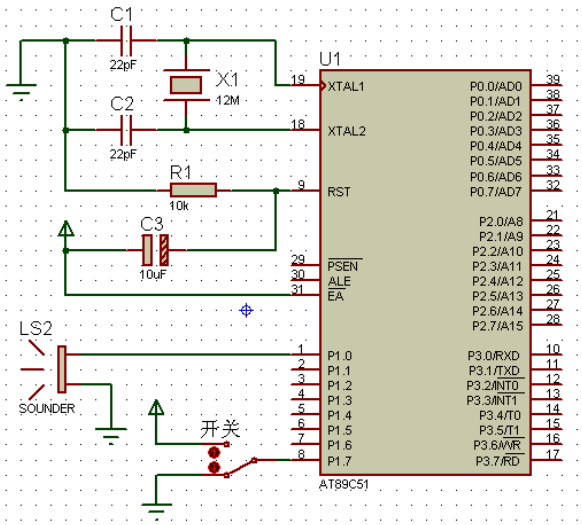
```
/*    名称：继电器控制照明设备
      说明：按下 K1 灯点亮，再次按下时灯熄灭
*/
#include<reg51.h>
#define uchar unsigned char
#define uint unsigned int
sbit K1=P1^0;
sbit RELAY=P2^4;
//延时
void DelayMS(uint ms)
{
    uchar t;
    while(ms--)for(t=0;t<120;t++);
}
//主程序
void main()
{
    P1=0xff;
    RELAY=1;
    while(1)
    {
        if(K1==0)
        {
            while(K1==0);
            RELAY=~RELAY;
            DelayMS(20);
        }
    }
}
```



20 数码管显示拨码开关编码

```
/*    名称：数码管显示拨码开关编码
      说明：系统显示拨码开关所设置的编码 000~255
*/
#include<reg51.h>
#include<intrins.h>
#define uchar unsigned char
```

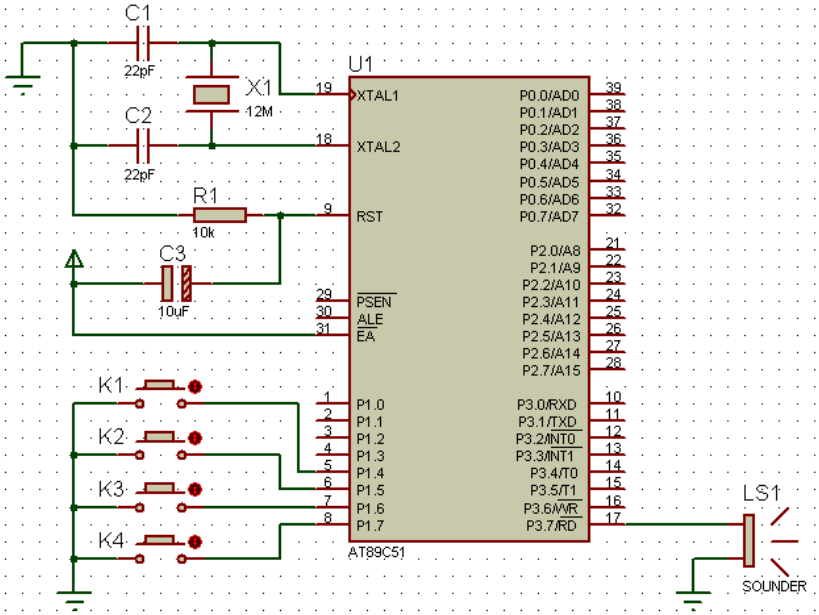
```
#define uint unsigned int
//各数字的数码管段码（共阴）
uchar code DSY_CODE[]={0x3f,0x06,0x5b,0x4f,0x66,0x6d,0x7d,0x07,0x7f,0x6f};
//显示缓冲
uchar DSY_Buffer[3]={0,0,0};
//延时
void DelayMS(uint ms)
{
    uchar t;
    while(ms--)for(t=0;t<120;t++);
}
//主程序
void main()
{
    uchar i,m,Num;
    P0=0xff;
    P2=0xff;
    while(1)
    {
        m=0xfe;
        Num=P1;//读取拨码开关的值
        DSY_Buffer[0]=Num/100;
        DSY_Buffer[1]=Num/10%10;
        DSY_Buffer[2]=Num%10;
        for(i=0;i<3;i++)    //刷新显示在数码管上
        {
            m=_crol_(m,1);
            P2=m;
            P0=DSY_CODE[DSY_Buffer[i]];
            DelayMS(10);
        }
    }
}
```



21　开关控制报警器

```
/*    名称：开关控制报警器
      说明：用 K1 开关控制报警器，程序控制 P1.0 输出两种不同频率的声音，模拟很逼真的报警效果
*/
#include<reg51.h>
#define uchar unsigned char
#define uint unsigned int
sbit SPK=P1^0;
sbit K1=P1^7;
//发声函数
```

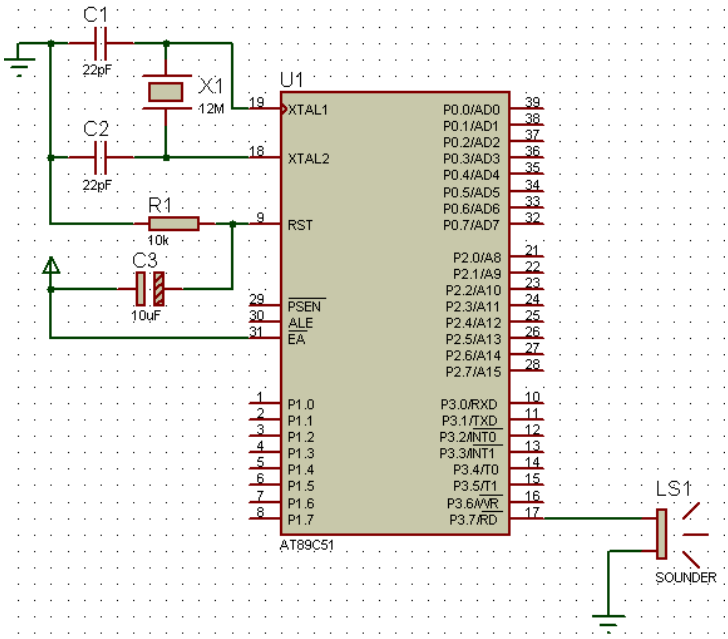
```
void Alarm(uchar t)
{
    uchar i,j;
    for(i=0;i<200;i++)
    {
        SPK=~SPK;
        for(j=0;j<t;j++);     //由参数 t 行成不同的频率
    }
}
void main()
{
    SPK=0;
    while(1)
    {
        if(K1==1)
        {
            Alarm(90);
            Alarm(120);
        }
    }
}
```



22 按键发音

```
/*    名称：按键发音
      说明：按下不同的按键会是 SOUNDER 发出不同频率的声音。本例使用延时函数实现不同频率的声音输出，以后也可使用定时器
*/
#include<reg51.h>
#define uchar unsigned char
#define uint unsigned int
sbit BEEP=P3^7;
sbit K1=P1^4;
sbit K2=P1^5;
sbit K3=P1^6;
sbit K4=P1^7;
//延时
void DelayMS(uint x)
{
    uchar t;
    while(x--) for(t=0;t<120;t++);
}
//按周期 t 发音
void Play(uchar t)
{
```

```
    uchar i;
    for(i=0;i<100;i++)
    {
        BEEP=~BEEP;
        DelayMS(t);
    }
    BEEP=0;
}
void main()
{
    P1=0xff;
    BEEP=0;
    while(1)
    {
        if(K1==0)    Play(1);
        if(K2==0)    Play(2);
        if(K3==0)    Play(3);
        if(K4==0)    Play(4);
    }
}
```

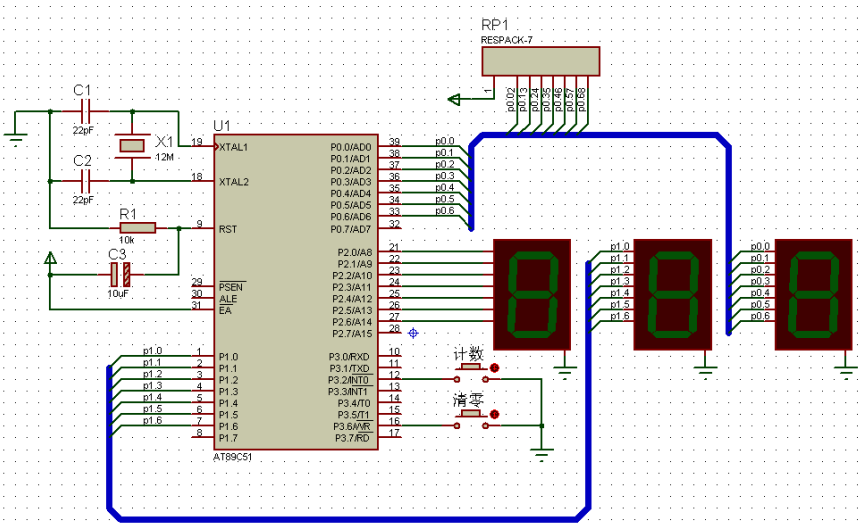


23 播放音乐

```
/*    名称：播放音乐
      说明：程序运行时播放生日快乐歌，  未使用定时器中断，所有频率完全用延时实现
*/
#include<reg51.h>
#define uchar unsigned char
#define uint unsigned int
sbit BEEP=P3^7;
//生日快乐歌的音符频率表，不同频率由不同的延时来决定
uchar code SONG_TONE[]={212,212,190,212,159,169,212,212,190,212,142,159,
                        212,212,106,126,159,169,190,119,119,126,159,142,159,0};
//生日快乐歌节拍表，节拍决定每个音符的演奏长短
uchar code SONG_LONG[]={9,3,12,12,12,24,9,3,12,12,12,24,
                        9,3,12,12,12,12,12,9,3,12,12,12,24,0};
//延时
void DelayMS(uint x)
{
    uchar t;
    while(x--) for(t=0;t<120;t++);
}
//播放函数
void PlayMusic()
{
```

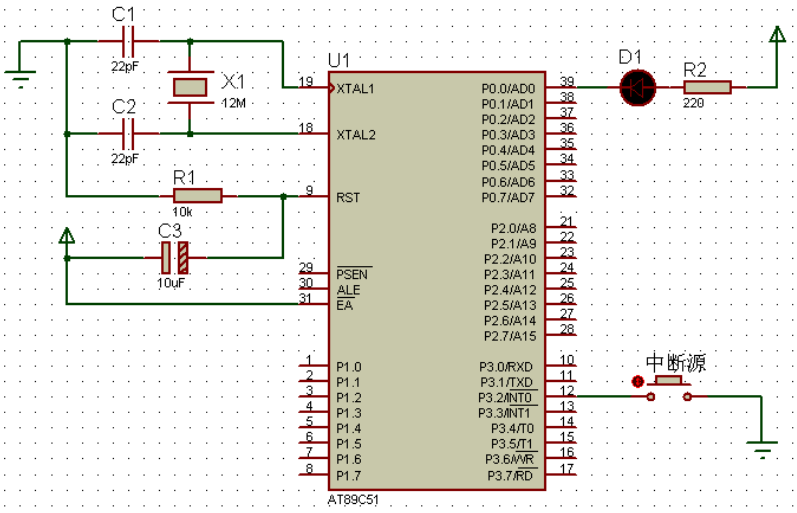
```
    uint i=0,j,k;
    while(SONG_LONG[i]!=0||SONG_TONE[i]!=0)
    {   //播放各个音符，SONG_LONG 为拍子长度
        for(j=0;j<SONG_LONG[i]*20;j++)
        {
            BEEP=~BEEP;
            //SONG_TONE 延时表决定了每个音符的频率
            for(k=0;k<SONG_TONE[i]/3;k++);
        }
        DelayMS(10);
        i++;
    }
}
void main()
{
    BEEP=0;
    while(1)
    {
        PlayMusic();  //播放生日快乐
        DelayMS(500);    //播放完后暂停一段时间
    }
}
```

24 INT0 中断计数




```
/*    名称：INT0 中断计数

    说明：每次按下计数键时触发 INT0 中断，中断程序累加计数，计数值显示在 3 只数码管上，按下清零键时数码管清零
*/
#include<reg51.h>
#define uchar unsigned char
#define uint unsigned int
//0~9 的段码
uchar code DSY_CODE[]={0x3f,0x06,0x5b,0x4f,0x66,0x6d,0x7d,0x07,0x7f,0x6f,0x00};
//计数值分解后各个待显示的数位
uchar DSY_Buffer[]={0,0,0};
uchar Count=0;
sbit Clear_Key=P3^6;
//数码管上显示计数值
void Show_Count_ON_DSY()
{
    DSY_Buffer[2]=Count/100;  //获取 3 个数
```

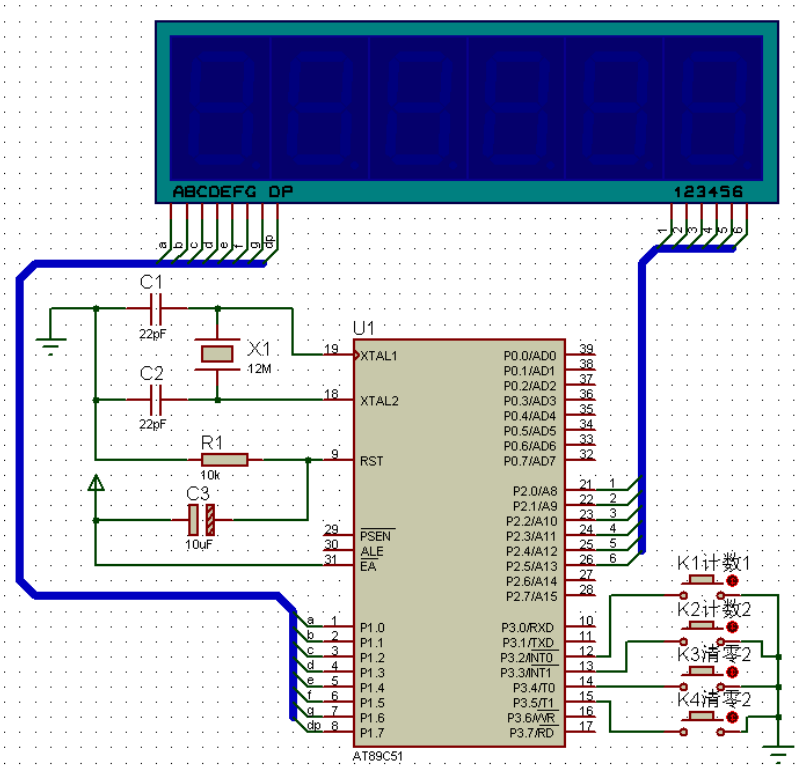
```
    DSY_Buffer[1]=Count%100/10;
    DSY_Buffer[0]=Count%10;
    if(DSY_Buffer[2]==0)     //高位为 0 时不显示
    {
        DSY_Buffer[2]=0x0a;
        if(DSY_Buffer[1]==0)     //高位为 0，若第二位为 0 同样不显示
            DSY_Buffer[1]=0x0a;
    }
    P0=DSY_CODE[DSY_Buffer[0]];
    P1=DSY_CODE[DSY_Buffer[1]];
    P2=DSY_CODE[DSY_Buffer[2]];
}
//主程序
void main()
{
    P0=0x00;
    P1=0x00;
    P2=0x00;
    IE=0x81; //允许 INT0 中断
    IT0=1;          //下降沿触发
    while(1)
    {
        if(Clear_Key==0) Count=0;  //清 0
        Show_Count_ON_DSY();
    }
}
//INT0 中断函数
void EX_INT0() interrupt 0
{
    Count++;       //计数值递增
}
```



25　外部 INT0 中断控制 LED

```
/*    名称：外部 INT0 中断控制 LED
      说明：每次按键都会触发 INT0 中断，中断发生时将 LED 状态取反，产生 LED 状态由按键控制的效果
*/
#include<reg51.h>
#define uchar unsigned char
#define uint unsigned int
sbit LED=P0^0;
//主程序
void main()
```

```
{
    LED=1;
    EA=1;
    EX0=1;
    IT0=1;
    while(1);
}
//INT0 中断函数
void EX_INT0() interrupt 0
{
    LED=~LED;  //控制 LED 亮灭
}
```



26　INT0 及 INT1 中断计数

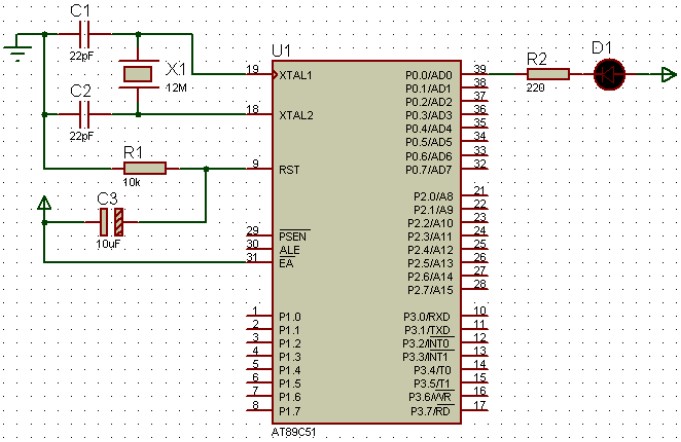
```
/*    名称：INT0 及 INT1 中断计数
      说明：每次按下第 1 个计数键时，第 1 组计数值累加并显示在右边 3 只数码管上， 每次按下第 2 个计数键时，第 2 组计数值累加并显示在左边 3 只数码管上，后两个按键分别清零。
*/
#include<reg51.h>
#define uchar unsigned char
#define uint unsigned int
sbit K3=P3^4; //2 个清零键
sbit K4=P3^5;
//数码管段码与位码
uchar code DSY_CODE[]={0xc0,0xf9,0xa4,0xb0,0x99,0x92,0x82,0xf8,0x80,0x90,0xff};
uchar code DSY_Scan_Bits[]={0x20,0x10,0x08,0x04,0x02,0x01};
//2 组计数的显示缓冲，前 3 位一组，后 3 位一组
uchar data Buffer_Counts[]={0,0,0,0,0,0};
uint Count_A,Count_B=0;
//延时
void DelayMS(uint x)
{
    uchar t;
    while(x--) for(t=0;t<120;t++);
}
//数据显示
void Show_Counts()
{
    uchar i;
    Buffer_Counts[2]=Count_A/100;
    Buffer_Counts[1]=Count_A%100/10;
    Buffer_Counts[0]=Count_A%10;
```

```
    if(   Buffer_Counts[2]==0)
    {
        Buffer_Counts[2]=0x0a;
        if(   Buffer_Counts[1]==0)
            Buffer_Counts[1]=0x0a;
    }
    Buffer_Counts[5]=Count_B/100;
    Buffer_Counts[4]=Count_B%100/10;
    Buffer_Counts[3]=Count_B%10;
    if(   Buffer_Counts[5]==0)
    {
        Buffer_Counts[5]=0x0a;
        if(   Buffer_Counts[4]==0)
            Buffer_Counts[4]=0x0a;
    }
    for(i=0;i<6;i++)
    {
        P2=DSY_Scan_Bits[i];
        P1=DSY_CODE[Buffer_Counts[i]];
        DelayMS(1);
    }
}
//主程序
void main()
{
    IE=0x85;
    PX0=1;   //中断优先
    IT0=1;
    IT1=1;
    while(1)
    {
        if(K3==0) Count_A=0;
        if(K4==0) Count_B=0;
        Show_Counts();
    }
}
//INT0 中断函数
void EX_INT0() interrupt 0
{
    Count_A++;
}
//INT1 中断函数
void EX_INT1() interrupt 2
{
    Count_B++;
```

```
}
```

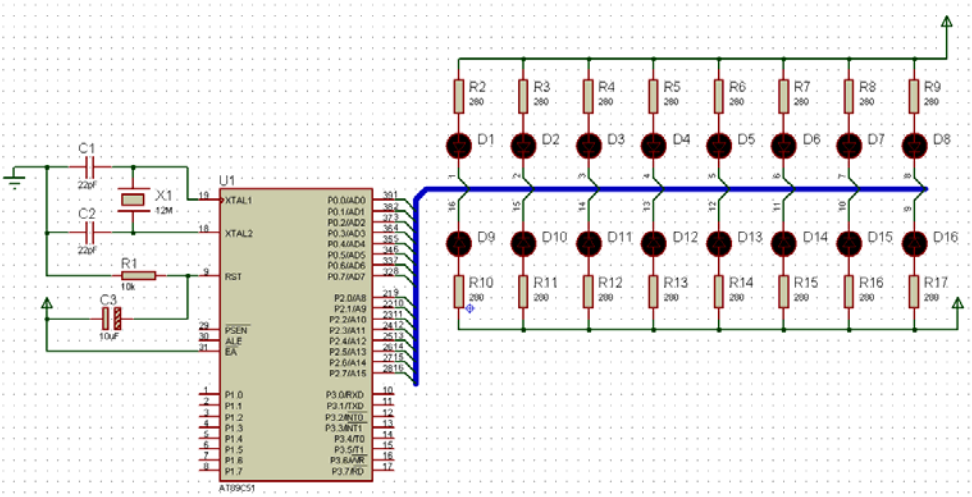
27　定时器控制单只 LED

```
/*    名称：定时器控制单只 LED
      说明：LED 在定时器的中断例程控制下不断闪烁。
*/
#include<reg51.h>
#define uchar unsigned char
#define uint unsigned int
sbit LED=P0^0;
uchar T_Count=0;
//主程序
void main()
{
      TMOD=0x00;              //定时器 0 工作方式 0
      TH0=(8192-5000)/32;    //5ms 定时
      TL0=(8192-5000)%32;
      IE=0x82;                //允许 T0 中断
      TR0=1;
      while(1);
}
//T0 中断函数
void LED_Flash() interrupt 1
{
      TH0=(8192-5000)/32;    //恢复初值
      TL0=(8192-5000)%32;
      if(++T_Count==100)     //0.5s 开关一次 LED
      {
            LED=~LED;
            T_Count=0;
      }
}
```



28　TIMER0 控制流水灯

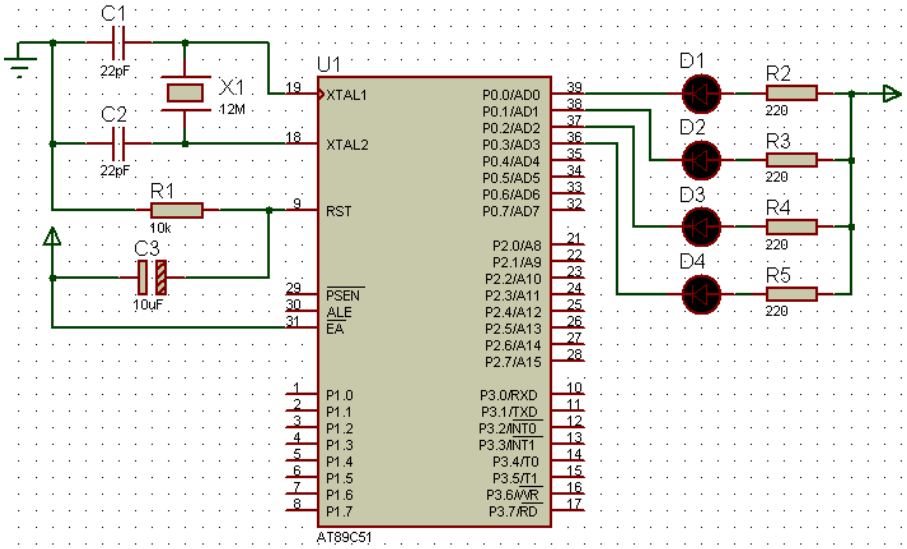
```
/*    名称：TIMER0 控制流水灯
      说明：定时器控制 P0、P2 口的 LED 滚动显示，本例未使用中断函数。
*/
#include<reg51.h>
#include<intrins.h>
#define uchar unsigned char
#define uint unsigned int
```



```
//主程序
void main()
{
    uchar T_Count=0;
    P0=0xfe;
    P2=0xfe;
    TMOD=0x01;                  //定时器 0 工作方式 1
    TH0=(65536-40000)/256;      //40ms 定时
    TL0=(65536-40000)%256;
    TR0=1;                      //启动定时器
    while(1)
    {
        if(TF0==1)
        {
            TF0=0;
            TH0=(65536-40000)/256;     //恢复初值
            TL0=(65536-40000)%256;
            if(++T_Count==5)
            {
                P0=_crol_(P0,1);
                P2=_crol_(P2,1);
                T_Count=0;
            }
        }
    }
}
```

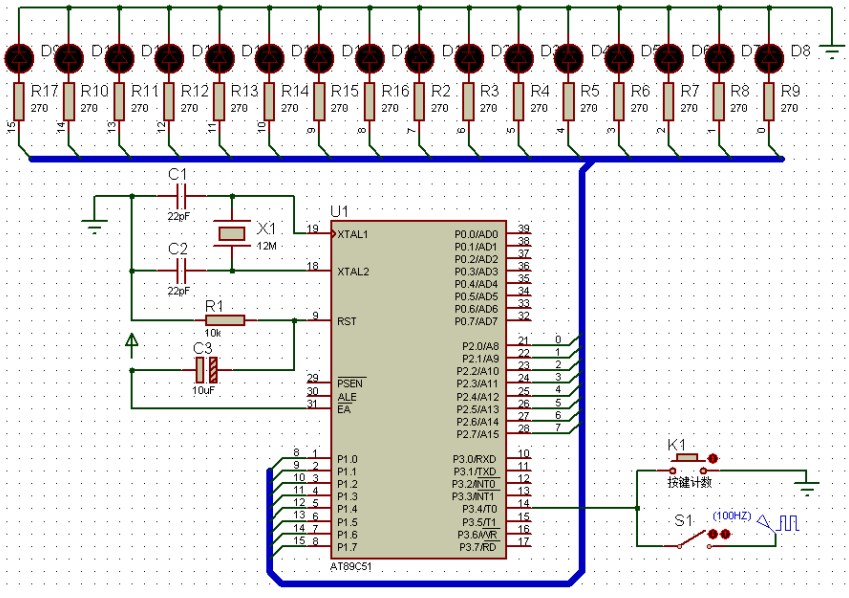
29　定时器控制 4 个 LED 滚动闪烁

```
/*    名称：定时器控制 4 个 LED 滚动闪烁
      说明：4 只 LED 在定时器控制下滚动闪烁。
*/
#include<reg51.h>
#define uchar unsigned char
#define uint unsigned int
sbit B1=P0^0;
sbit G1=P0^1;
sbit R1=P0^2;
sbit Y1=P0^3;
uint i,j,k;
//主程序
void main()
{
    i=j=k=0;
    P0=0xff;
```



```
    TMOD=0x02;              //定时器 0 工作方式 2
    TH0=256-200;            //200us 定时
    TL0=256-200;
    IE=0x82;
    TR0=1;                  //启动定时器
    while(1);
}
//T0 中断函数
void LED_Flash_and_Scroll() interrupt 1
{
    if(++k<35)    return;       //定时中断若干次后执行闪烁
    k=0;
    switch(i)
    {
        case 0:   B1=~B1;break;
        case 1:   G1=~G1;break;
        case 2:   R1=~R1;break;
        case 3:   Y1=~Y1;break;
        default:i=0;
    }
    if(++j<300) return;   //每次闪烁持续一段时间
    j=0;
    P0=0xff;  //关闭显示
    i++;   //切换到下一个 LED
}
```

30　T0 控制 LED 实现二进制计数

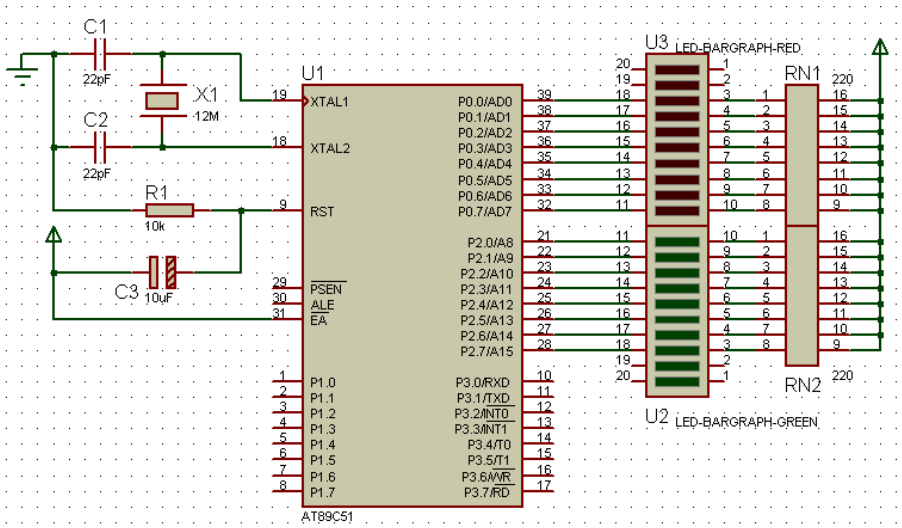



```
/*    名称：T0 控制 LED 实现二进制计数
      说明：本例对按键的计数没有使用查询法，没有使用外部中断函数，没有使用定时或计数中断函数。而是启用了计数器，连接在 T0 引脚的按键每次按下时，会使计数寄存器的值递增，其值通过 LED 以二进制形式显示
*/
#include<reg51.h>
//主程序
void main()
{
    TMOD=0x05; //定时器 0 为计数器，工作方式 1，最大计数值 65535
    TH0=0;          //初值为 0
    TL0=0;
    TR0=1;                  //启动定时器
    while(1)
```

```
    {
        P1=TH0;
        P2=TL0;
    }
}
```

31 TIMER0 与 TIMER1 控制条形 LED

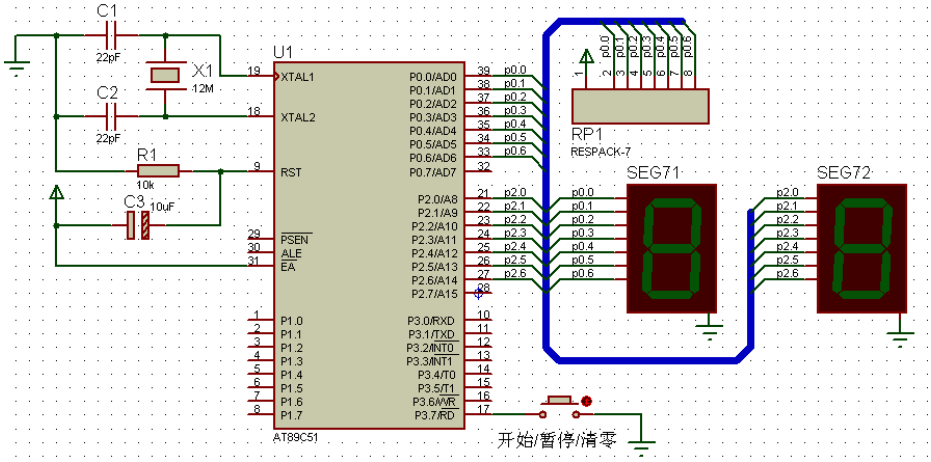
```
/*  名称：TIMER0 与 TIMER1 控制条形 LED
    说明：定时器 T0 定时控制上一组条形 LED，滚动速度较快
          定时器 T1 定时控制下一组条形 LED，滚动速度较慢
*/
#include<reg51.h>
#include<intrins.h>
#define uchar unsigned char
#define uint unsigned int
uchar tc0=0,tc1=0;
//主程序
void main()
{
    P0=0xfe;
    P2=0xfe;
    TMOD=0x11; //定时器 0、定时器 1 均工作于方式 1
    TH0=(65536-15000)/256;          //定时器 0：15ms
    TL0=(65536-15000)%256;
    TH1=(65536-50000)/256;          //定时器 1：50ms
    TL1=(65536-50000)%256;
    IE=0x8a;
    TR0=1;                  //启动定时器
    TR1=1;
    while(1);
}
//T0 中断函数
void Time0() interrupt 1
{
    TH0=(65536-15000)/256;          //恢复定时器 0 初值
    TL0=(65536-15000)%256;
    if(++tc0==10)                   //150ms 转换状态
    {
        tc0=0;
        P0=_crol_(P0,1);
    }
}
//T1 中断函数
```



```
void Time1() interrupt 3
{
    TH0=(65536-50000)/256;          //恢复定时器 1 初值
    TL0=(65536-50000)%256;
    if(++tc1==10)                   //500ms 转换状态
    {
        tc1=0;
        P2=_crol_(P2,1);
    }
}
```

32 10s 的秒表

```
/*  名称：10s 的秒表
    说明：首次按键计时开始，再次按键暂停，第三次按键清零。
*/
#include<reg51.h>
#define uchar unsigned char
#define uint unsigned int
sbit K1=P3^7;
uchar
i,Second_Counts,Key_Flag_Idx;
bit Key_State;
uchar
```



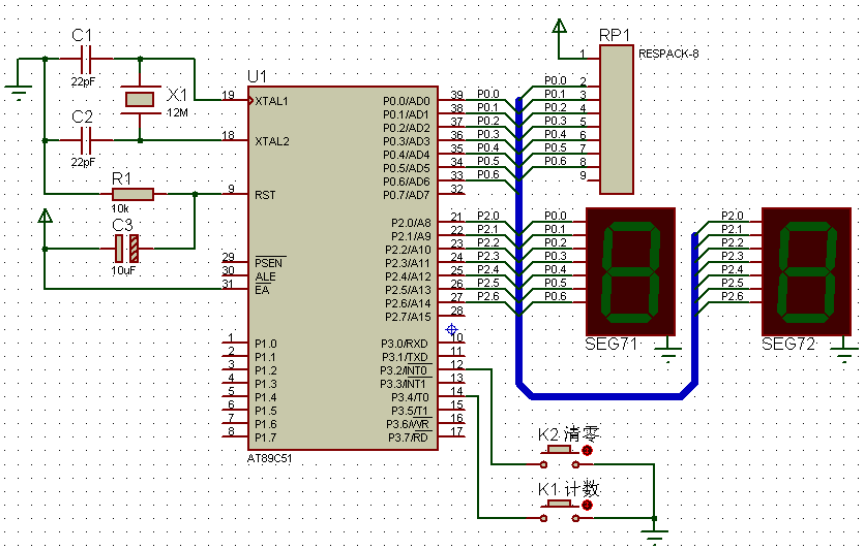

```
DSY_CODE[]={0x3f,0x06,0x5b,0x4f,0x66,0x6d,0x7d,0x07,0x7f,0x6f};
//延时
void DelayMS(uint ms)
{
    uchar t;
    while(ms--) for(t=0;t<120;t++);
}
//处理按键事件
void Key_Event_Handle()
{
    if(Key_State==0)
    {
        Key_Flag_Idx=(Key_Flag_Idx+1)%3;
        switch(Key_Flag_Idx)
        {
            case 1:   EA=1;ET0=1;TR0=1;break;
            case 2:   EA=0;ET0=0;TR0=0;break;
            case 0:   P0=0x3f;P2=0x3f;i=0;Second_Counts=0;
        }
```

```
    }
}
//主程序
void main()
{
    P0=0x3f;                //显示 00
    P2=0x3f;
    i=0;
    Second_Counts=0;
    Key_Flag_Idx=0;             //按键次数（取值 0，1，2，3）
    Key_State=1;            //按键状态
    TMOD=0x01;                  //定时器 0 方式 1
    TH0=(65536-50000)/256;          //定时器 0：15ms
    TL0=(65536-50000)%256;
    while(1)
    {
        if(Key_State!=K1)
        {
            DelayMS(10);
            Key_State=K1;
            Key_Event_Handle();
        }
    }
}
//T0 中断函数
void DSY_Refresh() interrupt 1
{
    TH0=(65536-50000)/256;          //恢复定时器 0 初值
    TL0=(65536-50000)%256;
    if(++i==2)                  //50ms*2=0.1s 转换状态
    {
        i=0;
        Second_Counts++;
        P0=DSY_CODE[Second_Counts/10];
        P2=DSY_CODE[Second_Counts%10];
        if(Second_Counts==100) Second_Counts=0;     //满 100（10s）后显示 00
    }
}
```

33 用计数器中断实现 100 以内的按键计数

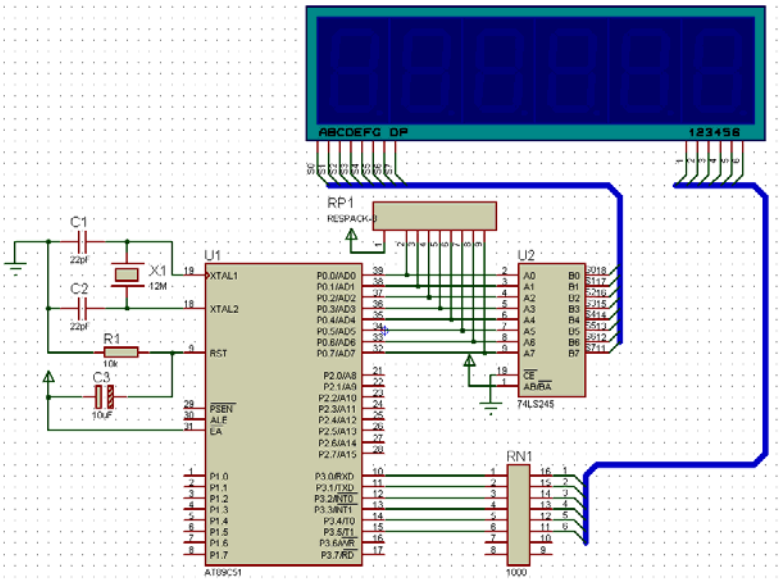
```
/*  名称：用计数器中断实现 100 以内的按键计数
    说明：本例用 T0 计数器中断实现按键技术，由于计数寄存器初值为 1，因此
          P3.4 引脚的每次负跳变都会触发 T0 中断，实现计数值累加。
          计数器的清零用外部中断 0 控制。
```

```
*/
#include<reg51.h>
#define uchar unsigned char
#define uint unsigned int
//段码
uchar                                code
```




```
DSY_CODE[]={0x3f,0x06,0x5b,0x4f,0x66,0x6d,0x7d,0x07,0x7f,0x6f,0x00};
uchar Count=0;
//主程序
void main()
{
    P0=0x00;
    P2=0x00;
    TMOD=0x06;              //计数器 T0 方式 2
    TH0=TL0=256-1;          //计数值为 1
    ET0=1;                  //允许 T0 中断
    EX0=1;                  //允许 INT0 中断
    EA=1;                   //允许 CPU 中断
    IP=0x02;              //设置优先级，T0 高于 INT0
    IT0=1;                  //INT0 中断触发方式为下降沿触发
    TR0=1;                  //启动 T0
    while(1)
    {
        P0=DSY_CODE[Count/10];
        P2=DSY_CODE[Count%10];
    }
}
//T0 计数器中断函数
void Key_Counter() interrupt 1
{
    Count=(Count+1)%100; //因为只有两位数码管，计数控制在 100 以内（00~99）
}
//INT0 中断函数
void Clear_Counter() interrupt 0
{
    Count=0;
}
```

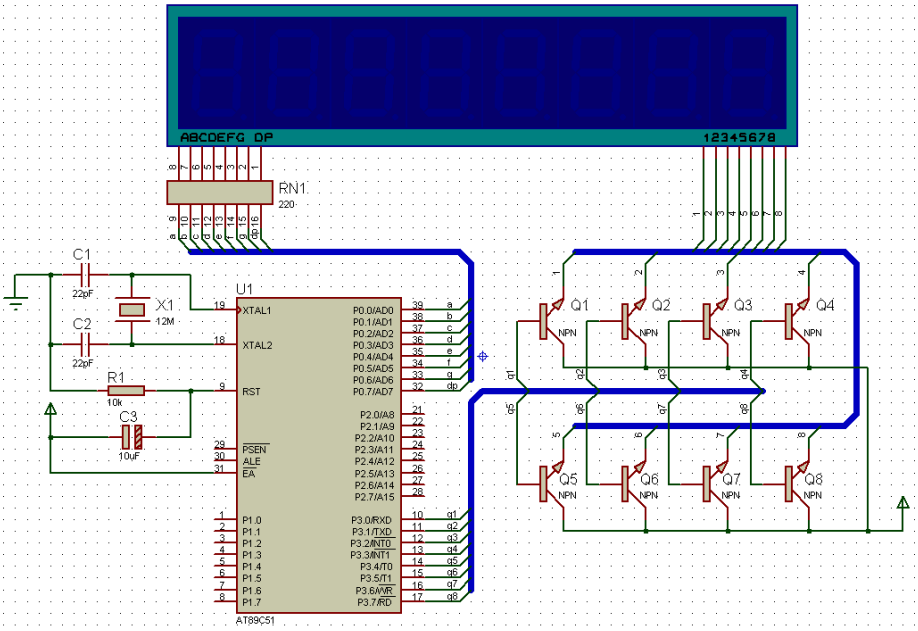
34　100 000s 以内的计时程序




```
/*    名称：100 000s 以内的计时程序
      说明：在 6 只数码管上完成 0~99 999.9s。
*/
#include<reg51.h>
#include<intrins.h>
#define uchar unsigned char
#define uint unsigned int
//段码
uchar                                    code
DSY_CODE[]={0x3f,0x06,0x5b,0x4f,0x66,0x6d,0x7d,0x07,0x7f,0x6f};
//6 只数码管上显示的数字
uchar Digits_of_6DSY[]={0,0,0,0,0,0};
uchar Count;
sbit Dot=P0^7;
//延时
void DelayMS(uint ms)
{
      uchar t;
      while(ms--) for(t=0;t<120;t++);
}
//主程序
void main()
{
      uchar i,j;
      P0=0x00;
      P3=0xff;
      Count=0;
      TMOD=0x01;                  //计数器 T0 方式 1
      TH0=(65536-50000)/256;      //50ms 定时
      TL0=(65536-50000)%256;
      IE=0x82;
      TR0=1;                      //启动 T0
      while(1)
      {
            j=0x7f;
            //显示 Digits_of_6DSY[5]~Digits_of_6DSY[0]的内容
            //前面高位，后面低位，循环中 i!=-1 亦可写成 i!=0xff
            for(i=5;i!=-1;i--)
            {
                  j=_crol_(j,1);
```

```
            P3=j;
            P0=DSY_CODE[Digits_of_6DSY[i]];
            if(i==1) Dot=1;          //加小数点
            DelayMS(2);
        }
    }
}
//T0 中断函数
void Timer0() interrupt 1
{
    uchar i;
    TH0=(65536-50000)/256;    //恢复初值
    TL0=(65536-50000)%256;
    if(++Count!=2) return;
    Count=0;
    Digits_of_6DSY[0]++;  //0.1s 位累加
    for(i=0;i<=5;i++)         //进位处理
    {
        if(Digits_of_6DSY[i]==10)
        {
            Digits_of_6DSY[i]=0;
            if(i!=5) Digits_of_6DSY[i+1]++;    //如果 0~4 位则分别向高一位进位
        }
        else break;           //若某低位没有进位，怎循环提前结束
    }
}
```

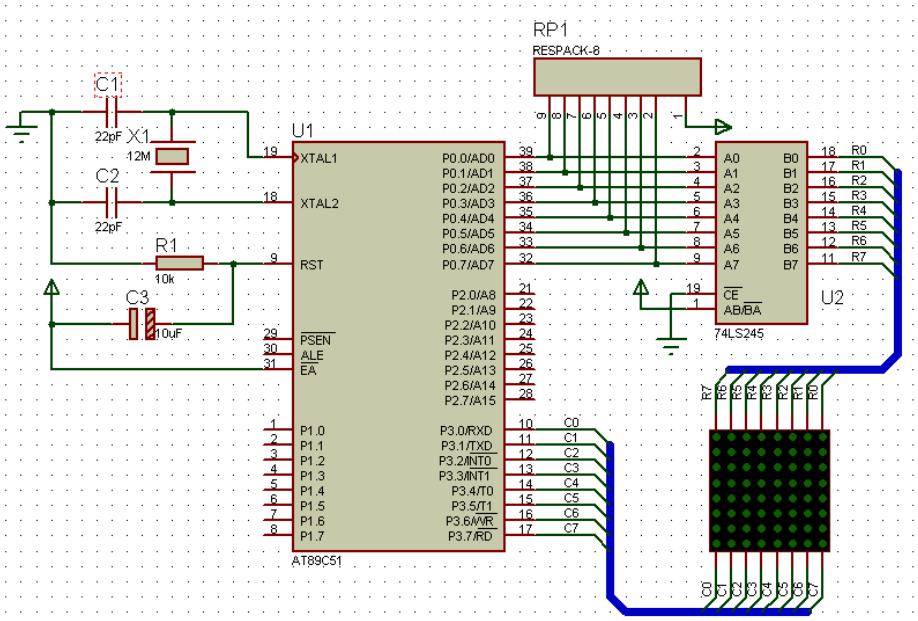
35　定时器控制数码管动态显示




```
/*    名称：定时器控制数码管动态显示
      说明：8 个数码管上分两组动态显示年月日与时分秒，本例的位显示延时用定时器实现。
*/
#include<reg51.h>
#include<intrins.h>
#define uchar unsigned char
#define uint unsigned int
//段码,最后一位是“-”的段码
uchar code DSY_CODE[]={0xc0,0xf9,0xa4,0xb0,0x99,0x92,0x82,0xf8,0x80,0x90,0xbf};
//待显示的数据：09-12-25 与 23-59-58（分两组显示）
uchar code Table_of_Digits[][8]={{0,9,10,1,2,10,2,5},{2,3,10,5,9,10,5,8}};
uchar i,j=0;
```

```
uint t=0;
//主程序
void main()
{
    P3=0x80;                //位码初值
    TMOD=0x00;              //计数器 T0 方式 0
    TH0=(8192-4000)/32;     //4ms 定时
    TL0=(8192-4000)%32;
    IE=0x82;
    TR0=1;                  //启动 T0
    while(1);
}
//T0 中断函数控制数码管刷新显示
void DSY_Show() interrupt 1
{
    TH0=(8192-4000)/32;     //恢复初值
    TL0=(8192-4000)%32;
    P0=0xff;                //输出位码和段码
    P0=DSY_CODE[Table_of_Digits[i][j]];
    P3=_crol_(P3,1);
    j=(j+1)%8;              //数组第 i 行的下一字节索引
    if(++t!=350) return;//保持刷新一段时间
    t=0;
    i=(i+1)%2;              //数组行 i=0 时显示年月日，i=1 时显示时分秒
}
```

36 8X8LED 点阵显示数字



/*　名称：8X8LED 点阵显示数字

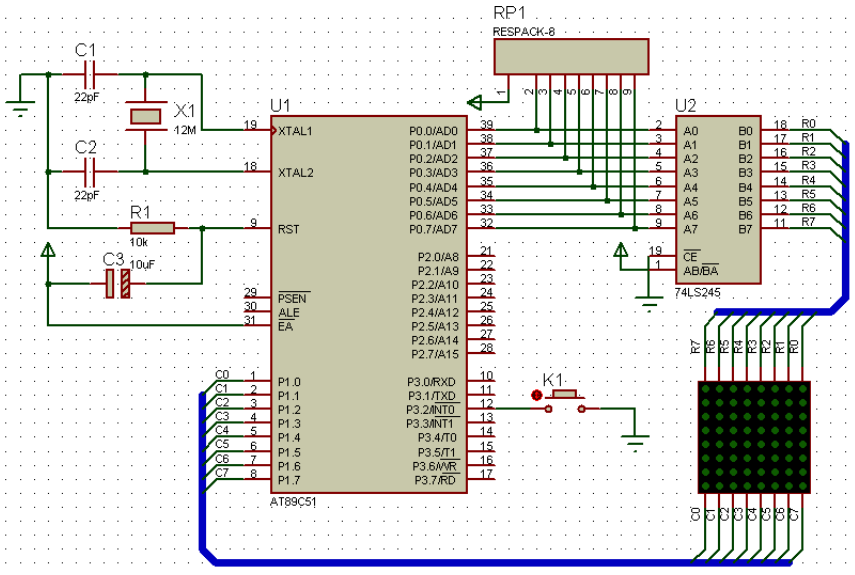
　　说明：8X8LED 点阵屏循环显示数字 0~9，刷新过程由定时器中断完成。

*/

```
#include<reg51.h>
#include<intrins.h>
#define uchar unsigned char
#define uint unsigned int
uchar code Table_of_Digits[]=
{
    0x00,0x3e,0x41,0x41,0x41,0x3e,0x00,0x00,    //0
    0x00,0x00,0x00,0x21,0x7f,0x01,0x00,0x00,    //1
    0x00,0x27,0x45,0x45,0x45,0x39,0x00,0x00,    //2
    0x00,0x22,0x49,0x49,0x49,0x36,0x00,0x00,    //3
    0x00,0x0c,0x14,0x24,0x7f,0x04,0x00,0x00,    //4
```

```
    0x00,0x72,0x51,0x51,0x51,0x4e,0x00,0x00,      //5
    0x00,0x3e,0x49,0x49,0x49,0x26,0x00,0x00,      //6
    0x00,0x40,0x40,0x40,0x4f,0x70,0x00,0x00,      //7
    0x00,0x36,0x49,0x49,0x49,0x36,0x00,0x00,      //8
    0x00,0x32,0x49,0x49,0x49,0x3e,0x00,0x00       //9
};
uchar i=0,t=0,Num_Index;
//主程序
void main()
{
    P3=0x80;
    Num_Index=0;                //从 0 开始显示
    TMOD=0x00;                  //T0 方式 0
    TH0=(8192-2000)/32;         //2ms 定时
    TL0=(8192-2000)%32;
    IE=0x82;
    TR0=1;                      //启动 T0
    while(1);
}
//T0 中断函数
void LED_Screen_Display() interrupt 1
{
    TH0=(8192-2000)/32;         //恢复初值
    TL0=(8192-2000)%32;
    P0=0xff;                    //输出位码和段码
    P0=~Table_of_Digits[Num_Index*8+i];
    P3=_crol_(P3,1);
    if(++i==8) i=0;             //每屏一个数字由 8 个字节构成
    if(++t==250)                //每个数字刷新显示一段时间
    {
        t=0;
        if(++Num_Index==10) Num_Index=0; //显示下一个数字
    }
}
```

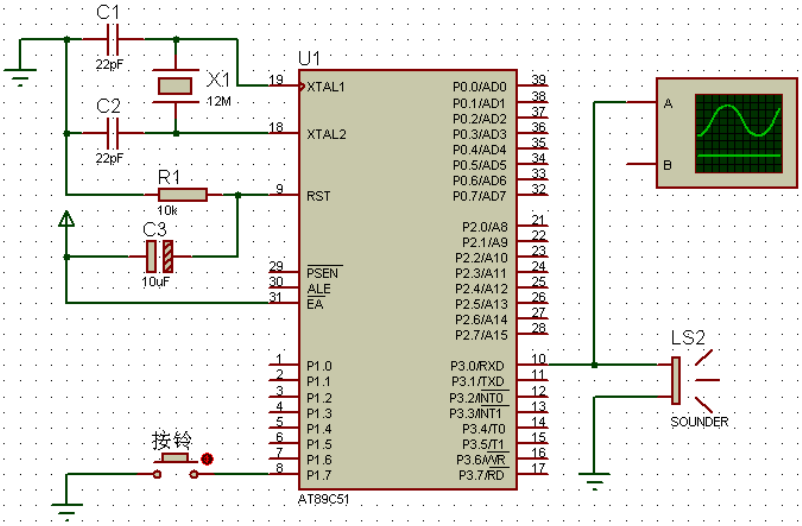
37　按键控制 8X8LED 点阵屏显示图形




```
/*    名称：按键控制 8X8LED 点阵屏显示图形
      说明：每次按下 K1 时，会使 8X8LED 点阵屏循环显示不同图形。
            本例同时使用外部中断和定时中断。
*/
#include<reg51.h>
```

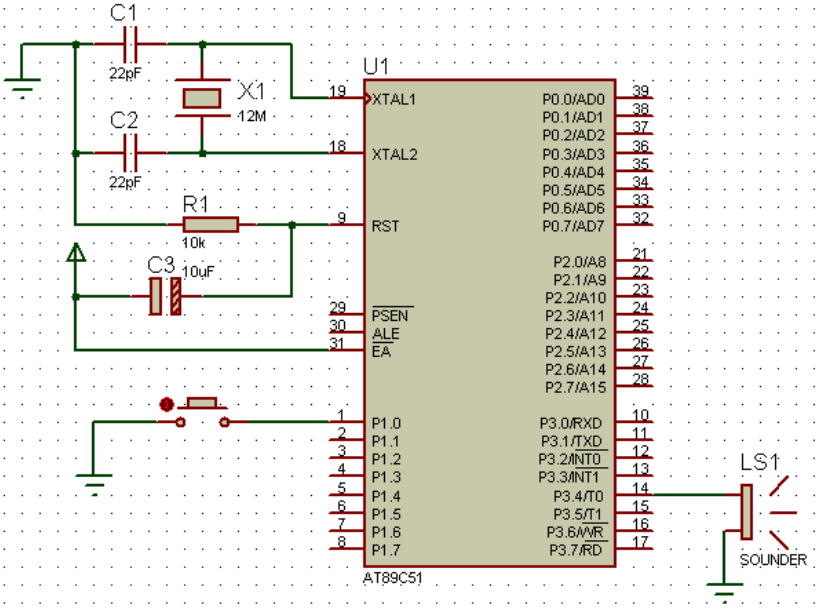
```
#include<intrins.h>
#define uchar unsigned char
#define uint unsigned int
//待显示图形编码
uchar code M[][8]=
{
    {0x00,0x7e,0x7e,0x7e,0x7e,0x7e,0x7e,0x00},        //图 1
    {0x00,0x38,0x44,0x54,0x44,0x38,0x00,0x00},        //图 2
    {0x00,0x20,0x30,0x38,0x3c,0x3e,0x00,0x00}         //图 3
};
uchar i,j;
//主程序
void main()
{
    P0=0xff;
    P1=0xff;
    TMOD=0x01;                  //T0 方式 1
    TH0=(65536-2000)/256;//2ms 定时
    TL0=(65536-2000)%256;
    IT0=1;                      //下降沿触发
    IE=0x83;                //允许定时器 0、外部 0 中断
    i=0xff;                     //i 的初值设为 0xff，加 1 后将从 0 开始
    while(1);
}
//T0 中断控制点阵屏显示
void Show_Dot_Matrix() interrupt 1
{
    TH0=(65536-2000)/256;//恢复初值
    TL0=(65536-2000)%256;
    P0=0xff;                //输出位码和段码
    P0=~M[i][j];
    P1=_crol_(P1,1);
    j=(j+1)%8;
}
//INT0 中断（定时器由键盘中断启动）
void Key_Down() interrupt 0
{
    P0=0xff;
    P1=0x80;
    j=0;
    i=(i+1)%3;                  //i 在 0，1，2 中取值，因为只要 3 个图形
    TR0=1;
}
```

38 用定时器设计的门铃




```
/*    名称：用定时器设计的门铃
      说明：按下按键时蜂鸣器发出叮咚的门铃声。
*/
#include<reg51.h>
#define uchar unsigned char
#define uint unsigned int
sbit Key=P1^7;
sbit DoorBell=P3^0;
uint p=0;
//主程序
void main()
{
      DoorBell=0;
      TMOD=0x00;              //T0 方式 0
      TH0=(8192-700)/32;      //700us 定时
      TL0=(8192-700)%32;
      IE=0x82;
      while(1)
      {
          if(Key==0)          //按下按键启动定时器
          {
              TR0=1;
              while(Key==0);
          }
      }
}
//T0 中断控制点阵屏显示
void Timer0() interrupt 1
{
      DoorBell=~DoorBell;
      p++;
      if(p<400)               //若需要拖长声音，可以调整 400 和 800
      {
          TH0=(8192-700)/32;      //700us 定时
          TL0=(8192-700)%32;
      }
      else if(p<800)
      {
          TH0=(8192-1000)/32;    //1ms 定时
          TL0=(8192-1000)%32;
      }
      else
      {
          TR0=0;
```

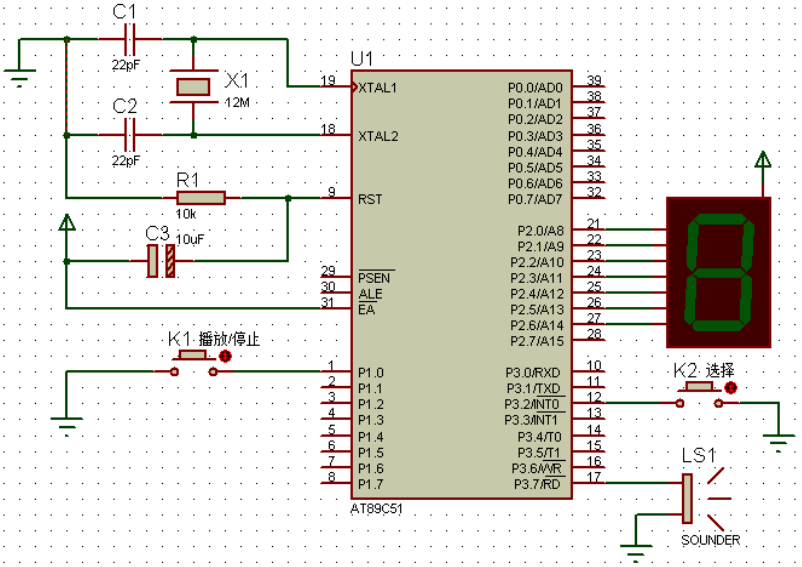
```
        p=0;
    }
}
```



39 演奏音阶

```
/*    名称：演奏音阶
      说明：本例使用定时器演奏一段音阶，播放由 K1 控制。
*/
#include<reg51.h>
#define uchar unsigned char
#define uint unsigned int
sbit K1=P1^0;
sbit SPK=P3^4;
uint i=0;                    //音符索引
//14 个音符放在方式 2 下的定时寄存器（TH0,TL0）
uchar code HI_LIST[]={0,226,229,232,233,236,238,240,241,242,244,245,246,247,248};
uchar code LO_LIST[]={0,4,13,10,20,3,8,6,2,23,5,26,1,4,3};
//定时器 0 中断函数
void T0_INT() interrupt 1
{
    TL0=LO_LIST[i];
    TH0=HI_LIST[i];
    SPK=~SPK;
}
//延时
void DelayMS(uint ms)
{
    uchar t;
    while(ms--) for(t=0;t<120;t++);
}
//主程序
void main()
{
    TMOD=0x00;               //T0 方式 0
    IE=0x82;
    SPK=0;
    while(1)
    {
        while(K1==1);        //未按键等待
        while(K1==0);        //等待释放
        for(i=1;i<15;i++)
        {
```

```
            TR0=1;              //播放一个音符
            DelayMS(500);       //播放延时
            TR0=0;
            DelayMS(50);
        }
    }
}
```



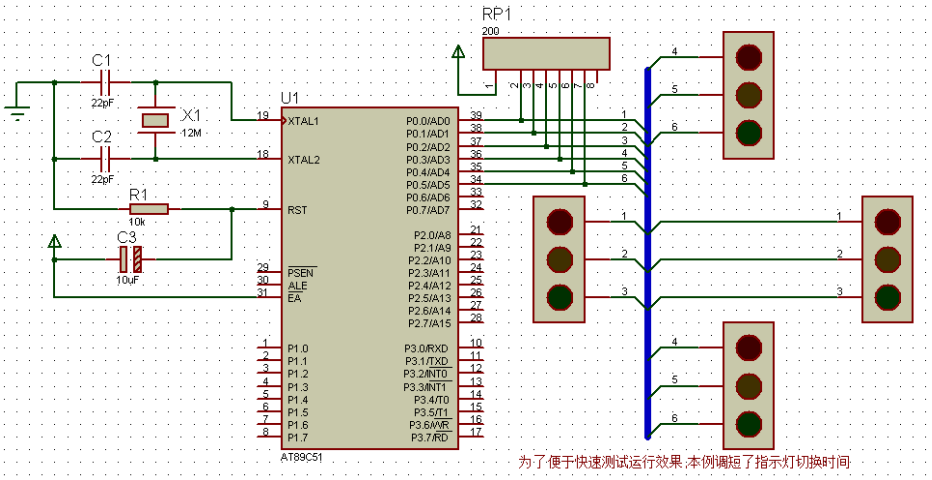
40 按键控制定时器选播多段音乐

```
/*  名称：按键控制定时器选播多段音乐
    说明：本例内置 3 段音乐，K1 可启动停止音乐播放，K2 用于选择音乐段。
*/
#include<reg51.h>
#include<intrins.h>
#define uchar unsigned char
#define uint unsigned int
sbit K1=P1^0;                   //播放和停止键
sbit SPK=P3^7;                  //蜂鸣器
uchar Song_Index=0,Tone_Index=0;    //当前音乐段索引，音符索引
//数码管段码表
uchar code DSY_CODE[]={0xc0,0xf9,0xa4,0xb0,0x99,0x92,0x82,0xf8,0x80,0x90};
//标准音符频率对应的延时表
uchar code HI_LIST[]={0,226,229,232,233,236,238,240,241,242,244,245,246,247,248};
uchar code LO_LIST[]={0,4,13,10,20,3,8,6,2,23,5,26,1,4,3};
//三段音乐的音符
uchar code Song[][50]=
{
    {1,2,3,1,1,2,3,1,3,4,5,3,4,5,5,6,5,3,5,6,5,3,5,3,2,1,2,1,-1},
    {3,3,3,4,5,5,5,5,6,5,3,5,3,2,1,5,6,53,3,2,1,1,-1},
    {3,2,1,3,2,1,1,2,3,1,1,2,3,1,3,4,5,3,4,5,5,6,5,3,5,3,2,1,3,2,1,1,-1}
};
//三段音乐的节拍
uchar code Len[][50]=
{
    {1,1,1,1,1,1,1,1,1,1,2,1,1,2,1,1,1,1,1,1,1,1,1,1,1,2,1,2,-1},
    {1,1,1,1,1,1,2,1,1,1,1,1,1,1,2,1,1,1,1,1,1,2,2,-1},
    {1,1,2,1,1,2,1,1,1,1,1,1,1,1,1,2,1,1,2,1,1,1,1,1,1,1,2,1,1,2,2,-1}
};
//外部中断 0
void EX0_INT() interrupt 0
{
    TR0=0;  //播放结束或者播放中途切换歌曲时停止播放
    Song_Index=(Song_Index+1)%3; //跳到下一首的开头
```

```
    Tone_Index=0;
    P2=DSY_CODE[Song_Index];          //数码管显示当前音乐段号
}
//定时器 0 中断函数
void T0_INT() interrupt 1
{
    TL0=LO_LIST[Song[Song_Index][Tone_Index]];
    TH0=HI_LIST[Song[Song_Index][Tone_Index]];
    SPK=~SPK;
}
//延时
void DelayMS(uint ms)
{
    uchar t;
    while(ms--) for(t=0;t<120;t++);
}
//主程序
void main()
{
    P2=0xc0;
    SPK=0;
    TMOD=0x00;                //T0 方式 0
    IE=0x83;
    IT0=1;
    IP=0x02;
    while(1)
    {
        while(K1==1);         //未按键等待
        while(K1==0);         //等待释放
        TR0=1;                //开始播放
        Tone_Index=0;         //从第 0 个音符开始
        //播放过程中按下 K1 可提前停止播放（K1=0）。
        //若切换音乐段会触发外部中断，导致 TR0=0，播放也会停止
        while(Song[Song_Index][Tone_Index]!=-1&&K1==1&&TR0==1)
        {
            DelayMS(300*Len[Song_Index][Tone_Index]); //播放延时（节拍）
            Tone_Index++;         //当前音乐段的下一音符索引
        }
        TR0=0;            //停止播放
        while(K1==0);     //若提前停止播放，按键未释放时等待
    }
}
```

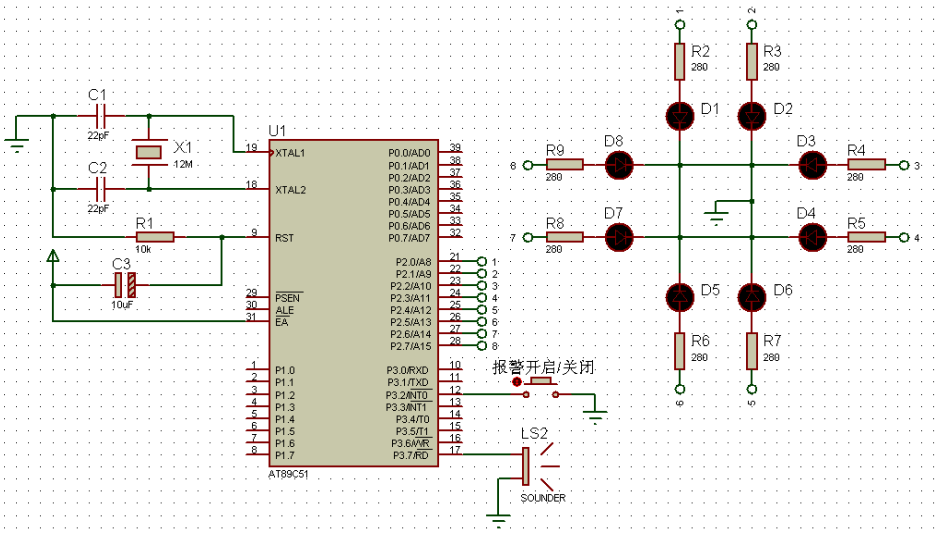
41 定时器控制交通指示灯

```
/*    名称：定时器控制交通指示灯
      说明：东西向绿灯亮 5s 后，黄灯闪烁，闪烁 5 次亮红灯，
            红灯亮后，南北向由红灯变成绿灯，5s 后南北向黄灯闪烁，
            闪烁 5 次后亮红灯，东西向绿灯亮，如此往复。
*/
#include<reg51.h>
#define uchar unsigned char
#define uint unsigned int
sbit RED_A=P0^0; //东西向指示灯
sbit YELLOW_A=P0^1;
sbit GREEN_A=P0^2;
sbit RED_B=P0^3; //南北向指示灯
sbit YELLOW_B=P0^4;
sbit GREEN_B=P0^5;
//延时倍数，闪烁次数，操作类型变量
```




```
uchar Time_Count=0,Flash_Count=0,Operation_Type=1;
//定时器 0 中断函数
void T0_INT() interrupt 1
{
    TL0=-50000/256;
    TH0=-50000%256;
    switch(Operation_Type)
    {
        case 1:   //东西向绿灯与南北向红灯亮 5s
                  RED_A=0;YELLOW_A=0;GREEN_A=1;
                  RED_B=1;YELLOW_B=0;GREEN_B=0;
                  if(++Time_Count!=100) return; //5s（100*50ms）切换
                  Time_Count=0;
                  Operation_Type=2;
                  break;
        case 2:   //东西向黄灯开始闪烁，绿灯关闭
                  if(++Time_Count!=8) return;
                  Time_Count=0;
                  YELLOW_A=~YELLOW_A;GREEN_A=0;
                  if(++Flash_Count!=10) return;     //闪烁
                  Flash_Count=0;
                  Operation_Type=3;
                  break;
        case 3:   //东西向红灯与南北向绿灯亮 5s
                  RED_A=1;YELLOW_A=0;GREEN_A=0;
                  RED_B=0;YELLOW_B=0;GREEN_B=1;
                  if(++Time_Count!=100) return; //5s（100*50ms）切换
                  Time_Count=0;
                  Operation_Type=4;
```

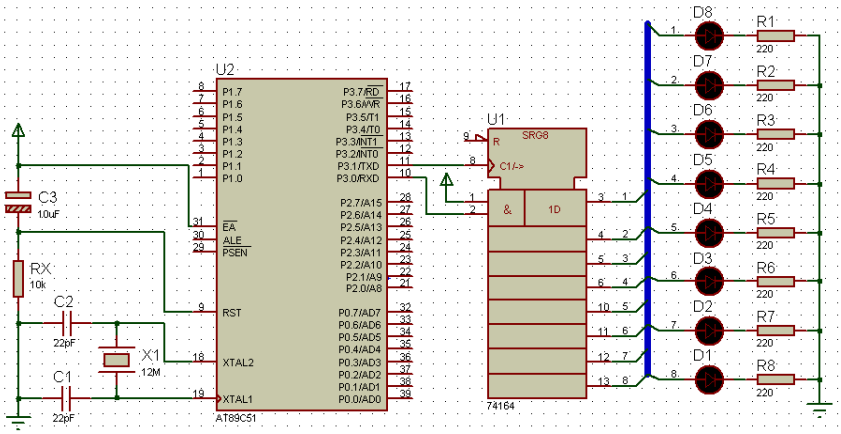
```
                break;
        case 4:  //南北向黄灯开始闪烁，绿灯关闭
                if(++Time_Count!=8) return;
                Time_Count=0;
                YELLOW_B=~YELLOW_B;GREEN_A=0;
                if(++Flash_Count!=10) return;      //闪烁
                Flash_Count=0;
                Operation_Type=1;
                break;
    }
}
//主程序
void main()
{
    TMOD=0x01;                  //T0 方式 1
    IE=0x82;
    TR0=1;
    while(1);
}
```



42　报警与旋转灯

```
/*    名称：报警与旋转灯
      说明：定时器控制报警灯旋转显示，并发出仿真警报声。
*/
#include<reg51.h>
#include<intrins.h>
#define uchar unsigned char
#define uint unsigned int
sbit SPK=P3^7;
uchar FRQ=0x00;
//延时
void DelayMS(uint ms)
{
    uchar i;
    while(ms--) for(i=0;i<120;i++);
}

//INT0 中断函数
void EX0_INT() interrupt 0
{
    TR0=~TR0;   //开启或停止两定时器，分别控制报警器的声音和 LED 旋转
    TR1=~TR1;
    if(P2==0x00)
```

```
            P2=0xe0;//开 3 个旋转灯
        else
            P2=0x00;//关闭所有 LED
}
//定时器 0 中断
void T0_INT() interrupt 1
{
    TH0=0xfe;
    TL0=FRQ;
    SPK=~SPK;
}
//定时器 1 中断
void T1_INT() interrupt 3
{
    TH1=-45000/256;
    TL1=-45000%256;
    P2=_crol_(P2,1);
}
//主程序
void main()
{
    P2=0x00;
    SPK=0x00;
    TMOD=0x11;      //T0、T1 方式 1
    TH0=0x00;
    TL0=0xff;
    IT0=1;
    IE=0x8b;        //开启 0，1，3 号中断
    IP=0x01;        //INT0 设为最高优先
    TR0=0;
    TR1=0;              //定时器启停由 INT0 控制，初始关闭
    while(1)
    {
        FRQ++;
        DelayMS(1);
    }
}
```

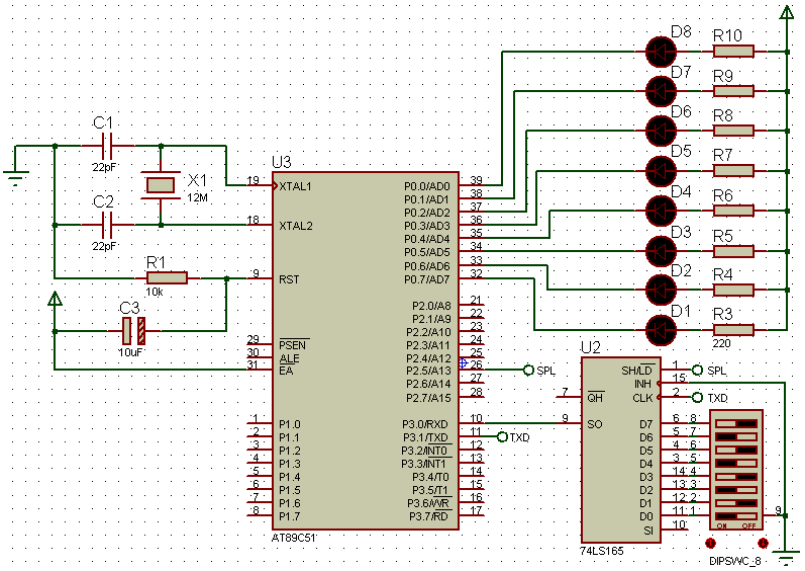


43 串行数据转换为并行数据

```
/*  名称：串行数据转换为并行数据
    说明：串行数据由 RXD 发送给串并转换芯片 74164，TXD 则用于输出移位时钟脉冲，74164 将串行输入的 1 字节转换为并行数据，并将转换的数据通过 8 只 LED 显示出来。本例串口工作模式 0，即移位寄存器 I/O 模式。
*/
```

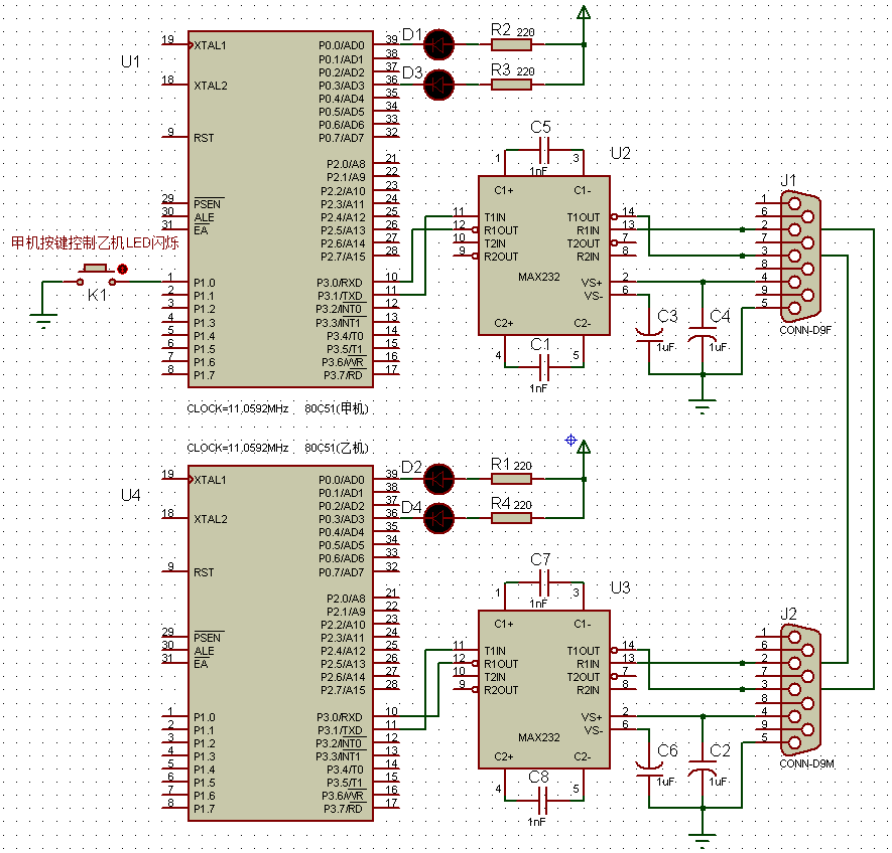
```
#include<reg51.h>
#include<intrins.h>
#define uchar unsigned char
#define uint unsigned int
sbit SPK=P3^7;
uchar FRQ=0x00;
//延时
void DelayMS(uint ms)
{
    uchar i;
    while(ms--) for(i=0;i<120;i++);
}
//主程序
void main()
{
    uchar c=0x80;
    SCON=0x00;      //串口模式 0，即移位寄存器输入/输出方式
    TI=1;
    while(1)
    {
        c=_crol_(c,1);
        SBUF=c;
        while(TI==0);      //等待发送结束
        TI=0;                    //TI 软件置位
        DelayMS(400);
    }
}
```

44 并行数据转换为串行数据




```
/*    名称：并行数据转换为串行数据
      说明：切换连接到并串转换芯片 74LS165 的拨码开关，该芯片将并行数据以串行方式发送到 8051 的 RXD 引脚，移位脉冲由 TXD 提供，显示在 P0 口。
*/
#include<reg51.h>
#include<intrins.h>
#include<stdio.h>
#define uchar unsigned char
#define uint unsigned int
sbit SPL=P2^5;          //shift/load
//延时
void DelayMS(uint ms)
{
```

```
    uchar i;
    while(ms--) for(i=0;i<120;i++);
}
//主程序
void main()
{
    SCON=0x10;        //串口模式 0，允许串口接收
    while(1)
    {
        SPL=0;        //置数(load)，读入并行输入口的 8 位数据
        SPL=1;        //移位(shift)，并口输入被封锁，串行转换开始
        while(RI==0); //未接收 1 字节时等待
        RI=0;                 //RI 软件置位
        P0=SBUF;              //接收到的数据显示在 P0 口，显示拨码开关的值
        DelayMS(20);
    }
}
```



45 甲机通过串口控制乙机 LED

```
/*  名称：甲机发送控制命令字符
    说明：甲单片机负责向外发送控制命令字符“A”、“B”、“C”，或者停止发送，乙机根据所接收到的字符完成 LED1 闪烁、LED2 闪烁、双闪烁、或停止闪烁。
*/
#include<reg51.h>
#define uchar unsigned char
#define uint unsigned int
sbit LED1=P0^0;
sbit LED2=P0^3;
sbit K1=P1^0;
//延时
void DelayMS(uint ms)
{
    uchar i;
    while(ms--) for(i=0;i<120;i++);
}
//向串口发送字符
void Putc_to_SerialPort(uchar c)
{
    SBUF=c;
    while(TI==0);
    TI=0;
```

```
}
//主程序
void main()
{
    uchar Operation_No=0;
    SCON=0x40;        //串口模式 1
    TMOD=0x20;        //T1 工作模式 2
    PCON=0x00;        //波特率不倍增
    TH1=0xfd;
    TL1=0xfd;
    TI=0;
    TR1=1;
    while(1)
    {
        if(K1==0)      //按下 K1 时选择操作代码 0，1，2，3
        {
            while(K1==0);
            Operation_No=(Operation_No+1)%4;
        }
        switch(Operation_No)   //根据操作代码发送 A/B/C 或停止发送
        {
            case 0:   LED1=LED2=1;
                      break;
            case 1:   Putc_to_SerialPort('A');
                      LED1=~LED1;LED2=1;
                      break;
            case 2:   Putc_to_SerialPort('B');
                      LED2=~LED2;LED1=1;
                      break;
            case 3:   Putc_to_SerialPort('C');
                      LED1=~LED1;LED2=LED1;
                      break;
        }
        DelayMS(100);
    }
}

/*   名称：乙机程序接收甲机发送字符并完成相应动作
     说明：乙机接收到甲机发送的信号后，根据相应信号控制 LED 完成不同闪烁动作。
*/
#include<reg51.h>
#define uchar unsigned char
#define uint unsigned int
sbit LED1=P0^0;
sbit LED2=P0^3;
```

```
//延时
void DelayMS(uint ms)
{
    uchar i;
    while(ms--) for(i=0;i<120;i++);
}
//主程序
void main()
{
    SCON=0x50;        //串口模式 1，允许接收
    TMOD=0x20;        //T1 工作模式 2
    PCON=0x00;        //波特率不倍增
    TH1=0xfd;         //波特率 9600
    TL1=0xfd;
    RI=0;
    TR1=1;
    LED1=LED2=1;
    while(1)
    {
        if(RI)     //如收到则 LED 闪烁
        {
            RI=0;
            switch(SBUF) //根据所收到的不同命令字符完成不同动作
            {
                case 'A':  LED1=~LED1;LED2=1;break;    //LED1 闪烁
                case 'B':  LED2=~LED2;LED1=1;break;    //LED2 闪烁
                case 'C':  LED1=~LED1;LED2=LED1;          //双闪烁
            }
        }
        else LED1=LED2=1;                              //关闭 LED
        DelayMS(100);
    }
}
```

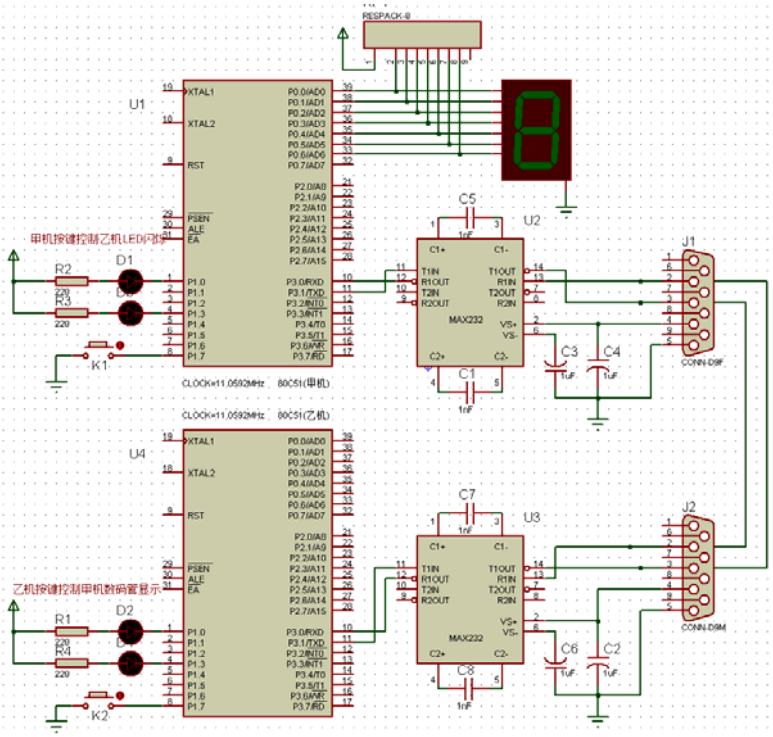
46　单片机之间双向通信

/*　名称：甲机串口程序

说明：甲机向乙机发送控制命令字符，甲机同时接收乙机发送的数字，并显示在数码管上。

*/

```
#include<reg51.h>
#define uchar unsigned char
#define uint unsigned int
sbit LED1=P1^0;
```

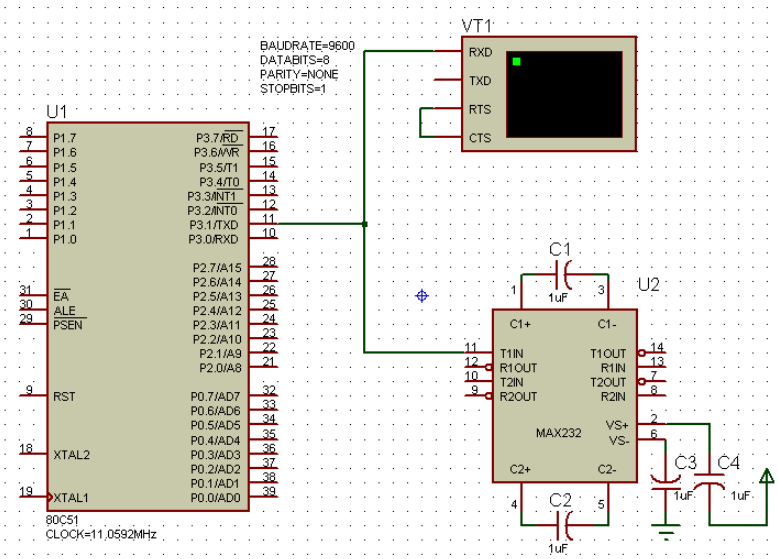


```
sbit LED2=P1^3;
sbit K1=P1^7;
uchar Operation_No=0; //操作代码
//数码管代码
uchar code DSY_CODE[]={0x3f,0x06,0x5b,0x4f,0x66,0x6d,0x7d,0x07,0x7f,0x6f};
//延时
void DelayMS(uint ms)
{
    uchar i;
    while(ms--) for(i=0;i<120;i++);
}
//向串口发送字符
void Putc_to_SerialPort(uchar c)
{
    SBUF=c;
    while(TI==0);
    TI=0;
}
//主程序
void main()
{
    LED1=LED2=1;
    P0=0x00;
    SCON=0x50;      //串口模式 1，允许接收
    TMOD=0x20;      //T1 工作模式 2
    PCON=0x00;      //波特率不倍增
    TH1=0xfd;
    TL1=0xfd;
    TI=RI=0;
    TR1=1;
    IE=0x90;      //允许串口中断
    while(1)
    {
        DelayMS(100);
        if(K1==0)     //按下 K1 时选择操作代码 0，1，2，3
        {
            while(K1==0);
            Operation_No=(Operation_No+1)%4;

            switch(Operation_No)   //根据操作代码发送 A/B/C 或停止发送
            {
                case 0:   Putc_to_SerialPort('X');
                          LED1=LED2=1;
                          break;
                case 1:   Putc_to_SerialPort('A');
```

```
                    LED1=~LED1;LED2=1;
                    break;
                case 2:  Putc_to_SerialPort('B');
                    LED2=~LED2;LED1=1;
                    break;
                case 3:  Putc_to_SerialPort('C');
                    LED1=~LED1;LED2=LED1;
                    break;
            }
        }
    }
}
//甲机串口接收中断函数
void Serial_INT() interrupt  4
{
    if(RI)
    {
        RI=0;
        if(SBUF>=0&&SBUF<=9) P0=DSY_CODE[SBUF];
        else P0=0x00;
    }
}

/*  名称：乙机程序接收甲机发送字符并完成相应动作
    说明：乙机接收到甲机发送的信号后，根据相应信号控制 LED 完成不同闪烁动作。
*/
#include<reg51.h>
#define uchar unsigned char
#define uint unsigned int
sbit LED1=P1^0;
sbit LED2=P1^3;
sbit K2=P1^7;
uchar NumX=-1;
//延时
void DelayMS(uint ms)
{
    uchar i;
    while(ms--) for(i=0;i<120;i++);
}
//主程序
void main()
{
    LED1=LED2=1;
    SCON=0x50;      //串口模式 1，允许接收
    TMOD=0x20;      //T1 工作模式 2
```

```
    TH1=0xfd;          //波特率 9600
    TL1=0xfd;
    PCON=0x00;         //波特率不倍增
    RI=TI=0;
    TR1=1;
    IE=0x90;
    while(1)
    {
        DelayMS(100);
        if(K2==0)
        {
            while(K2==0);
            NumX=++NumX%11;  //产生 0~10 范围内的数字，其中 10 表示关闭
            SBUF=NumX;
            while(TI==0);
            TI=0;
        }
    }
}
void Serial_INT() interrupt 4
{
    if(RI)     //如收到则 LED 则动作
    {
        RI=0;
        switch(SBUF) //根据所收到的不同命令字符完成不同动作
        {
            case 'X':  LED1=LED2=1;break;        //全灭
            case 'A':  LED1=0;LED2=1;break;      //LED1 亮
            case 'B':  LED2=0;LED1=1;break;      //LED2 亮
            case 'C':  LED1=LED2=0;              //全亮
        }
    }
}
```

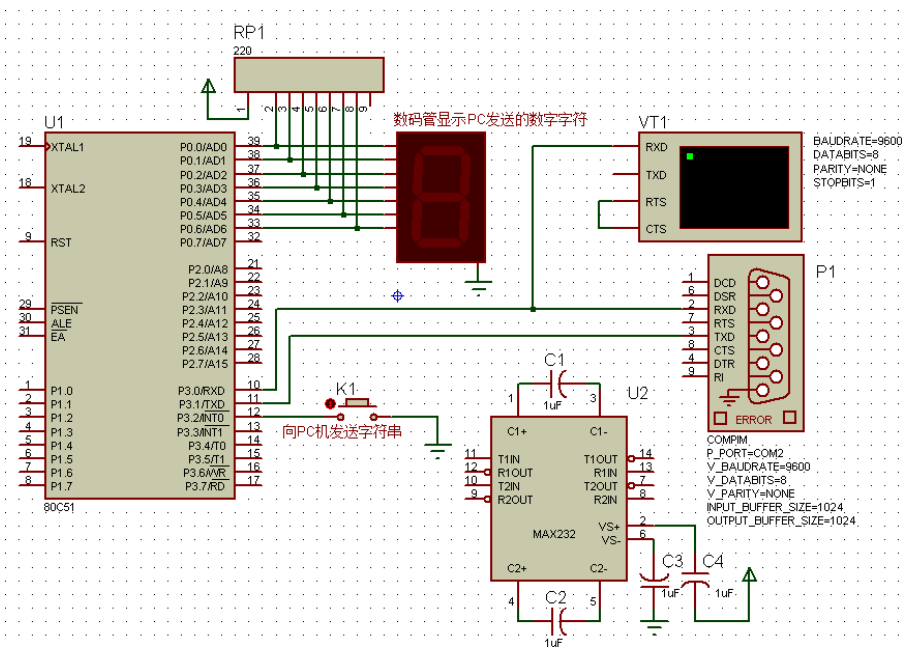
47　单片机向主机发送字符串




```
/*    名称：单片机向主机发送字符串
      说明：单片机按一定的时间间隔向主机发送字符串，发送内容在虚拟终端显示。
*/
#include<reg51.h>
#define uchar unsigned char
#define uint unsigned int
//延时
void DelayMS(uint ms)
```

```
{
    uchar i;
    while(ms--) for(i=0;i<120;i++);
}
//向串口发送字符
void Putc_to_SerialPort(uchar c)
{
    SBUF=c;
    while(TI==0);
    TI=0;
}
//向串口发送字符串
void Puts_to_SerialPort(uchar *s)
{
    while(*s!='\0')
    {
        Putc_to_SerialPort(*s);
        s++;
        DelayMS(5);
    }
}
//主程序
void main()
{
    uchar c=0;
    SCON=0x40;      //串口模式 1
    TMOD=0x20;      //T1 工作模式 2
    TH1=0xfd;       //波特率 9600
    TL1=0xfd;
    PCON=0x00;      //波特率不倍增
    TI=0;
    TR1=1;
    DelayMS(200);
    //向主机发送数据
    Puts_to_SerialPort("Receiving From 8051...\r\n");
    Puts_to_SerialPort("------------------------------\r\n");
    DelayMS(50);
    while(1)
    {
        Putc_to_SerialPort(c+'A');
        DelayMS(100);
        Putc_to_SerialPort(' ');
        DelayMS(100);
        if(c==25)//每输出一遍后加横线
        {
```

```
            Puts_to_SerialPort("\r\n------------------------------\r\n");
            DelayMS(100);
        }
        c=(c+1)%26;
        if(c%10==0)         //每输出 10 个字符后换行
        {
            Puts_to_SerialPort("\r\n");
            DelayMS(100);

        }
    }
}
```



48 单片机与 PC 通信

/* 名称：单片机与 PC 通信

说明：单片机可接收 PC 发送的数字字符，按下单片机的 K1 键后，单片机可向 PC 发送字符串。在 Proteus 环境下完成本实验时，需要安装 Virtual Serial Port Driver 和串口调试助手。本例缓冲 100 个数字字符，缓冲满后新数字从前面开始存放（环形缓冲）。

*/

```
#include<reg51.h>
#define uchar unsigned char
#define uint unsigned int
uchar Receive_Buffer[101];          //接收缓冲
uchar Buf_Index=0;                  //缓冲空间索引
//数码管编码
uchar code DSY_CODE[]={0x3f,0x06,0x5b,0x4f,0x66,0x6d,0x7d,0x07,0x7f,0x6f,0x00};
//延时
void DelayMS(uint ms)
{
    uchar i;
    while(ms--) for(i=0;i<120;i++);
}
//主程序
void main()
{
    uchar i;
    P0=0x00;
    Receive_Buffer[0]=-1;
    SCON=0x50;          //串口模式 1，允许接收
```

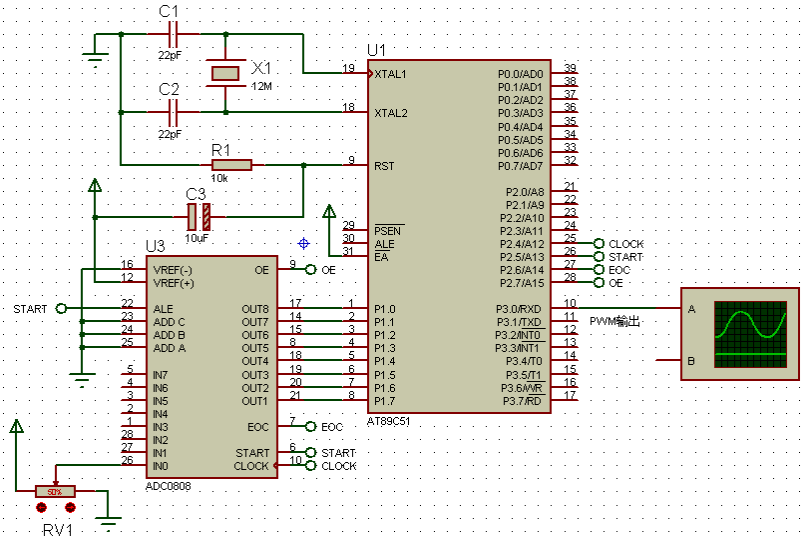
```
    TMOD=0x20;      //T1 工作模式 2
    TH1=0xfd;       //波特率 9600
    TL1=0xfd;
    PCON=0x00;      //波特率不倍增
    EA=1;EX0=1;IT0=1;
    ES=1;IP=0x01;
    TR1=1;
    while(1)
    {
        for(i=0;i<100;i++)
        {   //收到-1 为一次显示结束
            if(Receive_Buffer[i]==-1) break;
            P0=DSY_CODE[Receive_Buffer[i]];
            DelayMS(200);
        }
        DelayMS(200);
    }
}
//串口接收中断函数
void Serial_INT() interrupt 4
{
    uchar c;
    if(RI==0) return;
    ES=0;                   //关闭串口中断
    RI=0;                   //清接收中断标志
    c=SBUF;
    if(c>='0'&&c<='9')
    {   //缓存新接收的每个字符，并在其后放-1 为结束标志
        Receive_Buffer[Buf_Index]=c-'0';
        Receive_Buffer[Buf_Index+1]=-1;
        Buf_Index=(Buf_Index+1)%100;
    }
    ES=1;
}
void EX_INT0() interrupt 0      //外部中断 0
{
    uchar *s="这是由 8051 发送的字符串！ \r\n";
    uchar i=0;
    while(s[i]!='\0')
    {
        SBUF=s[i];
        while(TI==0);
        TI=0;
        i++;
    }
```

```
}
```

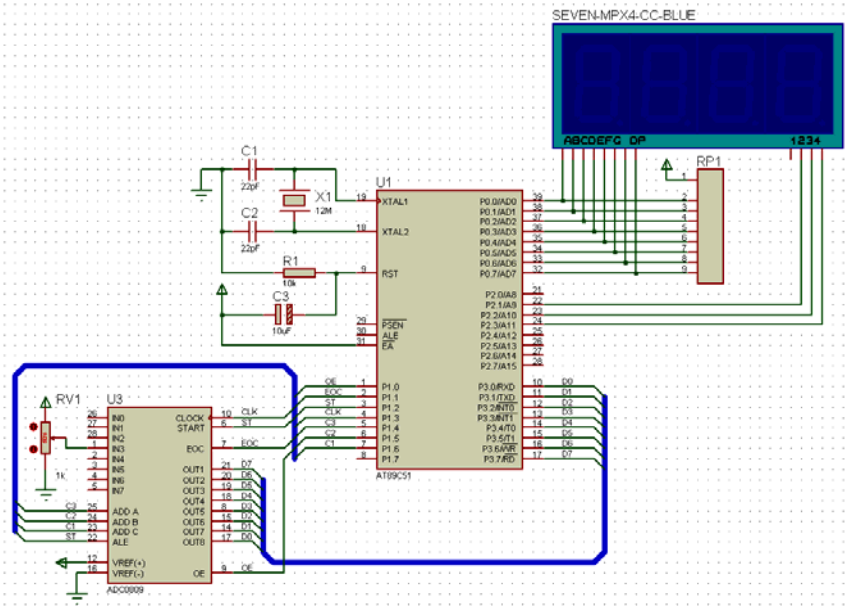
第 02 篇　硬件应用

19　用 ADC0808 控制 PWM 输出

```
/*    名称：用 ADC0808 控制 PWM 输出
      说明：使用数模转换芯片 ADC0808，通过调节可变电阻 RV1 来调节脉冲宽度，
      运行程序时，通过虚拟示波器观察占空比的变化。
*/
#include<reg51.h>
#define uchar unsigned char
#define uint unsigned int
sbit CLK=P2^4;          //时钟信号
sbit ST=P2^5;       //启动信号
sbit EOC=P2^6;          //转换结束信号
sbit OE=P2^7;       //输出使能
sbit PWM=P3^0;          //PWM 输出
//延时
void DelayMS(uint ms)
{
    uchar i;
    while(ms--) for(i=0;i<40;i++);
}
//主程序
void main()
{
    uchar Val;
    TMOD=0x02;      //T1 工作模式 2
    TH0=0x14;
    TL0=0x00;
    IE=0x82;
    TR0=1;
    while(1)
    {
        ST=0;ST=1;ST=0;     //启动 A/D 转换
        while(!EOC);        //等待转换完成
        OE=1;
        Val=P1;             //读转换值
        OE=0;
        if(Val==0)          //PWM 输出（占空比为 0%）
        {
            PWM=0;
            DelayMS(0xff);
            continue;
```



```
        }
        if(Val==0xff)          //PWM 输出（占空比为 100%）
        {
            PWM=1;
            DelayMS(0xff);
            continue;
        }
        PWM=1;                 //PWM 输出（占空比为 0%~100%）
        DelayMS(Val);
        PWM=0;
        DelayMS(0xff-Val);
    }
}
//T0 定时器中断给 ADC0808 提供时钟信号
void Timer0_INT() interrupt 1
{
    CLK=~CLK;
}
```



20 ADC0809 数模转换与显示

```
/*  名称：ADC0809 数模转换与显示
    说明：ADC0809 采样通道 3 输入的模拟量，转换后的结果显示在数码管上。
*/
#include<reg51.h>
#define uchar unsigned char
#define uint unsigned int
//各数字的数码管段码（共阴）
uchar code DSY_CODE[]={0x3f,0x06,0x5b,0x4f,0x66,0x6d,0x7d,0x07,0x7f,0x6f};
sbit CLK=P1^3;          //时钟信号
sbit ST=P1^2;       //启动信号
sbit EOC=P1^1;          //转换结束信号
sbit OE=P1^0;       //输出使能
//延时
void DelayMS(uint ms)
{
    uchar i;
    while(ms--) for(i=0;i<120;i++);
}
//显示转换结果
void Display_Result(uchar d)
{
    P2=0xf7;        //第 4 个数码管显示个位数
    P0=DSY_CODE[d%10];
```

```
    DelayMS(5);
    P2=0xfb;        //第 3 个数码管显示十位数
    P0=DSY_CODE[d%100/10];
    DelayMS(5);
    P2=0xfd;        //第 2 个数码管显示百位数
    P0=DSY_CODE[d/100];
    DelayMS(5);
}
//主程序
void main()
{
    TMOD=0x02;        //T1 工作模式 2
    TH0=0x14;
    TL0=0x00;
    IE=0x82;
    TR0=1;
    P1=0x3f;        //选择 ADC0809 的通道 3（0111）（P1.4~P1.6）
    while(1)
    {
        ST=0;ST=1;ST=0;        //启动 A/D 转换
        while(EOC==0);        //等待转换完成
        OE=1;
        Display_Result(P3);
        OE=0;
    }
}
//T0 定时器中断给 ADC0808 提供时钟信号
void Timer0_INT() interrupt 1
{
    CLK=~CLK;
}
```