东莞城市学院
DONGGUAN CITY COLLEGE

《单片机原理及应用》课程

实 验 指 导 书

人工智能学院

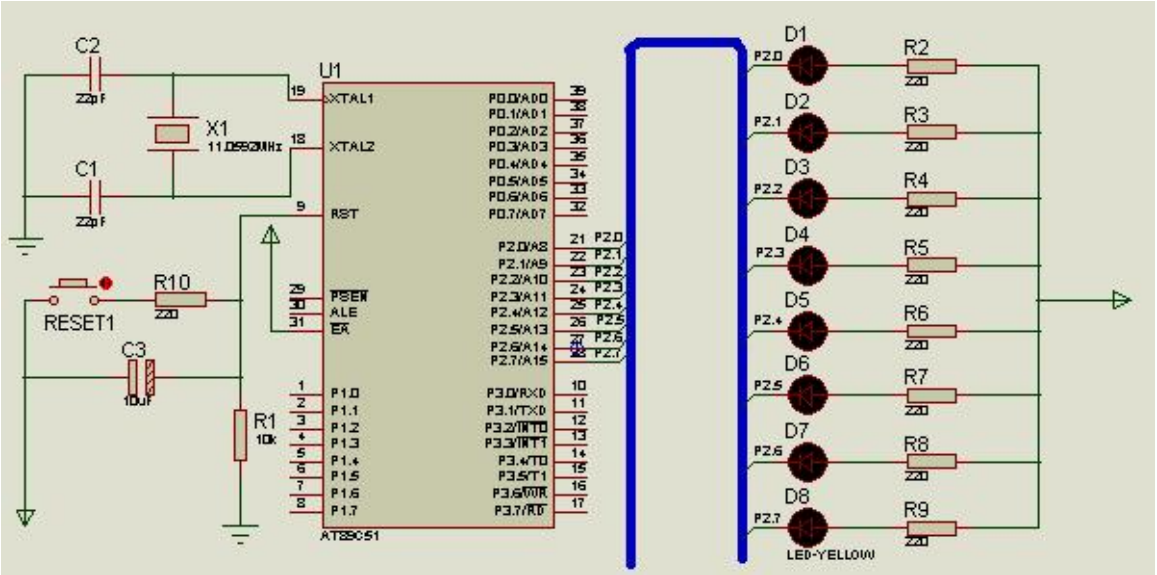
目　　录

实验一 Proteus 和 Keil 软件的使用

一、实验目的

1、熟悉 Proteus 界面，重点了解菜单栏、工具栏等基本功能；
2、学会选择元件、画导线、画总线、修改属性等基本操作；
3、学会可执行文件的加载及单片机系统程序仿真运行方法。

二、实验原理图



三、实验内容

1、利用 Proteus 完成实验原理图的绘制和参考程序的加载，仿真运行观察现象。
2、利用已给的实验原理图，修改参考程序代码，实现“流水灯从上到下然后从下到上循环显示 ”的功能。
3、利用已给的实验原理图，编写程序，实现“交叉闪烁，而且两次闪烁时间分别为 0.5s 和 2s 的功能。

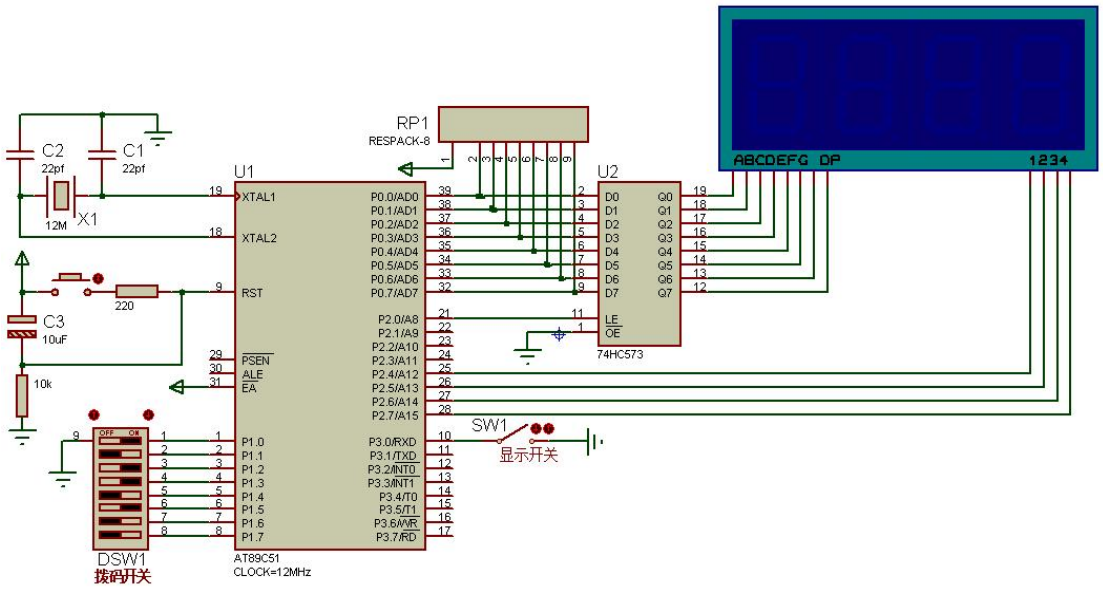
四、参考程序代码

```
/*************************************************
*说 明：      LED 从上到下流水点亮实验
-------------------------------------------------
*接线说明：     P20~P27 接 D1~D8     （各模块都要通电）
*************************************************/
#include "reg51.h"
#include "intrins.h"
```

```
#define   uchar     unsigned char  //数据类型宏定义
#define   uint       unsigned int
/*********************引脚定义*******************/
#define   out   P2
void   delayms (uint);   //延时函数声明
/*********************主函数*********************/
void main(void)
{     uchar i;
      while(1)
      {
            out=0xfe;   //初始化 P2 口
            for(i=0;i<8;i++)
            {
                  delayms(500);            //延时 500ms
                  out=_crol_(out,1);       //循环左移 1 位，点亮下一个 LED
            }
      }
}
/*********************延时函数*********************/
void delayms(uint y)
{
      uint i,j;
      for(i=y;i>0;i--)
            for(j=110;j>0;j--)
            {     ;     }
}
```

五、思考题

1、完成以下原理图的绘制



实验二　I/O 口应用（一）

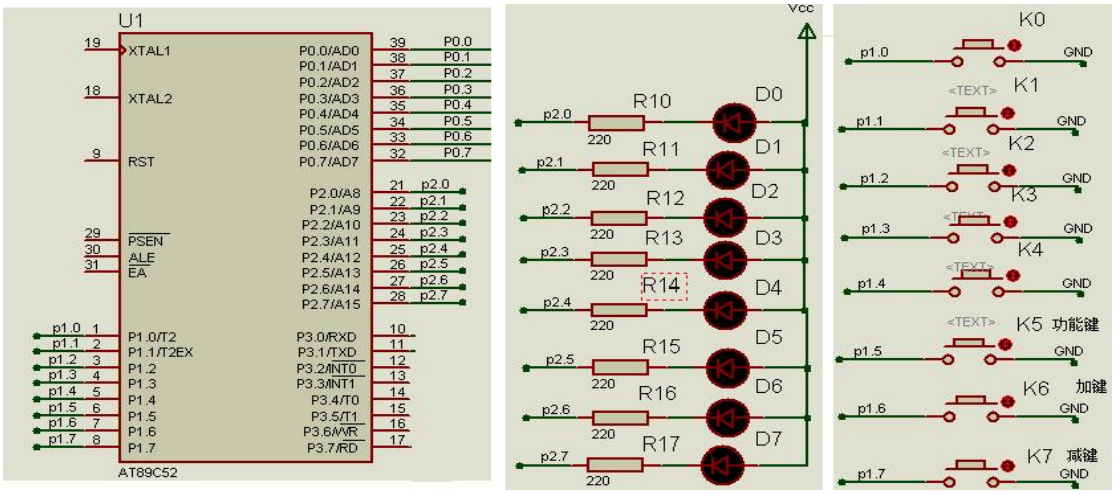
—— 根据按键、开关状态控制对应灯亮或灯灭

一、实验目的

1、学会利用 Keil 编写程序、加载可执行文件及仿真调试运行程序；
2、学会烧写器的使用与硬件电路的搭建；
3、掌握 51 单片机 IO 口基本输入输出功能的应用。

二、实验原理图

原理图 1：



三、实验内容

1、按照已给的实验原理图 1，在实验箱上合理连线，搭建实际的硬件电路，并烧写已给的参考程序代码，观察现象，**验证**其功能。

2、利用已给的实验原理图 1，**修改**参考程序代码，实现 **“软件消抖的 8 个独立式按键控制对应的 8 位 LED 灯亮灭”**的功能。

3、利用已给的实验原理图 1，**编写**程序实现 **“按键 K0 按下时，P2 口控制的 8 位发光二极管 LED 正向流水灯点亮；按键 K1 按下时，P2 口控制的 8 位发光二极管 LED 反向流水灯点亮；按键 K2 按下时，P2 口控制的 8 位发光二极管 LED 高、低 4 个交替点亮；按键 K3 按下时，P2 口控制的 8 位发光二极管 LED 闪烁”**的功能。

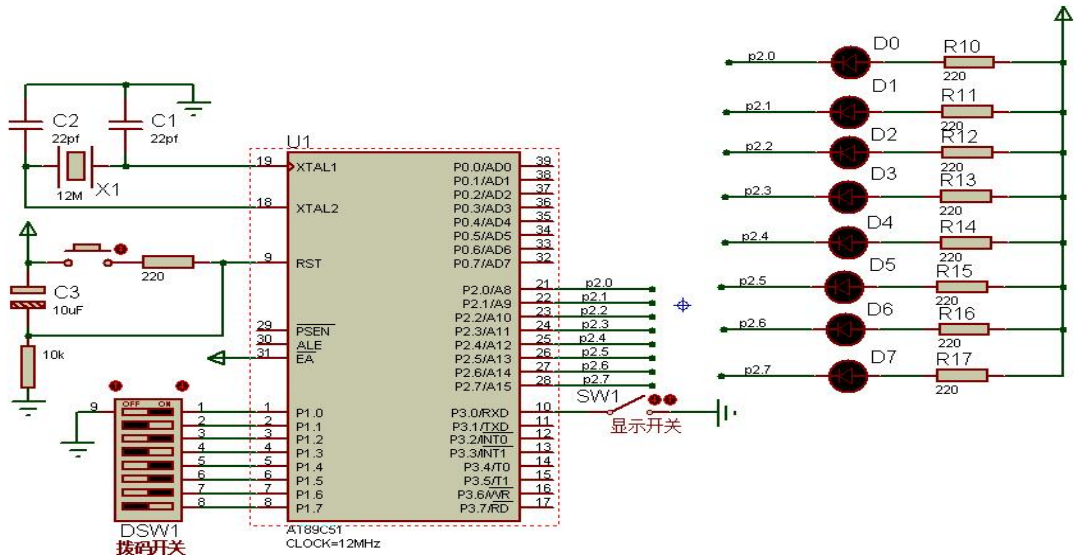
四、参考程序代码

1、软件消抖的独立式按键 K0 控制发光二极管 D0 的亮灭状态

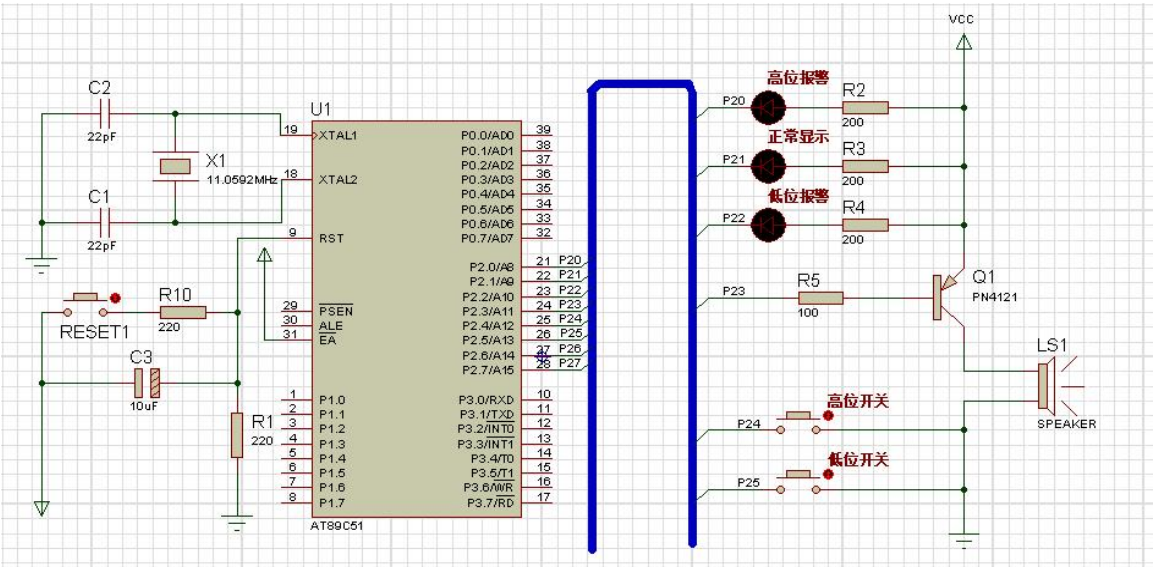
```
/*----------------------------------------------------------------------------------------------------------------第一
次按下按键 K0 后，发光二极管 D0 点亮；再次按下按键 K0 后，D0 熄灭，如此循环)
----------------------------------------------------------------------------------------------------------------*/

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

sbit D0 =P2^0;
sbit K0 =P1^0;

void delayms(uint y)
{
        uint i,j;
        for(i=y;i>0;i--)
          for(j=110;j>0;j--)
                ;
}

 main(void)
{
    D0=1;               //D0 初始是灭的
    while(1)
    {
        if(K0==0)                   //判断按键是否被按下
        {
            delayms(10)   ;          //延时 10ms
            if(K0==0)                //再次判断按键是否被按下
            {
                D0=!D0;              //确认按键是被按下，D0 状态取反
            }
            while(!K0);          //等待按键释放（松手）
        }

    }
}
```

五、思考题

1、设计以下单片机应用系统原理图，并编程实现“P3.0 口接的开关 SW1 合上，用 P1 接的 8 路拨码开关控制 P2 口接 8 只 LED 亮灭” 的功能。



2、请使用 Proteus 设计以下单片机控制水位高低限位音频报警的计算机应用系统硬件原理图。



实验三　I/O 口应用（二）

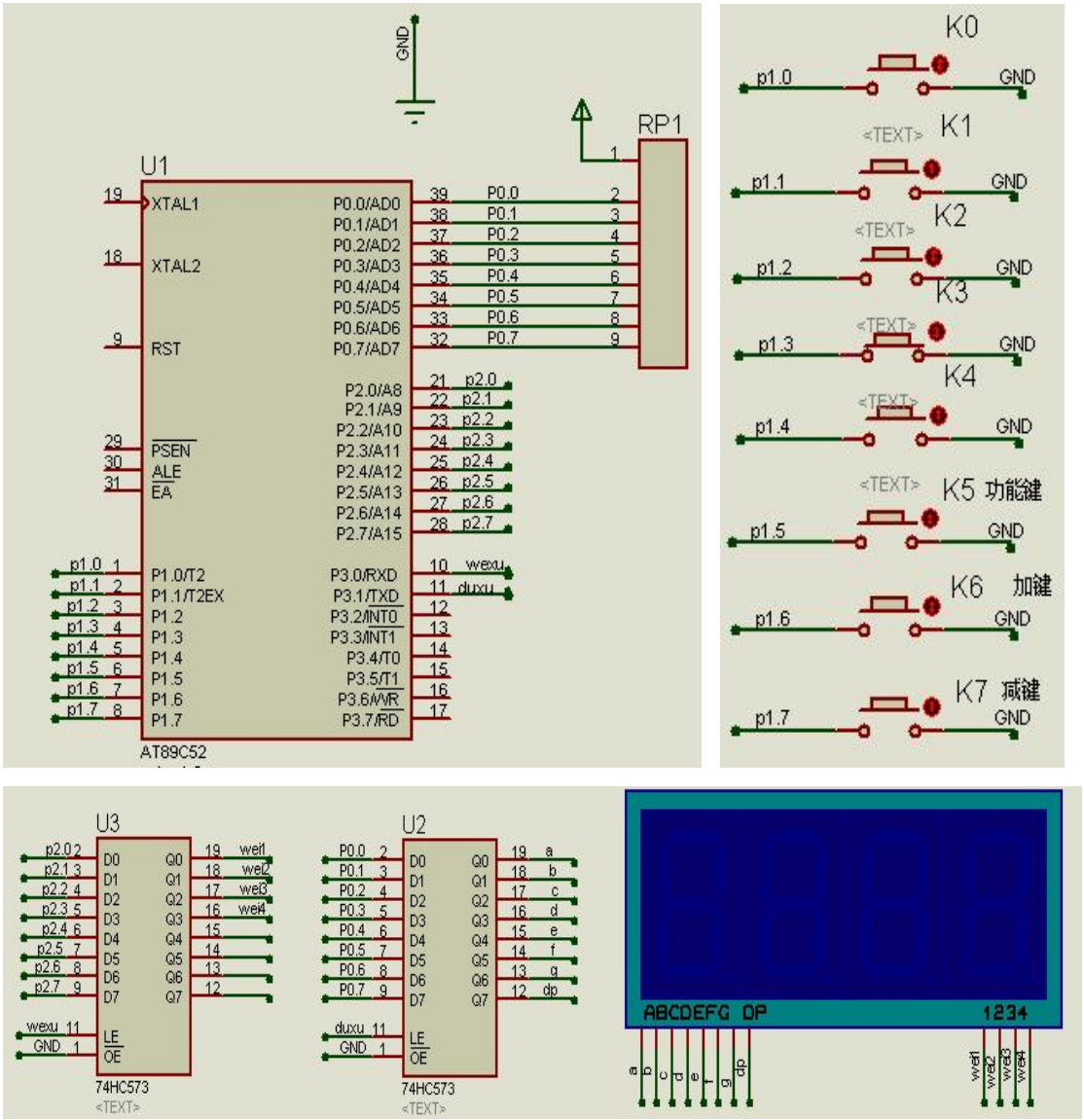
——根据按键、开关状态控制数码管显示

一、实验目的

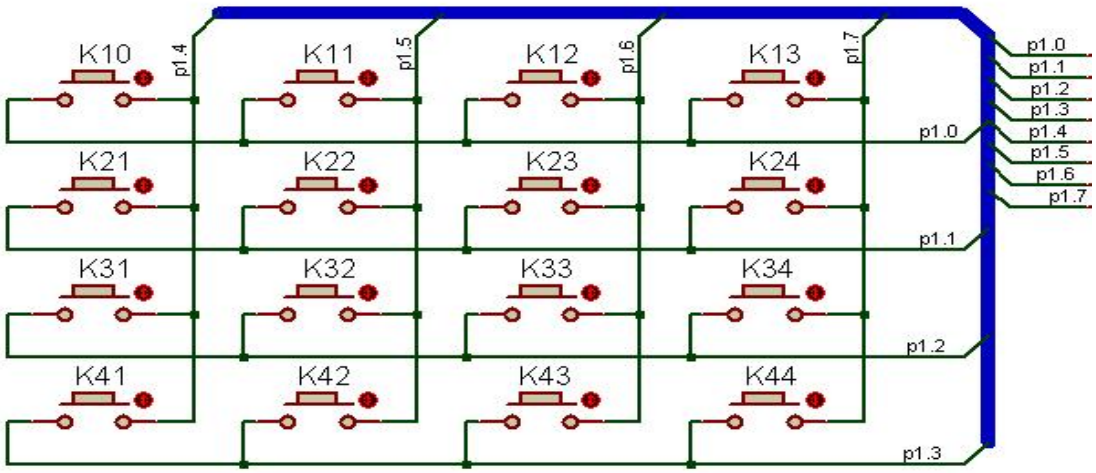
1、学会利用 Keil 编写程序、加载可执行文件及仿真调试运行程序；
2、学会烧写器的使用与硬件电路的搭建；
3、掌握 51 单片机 IO 口基本输入输出功能的应用。

二、实验原理图

原理图 1：



原理图 2：将原理图 1 中的 P1 口接的独立按键改成矩阵键盘



三、实验内容

1、按照教材 P97-98 页，例 4.4-例 4.5 设计硬件原理图与程序，并在 Proteus 仿真调试。

2、按照教材 P103-106 页，例 4.7-例 4.8 设计硬件原理图与程序，在 Proteus 仿真调试。

四、参考程序代码

1、四位一体（共阳）数码管动态显示独立式按键按下次数统计

//统计 K5 键按下次数，并数码管动态实时显示。

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

#define uchar unsigned char
#define uint    unsigned int
//0~9 的共阳数码管段码表
uchar code ledmod[] = {0xc0,0xf9,0xa4,0xb0,0x99,0x92,0x82,0xf8,0x80,0x90};

sbit    K5=P1^5;
sbit    wexu=P3^0;
sbit    duxu=P3^1;
void delayms(uint y)
{
        uint i,j;
        for(i=y;i>0;i--)
            for(j=110;j>0;j--)
                ;
}
```

```
void display(uchar shu)
{
        uchar wk=0x04;        //该变量用于存位选数据（0000 0100）
        uchar i;                    //改变位选的循环变量
        uchar buf[2];                //定义一个数组缓存要显示的 shu
        buf[0]=ledmod[shu/10];   //buf[0]存 shu 的十位对应的数码管段码
        buf[1]=ledmod[shu%10];   //buf[1]存 shu 的个位对应的数码管段码
        for(i=0;i<2;i++)
        {
            wexu=1;                  //打开位选选数据锁存器 U3
            P2=wk;               //打开第 3 个（共阳）数码管用于显示
        //    wexu=0;                  //关闭打开位选选数据锁存器 U3
            duxu=1;              //打开段选选数据锁存器 U2
            P0=buf[i];
        //    duxu=0;                  //关闭段选选数据锁存器 U2
            wk=_crol_(wk,1);    //改变位选值选通其他数码管
            delayms(10);
            P0 =0xff;                //消隐
        }
}

void main()
{
   char num;
   num=0;
   while(1)
   {
        display(num);
        if(K5==0)                    //K5 键按下
        {
             delayms(10);          //延时 10ms 再检测
             if(K5==0)             //确实有键按下
             {
                 num++;
                 if(num==100)
                 {
                     num=0;
                 }
             }
        }
       while(!K5);       //等待键释放
    }
}
```

2、四位一体（共阳）数码管动态显示矩阵键盘的对应的按键值（0-F）

```
#include<reg52.h>

#define uchar unsigned char
#define uint    unsigned int

sbit    wexu=P3^0;
sbit    duxu=P3^1;

uchar key;          //存储按键值

//0~F，灭的共阳数码管段码表
uchar code ledmod[] = {0xc0,0xf9,0xa4,0xb0,0x99,0x92,0x82,0xf8,0x80,
                       0x90,0x88,0x83,0xa7,0xa1,0x86,0x8e,0xff};

void delayms(uint y)
{
        uint i,j;
        for(i=y;i>0;i--)
             for(j=110;j>0;j--)
                  ;
}

void display(uchar num)     //显示按键位置
{
    wexu=1;
    P2=0x0f;
    duxu=1;
    P0=ledmod[num];
    delayms(10);
    P0=0xff;
}

uchar juzhen_keyscan()
{
      uchar temp;
      P1=0xf0;              //P1.0～P1.3 行线输出都为 0，P1.4～P1.7 列线都为 1，准备读列状态
      if((P1&0xf0)!=0xf0)     //如果 P1.4～P1.7 列线不全为 1，可能有键按下
      {
           delayms(10);         //延时去抖动
           if((P1&0xf0)!=0xf0)           //重读 P1.4～P1.7，若不全为 1，定有键按下
           {
```

```
P1=0xfe;        //把所有列置高电平，第一行置低电平，扫描第一行，确认是第
                  一行的哪一列有键按下
temp=P1;        //读取 P1 口值给 temp
temp=temp&0xf0;  //P1.0～P1.3 行线输出都为 0，P1.4～P1.7 列线都为 1，准
                   备读列状态
if(temp!=0xf0)       //如果 P1.4～P1.7 列线不全为 1，可能有键按下
{
    delayms(10);  //延时去抖动
    temp=P1;     //再次获取 P1 口的值，确认是否真的有键按下
    temp=temp&0xf0;  //再次让 P1.0～P1.3 行线输出都为 0，P1.4～P1.7 列线
                       都为 1，准备读列状态
    if(temp!=0xf0)       //重读 P1.4～P1.7，若不全为 1，定有键按下
    {
        temp=P1;  //确认是真的有键按下后，再次获取 P1 口的值，判断具
                    体是第一行的哪一列有键按下
        switch(temp)
        {
            case 0xee:   key =0;       //第一行第 1 列键按下，即 P1.0 和 P1.4 都为零
                 break;
            case 0xde:   key =1;       //第一行第 2 列键按下，即 P1.0 和 P1.5 都为零
                 break;
            case 0xbe:   key =2;       //第一行第 3 列键按下，即 P1.0 和 P1.6 都为零
                 break;
            case 0x7e:   key =3;       //第一行第 4 列键按下，即 P1.0 和 P1.7 都为零
                 break;
        }
        while(temp!=0xf0)      //等待按键释放
        {
            temp=P1;
            temp=temp&0xf0;
        }
    }
}


P1=0xfd;        //把所有列置高电平，第二行置低电平，扫描第二行，确认是第
                  二行的哪一列有键按下
temp=P1;        //读取 P1 口值给 temp
temp=temp&0xf0;  //P1.0～P1.3 行线输出都为 0，P1.4～P1.7 列线都为 1，准
                   备读列状态
if(temp!=0xf0)       //如果 P1.4～P1.7 列线不全为 1，可能有键按下
{
    delayms(10);  //延时去抖动
    temp=P1;     //再次获取 P1 口的值，确认是否真的有键按下
```

```
        temp=temp&0xf0;   //再次让 P1.0～P1.3 行线输出都为 0，P1.4～P1.7 列线
                            都为 1，准备读列状态
        if(temp!=0xf0)        //重读 P1.4～P1.7，若不全为 1，定有键按下
        {
            temp=P1;   //确认是真的有键按下后，再次获取 P1 口的值，判断具
                         体是第二行的哪一列有键按下
            switch(temp)
            {
                case 0xed:    key =4;       //第二行第 1 列键按下，即 P1.1 和 P1.4 都为零
                    break;
                case 0xdd:    key =5;       //第二行第 2 列键按下，即 P1.1 和 P1.5 都为零
                    break;
                case 0xbd:    key =6;       //第二行第 3 列键按下，即 P1.1 和 P1.6 都为零
                    break;
                case 0x7d:    key =7;       //第二行第 4 列键按下，即 P1.1 和 P1.7 都为零
                    break;
            }
            while(temp!=0xf0)       //等待按键释放
            {
                temp=P1;
                temp=temp&0xf0;
            }
        }
    }

    P1=0xfb;        //把所有列置高电平，第三行置低电平，扫描第三行，确认是第
                      三行的哪一列有键按下
    temp=P1;        //读取 P1 口值给 temp
    temp=temp&0xf0;   //P1.0～P1.3 行线输出都为 0，P1.4～P1.7 列线都为 1，准
                        备读列状态
    if(temp!=0xf0)        //如果 P1.4～P1.7 列线不全为 1，可能有键按下
    {
        delayms(10);   //延时去抖动
        temp=P1;     //再次获取 P1 口的值，确认是否真的有键按下
        temp=temp&0xf0;   //再次让 P1.0～P1.3 行线输出都为 0，P1.4～P1.7 列线
                            都为 1，准备读列状态
        if(temp!=0xf0)        //重读 P1.4～P1.7，若不全为 1，定有键按下
        {
            temp=P1;   //确认是真的有键按下后，再次获取 P1 口的值，判断具
                         体是第三行的哪一列有键按下
            switch(temp)
            {
                case 0xeb:    key =8;       //第三行第 1 列键按下，即 P1.2 和 P1.4 都为零
                    break;
```

```
                case 0xdb:    key =9;       //第三行第 2 列键按下，即 P1.2 和 P1.5 都为零
                    break;
                case 0xbb:    key =10;      //第三行第 3 列键按下，即 P1.2 和 P1.6 都为零
                    break;
                case 0x7b:    key =11;      //第三行第 4 列键按下，即 P1.2 和 P1.7 都为零
                    break;
            }
            while(temp!=0xf0)       //等待按键释放
            {
                temp=P1;
                temp=temp&0xf0;
            }
        }
    }

    P1=0xf7;        //把所有列置高电平，第四行置低电平，扫描第四行，确认是第
                     四行的哪一列有键按下
    temp=P1;        //读取 P1 口值给 temp
    temp=temp&0xf0;   //P1.0～P1.3 行线输出都为 0，P1.4～P1.7 列线都为 1，准
                       备读列状态
    if(temp!=0xf0)        //如果 P1.4～P1.7 列线不全为 1，可能有键按下
    {
        delayms(10);   //延时去抖动
        temp=P1;     //再次获取 P1 口的值，确认是否真的有键按下
        temp=temp&0xf0;   //再次让 P1.0～P1.3 行线输出都为 0，P1.4～P1.7 列线
                           都为 1，准备读列状态
        if(temp!=0xf0)        //重读 P1.4～P1.7，若不全为 1，定有键按下
        {
            temp=P1;   //确认是真的有键按下后，再次获取 P1 口的值，判断具
                        体是第四行的哪一列有键按下
            switch(temp)
            {
                case 0xe7:    key =12;      //第四行第 1 列键按下，即 P1.3 和 P1.4 都为零
                    break;
                case 0xd7:    key =13;      //第四行第 2 列键按下，即 P1.3 和 P1.5 都为零
                    break;
                case 0xb7:    key =13;      //第四行第 3 列键按下，即 P1.3 和 P1.6 都为零
                    break;
                case 0x77:    key =15;      //第四行第 4 列键按下，即 P1.3 和 P1.7 都为零
                    break;
            }
            while(temp!=0xf0)       //等待按键释放
            {
                temp=P1;
```

```
                    temp=temp&0xf0;
                }
            }
        }
    }
    return(key);
}

void main()
{
  while(1)
  {
     juzhen_keyscan();
     display(key);
  }
}
```

五、思考题

1、按照已给的实验原理图 1，在实验箱上合理连线，搭建实际的硬件电路，并烧写已给的参考程序代码 1，观察现象，验证其功能。

2、按照已给的实验原理图 2，在实验箱上合理连线，搭建实际的硬件电路，并烧写已给的参考程序代码 2，观察现象，验证其功能。

3、利用已给的实验原理图 1，修改参考程序代码，实现 “四位一体（共阳）数码管动态显示独立式按键加减操作按（K6 键数码管当前值+1，从 0 加到 9，然后回 0; 按 K7 键数码管当前值-1，从 9 减到 0，然后回 9”的功能。

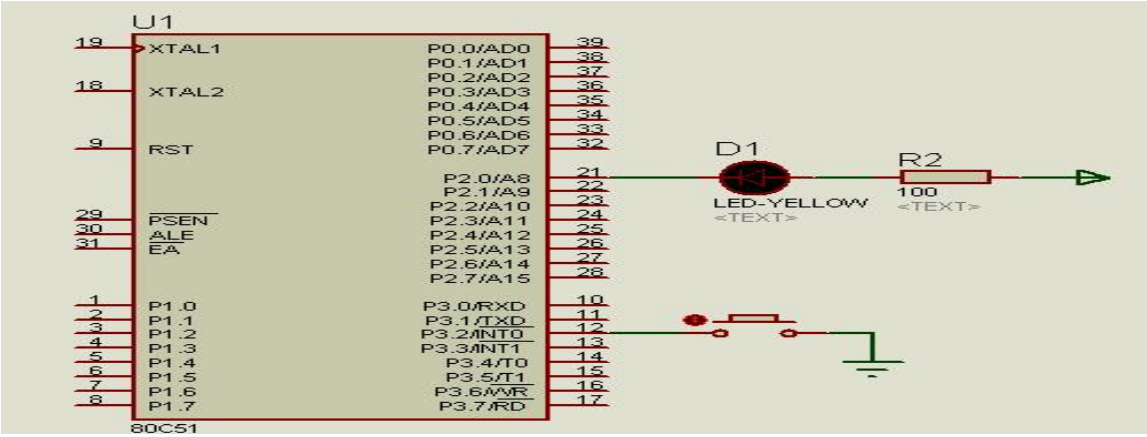
实验四　中断控制

一、实验目的

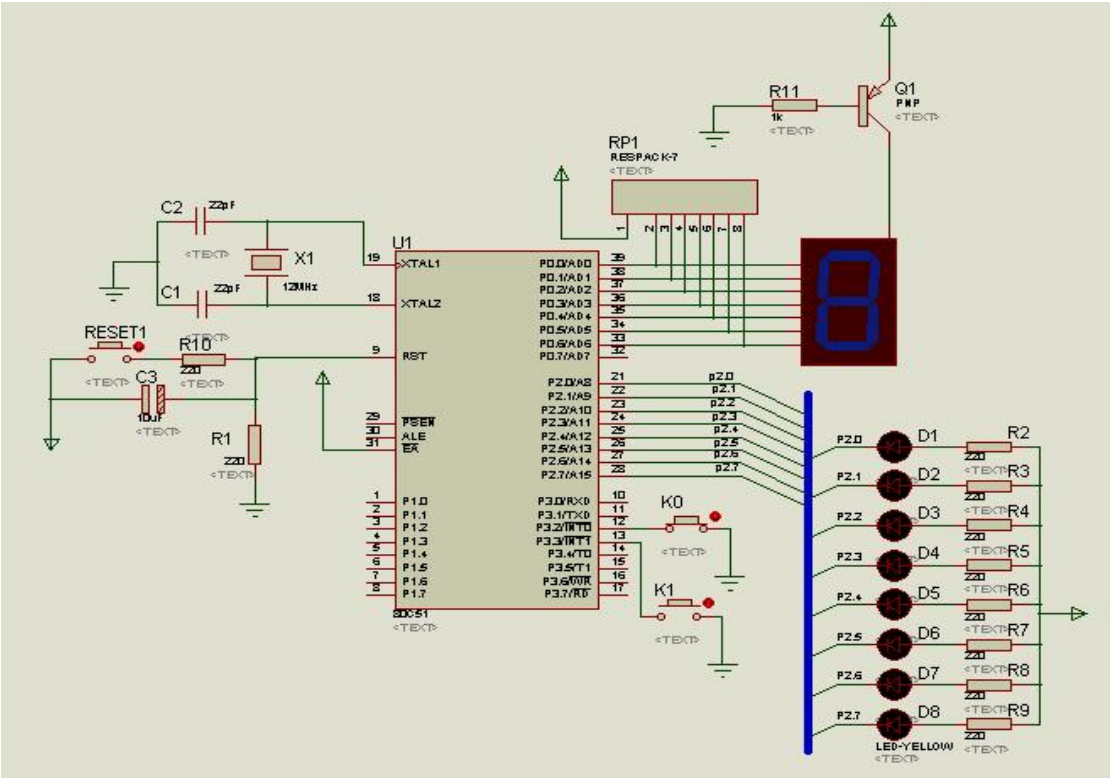
1、学会利用 Keil 编写程序、加载可执行文件及仿真调试运行程序；
2、学会烧写器的使用与硬件电路的搭建；
3、掌握 51 单片机外部中断功能的应用。

二、实验原理图

原理图 1：



原理图 2：

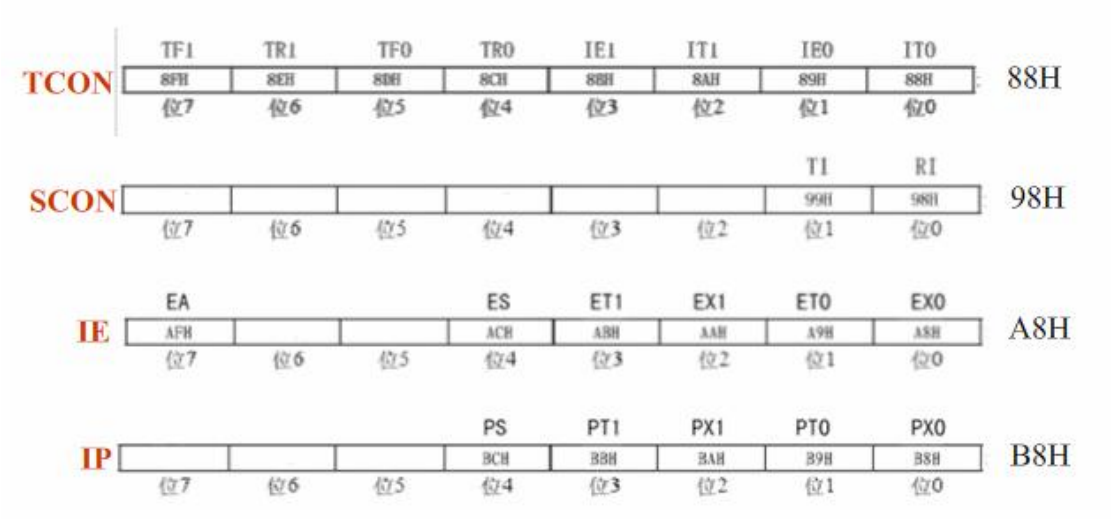


三、实验内容

1、按照已给的实验原理图 1，在实验箱上合理连线，搭建实际的硬件电路，并烧写已给的参考程序代码，观察现象，**验证**其功能。

2、利用已给的实验原理图 2，**修改**参考程序代码，实现 **“程序启动后，8 个灯交叉闪烁；单击 K0，可使 8 个灯上下循环点亮一次”**的功能。

3、利用已给的实验原理图 2，**编写**程序，实现**“程序启动后，8 个灯交叉闪烁；仅单击 K0，可使 8 个灯上下循环点亮一次；仅单击 K1，可使数码管从 0 显示到 F 显示一次后黑屏；既单击 K0 又单击 K1，则先相应 K1，再相应 K0（即 K1 优先级较高）”**的功能。（相关控制寄存器如下图所示）



四、参考程序代码

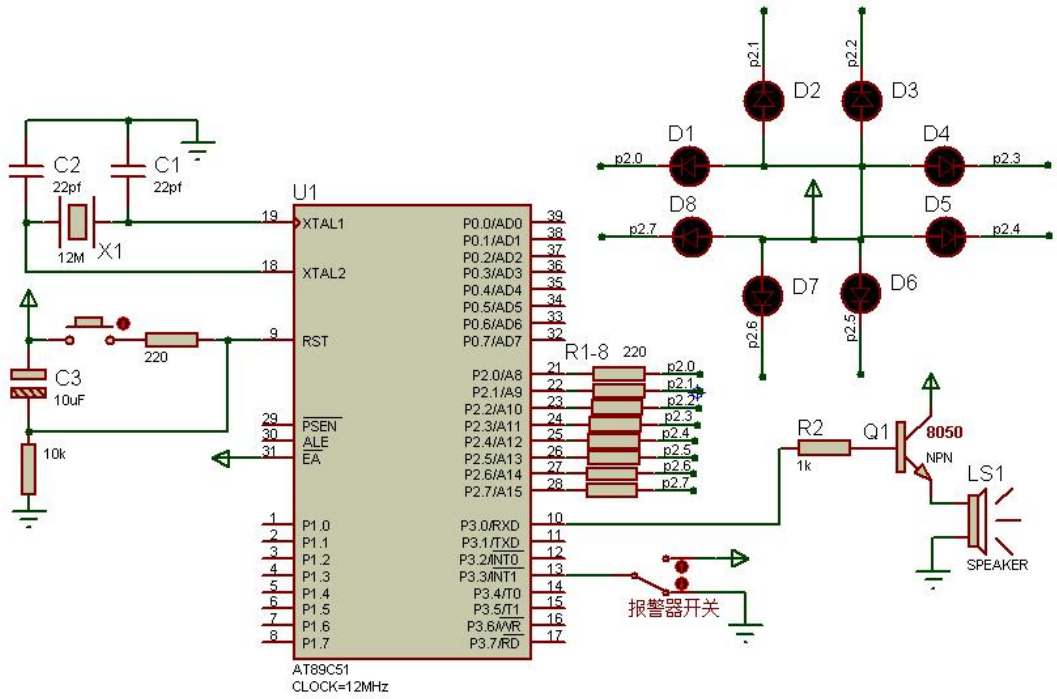
1、按键每按下一次提一个外部中断，让 D1 发光状态反转一次

```
#include <reg51.h>

sbit D1=P2^0;        //定义位变量

int0_srv () interrupt 0          //外部中断 0 对应的中断服务函数
{
  D1 = !D1;                      //输出反转电平，让灯状态反转
}

main()
{
   IE=0X81;                 /*中断初始化，也可分开写成两句“ EA=1; //开总中断允许”
                                  和  “EX0=1; //开外部中断 0 中断允许”     */
   IT0=1;     //外部中断 0 选择边沿触发方式
   while(1);
}
```

五、思考题

1、INT1 实现开关控制防盗报警器程序设计

用开关产生一个负脉冲模拟启动报警：蜂鸣器发出警笛声，8 个灯循环点亮



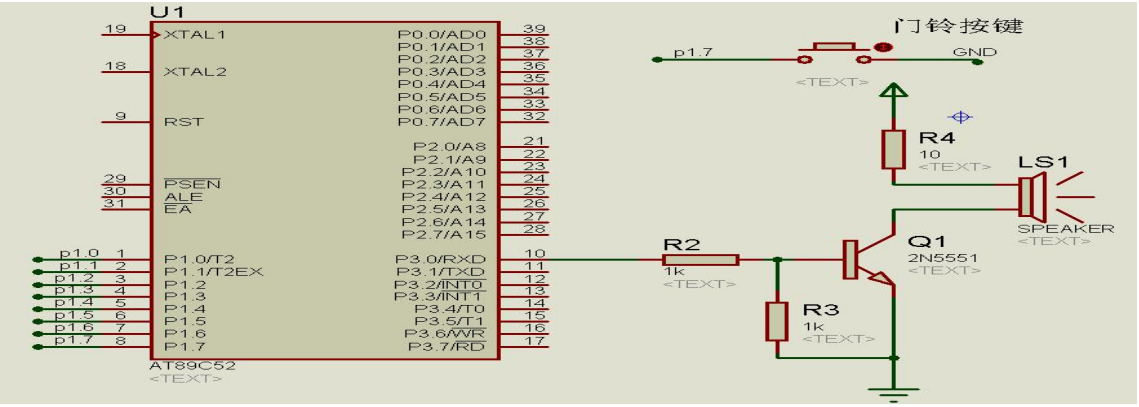
实验五　定时器的应用

一、实验目的

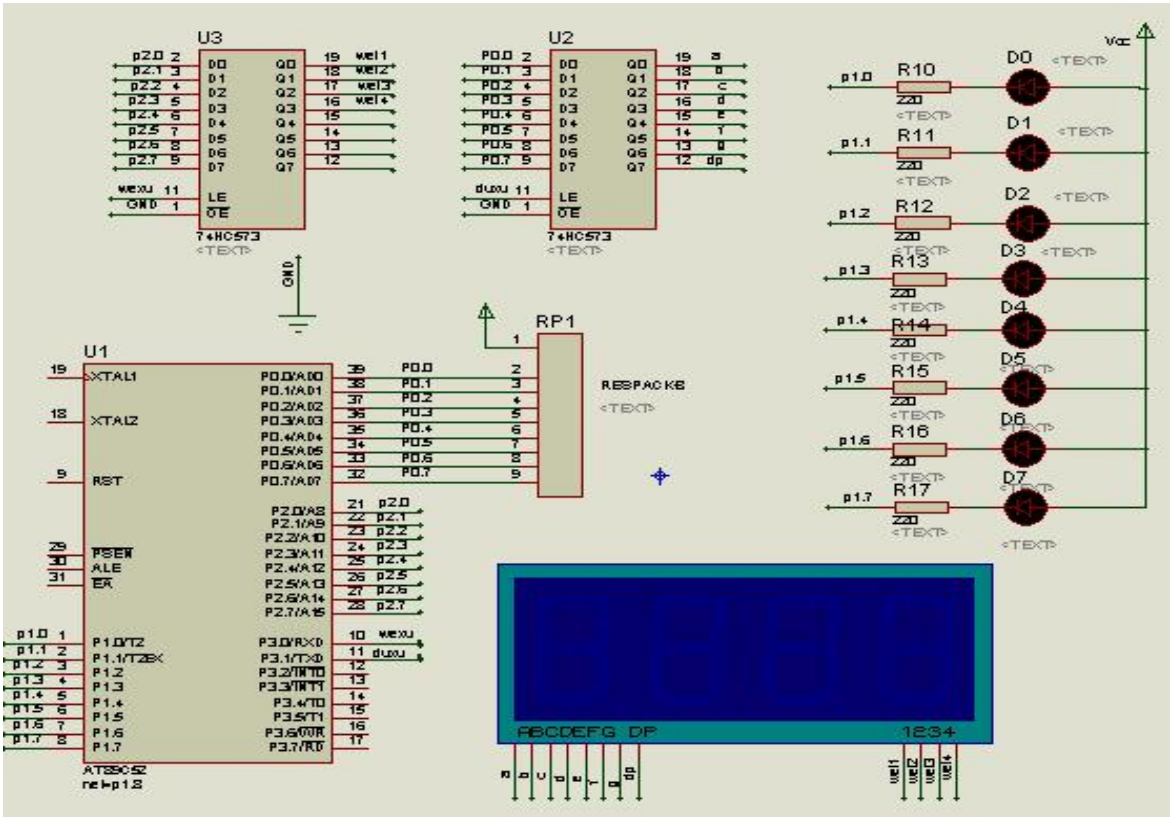
1、学会利用 Keil 编写程序、加载可执行文件及仿真调试运行程序；
2、学会烧写器的使用与硬件电路的搭建；
3、掌握 51 单片机定时/计数器功能的应用。

二、实验原理图

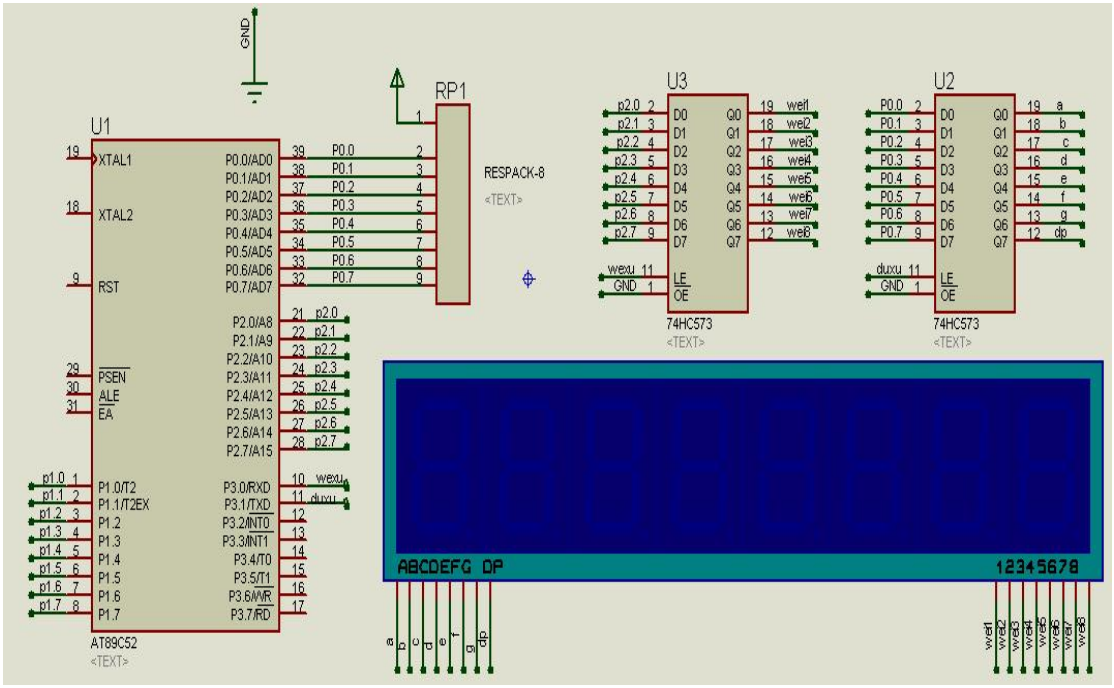
原理图 1：



原理图 2：



原理图 3：



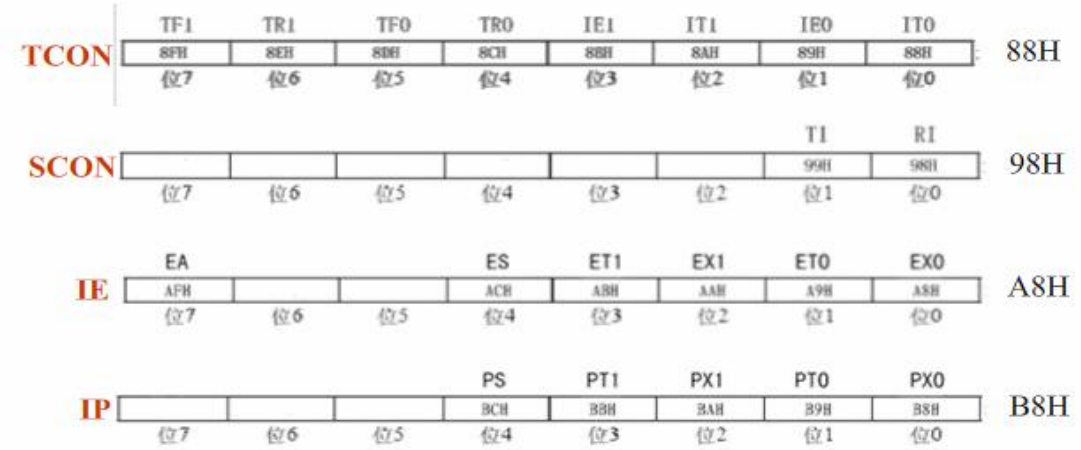
三、实验内容

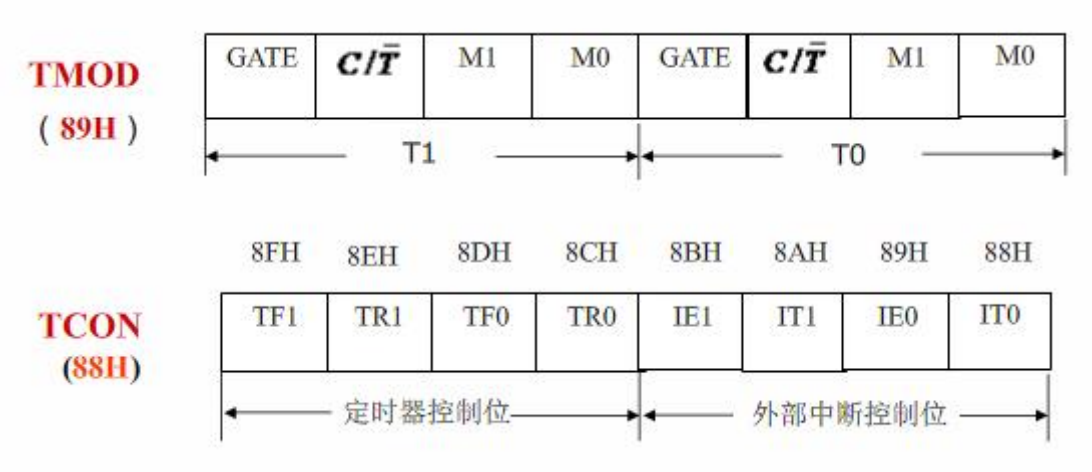
1、按照已给的实验原理图 1，在实验箱上合理连线，搭建实际的硬件电路，并烧写已给的参考程序代码 1，观察现象，**验证**其功能。

2、按照已给的实验原理图 2，在实验箱上合理连线，搭建实际的硬件电路，并烧写已给的参考程序代码 2，观察现象，**验证**其功能。

3、利用已给的实验原理图 3，**修改**参考程序代码，实现 **“集成 8 位一体共阳数码管实时显示时分秒”**的功能。具体而言，**程序启动后，数码管初始显示值为“23-59-52”；当 1S 产生时，秒加 1；秒计数到 60 时变为 00，同时分加 1；分计数到 60 时变为 00，同时时加 1；分计数到 24 时变为 00，如此周而复始进行。（提示：采用 T0 定时方式 1 中断法编程，其中 1S 定时可采用 20 次 50ms 定时中断的方案实现）**

（相关控制寄存器如下图所示）





四、参考程序代码

1、定时器设计一个叮咚门铃，当按键按下，蜂鸣器发出“叮咚”声

（叮咚门铃实际上是两个单音频声音的组合，先发出音频较高的声音，再发出音频较低的声音。本实例中先使用定时器在 P3.0 口线输出约 714Hz 的方波，然后再输出 500Hz 的方波，使扬声器发出“叮咚”的声音 ）

```
#include <reg51.h>
sbit k7 = P1^7;
sbit sp = P3^0;            //扬声器
unsigned int a=0;          //叮咚声音持续多长的变量

delay(unsigned int i)
{
    while(i--);
}

void Timer0() interrupt 1
{
    sp=!sp;
    a++;
    if(a<500)
    {
        TH0=(65536-700)/256;  //先高音，约 714Hz
        TL0=(65536-700)%256;
    }
    else if(a<1200)
    {
        TH0=(65536-1000)/256;//后低音，500Hz
```

```
            TL0=(65536-1000)%256;
        }
        else
        {
            TR0=0;
            a=0;
        }
}

int main()
{
        sp=0;   //蜂鸣器关
        IE=0x82;            //开中断
        TMOD=0x01;
        TH0=(65536-700)/256;  //先高音，约 714Hz
        TL0=(65536-700)%256;
        while(1)
        {
            k7=1;                   //作输入时须先写 1
            if(k7==0)
            {
                delay(1000);
                if(k7==0)
                {
                    TR0=1;
                    while(k7==0);
                }
            }
        }
}
```

2、定时器定时每隔 1s 数码管动态显示 0-99 和对应 LED 亮灭

```
/*****************************************
//   名称: 集成 4 位一体共阳数码管每隔约 1 秒循环显示 00~99，
              同时，P2 口控制 8 只灯从 0-99 每间隔为 1s 显示,
               共显示 0 到 99 的 100 种状态，重复上述过程。
*****************************************/
#include <reg51.h>
#define uchar  unsigned char    //宏定义
#define uint     unsigned int
sbit wexu =    P3^0;            //位选定义（74hc573 的 LE）
sbit duxu =    P3^1;            //段选定义（74hc573 的 LE）
```

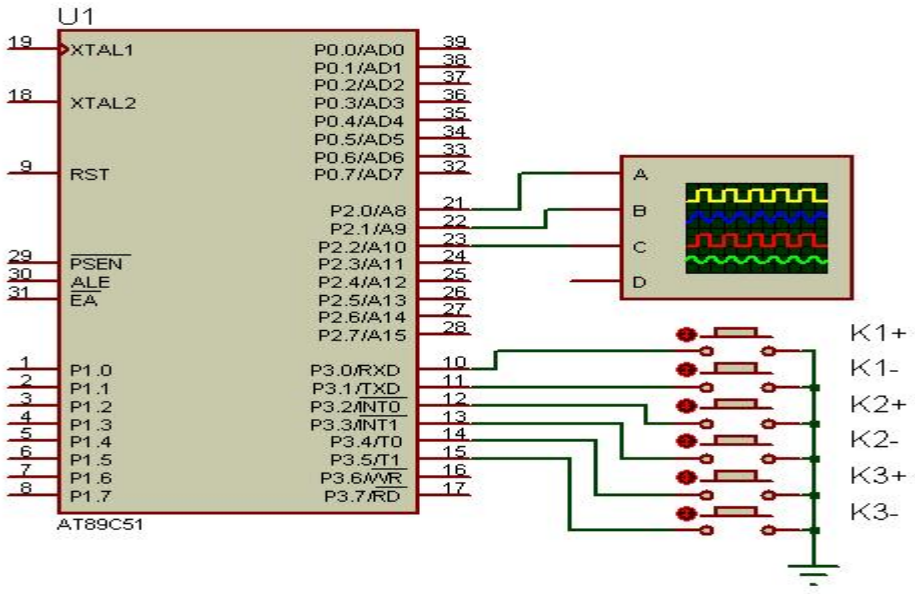
```
uchar time=0;                //中断次数统计变量初始化
uchar shu=0;
uchar shi;
uchar ge;
//0~9 的共阳数码管段码表
uchar code ledmod[] = {0xc0,0xf9,0xa4,0xb0,0x99,0x92,0x82,0xf8,0x80,0x90};

void delayms(uint y)
{
        uint i,j;
        for(i=y;i>0;i--)
          for(j=110;j>0;j--)
                ;
}

display(uchar shu)
{
          shi=shu/10;       //获取十位
          ge= shu%10;       //获取各位

          wexu=1;               //打开位选选数据锁存器 U3
          P2=0x04;          //选中第 3 个（共阳）数码管显示即 wei3     （0000 0100）
          duxu=1;               //打开段选选数据锁存器 U2
          P0=ledmod[shi];       //显示十位
          delayms(5);
          P0=0xff;                  //消隐

          wexu=1;               //打开位选选数据锁存器 U3
          P2=0x08;          //选中第 4 个（共阳）数码管显示即 wei4     （0000 1000）
          duxu=1;               //打开段选选数据锁存器 U2
          P0=ledmod[ge];        //显示个位
          delayms(5);
          P0=0xff;        //消隐
}

 main(void)
{
     TMOD=0x01;                     //使用定时器 T0，方式 1
     TH0=(65536-50000)/256;         //将定时器计时时间设定为 50ms=50000us
     TL0=(65536-50000)%256;
     EA=1;                           //启动总中断
     ET0=1;                          //定时器 T0 中断允许
     TR0=1;                          //启动定时器 T0
     while(1)
```

```
    {
        display(shu);
        P1=shu;  //P2 口控制 8 只灯从 0-99 每间隔为 1s 显示,共显示 0-99 的 100 种状态
    }
}

void time0() interrupt 1 using 1       //定时器 T0 中断服务函数（1s 计时）
{
    TR0=0;                  //关闭定时器 T0
    time++;                 //每 50ms 来一次中断，中断次数 time 加 1
    if(time==20)           //time=20 时，说明每 1s 到了
    {
        time=0;            //先清零 time 次数的计数，为下一秒计数准备
        shu++;              //1s 到了，shu 加 1
        if(shu==100)       //shu=100，就清零
        shu=0;
    }
    TH0=(65536-50000)/256;            //重新给计数器 T0 赋初值
    TL0=(65536-50000)%256;
    TR0=1;                            //启动定时器 T0
}
```

五、思考题

1、利用一个定时/计数器实现同时多路 PWM 输出程序设计

（P2.0、P2.1、P2.2 引脚输出三路 PWM 波，六只按键分别对其占空比进行调节）



实验六 串口通信的应用

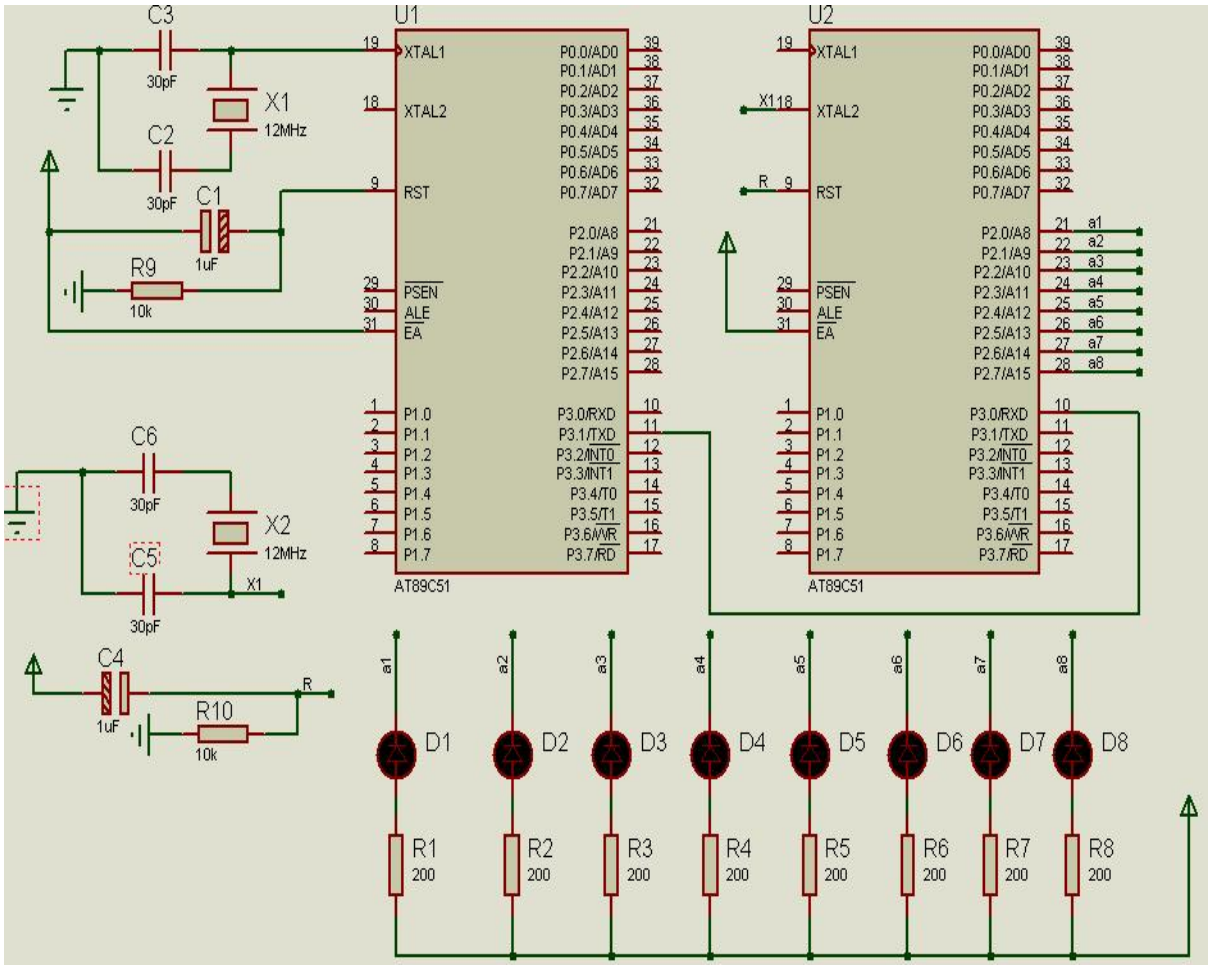
—— 单片机与单片机、单片机与 PC

一、实验目的

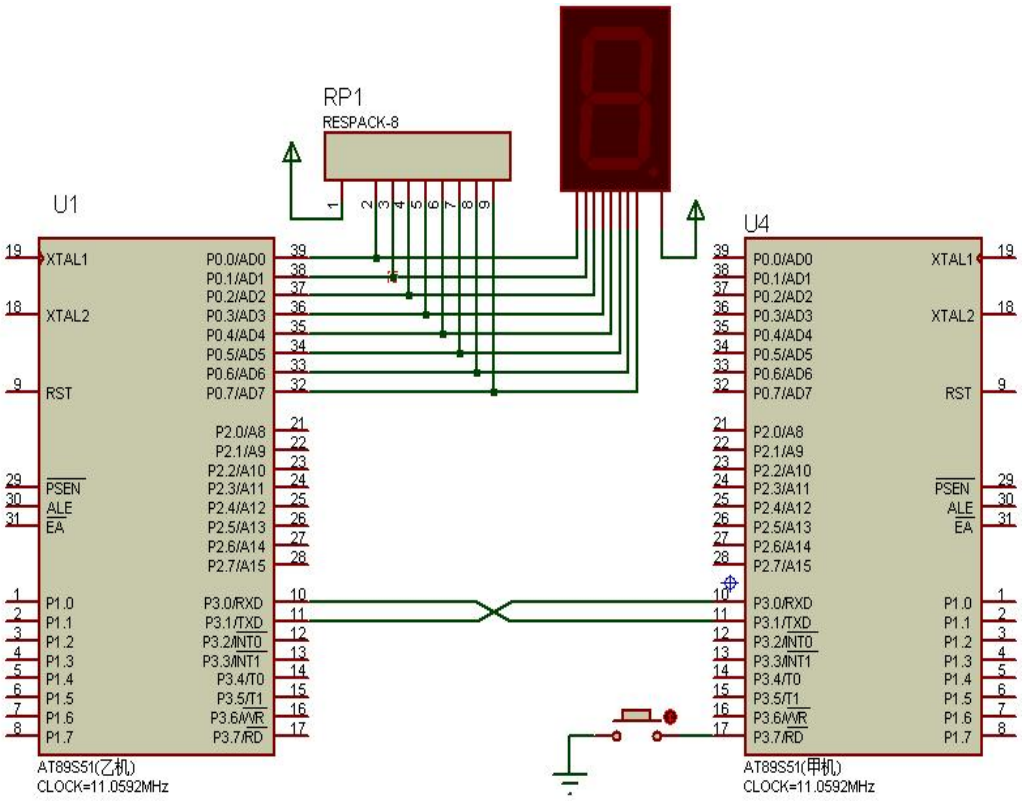
1、学会利用 Keil 编写程序、加载可执行文件及仿真调试运行程序；
2、学会烧写器的使用与硬件电路的搭建；
3、了解 Proteus 虚拟终端的使用；
4、了解 PC 超级终端（串口调试助手）和 RS232 的使用；
5、掌握 80C51 串行口的通信方式设置及波特率设置方法；
6、掌握 80C51 单片机与 PC 机的通信软件编程和硬件使用方法。

二、实验原理图

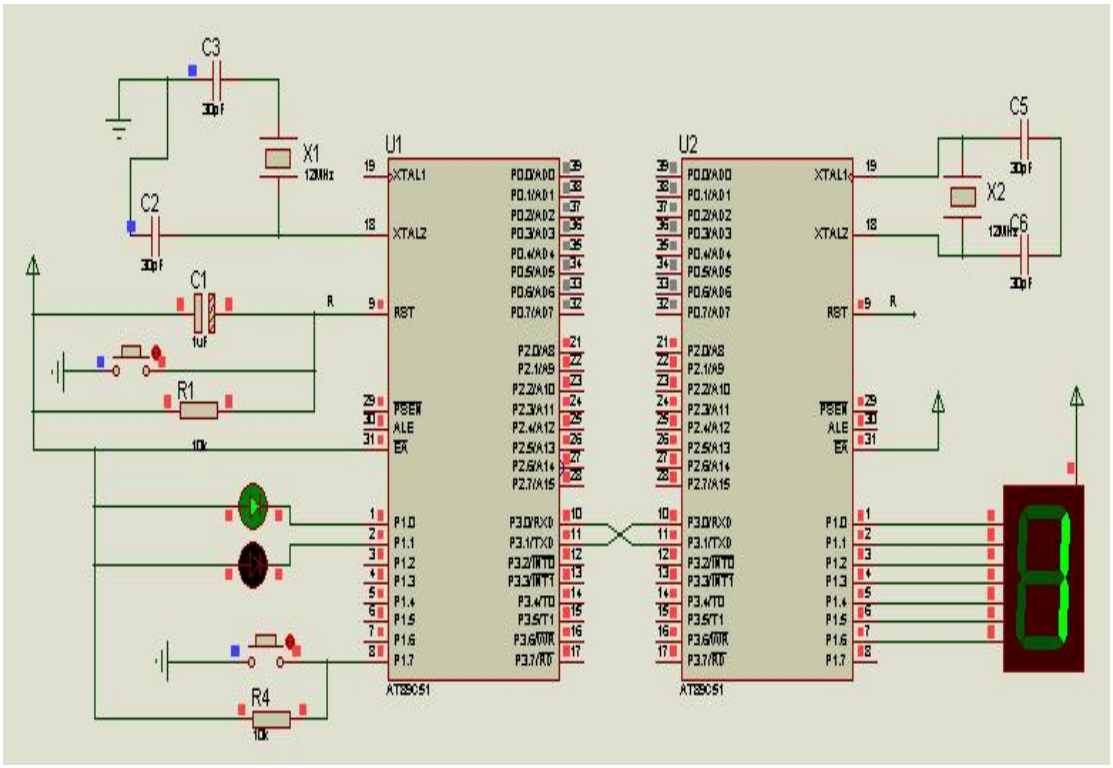
原理图 1：



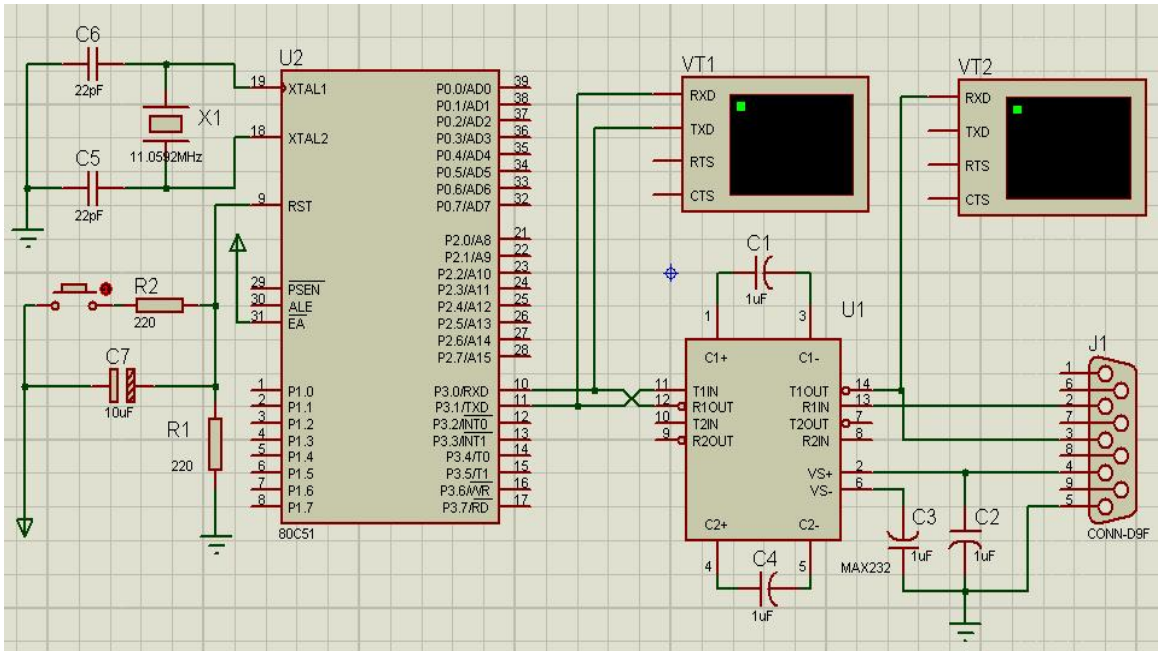
原理图 2：


RP1
RESPACK-8
U1
U4
AT89S51(乙机)
CLOCK=11.0592MHz
AT89S51(甲机)
CLOCK=11.0592MHz

原理图 3：


C3
X1
C2
C1
R1
R4
U1
U2
C5
X2
C6
AT89C51

原理图 4：



硬件连接表

MCU 模块	PB-EDU-005	PB-EDU-005
P30	RXD	
P31	TXD	
+5V	+5V	+5V
GND	GND	GND

1、注意事项：

（1）实验箱上各模块是独立供电，实验时需要用到的模块都要给它提供电源，即+5V 接口都要接到电源模块的+ 5V 电源接口，GND 接口可以不用接（默认实验箱上的 GND 网络都接在一起了），千万不要把+5V 接口接到 GND 接口上，短路烧坏保险管。

（2）RS232 接口通过串口线与 PC 相连，打开串口调试助手，正确设置波特率，在串口调试助手界面观看实验现象。

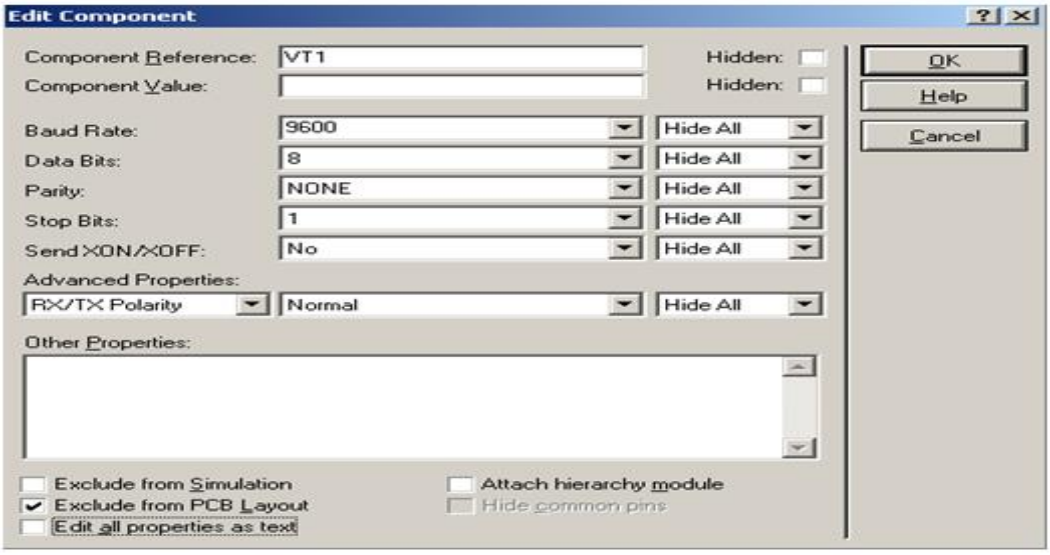
2、实验说明

本实验用到的主要知识点是：MAX232 工作原理和 Proteus 虚拟终端使用。

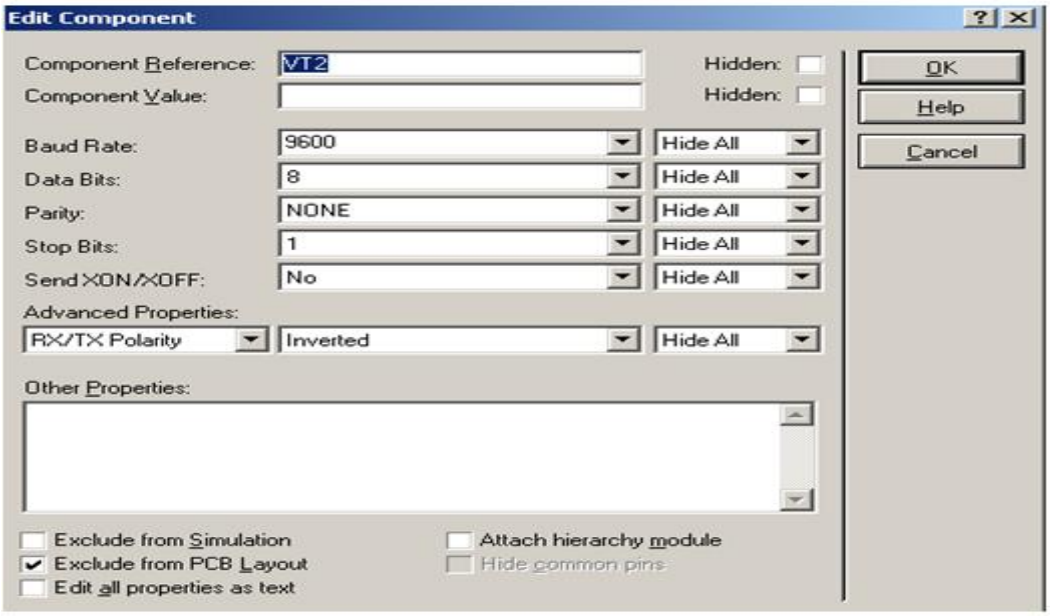
（1）在简单的应用中，最常用的是 MAX232 电路。它只需要有 3 条线即可完成通信，分别是 第二脚 RXD，第 3 脚 TXD ，第 5 脚 GND。串行通信与单片机之间的接口：RS-232C 采用负逻辑规定逻辑电平，-5V—-15V 为逻辑“1”电平，5V—+15V 为“0”电平。由于串行通信的电平逻辑定义是+15V（低电平 0），-15V(高电平 1) 而单片机中分别用 5V ,0V 来表示 1,0 它们之间必须通过电平转换才可以完成通信。

（2）此设计中将两个虚拟终端按图示挂于电路中，属性分别设置如下：

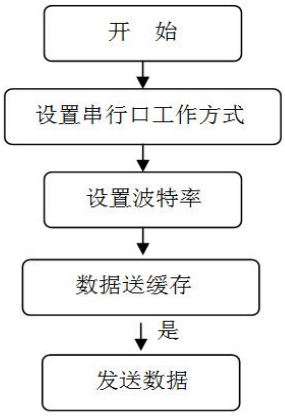
VT1：



VT2：



3、实验流程图



三、实验内容

1、按照已给的实验原理图 1，在实验箱上合理连线，搭建实际的硬件电路，并烧写已给的参考程序代码 1，观察现象，**验证**其功能。

2、按照已给的实验原理图 2，在实验箱上合理连线，搭建实际的硬件电路，并烧写已给的参考程序代码 2，观察现象，**验证**其功能。

3、利用已给的实验原理图 3，修改参考程序代码，实现 “双机串行通信（波特率为 4800b/s ，串口方式 1）”的功能。具体而言，程序启动后，按键按下， D1（红）亮，单片机 U1 开始向单片机 U2 发送数据代码 ； 单片机 U2 接收到代码后驱动数码管显示，当数码管显示数据后，再向单片机 U1 回送一数据，使 D2 （绿）灯亮时，证明两单片机之间的通信成功 ； 再返回键扫描，查询键是否按下，重复上述过程。

4、按照已给的实验原理图 4，在实验箱上合理连线，搭建实际的硬件电路，并烧写已给的参考程序代码 3，观察现象，**验证**其功能。

5、按照已给的实验原理图 4，在实验箱上合理连线，搭建实际的硬件电路，并烧写已给的参考程序代码 4，观察现象，**验证**其功能。

6、在已给的实验原理图 4 上，将 P2 口接 8 个 LED 灯，在实验箱上合理连线，搭建实际的硬件电路，并烧写已给的参考程序代码 5，观察现象，**验证**其功能。

四、参考程序代码

1、两单片机之间进行通信

要求：单片机 U1 通过串行口 TXD（P3.1 引脚）端，将控制码以方式 1 发至单片机 U2 的 RXD（P3.0 引脚）端，U2 单片机接收后把控制码送 8 位 LED 显示。

①单片机 U1 数据发送程序

```
#include<reg51.h>        //包含单片机寄存器的头文件

unsigned char code Tab[ ]={0xFE,0xFD,0xFB,0xF7,0xEF,0xDF,0xBF,0x7F,0xaa,0x0f,0xf0,0x55};
                          //流水灯控制码，该数组被定义为全局变量

void Send(unsigned char dat)          //发送一个字节数据
{
    SBUF=dat; //将数据写入发送缓冲器，启动发送
    while(TI==0) //若没有发送完毕，等待
       ;
     TI=0; //发送完毕，TI被置"1"，需将其清0
}

void delay(void)          // 延时约150ms
{
    unsigned int m;
      for(m=0;m<50000;m++)
      ;
}
```

```
20
21  void main(void)
22  {
23      unsigned char i;
24      TMOD=0x20;   //TMOD=0010 0000B，定时器T1工作于方式2
25      SCON=0x40;   //SCON=0100 0000B，串口工作方式1
26      PCON=0x00;    //PCON=0000 0000B，波特率9600
27      TH1=0xfd;    //根据规定给定时器T1高8位赋初值
28      TL1=0xfd;    //根据规定给定时器T1低8位赋初值
29      TR1=1;       //启动定时器T1
30      while(1)
31        {
32            for(i=0;i<12;i++)    //模拟检测数据
33            {
34            Send(Tab[i]);          //发送数据i
35            delay();    //150ms发送一次检测数据
36            }
37        }
38  }
```

② 单片机 U2 数据接收程序

```
01 #include<reg51.h>          //包含单片机寄存器的头文件
02
03 unsigned char Receive(void)        //  接收一个字节数据
04 {
05     unsigned char dat;
06     while(RI==0)          //只要接收中断标志位RI没有被置"1"
07           ;               //等待，直至接收完毕（RI=1）
08      RI=0;                //为了接收下一帧数据，需用软件将RI清0
09       dat=SBUF;           //将接收缓冲器中的数据存于dat
10      return dat;          //将接收到的数据返回
11 }
12
13 void main(void)
14 {
15     TMOD=0x20;   //定时器T1工作于方式2
16     SCON=0x50;   //SCON=0101 0000B，串口工作方式1,允许接收（REN=1）
17     PCON=0x00;   //PCON=0000 0000B，波特率9600
18     TH1=0xfd;     //根据规定给定时器T1高8位赋初值
19     TL1=0xfd;     //根据规定给定时器T1低8位赋初值
20     TR1=1;       //启动定时器T1
21     REN=1;        //允许接收
22     while(1)
23       {
24          P2=Receive(); //将接收到的数据送P2口显示
25       }
26 }
```

2、两单片机之间进行通信

要求：甲、乙两个单片机系统，甲机通过按下接在 P3.7 口线的按键，依次向乙机发送 0～9 十个数字；乙机以中断的方式接收甲机发来的数据，并输出到接在 P0 口的数码管进行显示。

①单片机 U1 数据发送程序

```
#include<reg51.h>

sbit key=P3^7;
unsigned char a;

delay()
{
    unsigned int i;
    for (i=0;i<200;i++);
}

sendB(unsigned char da)   //发送单字节数据子函数
{
    SBUF=da;              //待发送的数据送到SBUF，触发发送
    while(!TI);           //等待发送结束
    TI=0;                 //必须软件清除TI
}

int main()
{
    TMOD=0x20;   //定时器T1方式2作波特率发生器
    TH1=0xfd;    //波特率9600，和甲机一致
    TL1=0xfd;
    SCON=0x40;   //串行口方式1，禁止接收
    TR1=1;
    while(1)
    {
        if (key==0)
        {
            delay();
            if (key==0)
            {
                sendB(a);    //发送数据
                a=(a+1)%10;
                while(key==0)delay();
            }
        }
    }
}
```

②单片机 U2 数据接收程序

```
#include<reg51.h>

unsigned char a;
unsigned char code seg[]=
{    0xC0,0xF9,0xA4,0xB0,0x99,0x92,0x82,0xF8,0x80,0x90};

int main()
{
    TMOD=0x20;                          //定时器T1方式2作波特率发生器
    TH1=0xfd;                           //波特率9600，和甲机一致
    TL1=0xfd;
    SCON=0x50;                          //串行口方式1，允许接收
    EA=1;                               //开中断
    ES=1;
    TR1=1;
    while(1)
    {
                                        //动态停机
    }
}

void serial() interrupt 4
{
    if (RI)
    {
        RI=0;                           //必须软件清除RI
        a=SBUF;                         //接收并显示
        P0=seg[a];
    }
}
```

3、51 和 PC 通信实验（MCU 发， PC 收）

/**

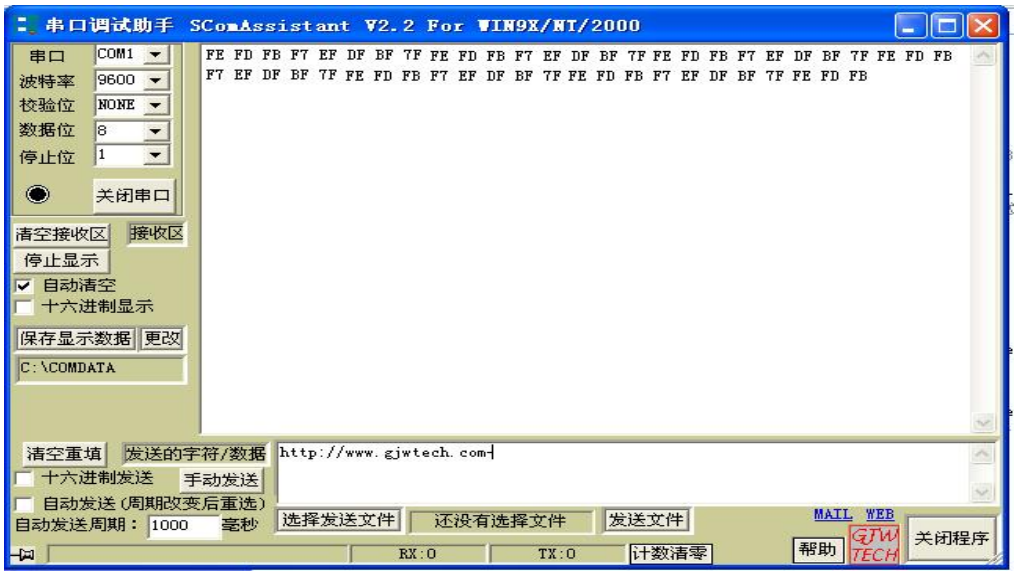
程序功能：MCU 不停向 PC 机发送数据,在屏幕上显示 。

通信格式：N.8.1, 9600

测试说明：打开串口调试助手，正确设置通信格式，观察屏幕

接线说明：P30-RXD，P31-TXD，

**/



```c
#include<reg51.h>        //包含单片机寄存器的头文件

unsigned char code Tab[ ]={0xFE,0xFD,0xFB,0xF7,0xEF,0xDF,0xBF,0x7F};
                                  //流水灯控制码，该数组被定义为全局变量

void Send(unsigned char dat)      //  向PC发送一个字节数据
{
   SBUF=dat; //将数据写入发送缓冲器，启动发送
   while(TI==0) //若没有发送完毕，等待
      ;                    //空操作
    TI=0; //发送完毕，TI被置"1"，需将其清0
}

void delay(void)      //   延时约150ms
{
   unsigned char m,n;
   for(m=0;m<200;m++)
   {
        for(n=0;n<250;n++)
            ;
    }
}
```

```c
void main(void)
{
    unsigned char i;
    TMOD=0x20;   //TMOD=0010 0000B，定时器T1工作于方式2
    SCON=0x40;   //SCON=0100 0000B，串口工作方式1
    PCON=0x80;   //PCON=1000 0000B，波特率9600
    TH1=0xfa;     //根据规定给定时器T1高8位赋初值
    TL1=0xfa;     //根据规定给定时器T1低8位赋初值
    TR1=1;       //启动定时器T1
   while(1)
    {
           for(i=0;i<8;i++)    //一共8位数据
          {
         Send(Tab[i]);          //发送数据i
          delay();    //150ms发送一次数据
           }
     }
}
```

4、51 和 PC 通信实验（MCU 发， PC 收）

/***

程序功能：MCU 不停向 PC 机发送数据,在屏幕上显示 **如“MCU 123 ”和“Welcome to you!”。**

通信格式：N.8.1, 9600

测试说明：打开串口调试助手，正确设置通信格式，观察屏幕

接线说明：P30-RXD，P31-TXD，

***/

```
#include <reg52.h>
#define uchar unsigned char
#define uint unsigned int
uchar a[]=" MCU 123 \r";
uchar b[]="Welcome to you! \r ";

void UART_Init(void);
void UART_SendData(uchar dat);
void UART_SendString(uchar *p);
void Delayms(uint j);

void main(void)
{
     UART_Init();
     while(1)
     {
          UART_SendString(a);      //发送字符串
          //UART_SendData('\n');     //换行
          Delayms(100);            //延时
          UART_SendString(b);      //发送字符串
          //UART_SendData('\n');     //换行
          Delayms(100);            //延时
     }
}
```

```
void UART_Init(void)     //    串口初始化
{
    SCON=0x50;
    TMOD=0x20;
    PCON=0x00;
    TH1 =0xfd;
    TL1 =0xfd;  //预置初值，设波特率为9600
    TR1 =1;
    ES  =1;
    EA  =1;     //开中断

}

void UART_SendData(uchar dat)          //    串口发送一个字节的数据
{
    SBUF=dat;       //发送数据
    while(TI==0);   //判断是否发送完
    TI=1;
}
```

```
void UART_SendString(uchar *p)
{
  while(*p!='\0')
  {
   UART_SendData(*p++);
  }

}
void Delayms(uint j)
{
    uchar i;
    for(;j>0;j--)
    {
        i=250;
        while(--i);
        i=249;
        while(--i);
    }
}
void INT_UART_Rev() interrupt 4          // 串口接收中断函数
{
    uchar *temp;
    if(RI)
     {
       RI=0;
       *temp++=SBUF;
     }
}
```

5、51 和 PC 通信实验（PC 发，MCU 收）

/**

程序功能：PC 机不停向 MCU 发送数据

通信格式：N.8.1, 4800

测试说明：打开串口调试助手，正确设置通信格式，发送数据，观察 P2 灯的状态

接线说明：P30-RXD，P31-TXD，

**/

```
#include<reg51.h>       //包含单片机寄存器的头文件

unsigned char Receive(void)      //    接收一个字节数据
{
  unsigned char dat;
  while(RI==0)   //只要接收中断标志位RI没有被置"1"
        ;        //等待，直至接收完毕（RI=1）
  RI=0;       //为了接收下一帧数据，需将RI清0
  dat=SBUF;   //将接收缓冲器中的数据存于dat
  return dat; //将接收到的数据返回
}

void main(void)
{
   TMOD=0x20;   //定时器T1工作于方式2
   SCON=0x50;   //SCON=0101 0000B，串口工作方式1,允许接收（REN=1）
   PCON=0x00;   //PCON=0000 0000B，波特率4800
   TH1=0xfa;   //根据规定给定时器T1高8位赋初值
   TL1=0xfa;   //根据规定给定时器T1低8位赋初值
   TR1=1;     //启动定时器T1
   REN=1;     //允许接收
   while(1)
   {
        P2=Receive(); //将接收到的数据送P2口显示
   }
}
```

五、思考题

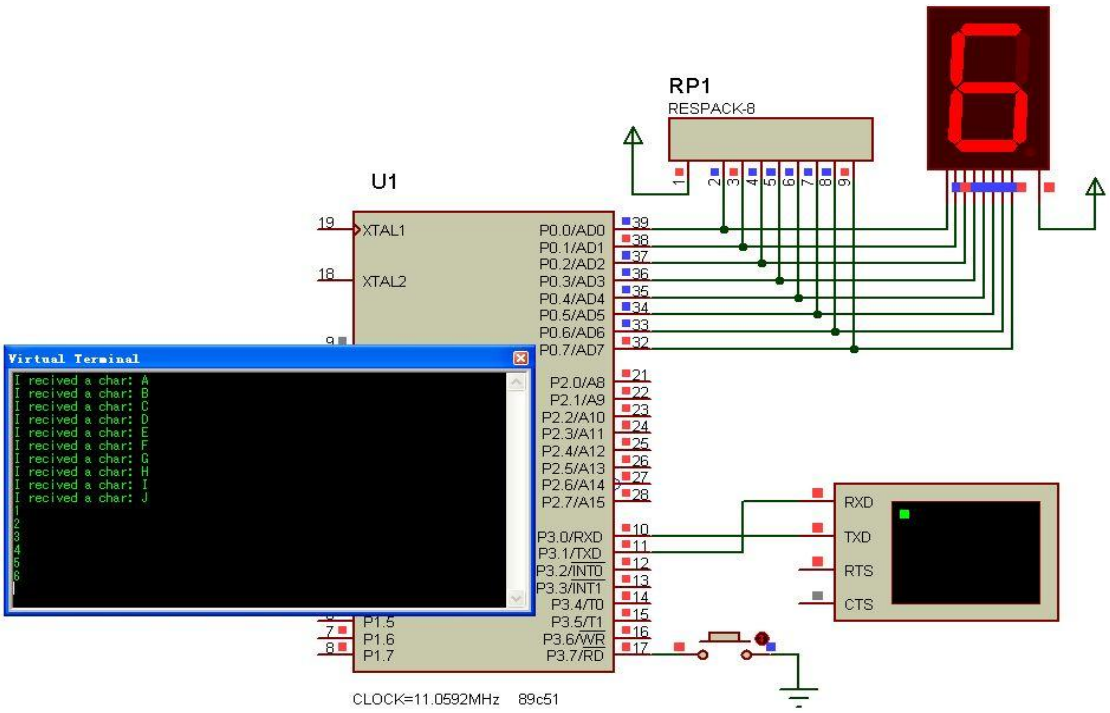
1、单片机与 PC 机之间的双机通信

/***

每按一次接在单片机 P3.7 脚的按键，便向 PC 机发送字符串“I seng a char: ”和一个大写字母，这个字母依次从 A 到 Z；单片机以中断的方式接收 PC 机发送来的数字 0～9 并在数码管上显示，同时再把这个数字回送给 PC 机。

***/

单片机与 PC 机通信系统中单片机的程序与双机通信中单片机的程序相似，不同的是单片机发送时要以 ASCII 码的形式发送，比如大写字母 A 的 ASCII 码是 0x41 或者写成’A’，则 B 的 ASCII 码可以写作 0x41+1 或者’A’+1，C 的 ASCII 码可以写作 0x41+2 或者’A’+2 等等。同时，单片机接收到的是 ASCII 码，需要转换为十六进制才能在数码管上显示，由于 0 的 ASCII 码是 0x30，所以 0～9 这 10 个数字的 ASCII 码只要减去 0x30 或者’0’即可转换为相应的十六进制编码，比如，6 的 ASCII 码是 0x36，减去 0x30 或者’0’即得 6。



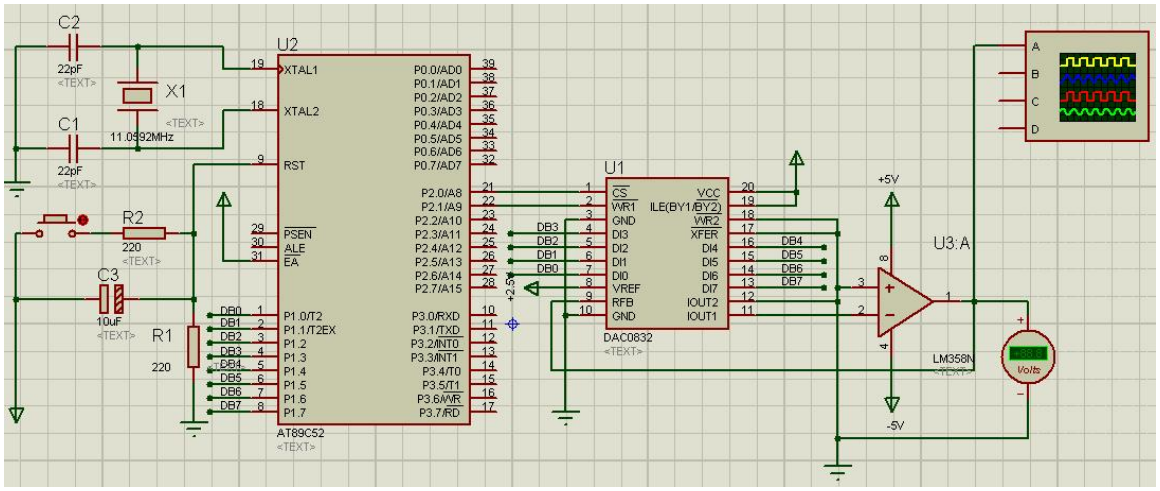
实验七　A/D、D/A 转换的应用

一、实验目的

1、学会利用 Keil 编写程序、加载可执行文件及仿真调试运行程序；
2、学会烧写器的使用与硬件电路的搭建；
3、掌握 DA、AD 转换的原理和使用方法。

二、实验原理图

1、DAC0832 模数转换实验



硬件连接表

MCU 模块	PB-EDU-010
P10~P17	**DB0~DB7**
P20	**0832_CS**
P21	**0832_WR**
-5V	-5V
+5V	+5V
GND	GND

DAC0832 的 8 位输入口连接到单片机的 P1 口，CS 和 WR1 引脚分别是片选信号和数据信号，都是低电平有效。WR2 和 XFER 直接连接到地，ILE 连接到 VCC。IOUT1 和 IOUT2 为 DAC0808 的输出口，输出的是电流值，还需要通过运算放大器 LM358 把它转换为电压值。在硬件实验中，使用电压表测最后转换得到的电压值，并与理论值作比较。

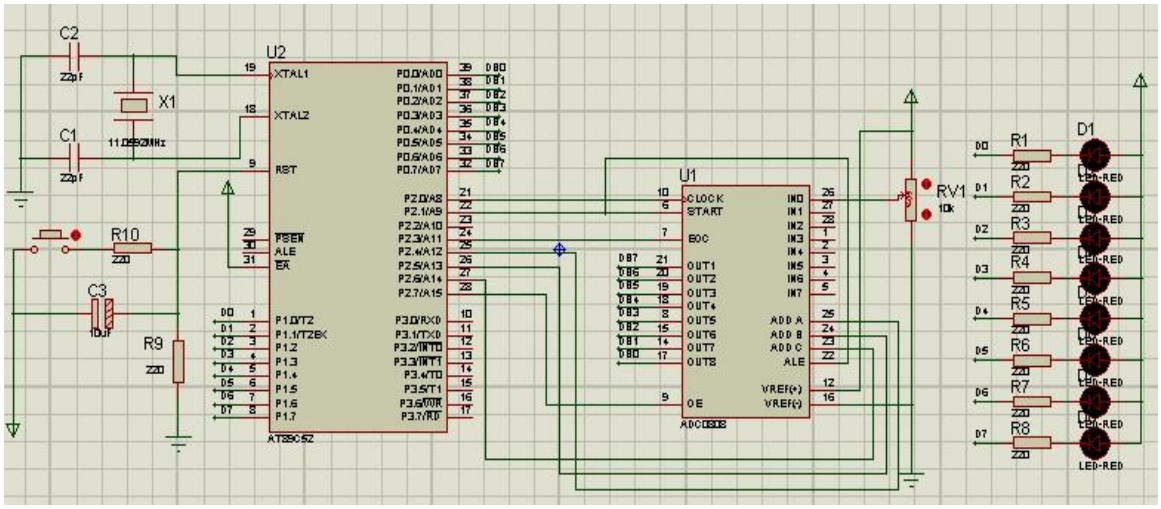
注意事项：

（1）实验箱上各模块是独立供电，实验时需要用到的模块都要给它提供电源，即+5V 接口都要接到电源模块的+ 5V 电源接口，GND 接口可以不用接（默认实验箱上的 GND 网络都接在一起了），千万不要把+5V 接口接到 GND 接口上，短路烧坏保险管。
（2）**参考电压 DA_AREF 短路帽短接右边，选中 0832，可以调节电位器选择参考电压**。

实验说明：

本实验用到的主要知识点是：DAC0832 的工作原理。DAC0832 是采用先进的 CMOS 工艺制成的单片电流输出型 8 位 D/A 转换器。它采用的是 R-2R 电阻梯级网络进行 DA 转换。电平接口与 TTL 兼容。具有两级缓存。通过电压表测量 DAC 转换出来的电压值

2、ADC0808 模数转换实验



硬件连接表

MCU 模块	PB-EDU-010	PB-EDU-011
P00~P07		**DB0~DB7**
P10~P17	**D1~D8**	
P20	**0809_CLK**	
P21	**0809_ST 和 ALE**	
P23	**0809_EOC**	
P24	**0809_A**	
P25	**0809_B**	
P26	**0809_C**	
P27	**0809_OE**	
+5V	**+5V**	**+5V**
GND	**GND**	**GND**

ADC0808 是 8 位的 A/D 转换器件，在本实验中，**它的输出口连接到单片机的 P0 口，控制信号 ALE 和 START 连接到 P21，时钟信号 CLOCK 连接到 P20，EOC 连接到 P23，输出控制信号 OE 到 P27，输入选择地址 ADD_A、ADD_B 和 ADD_C 连接到 P24-P26。**

注意事项：

（1）实验箱上各模块是独立供电，实验时需要用到的模块都要给它提供电源，即+5V 接口都要接到电源模块的+ 5V 电源接口，GND 接口可以不用接（默认实验箱的 GND 网络都接在一起了），千万不要把+5V 接口接到 GND 接口上，短路烧坏保险管。

（2）**先下载程序在单片机中，下载完成后再按照上表连接硬件，否则下载程序时会出现 FLASH 校验失败。在做实验时电位器下面的短路帽短接到右边，选中 ADC0809 的通道 0。**

实验说明：

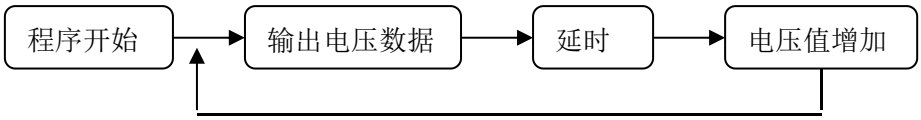
A/D 转换器大致有三类：一是双积分 A/D 转换器，优点是精度高，抗干扰性好，价格便宜，但速度慢；二是逐次逼近 A/D 转换器，精度、速度、价格适中；三是并行 A/D 转换器，速度快，价格也昂贵。本实验用的 ADC0808 属第二类，是 8 位 A/D 转换器，每采集一次一般需100 μs 。本实验可采用延时方式或查询方式读入 A/D 转换结果，也可以采用中断方式读入结果，在中断方式下，A/D 转换结束后会自动产生 EOC 信号，将其与 CPU 的外部中断相接。调整电位计，得到不同的电压值，转换后的数据通过发光二级管输出。

三、实验内容

1、 按照已给的 **DAC0832 模数转换**实验原理图，在实验箱上合理连线，搭建实际的硬件电路，并烧写已给的 **DAC0832 模数转换**参考程序代码，观察现象，**验证**其功能。
2、 按照已给的 **ADC0808 模数转换**实验原理图，在实验箱上合理连线，搭建实际的硬件电路，并烧写已给的 **ADC0808 模数转换**参考程序代码，观察现象，**验证**其功能。

四、参考程序流程图和代码

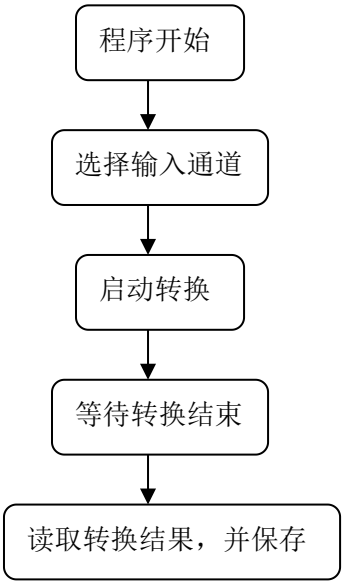
1、 DAC0832 模数转换实验程序流程图和代码




```
/*************************************************
  *Descriptoon: 数模转换，产生一锯齿波，输出 0－2.5V 电压，可用电压表查看变化
*************************************************/
    #include "reg51.h"
    #define uchar unsigned char       //数据类型宏定义
    #define uint unsigned int

    #define     out     P1      // 引脚定义
    sbit dac_cs=P2^0;
    sbit dac_wr=P2^1;
    void delayms(uint j);

    void main(void)     //主函数
    {
    uchar i;
    while(1)
            {
             for(i=0;i<255;i++)
             {
                 out=i;
```

```
                dac_cs=0;
                dac_wr=0;
                dac_cs=1;
                dac_wr=1;
                delayms(200);
             }
            }

}

void delayms(uint j)     //   延时函数
{
uchar i;
for(;j>0;j--)
            {
             i=250;
             while(--i);
             i=249;
             while(--i);
            }
}
```

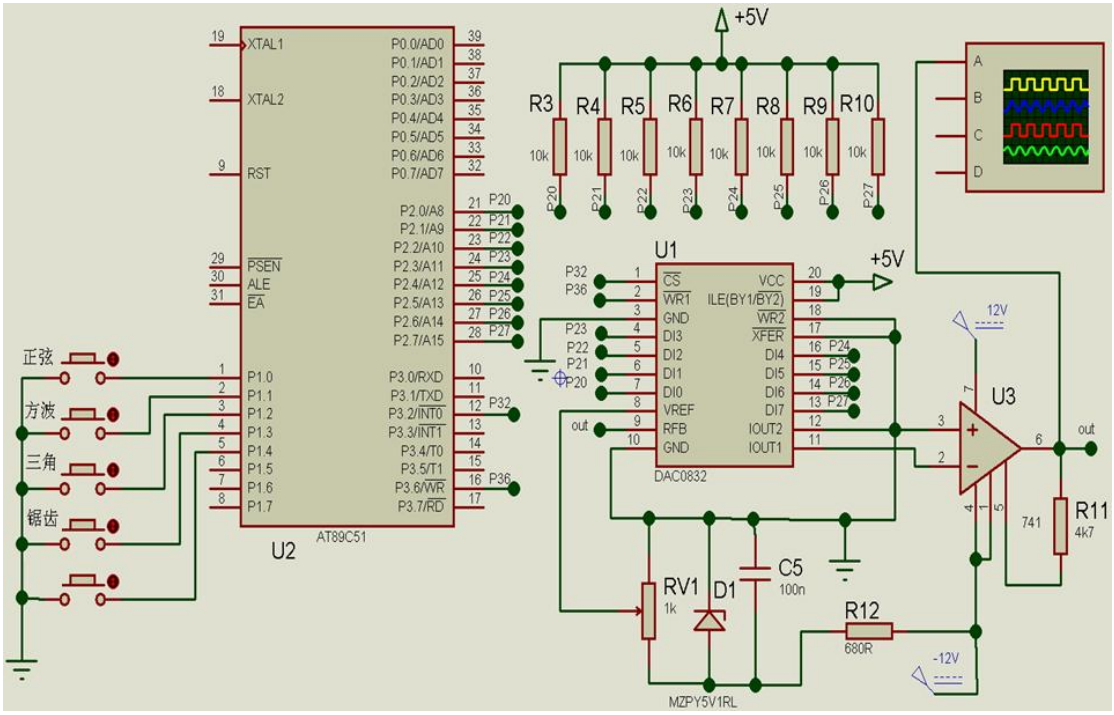
2、ADC0808 模数转换实验程序流程图和代码



```
/*************************************************************************
    *Descriptoon:利用 ADC0808/(0809)做 A/D 转换器，由实验板上的电位器提供模拟量输入，
                将模拟量转换成二进制数字量，用 P1 口接发光二极管来显示转换的结果实物
                板中使用 ADC0809 芯片，它与 ADC0808 是全兼容的（转换精度低些）。
************************************************************************* /
        #include "reg51.h"
        #define uchar unsigned char//数据类型宏定义
        #define uint unsigned int
        /*********************引脚定义*******************/
        #define led   P1
        #define out   P0
        sbit start=P2^1;
        sbit oe=P2^7;
        sbit eoc=P2^3;
        sbit clock=P2^0;
        sbit add_a=P2^4;
        sbit add_b=P2^5;
        sbit add_c=P2^6;
        /*********************主函数**********************/
        void main(void)
        {
        uchar   temp;
        add_a=0;add_b=0;add_c=0;   //选择 ADC0808 的通道 0
        while(1)
                {
                start=0;
                start=1;
                start=0;   //启动转换
                while(1){clock=!clock;if(eoc==1)break;}//等待转换结束
                oe=1;   //允许输出
                temp=out; //暂存转换结果
                oe=0;   //关闭输出
                led=temp; //采样结果输出到 LED
                }
        }
```

五、思考题

1、 简易波形发生器（正弦波、三角波、锯齿波、方波）



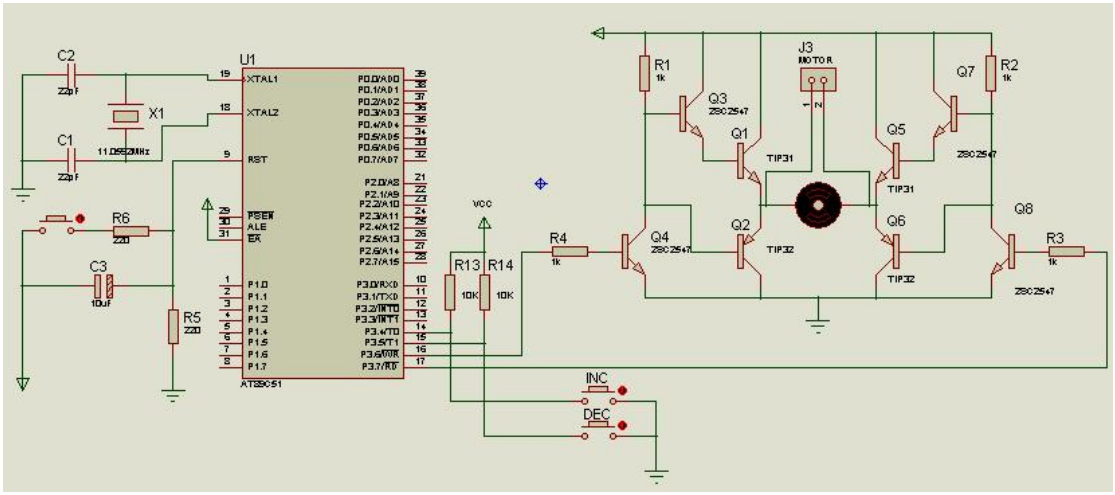
实验八　步进电机的应用

一、实验目的

1、学会利用 Keil 编写程序、加载可执行文件及仿真调试运行程序；
2、学会烧写器的使用与硬件电路的搭建；
3、了解脉宽调制（PWM）的原理；
4、掌握控制直流、步进电机转动的编程方法；
5、掌握单片机控制直流、步进电机的基本原理。

二、实验原理图

1、直流电机实验



硬件连接表

MCU 模块	PB-EDU-011	PB-EDU-009
P34	K1	
P35	K2	
P36		DIR
P37		PWM
+5V	+5V	+5V
GND	GND	GND

注意事项：

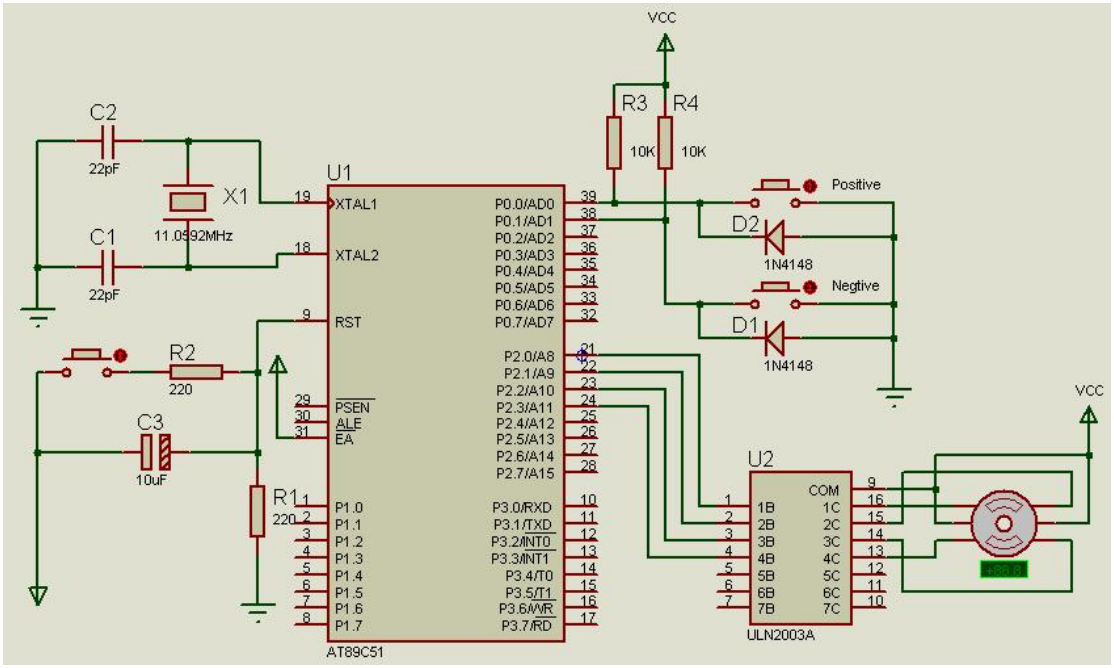
（1）实验箱上各模块是独立供电，实验时需要用到的模块都要给它提供电源，即+5V 接口都要接到电源模块的+ 5V 电源接口，GND 接口可以不用接（默认实验箱上的 GND 网络都接在一起了），千万不要把+5V 接口接到 GND 接口上，短路烧坏保险管。

实验说明：

本实验用到了两个主要知识点是：达林顿管的应用、PWM 波的产生方法。在实验中，我们改变 PWM 的占空比，然后查看对电机速度的影响。

实验流程图：

2、步进电机实验



硬件连接表

MCU 模块	PB-EDU-011	MCU 模块
P00	K1	
P01	K2	
P20~P23		B1~B4
+5V	+5V	+5V
GND	GND	GND

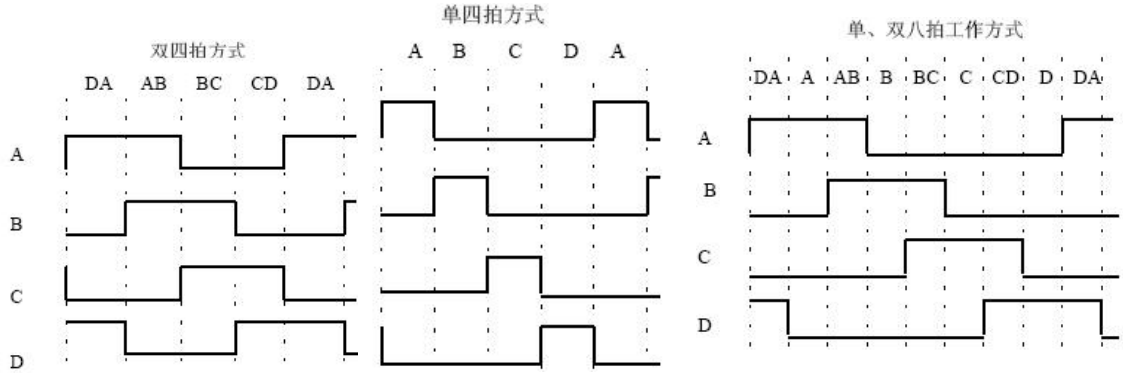
注意事项：

（1）实验箱上各模块是独立供电，实验时需要用到的模块都要给它提供电源，即+5V 接口都要接到电源模块的+ 5V 电源接口，GND 接口可以不用接（默认实验箱上的 GND 网络都接在一起了），千万不要把+5V 接口接到 GND 接口上，短路烧坏保险管。

实验说明：

步进电机驱动原理是通过对每组线圈中的电流的顺序切换来使电机作步进式旋转。切

换是通过单片机输出脉冲信号来实现的。所以调节脉冲信号的频率就可以改变步进电机的转速，改变各相脉冲的先后顺序，就可以改变电机的转向。步进电机的转速应由慢到快逐步加速。电机驱动方式可以采用双四拍（$AB \to BC \to CD \to DA \to AB$）方式，也可以采用单四拍（$A \to B \to C \to D \to A$）方式。为了旋转平稳，还可以采用单、双八拍方式（$A \to AB \to B \to BC \to C \to CD \to D \to DA \to A$）。各种工作方式的时序图如下：（高电平有效）：



上图中示意的脉冲信号是高电平有效，但实际控制时公共端是接在 VCC 上，所以实际控制脉冲是低电平有效。

三、实验内容

1、按照已给的直流电机实验原理图，在实验箱上合理连线，搭建实际的硬件电路，并烧写已给的直流电机参考程序代码，观察现象，验证其功能。

2、按照已给的步进电机实验原理图，在实验箱上合理连线，搭建实际的硬件电路，并烧写已给的步进电机参考程序代码，观察现象，验证其功能。

四、参考程序代码

1、直流电机实验

```
/************************************************
*Descriptoon: 采用单片机的 2 个 IO 口来控制直流电机，其中一个 IO 口使用脉宽调制（PWM）
             对电机转速进行控制,另一个 IO 口控制电机的转动方向。当 INC 按下，电机加
             速，当 DNC 按下电机减速。
*接线说明：    P34-K1，P35-K2，P36-DIR，P37-PWM
************************************************/
#include "reg51.h"
#include "intrins.h"
#define uchar unsigned char       //数据类型宏定义
#define uint unsigned int
/*********************引脚定义*******************/
sbit Inc = P3^4;
sbit Dec = P3^5;
sbit Dir = P3^6;
```

```
sbit PWM = P3^7;
void delay(uint);

int   pwm = 900;
/*********************主函数*********************/
void main(void)
{
    Dir=1;
    while(1)
    {
        if(!Inc)    //K1 加速
            pwm = pwm > 0 ? pwm - 1 : 0;
        if(!Dec)    //K2 减速
            pwm = pwm < 1000 ? pwm + 1 : 1000;

        PWM=1;
        delay(pwm);
        PWM=0;
        delay(1000-pwm);
    }
}
/*********************延时函数*********************/
void delay(uint j)
{
    for(;j>0;j--)
    {
        _nop_();
    }
}
```

2、步进电机实验

```
/************************************************
 *Descriptoon: 用单片机四路 IO 口实现环形脉冲的分配，控制步进电机按固定方向连续转
              动。同时，要求按下 POS 键时，控制步进电机正转；按下 NEG 键时，控制
              步进电机反转；放开按键时，电机停止转动。
*接线说明：    P00-K1，P01-K2,P20~P23-B1~B4
************************************************/
#include "reg51.h"
#define uchar unsigned char          //数据类型宏定义
#define uint unsigned int
/*********************引脚定义*********************/
#define out   P2
sbit pos=P0^0;
```

```
sbit neg=P0^1;
void delayms(uint);
uchar code turn[]={0x02,0x06,0x04,0x0c,0x08,0x09,0x01,0x03};

/*********************主函数**********************/
void main(void)
{
    uchar i;
    out=0x03;
    while(1)
    {
        if(!pos)            //K1 正转
        {
            i = i < 8 ? i+1 : 0;
            out=turn[i];
            delayms(50);
        }
        else if(!neg)    //K2 反转
        {
            i = i > 0 ? i-1 : 7;
            out=turn[i];
            delayms(50);
        }
    }
}
/*********************延时函数*********************/
void delayms(uint j)
{
    uchar i;
    for(;j>0;j--)
    {
        i=250;
        while(--i);
        i=249;
        while(--i);
    }
}
```