

## 《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()

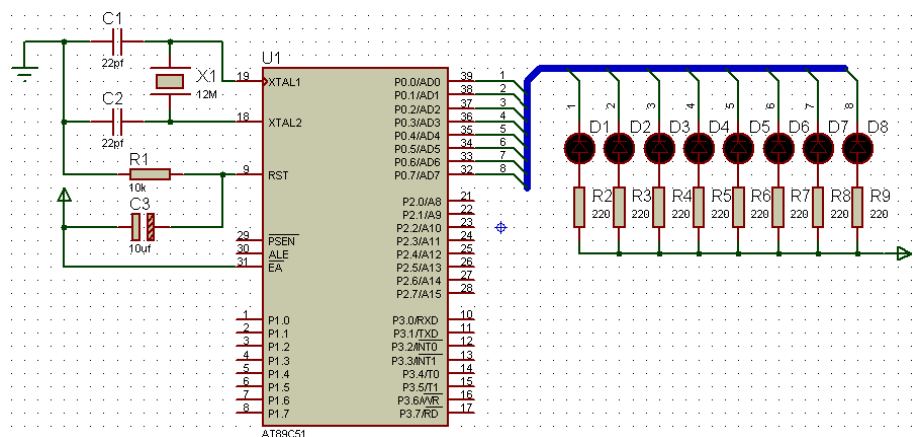
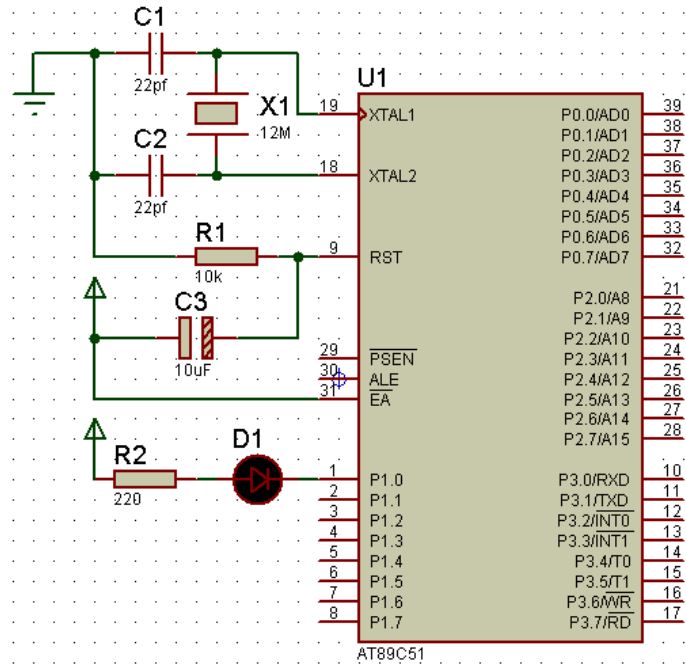
```
{
    while(1)
    {
        LED=~LED;
        DelayMS(150);
    }
}
```

## 02 从左到右的流水灯

/\*

名称:从左到右的流水灯

工程师社群371516244



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

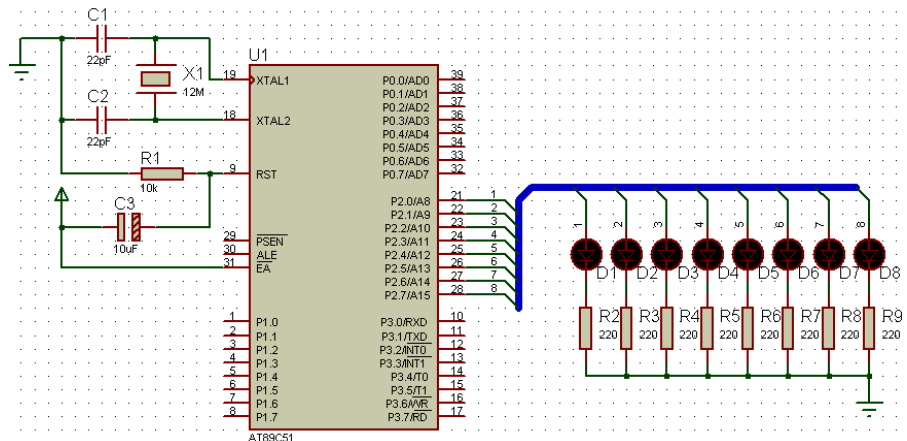
```

*/
#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

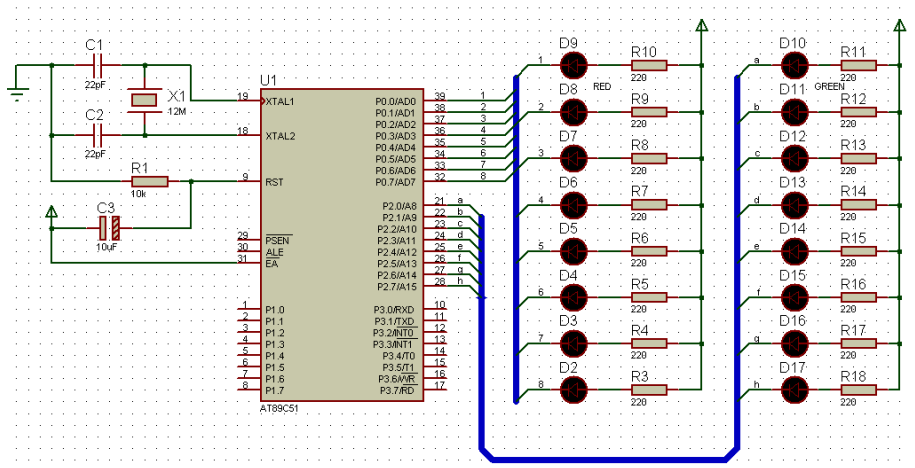
#define uint unsigned int

uchar code Pattern\_P0[] =

{

0xfc,0xf9,0xf3,0xe7,0xcf,0x9f,0x3f,0x7f,0xff,0xff,0xff,0xff,0xff,0xff,0xff,

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,0x7f,0xbf,0xdf,0xef,0xf7,0xfb,0xfd,0xfe,
0xfe,0xfc,0xf8,0xf0,0xe0,0xc0,0x80,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x80,0xc0,0xe0,0xf0,0xf8,0xfc,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,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
```

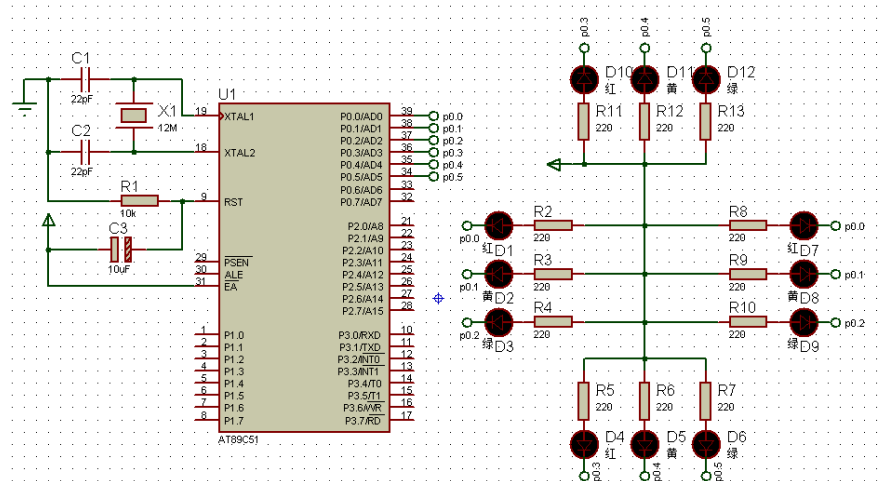
```
void DelayMS(uint x)
```

```
{
    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)
    { /* for(i<11;i++){
        P0=~DSY_CODE[i]; DelayMS(300);} //注:另一方案 */
        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>
```

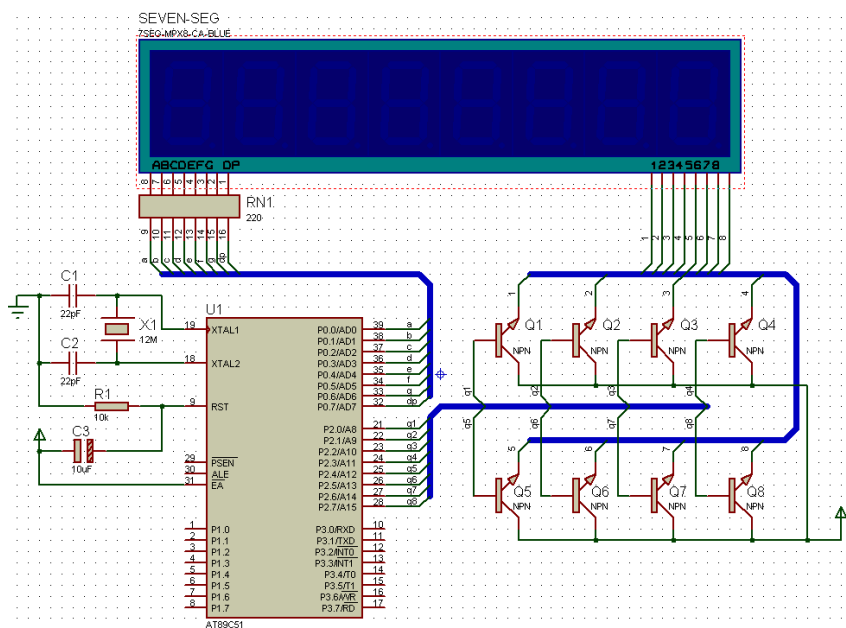
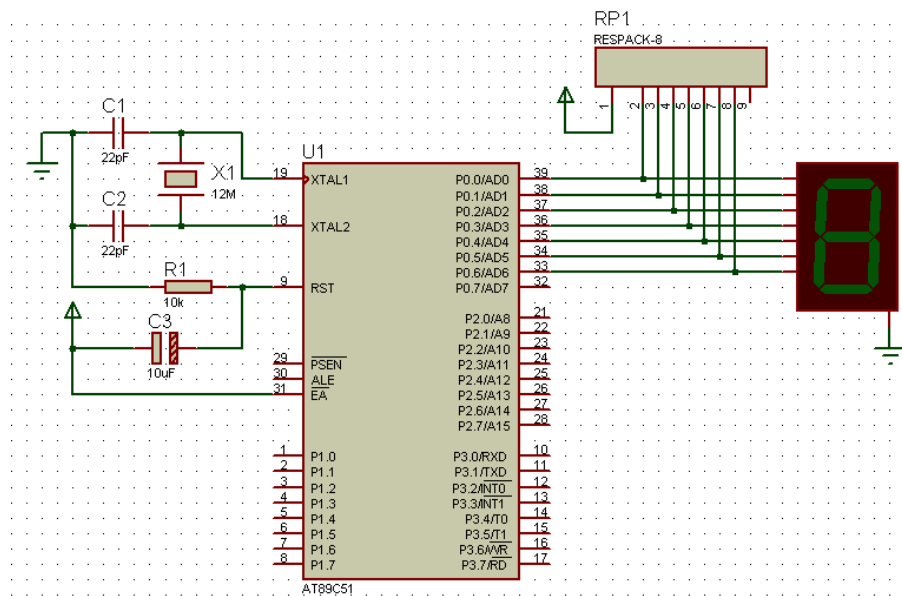
```
#include<intrins.h>
```

```
#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++)
        {
            P2=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-->0) 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;

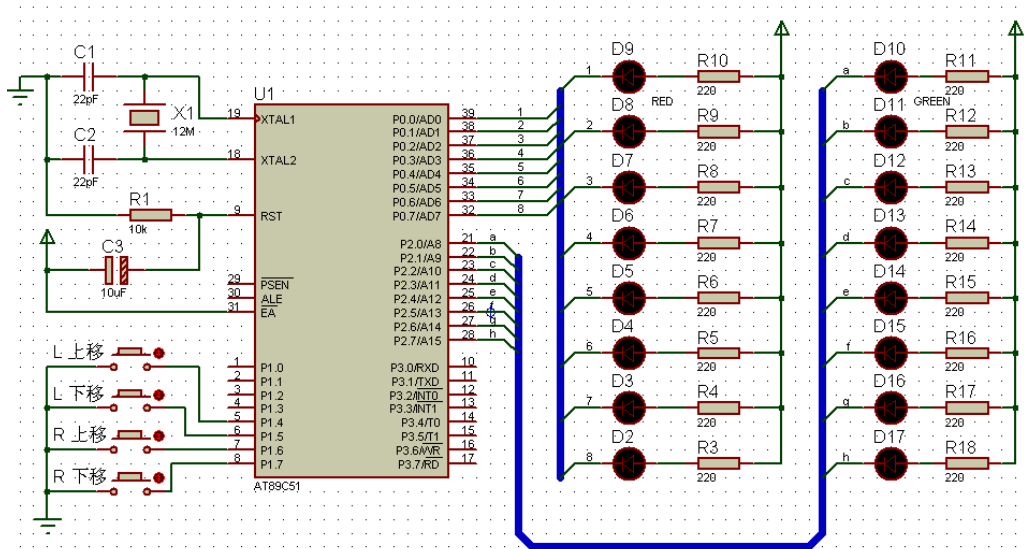
          按下K4时, P2口LED下移一位;

```

*/
#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;
```

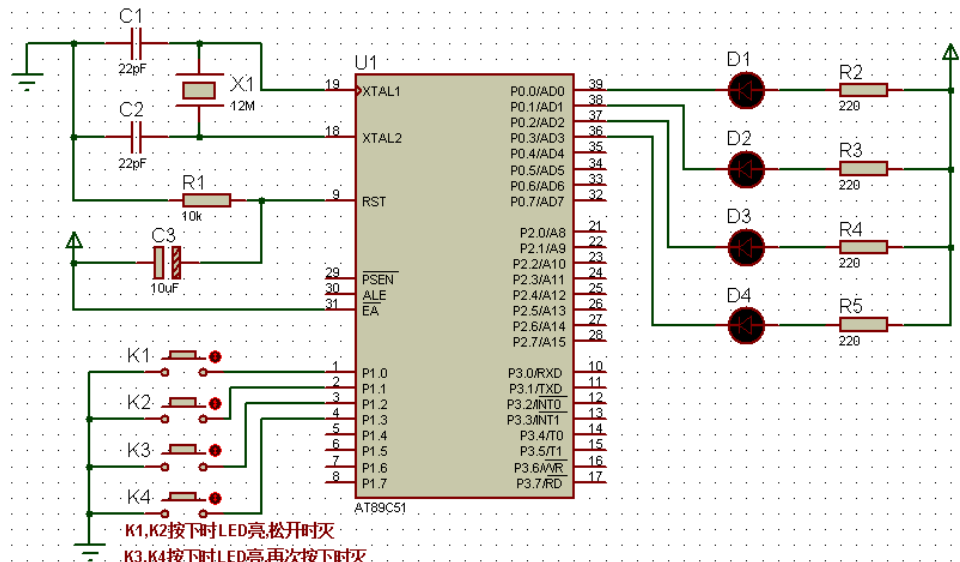
```
        }
```

```
        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;

```

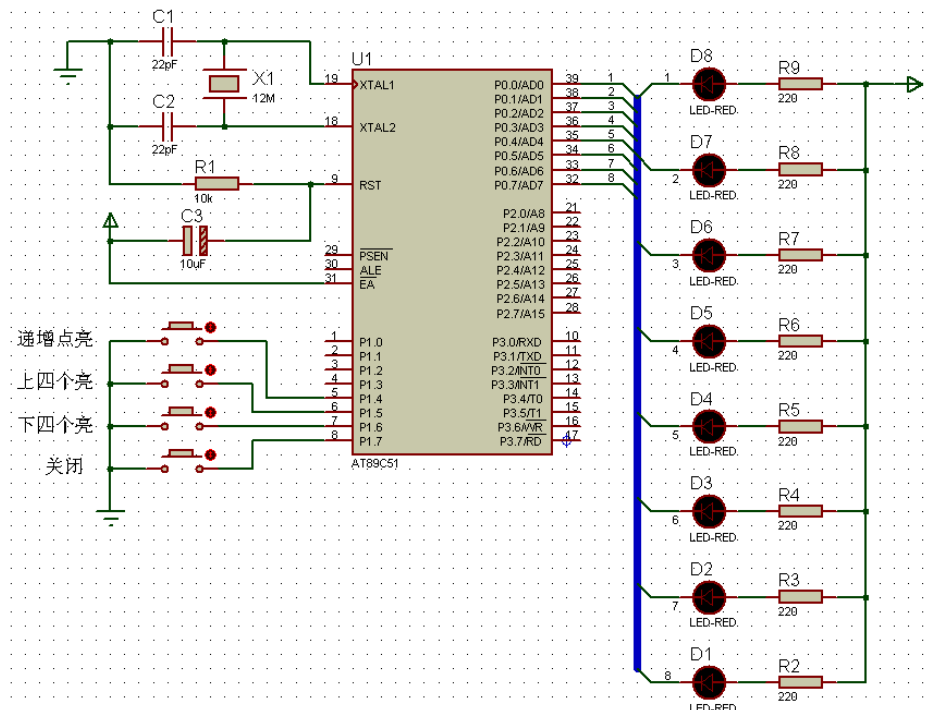
//取得4位按键值, 由模式XXXX1111(X中有一位为0, 其他均为1)

//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_DS7()
```

```
{
```

```
    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_DS7();
```

```
        P1=0xff;
```

```
        Key_NO=P1;
```

```
//P1口按键状态分别为K1-0xfe, K2-0xfd,K3-0xfb
```

```
switch(Key_NO)
```

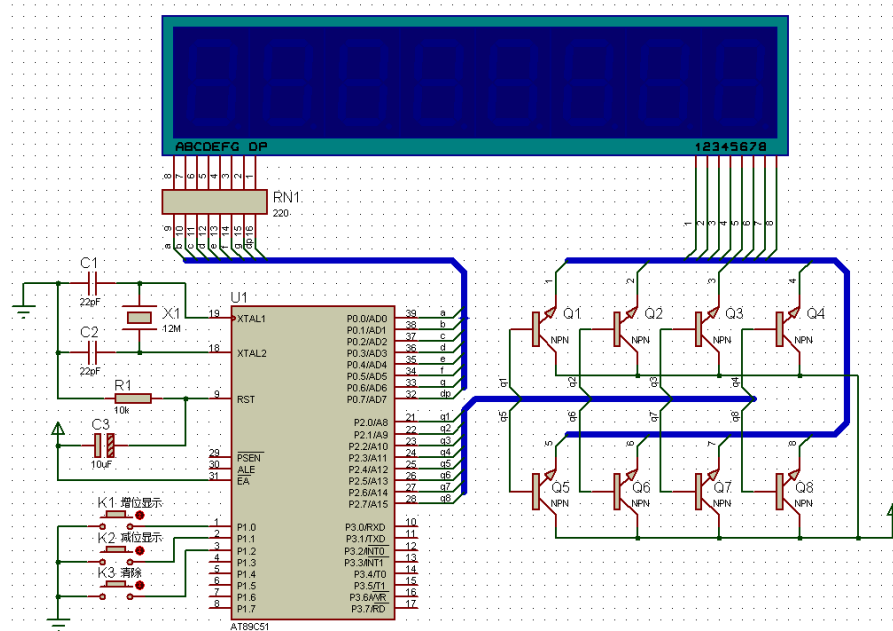
```
{
```

```
    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)
```

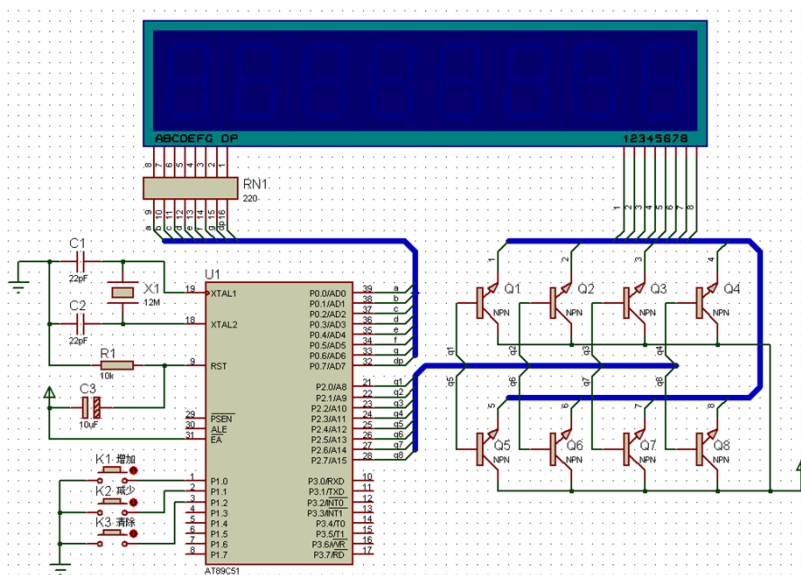
```
{
```

```
    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&lt;reg51.h&gt;

#include&lt;intrins.h&gt;

#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++);
}
```

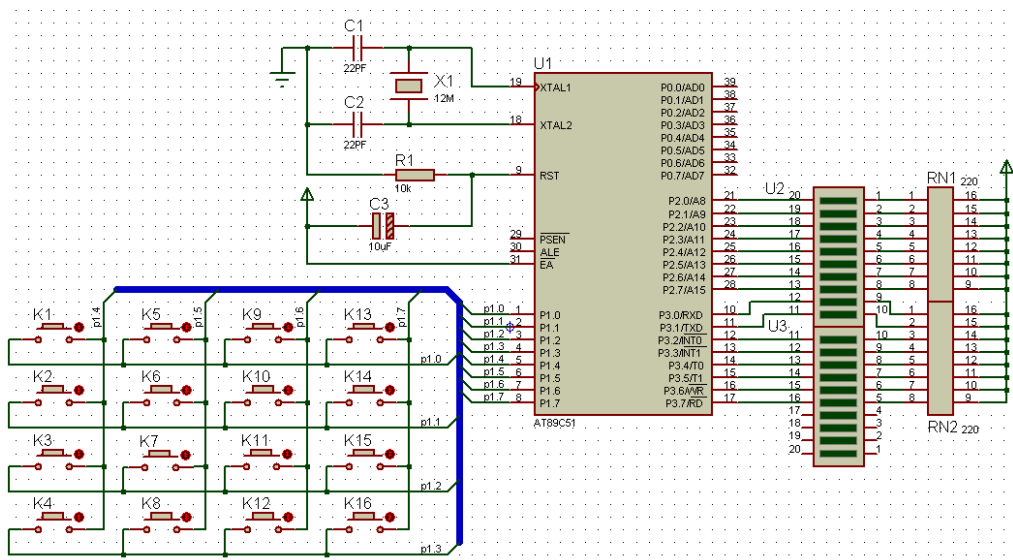
//键盘扫描

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;
```

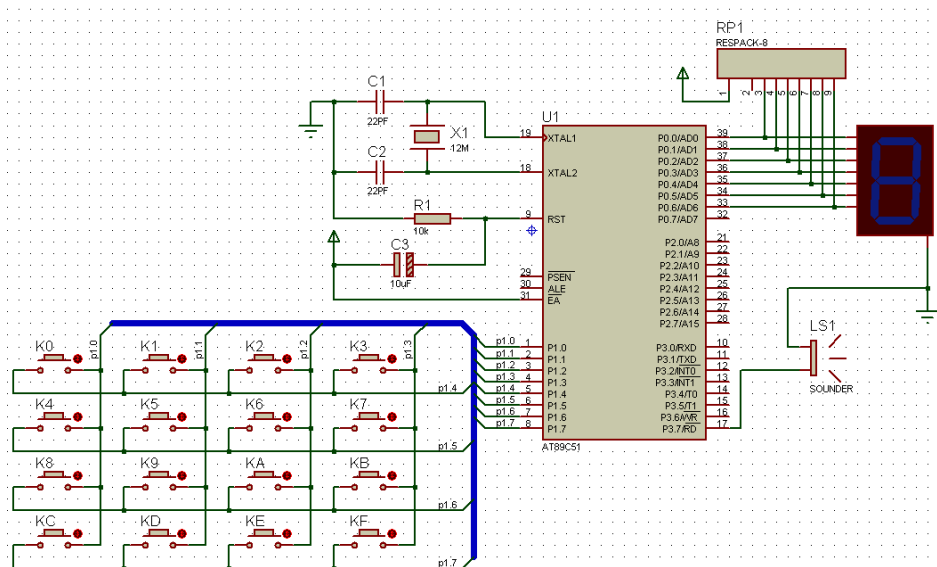
```
    }
}
```

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&lt;reg51.h&gt;

#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 LED1=P0^0;
sbit LED2=P0^1;
//主程序
void main()
{

```

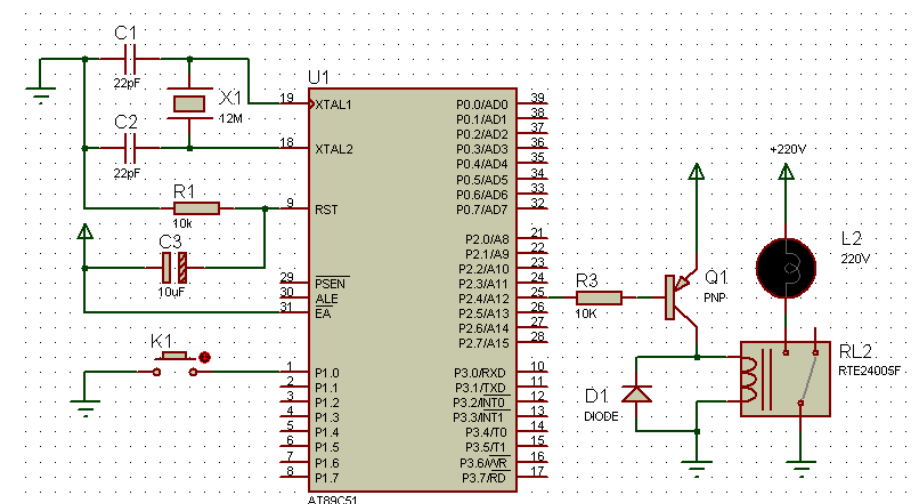
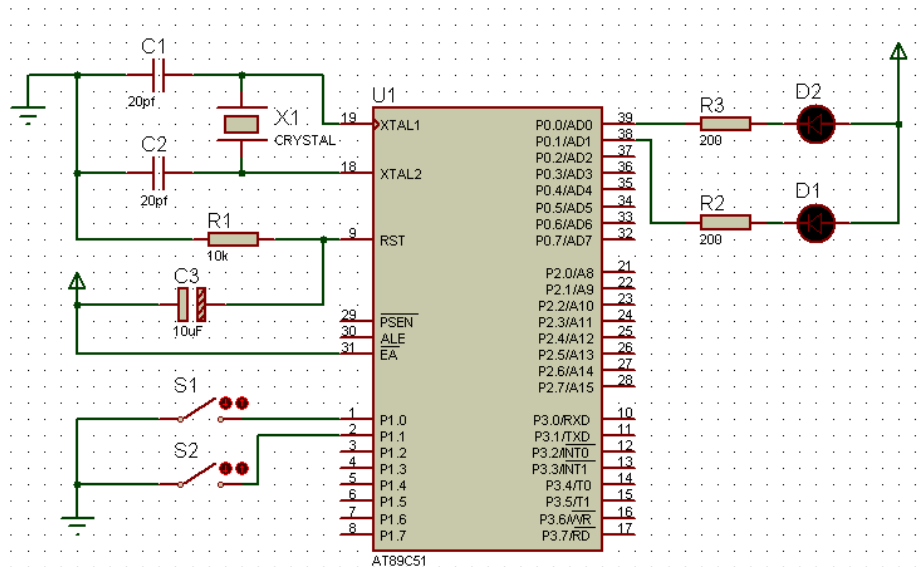
```

    while(1)
    {
        LED1=S1;
        LED2=S2;
    }
}

```

## 19 继电器控制照明设备

工程师社群371516244



```
/* 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);
```

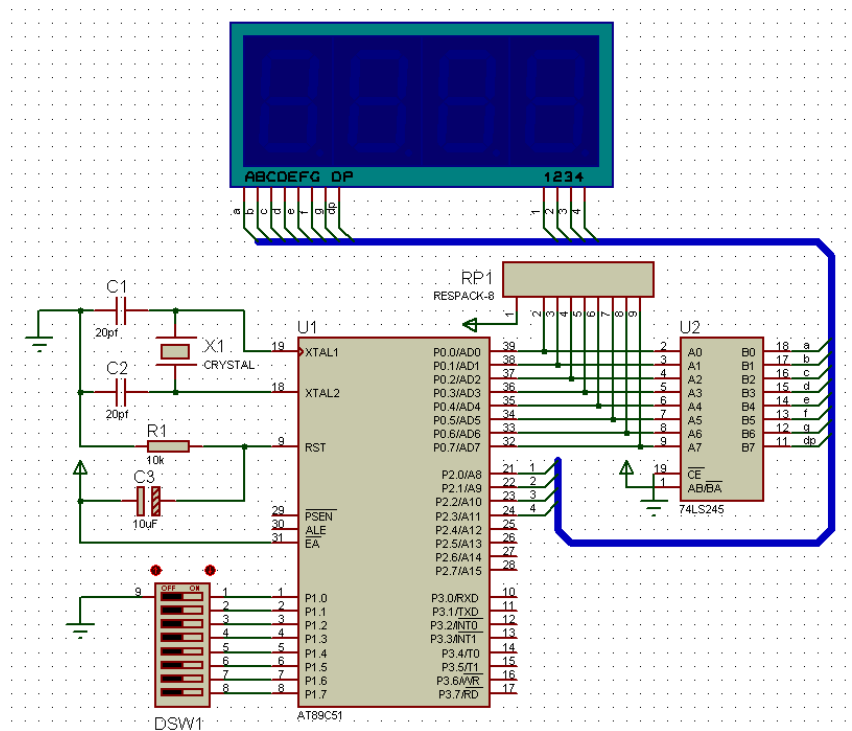
```
            RELAY=~RELAY;
```

```
            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-->0)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++)

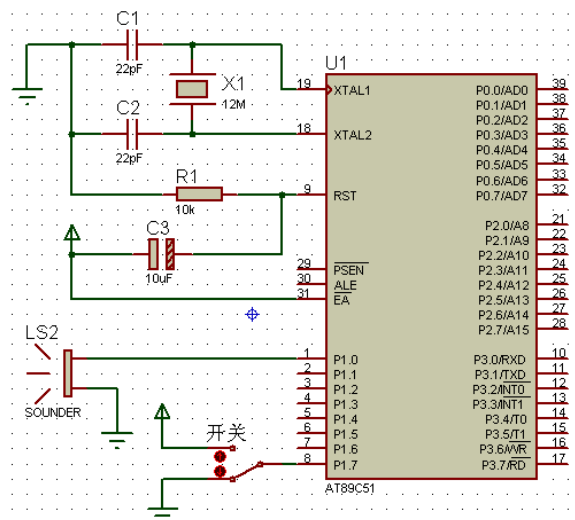
        //刷新显示在数码管上

        {
            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)
```

```
        {
```

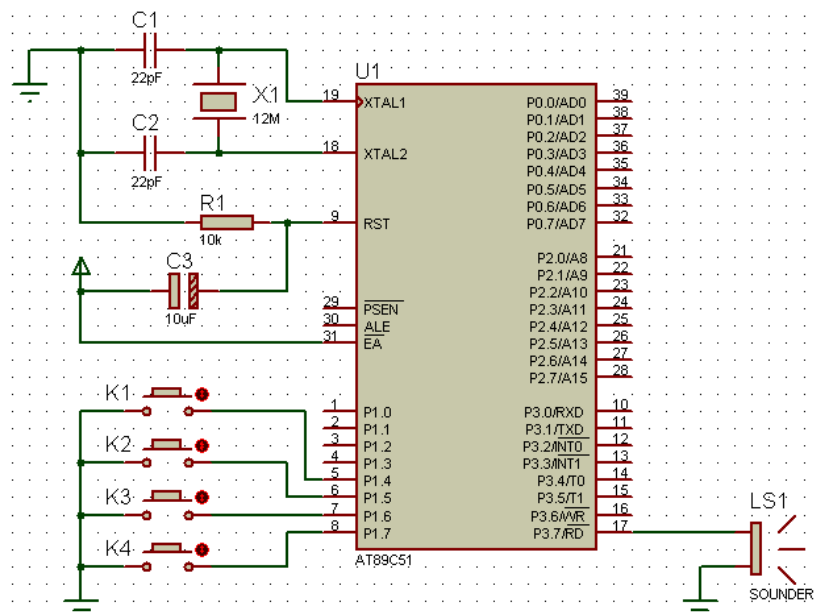
```
            Alarm(90);
```

```
            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;
}

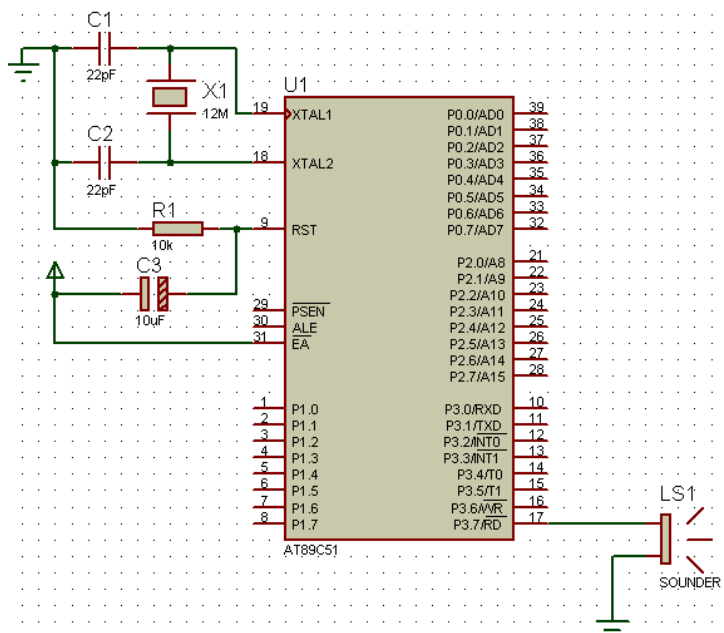
void main()
{

```

```

    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-->0) 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++);
        }
        i++;
    }
}

```

```

    }
    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
    }
}

```

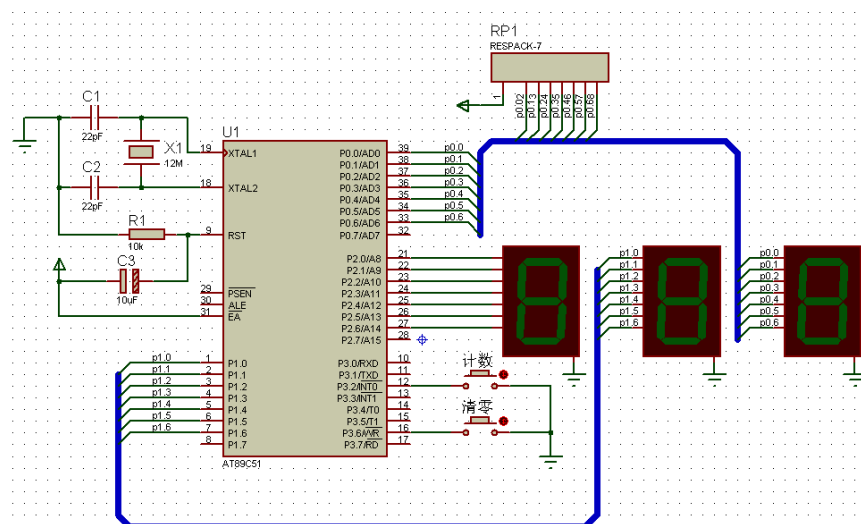
## 24 INT0中断计数

/\* 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
```

```
{
```

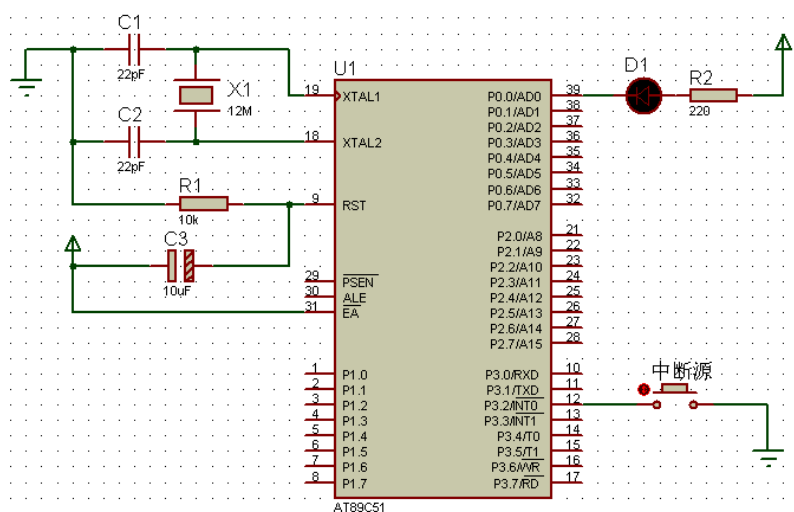
```
    Count++;    //计数值递增
```

```
}
```

## 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

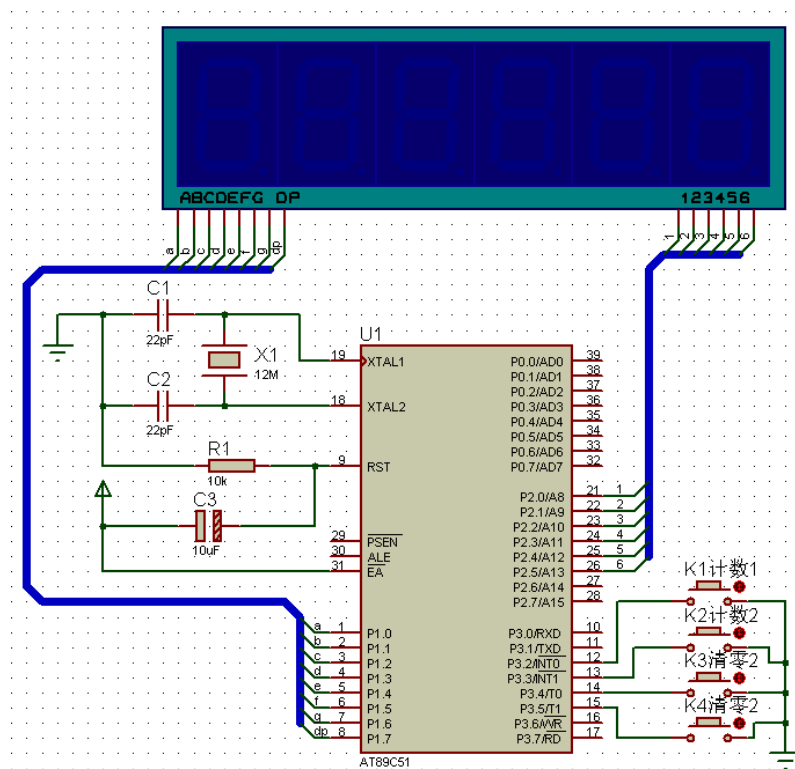
```

*/
#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。

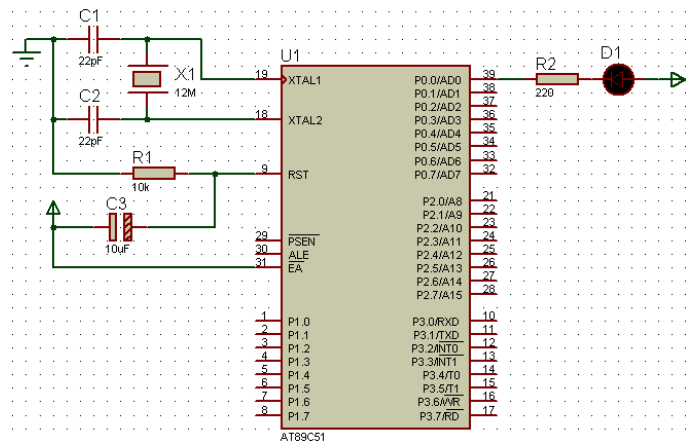
```

*/
#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

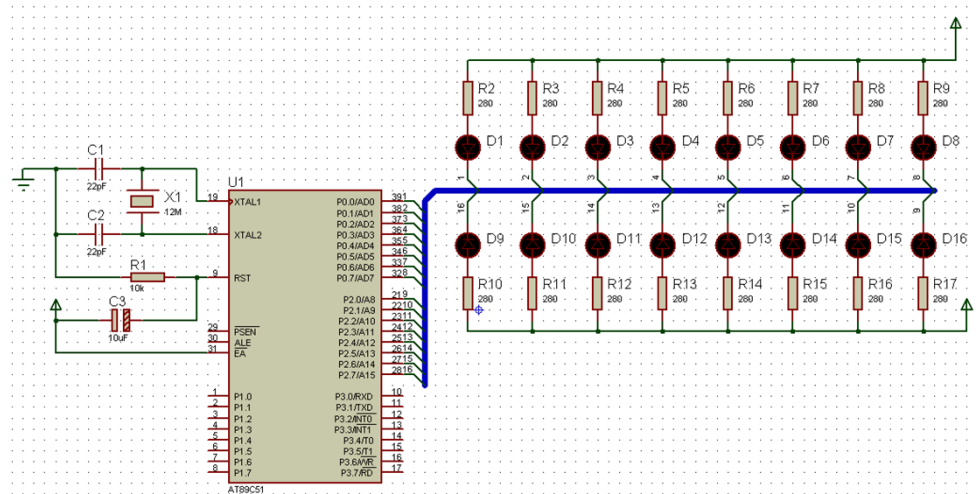
    TL0=(8192-5000)%32;
    if(++T_Count==100) //0.5s开关一次LED
    {
        LED=~LED;
        T_Count=0;
    }
}

```

## 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;

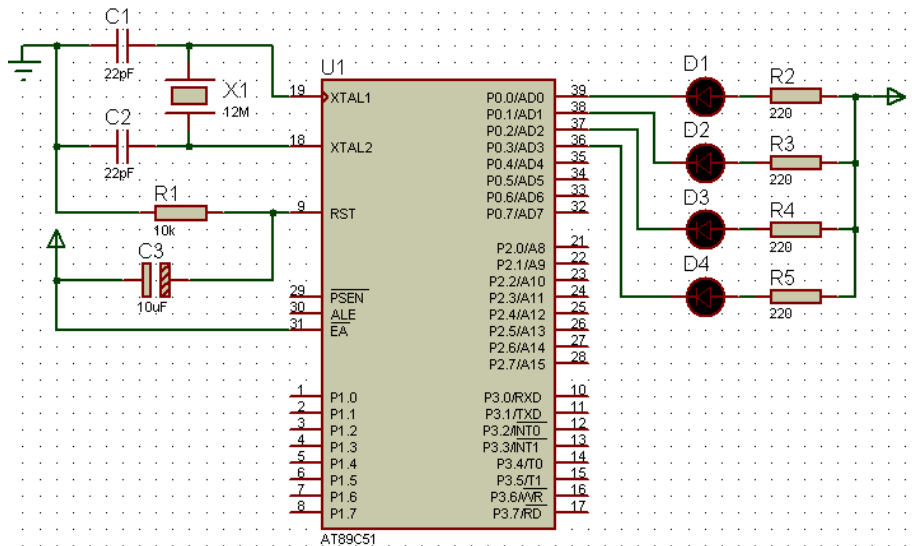
```

//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

/\*

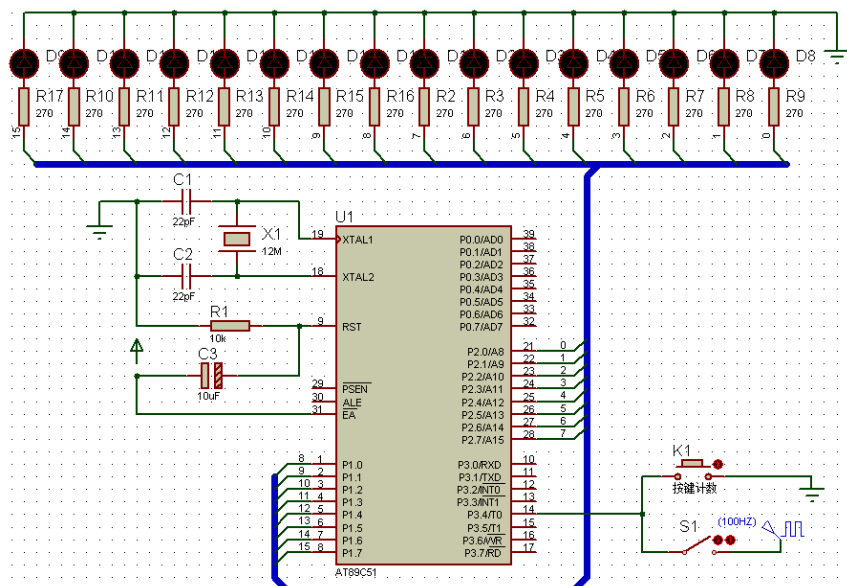
名称:T0控制LED实现二进制

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

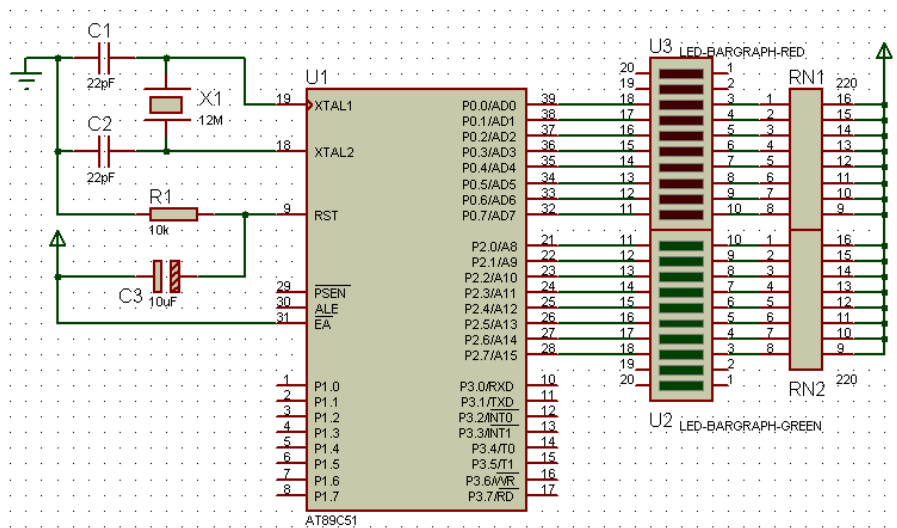
```

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);
    }
}

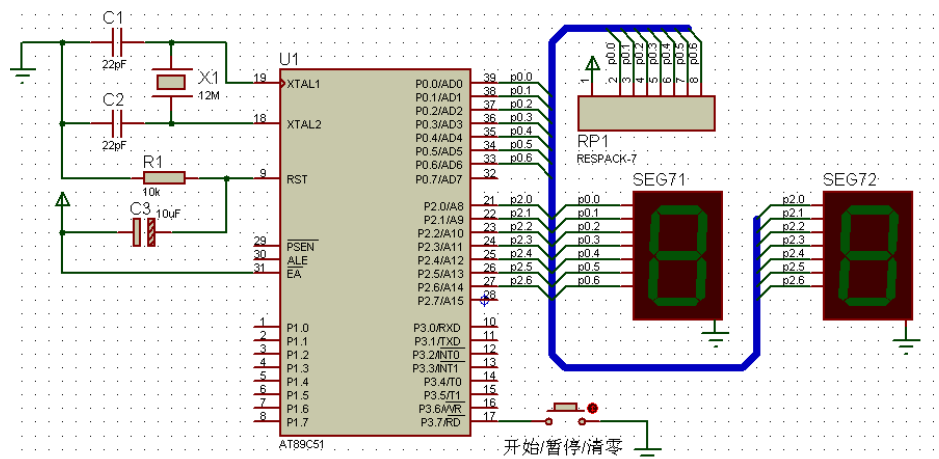
```

## 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>

工程师社群371516244



```

#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。

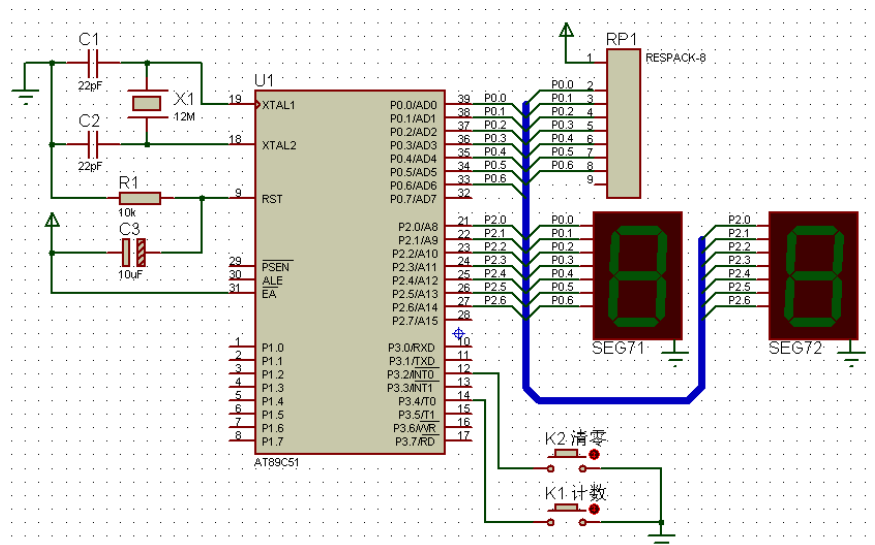
```

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

//段码

```

工程师社群371516244





```

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
{

```

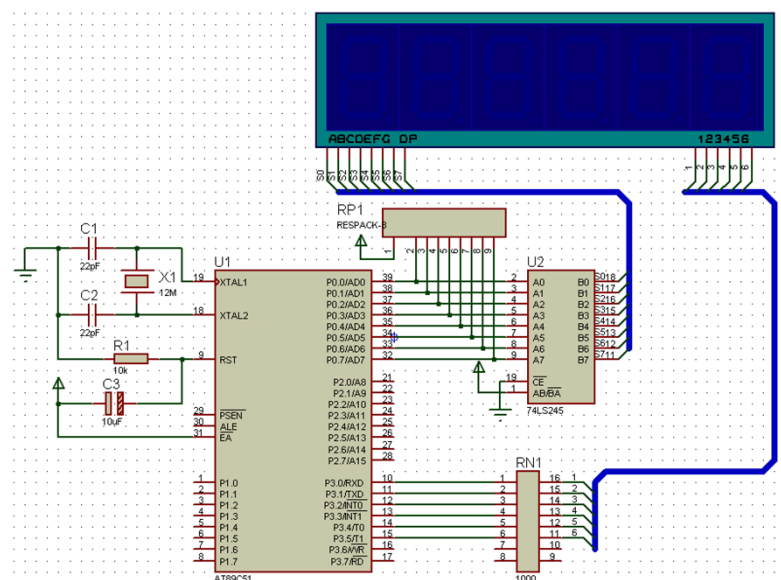
Count=0;

```

}

```

### 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

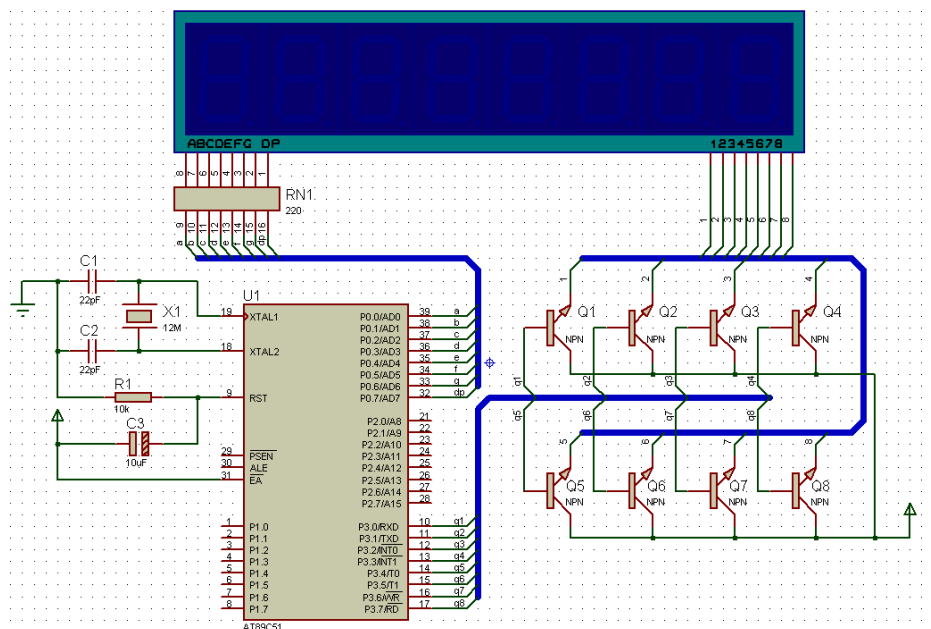
## 定时器控制数码管动态显示

/\*

名称:定时器控制数码

管动态显示

工程师社群371516244



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
```

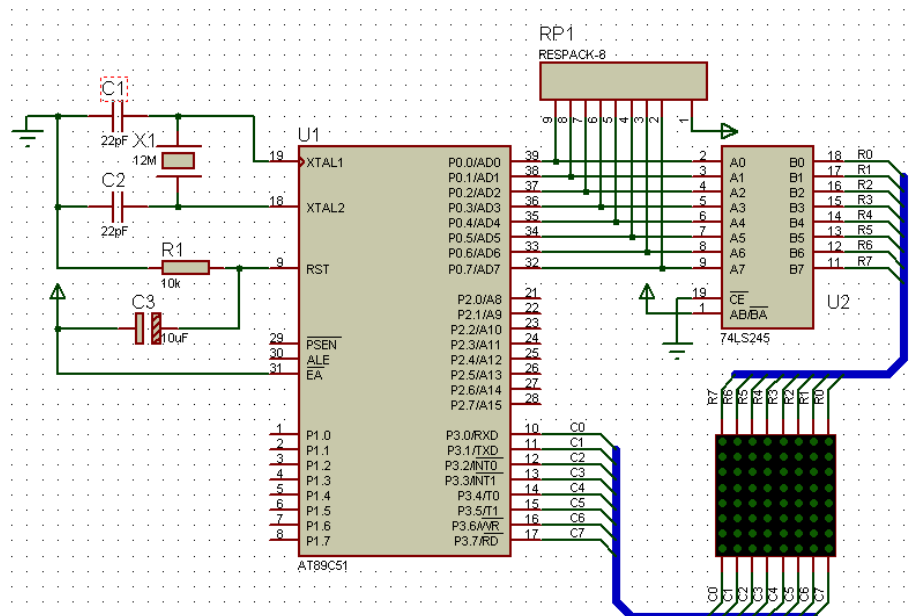
```
void main()
```

```
{
```

```
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

    }

}

```

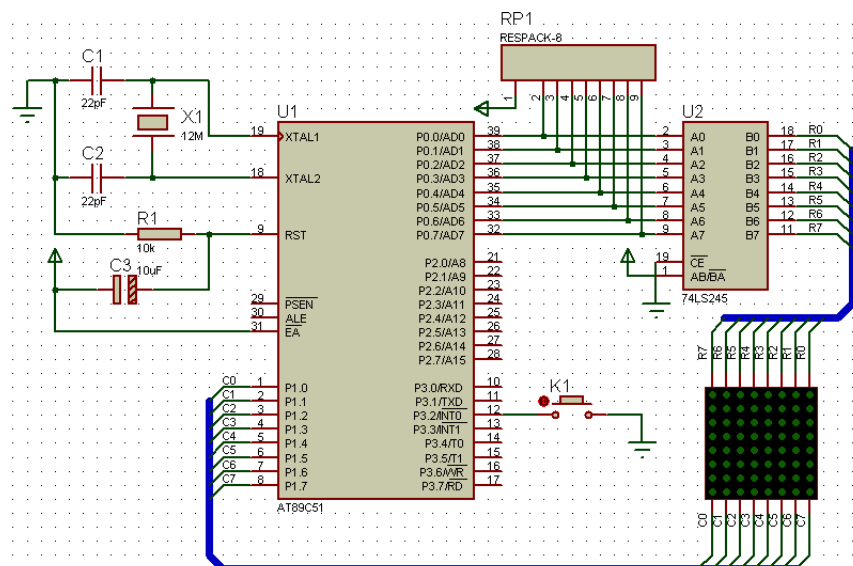
37

## 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;

    TR0=1;
}

```

//i在0, 1, 2中取值, 因为只要3个图形

### 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;
```

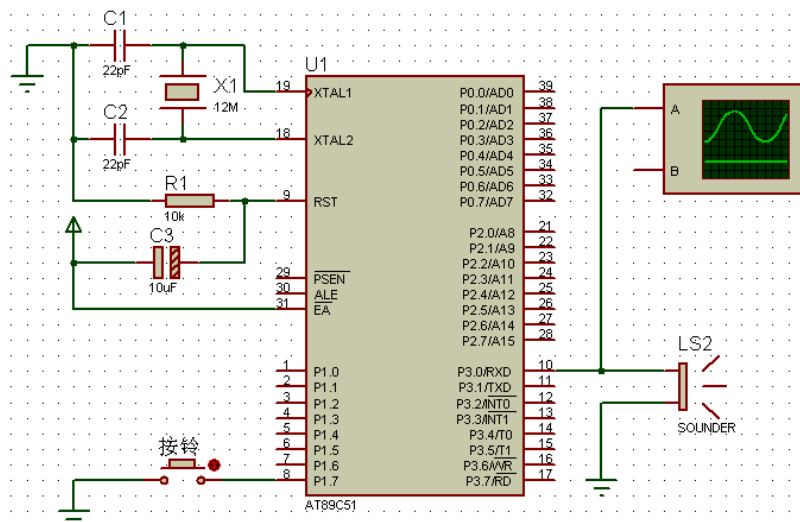
```
            while(Key==0);
```

```
        }
```

```
    }
```

```
}
```

```
//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;
```

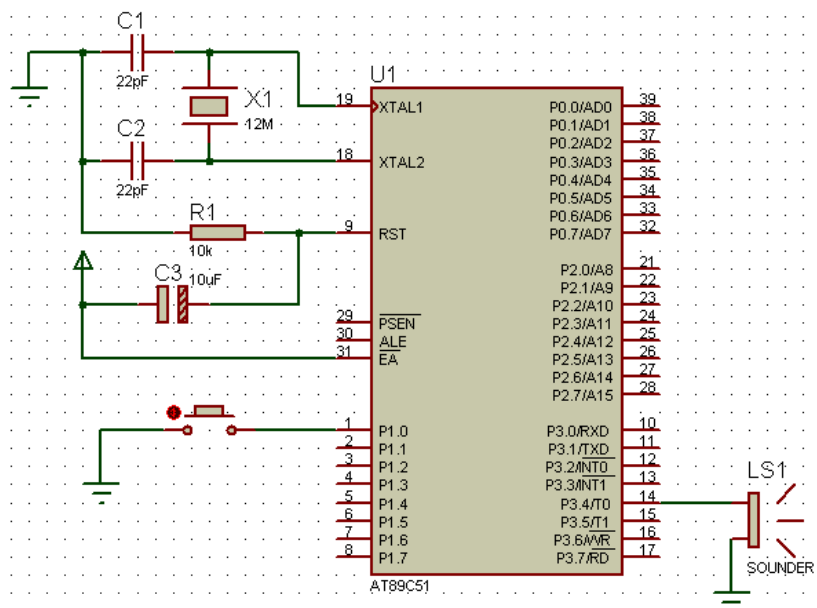
```
uint i=0;                                //音符索引
```

```
//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);

        }
    }
}

```

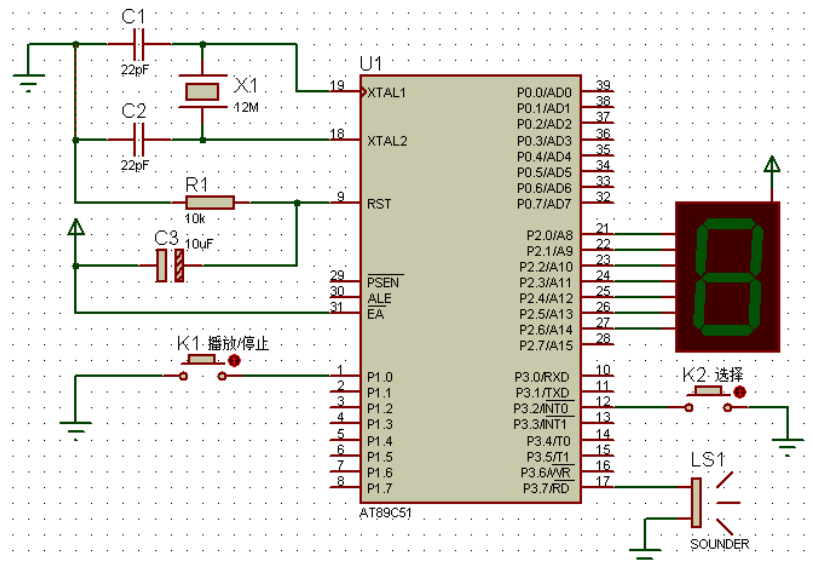
#### 40 按键控制定时器选播多段音乐

/\*

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},
    {3,3,3,4,5,5,5,6,5,3,5,3,2,1,5,6,5,3,3,2,1,1,-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]=
{
    {1,1,1,1,1,1,1,1,1,2,1,1,2,1,1,1,1,1,1,1,1,1,1,1,2,1,2,-1},
    {1,1,1,1,1,1,2,1,1,1,1,1,1,2,1,1,1,1,1,2,2,-1},
    {1,1,2,1,1,2,1,1,1,1,1,1,1,1,1,2,1,1,1,1,1,1,1,1,1,2,1,1,2,2,-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-->0) 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)
```

```
    {
```

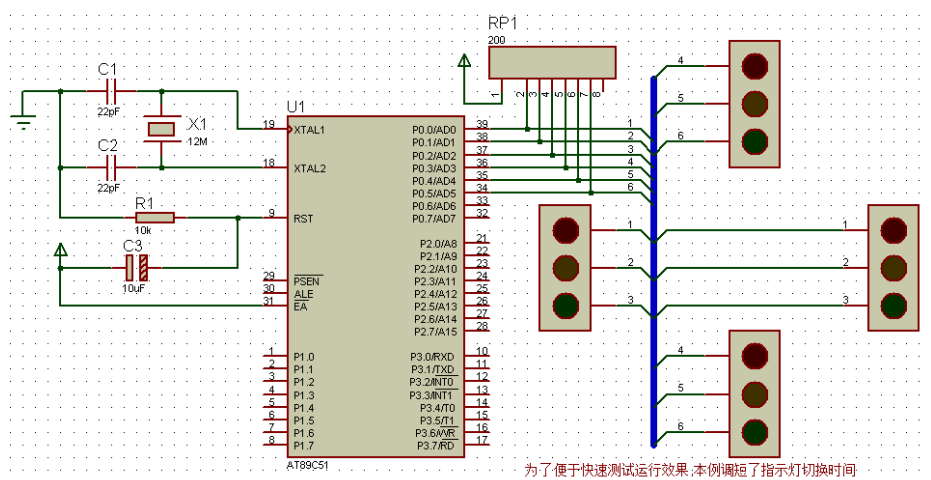
```
        case 1: //东西向绿灯与南北向红灯亮5s
```

```
            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;
```



为了便于快速测试运行效果,本例调短了指示灯切换时间

```
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;
```

```
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;
```

**//T0方式1**

## /\* 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

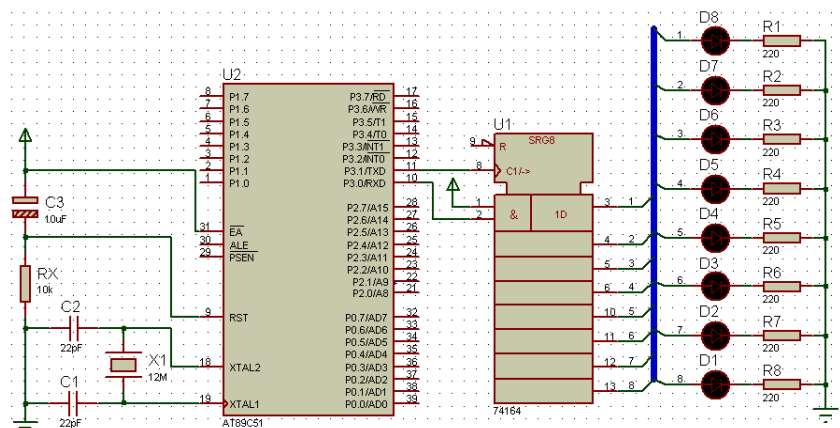
    TR0=0;
    TR1=0;             //定时器启停由INT0控制, 初始关闭

    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, 741

64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

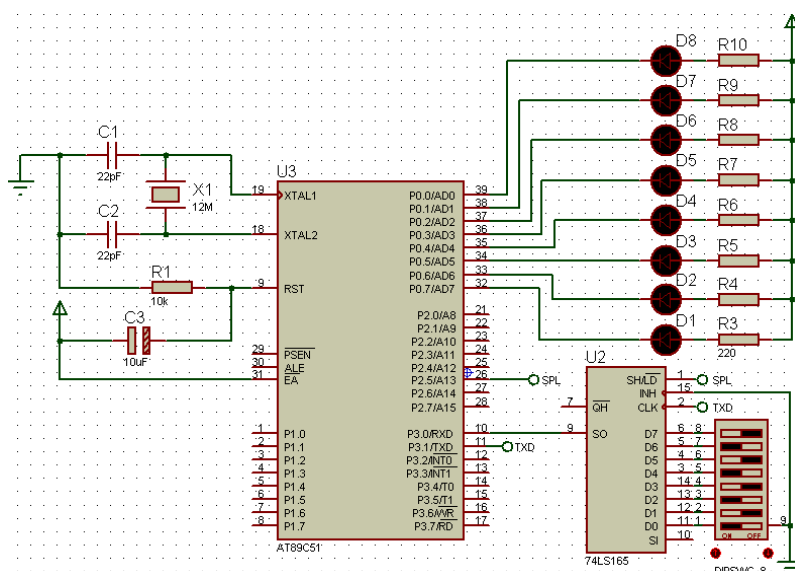
/ \*

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

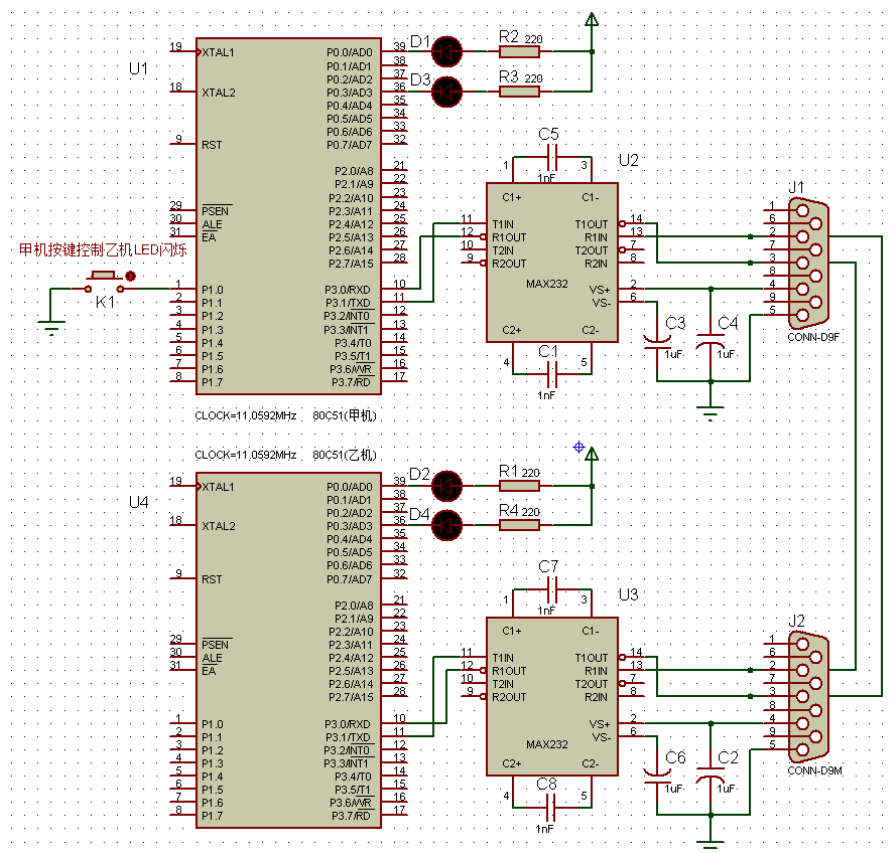
```

/*

```

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-->0) 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-->0) 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。

```

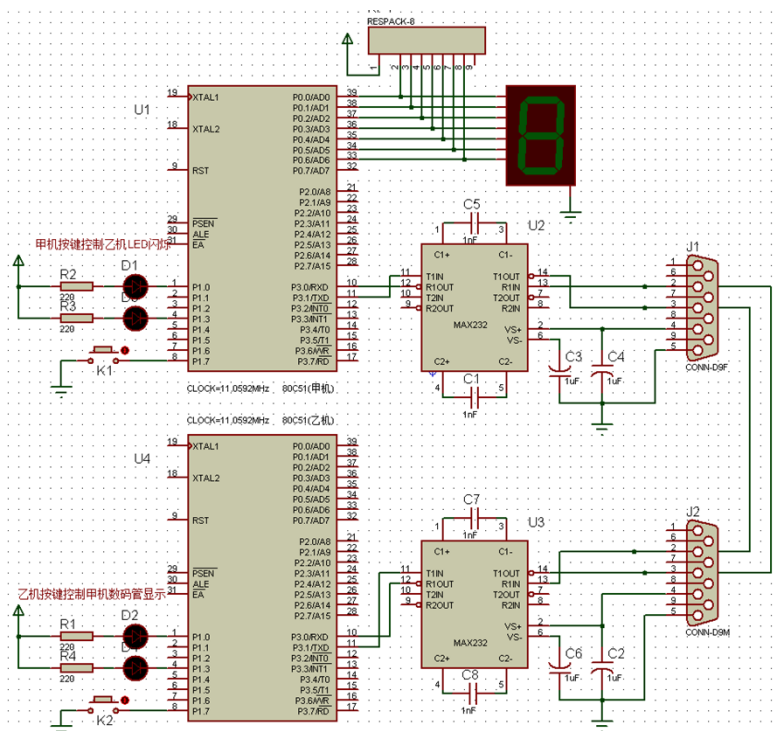
*/
#include<reg51.h>
#define uchar unsigned char
#define uint unsigned int
sbit LED1=P1^0;
sbit LED2=P1^3;
sbit K1=P1^7;

uchar Operation_No=0;    //操作代码

```

//数码管代码

工程师社群371516244



```

uchar code DSY_CODE[]={0x3f,0x06,0x5b,0x4f,0x66,0x6d,0x7d,0x07,0x7f,0x6f};

//延时

void DelayMS(uint ms)
{
    uchar i;
    while(ms-->0) 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-->0) 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;
}

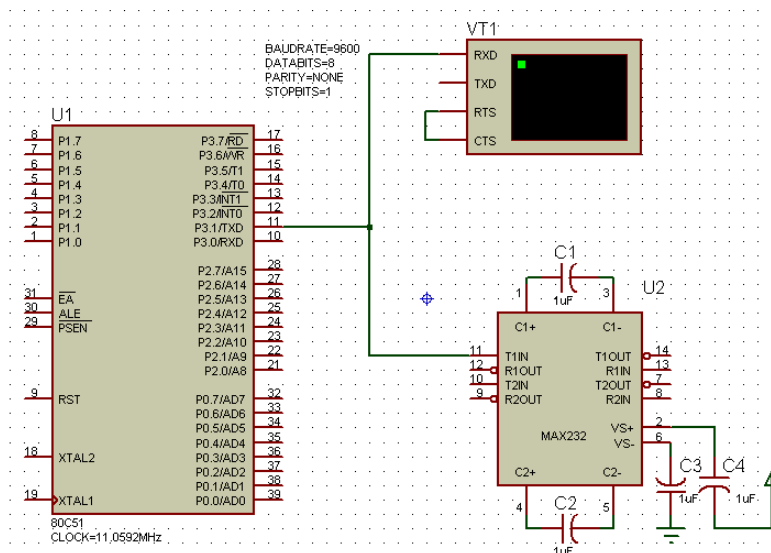
```

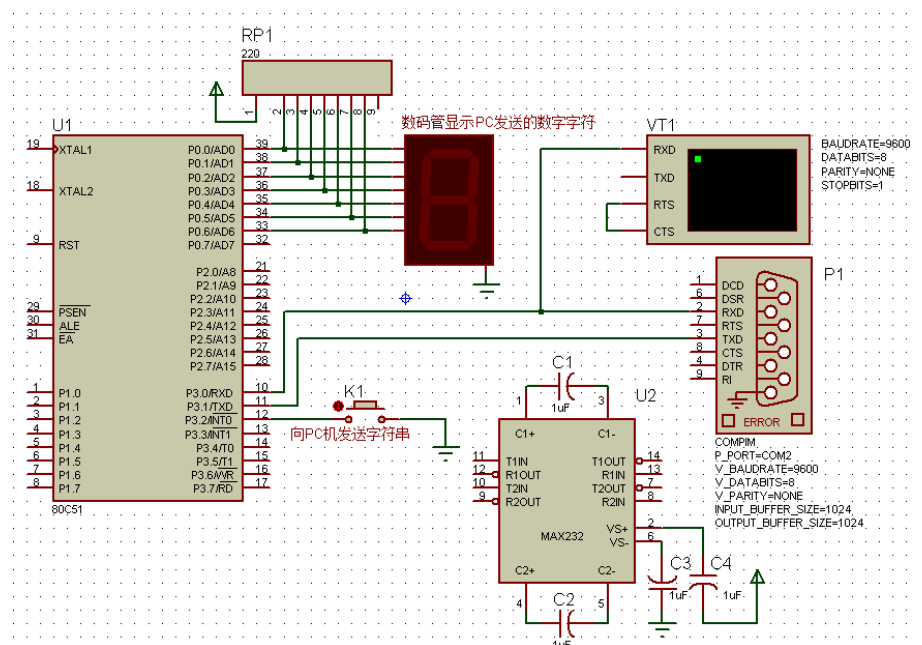
//向串口发送字符串

```

void Puts_to_SerialPort(uchar *s)
{
    while(*s!='\0')
    {
        Putc_to_SerialPort(*s);
        s++;
        DelayMS(5);
    }
}

```





/\* 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

## 01 74LS138译码器应用

/\* 名称:74LS138译码器应用

说明:本例通过74LS138译码器, 仅用P2口3个引脚来控制8只LED滚动显示。

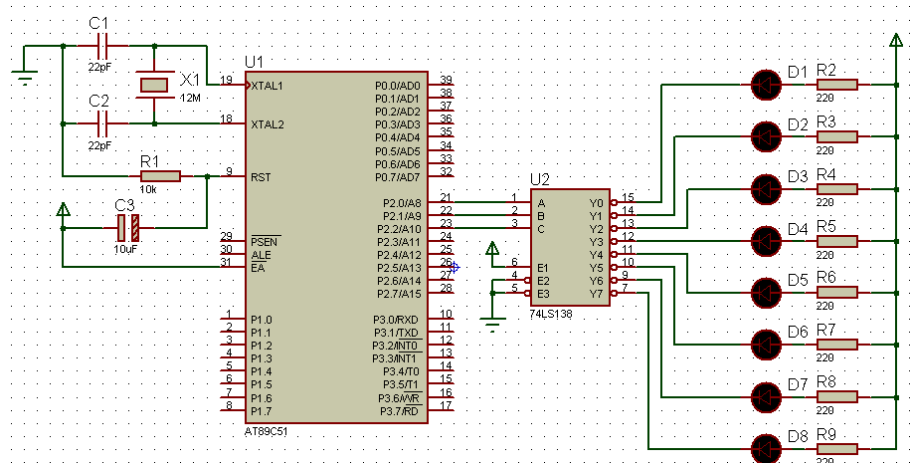
```

*/
#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<40;i++);
}

//主程序
void main()
{
    P2=0x00;
    while(1)
    {
        P2=(P2+1)%8;
        DelayMS(500);
    }
}

```



## 02 74HC154译码器应用

/\* 名称:74HC154译码器应用

说明:74HC154是4-16译码器, 本例利用P2口输出4位二进制数,

经译码后使相应的LED被点亮, 形成滚动显示效果。

```

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

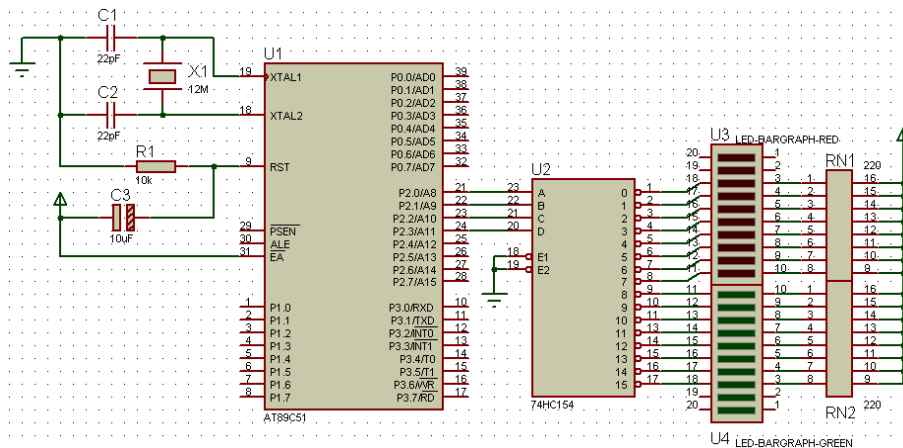
//延时
void DelayMS(uint ms)
{
    uchar i;
    while(ms--)
    for(i=0;i<40;i++);
}

//主程序
void main()
{
    while(1)
    {
        P2=(P2+1)%16; //P2口低4位在0~15取值, 使154译码器输入4位为0000~1111

        DelayMS(500); //经译码器输出0~15中对应引脚输出0, LED点亮

    }
}

```



### 03 74HC595串入并出芯片应用

/\* 名称:74HC595串入并出芯片应用

说明:74HC595是具有一个8位串入并出的移位寄存器和一个8位输出寄存器,

本例利用74HC595, 通过串行输入数据来控制数码管的显示。

```

*/
#include<reg51.h>
#include<intrins.h>
#define uchar unsigned char
#define uint unsigned int

sbit SH_CP=P2^0; //移位时钟脉冲

sbit DS=P2^1; //串行数据输入

sbit ST_CP=P2^2; //输出锁存器控制脉冲

uchar temp;

```

```
uchar code DSY_CODE[]={0xc0,0xf9,0xa4,0xb0,0x99,0x92,0x82,0xf8,0x80,0x90};
```

```
//延时
```

```
void DelayMS(uint ms)
```

```
{
    uchar i;
    while(ms--)
    for(i=0;i<120;i++);
}
```

```
//串行输入子程序
```

```
void In_595()
```

```
{
    uchar i;
    for(i=0;i<8;i++)
    {
        temp<=<1;DS=CY;
        SH_CP=1;           //移位时钟脉冲上升沿移位
        _nop_();_nop_();
        SH_CP=0;
    }
}
```

```
//并行输出子程序
```

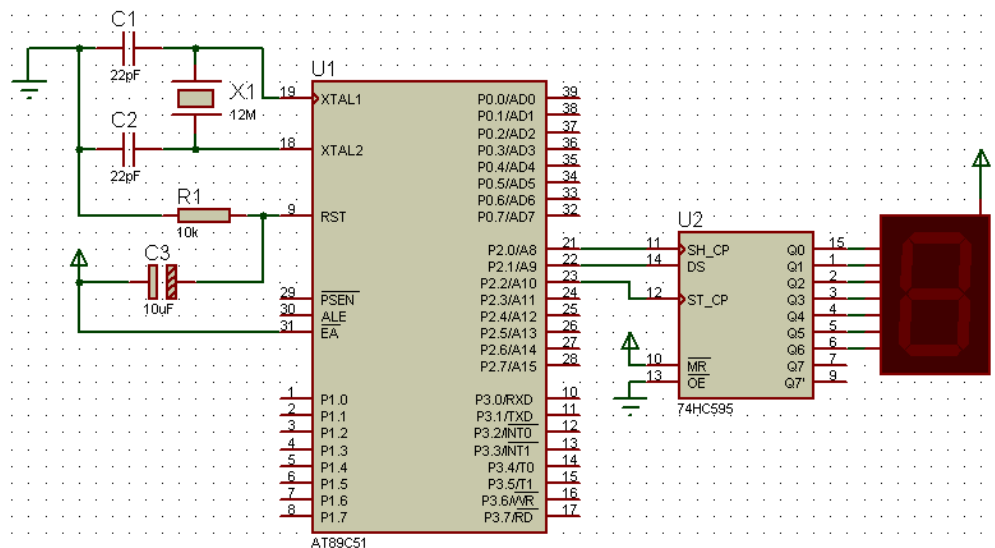
```
void Out_595()
```

```
{
    ST_CP=0;_nop_();
    ST_CP=1;           //上升沿将数据送到输出锁存器
    _nop_();
    ST_CP=0;           //锁存显示数据
}
```

```
//主程序
```

```
void main()
```

```
{
    uchar i;
    while(1)
    {
        for(i=0;i<10;i++)
        {
            temp=DSY_CODE[i];
            In_595();           //temp中的一字节数据串行输入74HC595
        }
    }
}
```



$$\left\{ \begin{array}{l} \\ \\ \end{array} \right\}$$

## 04 74LS148扩展中断

/\* 名称:74LS148扩展中断

说明:本例利用74LS148扩展中断,对于外部的8个控制开关,

任意合上一个都将在GS引脚输出低电平，触发外部中断，

优先级最高的输入引脚7，最低的是引脚0。中断触发后，

中断例程通过读取A2, A1, A0的输出, 判断是哪一路开关触发中断。

```

*/
#include<reg51.h>
#include<intrins.h>
#define uchar unsigned char
#define uint unsigned int
sbit LED=P1^0;
//INT0中断程序
void EX_INT0() interrupt 0
{
    uchar bi=P2&0x07;

    //中断控制点亮按键对应的

LED

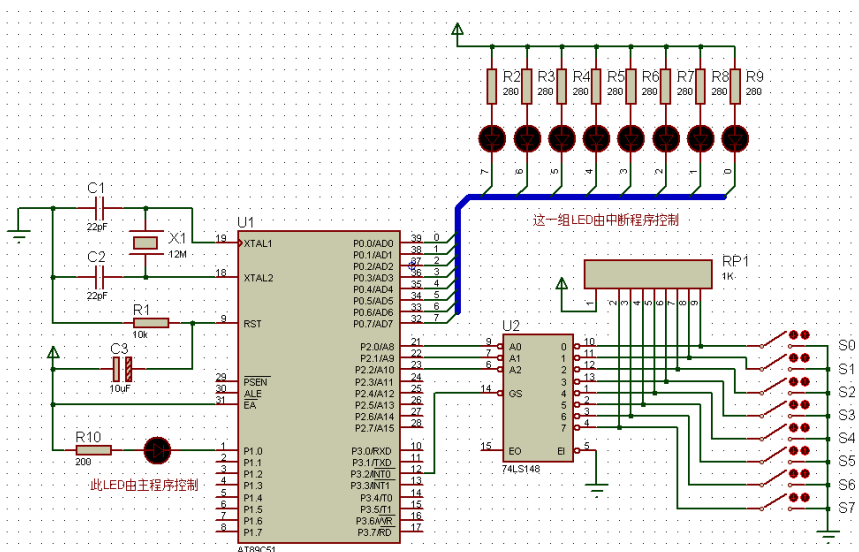
    P0=_cror_(0x7f,bi);
}

```

//主程序。说明:由于Proteus中74LS148存在问题,输入引脚0对应的开关无效。

```
void main()
```

```
{
    uint i;
    IE=0x81;
    IT0=0;
    while(1)
    {
```





LED=~LED;

//主程序控制1个LED

for(i=0;i&lt;30000;i++); //延时

if(INT0==0) P0=0xff; //INT0为1(即GS为1), 无按键合上, 关闭LED

}

}

05 I<sup>2</sup>C-24C04与蜂鸣器/\* 名称:I<sup>2</sup>C-24C04与蜂鸣器

说明:程序首先向24C04写入一段音符, 然后读取并播放。

\*/

#include&lt;reg51.h&gt;

#include&lt;intrins.h&gt;

#define uchar unsigned char

#define uint unsigned int

#define NOP4() {\_nop\_();\_nop\_();\_nop\_();\_nop\_();}

sbit SCL=P1^0;

sbit SDA=P1^1;

sbit SPK=P3^0;

//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};

//待写入24C04的音符

uchar code Song\_24C04[]={1,2,3,1,1,2,3,1,3,4,5,3,4,5};

uchar sidx; //读取音符索引

//延时

void DelayMS(uint ms)

{

uchar i;

while(ms--) for(i=0;i&lt;120;i++);

}

//IIC开始

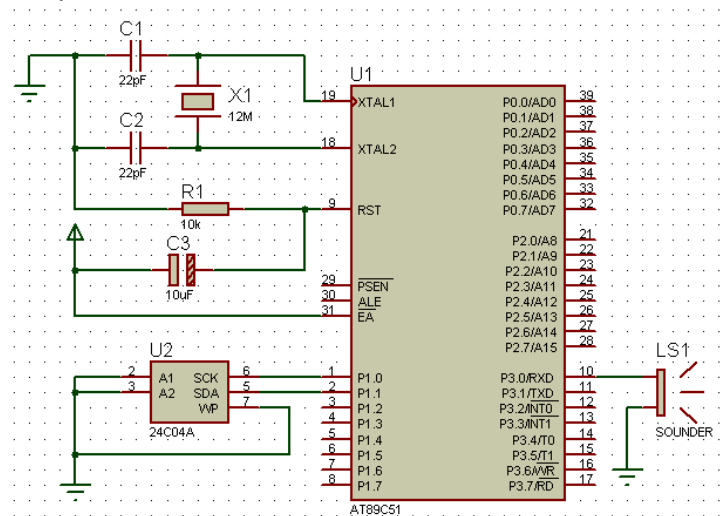
void Start()

{

SDA=1;SCL=1;NOP4();SDA=0;NOP4();SCL=0;

}

//IIC停止



```

void Stop()
{
    SDA=0;SCL=0;NOP4();SCL=1;NOP4();SDA=1;
}
//读取应答

void RACK()
{
    SDA=1;NOP4();SCL=1;NOP4();SCL=0;
}
//发送非应答信号

void NO_ACK()
{
    SDA=1;SCL=1;NOP4();SCL=0;SDA=0;
}
//向24C04中写一个字节数据

void Write_A_Byte(uchar b)
{
    uchar i;
    for(i=0;i<8;i++)
    {
        b<<=1;SDA=C_Y;_nop_();SCL=1;NOP4();SCL=0;
    }
    RACK();
}
//向指定地址写数据

void Write_IIC(uchar addr,uchar dat)
{
    Start();
    Write_A_Byte(0xa0);Write_A_Byte(addr);Write_A_Byte(dat);
    Stop();
    DelayMS(10);
}
//从24C04中读一个字节数据

uchar Read_A_Byte()
{
    uchar i,b;
    for(i=0;i<8;i++)
    {
        SCL=1;b<<=1;b|=SDA;SCL=0;
    }
    return b;
}

```

}

//从当前地址读取数据

uchar Read\_Current()

{

uchar d;

Start();

Write\_A\_Byte(0xa1);d=Read\_A\_Byte();NO\_ACK();

Stop();

return d;

}

//从任意地址读取数据

uchar Random\_Read(uchar addr)

{

Start();

Write\_A\_Byte(0xa0);Write\_A\_Byte(addr);

Stop();

return Read\_Current();

}

//定时器0中断

void T0\_INT() interrupt 1

{

SPK=~SPK;

TH0=HI\_LIST[sidx];

TL0=LO\_LIST[sidx];

}

//主程序

void main()

{

uint i;

IE=0x82;

TMOD=0x00;

for(i=0;i&lt;14;i++)

//向24C04写入音符表

{

Write\_IIC(i,Song\_24C04[i]);

}

while(1)

//反复读取音符并播放

{

for(i=0;i&lt;15;i++)

//从24C04中读取音符

{

sidx=Random\_Read(i);

//从指定地址读取

```
TR0=1;                                //播放
DelayMS(300);
```

```
    }
}
}
```

## 06 24C04与数码管

/\* 名称:24C04与数码管

说明:每次运行时,程序将24C04芯片内的计数字节值

递增并显示在数码管上,反复运行,实现计数。

```
*/
#include<reg51.h>
#include<intrins.h>
#define uchar unsigned char
#define uint unsigned int
#define Delay4us() {_nop_();_nop_();_nop_();_nop_();}
sbit SCL=P1^0;
sbit SDA=P1^1;

//数码管段码

uchar code DSY_CODE[]={0xc0,0xf9,0xa4,0xb0,0x99,0x92,0x82,0xf8,0x80,0x90,0xff};

//三位数显示缓冲

uchar DISP_Buffer[]={0,0,0};
uchar Count=0;

//延时

void DelayMS(uint ms)
{
    uchar i;
    while(ms--) for(i=0;i<120;i++);
}

//IIC启动

void Start()
{
    SDA=1;SCL=1;Delay4us();SDA=0;Delay4us();SCL=0;
}

//IIC停止

void Stop()
```

```

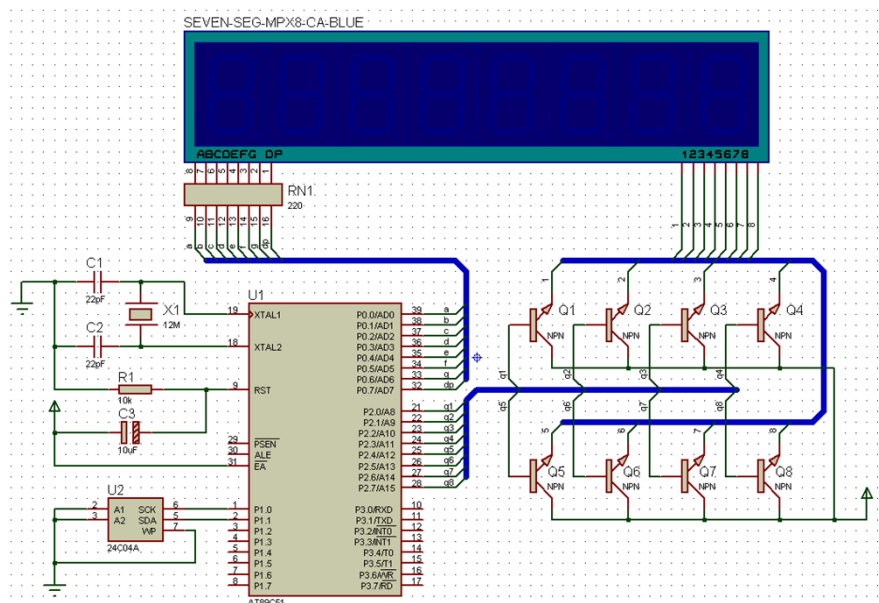
{
    SDA=0;SCL=0;Delay4us();SCL=1;Delay4us();SDA=1;
}
//读取应答
void RACK()
{
    SDA=1;Delay4us();SCL=1;Delay4us();SCL=0;
}
//发送非应答信号
void NO_ACK()
{
    SDA=1;SCL=1;Delay4us();SCL=0;SDA=0;
}
//向24C04中写一个字节数据
void Write_A_Byte(uchar byte)
{
    uchar i;
    for(i=0;i<8;i++)
    {
        byte<<=1;SDA=CY;_nop_();SCL=1;Delay4us();SCL=0;
    }
    RACK();
}
//向指定地址写数据
void Write_Random_Adress_Byte(uchar addr,uchar dat)
{
    Start();
    Write_A_Byte(0xa0);Write_A_Byte(addr);Write_A_Byte(dat);
    Stop();
    DelayMS(10);
}
//从24C04中读一个字节数据

```

```

uchar Read_A_Byte()
{
    uchar i,b;
    for(i=0;i<8;i++)
    {

```



```

        SCL=1;b<<=1;b|=SDA;SCL=0;
    }
    return b;
}

//从当前地址读取数据
uchar Read_Current_Address_Data()
{
    uchar dat;
    Start();
    Write_A_Byte(0xa1);dat=Read_A_Byte();NO_ACK();
    Stop();
    return dat;
}

//从任意地址读取数据
uchar Random_Read(uchar addr)
{
    Start();
    Write_A_Byte(0xa0);Write_A_Byte(addr);
    Stop();
    return Read_Current_Address_Data();
}

//数据转换与显示
void Convert_And_Display()
{
    DISP_Buffer[2]=Count/100;
    DISP_Buffer[1]=Count%100/10;
    DISP_Buffer[0]=Count%100%10;

    if(DISP_Buffer[2]==0)                //高位为0不显示
    {
        DISP_Buffer[2]=10;
        if(DISP_Buffer[1]==0)            //高位为0, 次高位为0也不显示
            DISP_Buffer[1]=10;
    }
    P0=0xff;
    P2=0x80;                            //个位
    P0=DSY_CODE[DISP_Buffer[0]];
    DelayMS(2);
    P0=0xff;
    P2=0x40;                            //十位
    P0=DSY_CODE[DISP_Buffer[1]];

```

```

DelayMS(2);
P0=0xff;
P2=0x20;                                     //百位
P0=DSY_CODE[DISP_Buffer[2]];
DelayMS(2);
}
//主程序
void main()
{
    Count=Random_Read(0x00)+1;
    //从24C04的0x00地址读取数据并递增

    Write_Random_Adress_Byte(0x00,Count); //将递增后的计数值写入24C04

    while(1) Convert_And_Display();        //转换并持续刷新数码管显示
}

```

## 07 用6264扩展内存

/\* 名称:用6264扩展内存

说明:本例先向6264中写入整数1~200, 然后将其逆向复制到0x0100处。

\*/

```

#include<reg51.h>
#include<absacc.h>

```

```

#define uchar unsigned char
#define uint unsigned int
sbit LED=P1^0;

```

//主程序

```
void main()
```

```
{
```

```
    uint i;
```

```
    LED=1;
```

```
    for(i=0;i<200;i++)
```

```
    {
```

```
        XBYTE[i]=i+1;
```

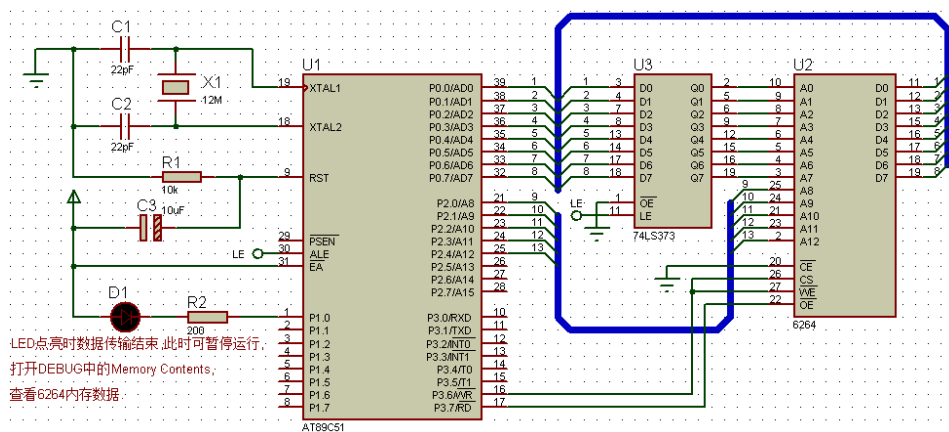
```
    }
```

```
    for(i=0;i<200;i++)
```

```
    {
```

//向6264的0x0000地址开始写入1~200

//将6264中的1~200逆向复制到0x0100开始处



```

XBYTE[i+0x0200]=XBYTE[199-i];
}

LED=0;                                     //扩展内存数据处理完后LED点亮

while(1);
}

```

## 08 用8255实现接口扩展(仿真未成功)

/\* 名称:用8255实现接口扩展

说明:8255的PA、PB端口分别连接8位数码管的段码和位码, 程序控制数码管滚动显示一串数字。

\*/

```
#include<reg51.h>
```

```
#include<absacc.h>
```

```
#define uchar unsigned char
```

```
#define uint unsigned int
```

```
//PA、PB、PC端口及命令端口地址定义
```

```
#define PA XBYTE[0x0000]
```

```
#define PB XBYTE[0x0001]
```

```
#define PC XBYTE[0x0002]
```

```
#define COM XBYTE[0x0003]
```

```
//待显示字符编码队列
```

```
uchar code DSY_CODE_Queue[]={
```

```

0xff,0xff,0xff,0xff,0xff,0xff,0xff,
0xa4,0xc0,0xc0,0x80,0xc0,0x80,0xf9,
0x80,0xff,0xff,0xff,0xff,0xff,0xff
};

```

```
//数码管选通
```

```
uchar code DSY_Index[]={0x01,0x02,0x04,0x08,0x10,0x20,0x40,0x80};
```

```
//延时
```

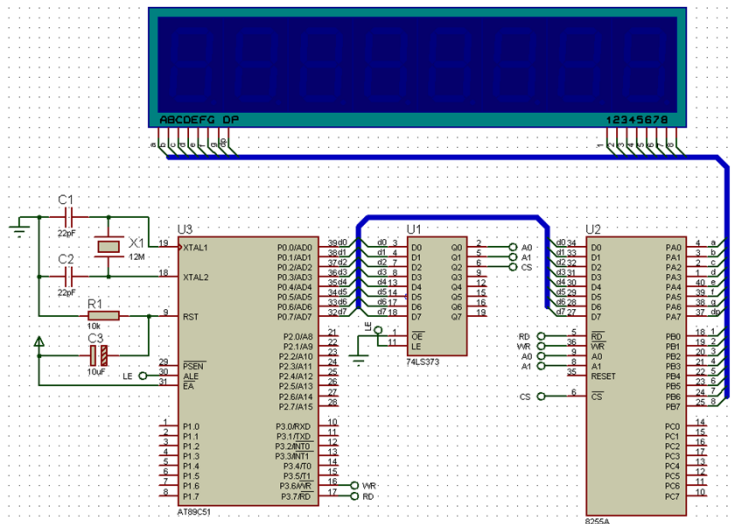
```
void DelayMS(uint ms)
```

```

{
    uchar i;
    while(ms--) for(i=0;i<120;i++);
}

```

```
//主程序
```





```

void main()
{
    uint i,j,k;

    COM=0x80;          //8255工作方式选择:PA、PB均输出, 工作方式0

    while(1)
    {
        for(j=0;j<40;j++)          //刷新显示一段时间
        {
            for(k=0;k<8;k++)        //在8个数码管上显示字符
            {
                PB=DSY_Index[k];    //位码

                PA=DSY_CODE_Queue[k+1];    //段码

                DelayMS(1);
            }
            i=(i+1)%15;
            //刷新显示一段时间后递增i, 形成滚动效果, 最大索引为14
        }
    }
}

```

## 09 555定时器实验

/\* 名称:555定时器实验

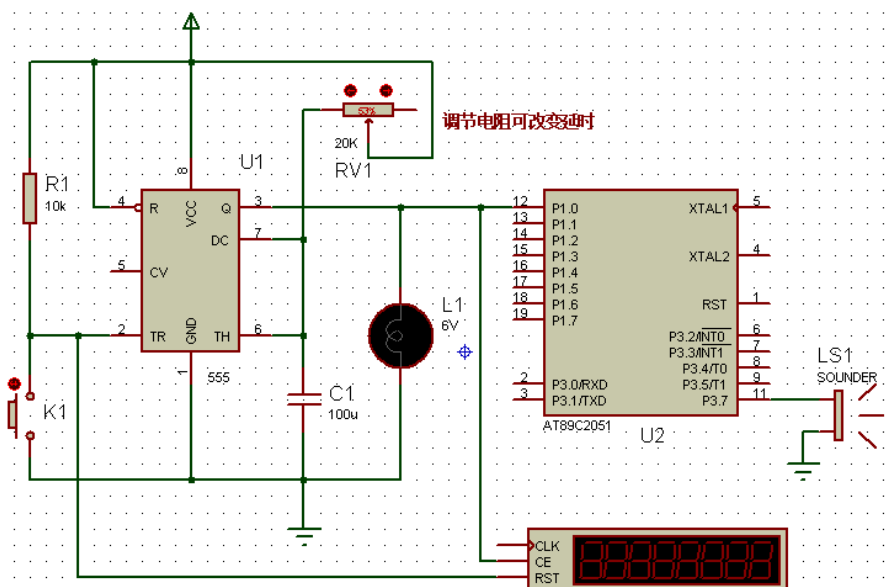
说明:调节外部电阻RV1可改变延时值, 从而影响灯点亮延时和发声延时。

```

*/
#include<reg51.h>
#define uchar unsigned char
#define uint unsigned int
sbit Signal=P1^0;
sbit BEEP=P3^7;

//延时
void DelayMS(uint ms)
{
    uchar i;

```



```

while(ms--) for(i=0;i<120;i++);
}
//主程序
void main()
{
    while(1)
    {
        if(Signal)
        {
            BEEP=~BEEP;
            DelayMS(3);
        }
    }
}

```

## 10 BCD译码数码管显示数字

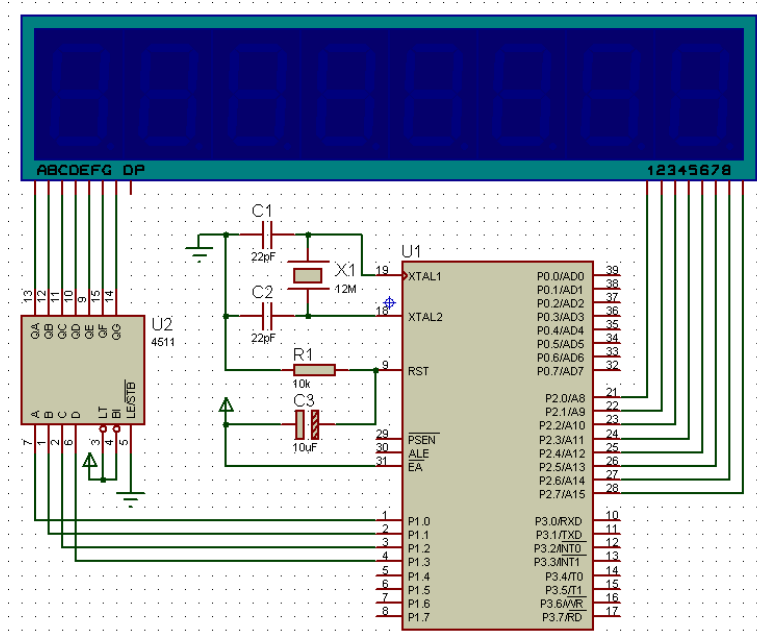
/\* 名称:BCD译码数码管显示数字

说明:BCD码经4511译码后输出数码管段码, 实现数码管显示(4511驱动数码管)。

```

*/
#include<reg51.h>
#define uchar unsigned char
#define uint unsigned int
//数码管位码
uchar code DSY_Index[]={0xfe,0xfd,0xfb,0xf7,0xef,0xdf,0xbf,0x7f};
//待显示数字(10为不显示)
uchar code
BCD_CODE[]={2,0,1,0,10,3,10,5};
//延时
void DelayMS(uint ms)
{
    uchar i;
    while(ms--) for(i=0;i<120;i++);
}
//主程序
void main()
{
    uchar k;
    while(1)

```



```

{
    for(k=0;k<8;k++)
    {
        P2=DSY_Index[k];
        P1=BCD_CODE[k];
        DelayMS(1);
    }
}
}

```

## 11 MAX7221控制数码管动态显示

/\* 名称:MAX7221控制数码管动态显示

说明:本例用MAX7221控制8只数码管动态显示,这样大大减少了单片机引脚和机器时间的占用。

```

*/
#include<reg51.h>
#include<intrins.h>
#define uchar unsigned char
#define uint unsigned int
sbit DIN=P2^0;
sbit CSB=P2^1;
sbit CLK=P2^2;

uchar Disp_Buffer[]={2,0,1,5,10,5,10,9}; //显示缓冲,10为“-”

```

//延时

```
void DelayMS(uint ms)
```

```

{
    uchar i;
    while(ms--)
    for(i=0;i<120;i++);
}

```

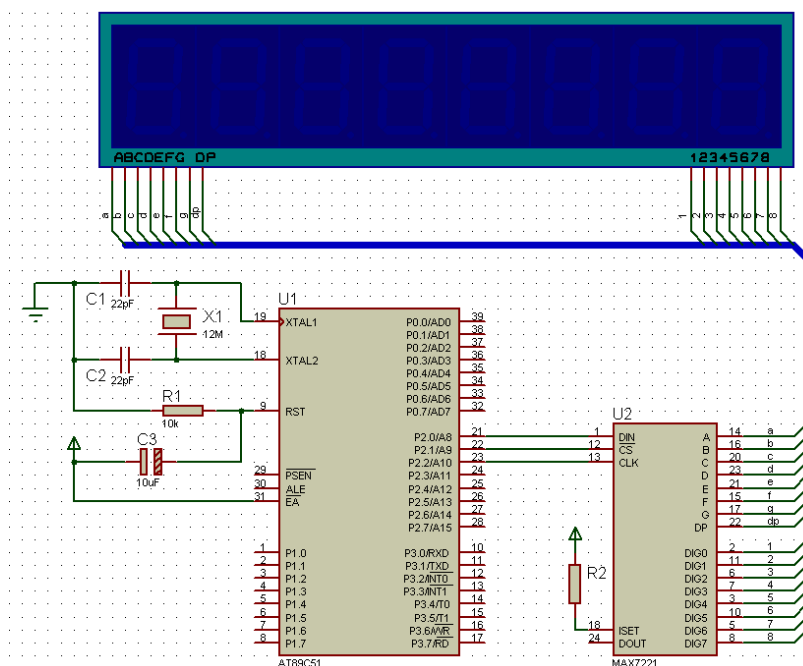
//写数据

```
void Write(uchar Addr,uchar Dat)
```

```

{
    uchar i;
    CSB=0;
    for(i=0;i<8;i++)
    {

```



```

        CLK=0;Addr<=1;DIN=CY;
        CLK=1;_nop();_nop();CLK=0;
    }
    for(i=0;i<8;i++)
    {
        CLK=0;Dat<=1;DIN=CY;
        CLK=1;_nop();_nop();CLK=0;
    }
    CSB=1;
}
//MAX7221初始化
void Initialise()
{
    Write(0x09,0xff);    //编码模式地址0x09 0x00~0xff, 为1的则位选通
    Write(0x0a,0x07);    //亮度地址0x0a 0x00~0x0f, 0x0f最亮
    Write(0x0b,0x07);    //扫描数码管个数地址0x0b, 最多扫描8只数码管

    Write(0x0c,0x01);    //工作模式地址0x0c 0x00:关闭;0x01:正常
}
//主程序
void main()
{
    uchar i;
    Initialise();        //初始化
    DelayMS(1);
    for(i=0;i<8;i++)    //显示8个数码管
    {
        Write(i+1,Disp_Buffer[i]);
    }
    while(1);
}

```

## 12 LCD1602字符液晶滚动演示程序

//main.c

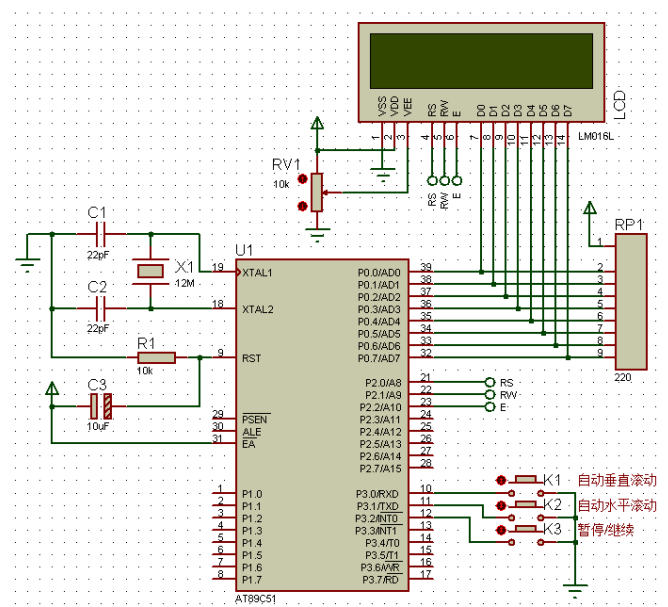
/\* 名称:LCD1602字符液晶滚动演示程序

说明:K1~K3按钮分别实现液晶垂直或

水平滚动显示及暂停与继续控制。

\*/

工程师社群371516244



```

#include<reg51.h>
#include<string.h>
#define uchar unsigned char
#define uint unsigned int
void Initialize_LCD();
void DelayMS(uint ms);
void ShowString(uchar,uchar,uchar *);
sbit K1=P3^0;
sbit K2=P3^1;
sbit K3=P3^2;
uchar code Prompt[]="Press K1 - K3 To Start Demo Prog";

//待滚动显示的信息段落, 每行不超过80个字符, 共6行

uchar const Line_Count=6;
uchar code Msg[][80]=
{
    "Many CAD users dismiss schematic capture as a necessary evil in the ",
    "process of creating PCB layout but we have always disputed this point ",
    "of view. With PCB layout now offering automation of both component ",
    "can often be the most time consuming element of the exercise.",
    "And if you use circuit simulation to develop your ideas, ",
    "you are going to spend even more time working on the schematic."
};

//显示缓冲(2行)

uchar Disp_Buffer[32];

//垂直滚动显示

void V_Scroll_Display()
{
    uchar i,j,k=0;
    uchar *p=Msg[0];
    uchar *q=Msg[Line_Count]+strlen(Msg[Line_Count]);

    //以下仅使用显示缓冲的前16字节空间

    while(p<q)
    {
        for(i=0;i<16&&p<q;i++)
        {
            //消除显示缓冲中待显示行首尾可能出现的空格

            if((i==0||i==15)&&*p==' ') p++;
            if(*p!='\0')
            {
                Disp_Buffer[i]=*p++;
            }
        }
    }
}

```

```

else
{
    if(++k>Line_Count) break;
    p=Msg[k];
    Disp_Buffer[i]=*p++;
}
}

//不足16个字符时空格补充
for(j=i;j<16;j++) Disp_Buffer[j]=' ';

//垂直滚动显示

while(F0) DelayMS(5);
ShowString(0,0,"");
DelayMS(150);
while(F0) DelayMS(5);
ShowString(0,1,Disp_Buffer);
DelayMS(150);
while(F0) DelayMS(5);
ShowString(0,0,Disp_Buffer);
ShowString(0,1,"");
DelayMS(150);
}

//最后清屏
ShowString(0,0,"");
ShowString(0,1,"");
}

//水平滚动显示

void H_Scroll_Display()
{
    uchar i,j,k=0,L=0;
    uchar *p=Msg[0];
    uchar *q=Msg[Line_Count]+strlen(Msg[Line_Count]);

    //将32个字符的显示缓冲前16个字符设为空格

    for(i=0;i<16;i++) Disp_Buffer[i]=' ';
    while(p<q)
    {
        //忽略缓冲中首尾可能出现的空格

        if((i==16||i==31)&&*p==' ') p++;
        for(i=16;i<32&&p<q;i++)
        {
            if(*p!='\0')

```

```

    {
        Disp_Buffer[i]=*p++;
    }
    else
    {
        if(++k>Line_Count) break;
        p=Msg[k];
        Disp_Buffer[i]=*p++;
    }
}

//不足32个字符时空格补充
for(j=i;j<32;j++) Disp_Buffer[j]=' ';

//水平滚动显示
for(i=0;i<=16;i++)
{
    while(F0) DelayMS(5);
    ShowString(0,L,Disp_Buffer+i);
    while(F0) DelayMS(5);
    DelayMS(20);
}

L=(L==0)?1:0;           //行号在0, 1间交替

DelayMS(300);
}

//如果显示结束时停留在第0行, 则清除第1行的内容
if(L==1) ShowString(0,1,"");
}

//外部中断0, 由K3控制暂停与继续显示
void EX_INT0() interrupt 0
{
    F0=!F0;              //暂停与继续显示控制标志位
}

//主程序
void main()
{
    uint Count=0;

    IE=0x81;             //允许外部中断0

    IT0=1;               //下降沿触发

```

```

F0=0;                //暂停与继续显示控制标志位

Initialize_LCD();
ShowString(0,0,Prompt);
ShowString(0,1,Prompt+16);
while(1)
{
    if(K1==0)
    {
        V_Scroll_Display();
        DelayMS(300);
    }
    else
    if(K2==0)
    {
        H_Scroll_Display();
        DelayMS(300);
    }
}
}
//LCD1602.c

```

/\* 名称:液晶控制与显示程序

说明:本程序是通用的1602液晶控制程序。

```

*/
#include<reg51.h>
#include<intrins.h>
#define uchar unsigned char
#define uint unsigned int
sbit RS=P2^0;
sbit RW=P2^1;
sbit EN=P2^2;

//延时

void DelayMS(uint ms)
{
    uchar i;
    while(ms-->0) for(i=0;i<120;i++);
}

//忙检查

uchar Busy_Check()
{
    uchar LCD_Status;

```



```

    RS=0;                //寄存器选择

    RW=1;                //读状态寄存器

    EN=1;                //开始读

    DelayMS(1);
    LCD_Status=P0;
    EN=0;
    return LCD_Status;
}
//写LCD命令
void Write_LCD_Command(uchar cmd)
{
    while((Busy_Check() & 0x80) == 0x80);    //忙等待

    RS=0;        //选择命令寄存器

    RW=0;        //写
    EN=0;
    P0=cmd; EN=1; DelayMS(1); EN=0;
}
//发送数据
void Write_LCD_Data(uchar dat)
{
    while((Busy_Check() & 0x80) == 0x80);    //忙等待
    RS=1; RW=0; EN=0; P0=dat; EN=1; DelayMS(1); EN=0;
}
//LCD初始化
void Initialize_LCD()
{
    Write_LCD_Command(0x38); DelayMS(1);
    Write_LCD_Command(0x01); DelayMS(1); //清屏

    Write_LCD_Command(0x06); DelayMS(1); //字符进入模式: 屏幕不动, 字符后移

    Write_LCD_Command(0x0c); DelayMS(1); //显示开, 光标关
}
//显示字符串
void ShowString(uchar x, uchar y, uchar *str)
{
    uchar i=0;

    if(y==0) Write_LCD_Command(0x80|x);    //设置显示起始位置

```

```

if(y==1) Write_LCD_Command(0xc0|x);
for(i=0;i<16;i++)                                //输出字符串
{
    Write_LCD_Data(str[i]);
}

```

13

## 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)

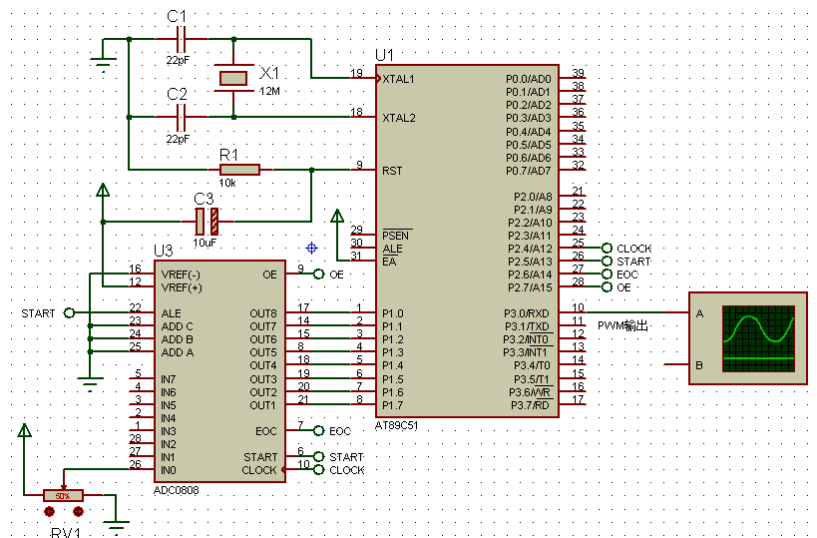
```

{
    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

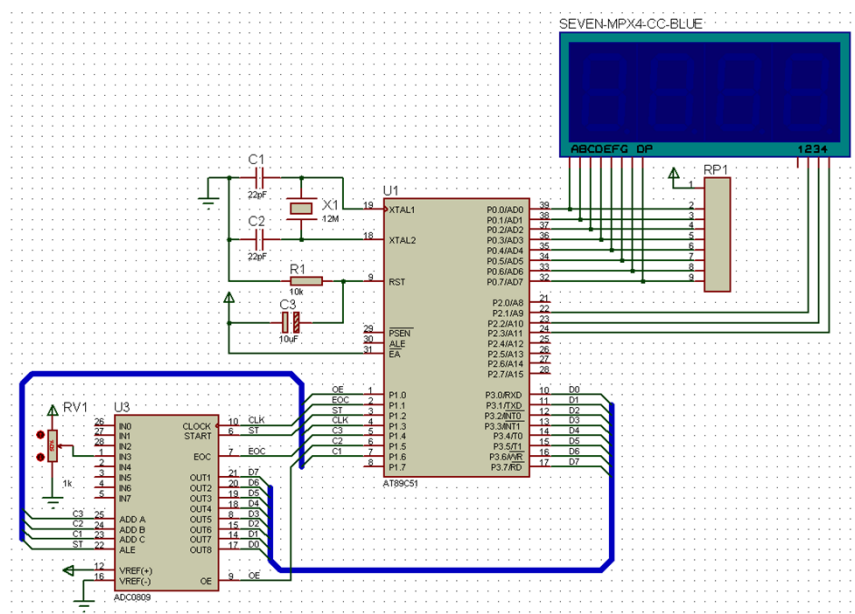
```
void Timer0_INT() interrupt 1
```

```

{
    CLK=~CLK;
}

```

工程师社群371516244



## 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-->0) 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;
}
```