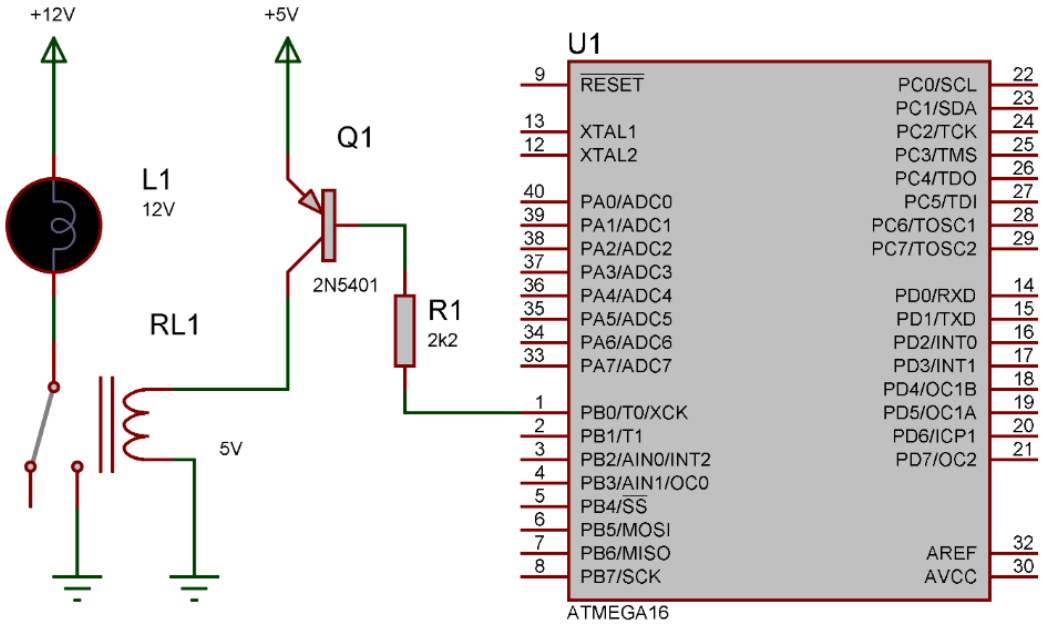
实验 2：继电器控制

1. 试验描述：

继电器部分由 1 只 PNP 型三极管 2N5401 驱动 5V 继电器。在继电器没有动作时，继电器触点断开；在继电器动作时，继电器触点闭合。把继电器触点引出，即可控制外部的设备。本实验使用继电器来控制 12V 的小灯。

2.系统框图：

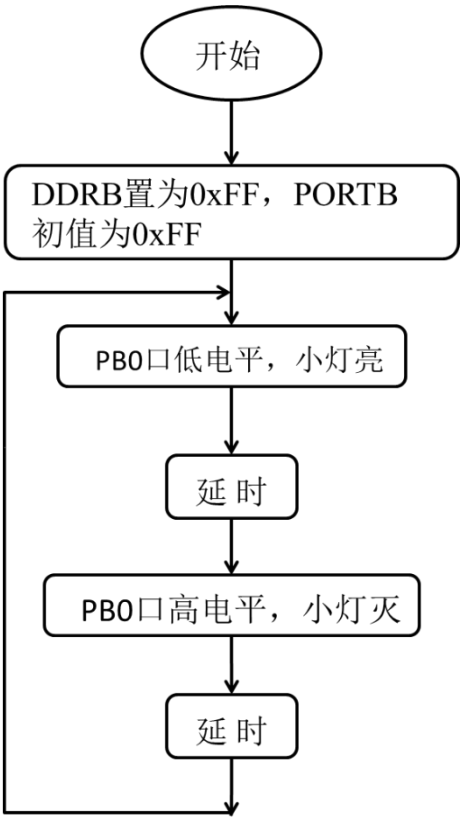
- **硬件电路**



- **元件清单**

单片机 ATmega16	电阻 2k2 欧姆	三极管 2N5401
继电器 RELAY	小灯 LAMP	

➢ 软件流程



3. 程序代码：

➢ ICC 程序

```
#include <iom16v.h>

#define uchar unsigned char
#define uint unsigned int

void delay(uint ms) //毫秒级延时程序
{
  uint i, j;
  for (i = 0; i < ms; i++)
  {
    for (j = 0; j < 1141; j++)
      ;
  }
}
int main(void) //主程序
{
```

```
    DDRB = 0xFF;  //PB 口设置为输出
    PORTB = 0xFF; //PB 设置高电平
    while (1)
    {
      PORTB = 0xFE; //PB0 低电平
      delay(100);   //延时
      PORTB = 0xFF; //PB0 高电平
      delay(100);   //延时
    }
    return 0;
}
```

- **CVAVR 程序**

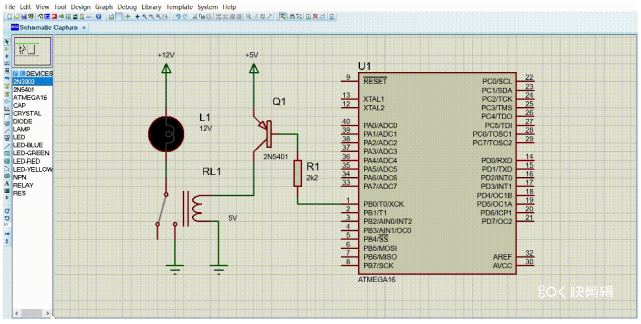
```
#include <mega16.h>

#define uchar unsigned char
#define uint unsigned int

void delay(uint ms) //毫秒级延时程序
{
  uint i, j;
  for (i = 0; i < ms; i++)
  {
    for (j = 0; j < 1141; j++)
      ;
  }
}
void main(void) //主程序
{
  DDRB = 0xFF;  //PB 口设置为输出
  PORTB = 0xFF; //PB 设置高电平
  while (1)
  {
    PORTB = 0xFE; //PB0 低电平
    delay(100);   //延时
    PORTB = 0xFF; //PB0 高电平
    delay(100);   //延时
  }
}
```

4.仿真结果：

继电器的状态随着 PB0 的输出电平交替变化，小灯的亮灭随着继电器的的状态改变。



（双击图片打开动态仿真结果）

5.改进：

因为使用循环空指令的方式得到的延时不准确，所以使用 AVR 的定时计数器 0 获得更加准确的时间间隔。为使时间进一步准确，使用 1M 的外部晶振和修改 AVR 熔丝位。同时，将定时器延时的方法使用函数封装并新建 mydelay.c、mydelay.h 文件存放，方便以后使用。

- main.c 文件：

```
#include <mega16.h>
#include <mydelay.h>

void Lamp(void);

void main(void)
{
  DDRB = 0x01;    //PB0 设置为输出
  PORTB |= 0x01; //PB0 高电平

  DelayInit();

  while (1)
  {
    DelayMs(5000);
    Lamp();
  }
```

```
}

void Lamp(void) //切换灯泡状态
{
  PORTB = (!(PORTB & 0x01)) | (PORTB & 0xfe); //PB0 取
}
```

- **mydelay.h 文件：**

```
#ifndef __MYDELAY_H
#define __MYDELAY_H

#include <mega16.h>

/* 使用定时器实现延时功能 */

#define DELAY_TIMER 0      //默认使用定时器 0,可以修改为 2
#define CLOCK_FREQUENCY 1 //晶振频率，默认 1M

void DelayInit(void); //初始化

void DelayMs(unsigned int ms); //毫秒级延时

void DelayS(unsigned int s); //秒级延时

#endif
```

- **mydelay.c 文件**

```
#include <mydelay.h>

void DelayInit(void) //初始化
{
#if DELAY_TIMER == 0
  TCCR0 = 0x42; //定时计数器 0，CTC 模式，8 分频
  OCR0 = 124;   //1M/8/125 = 1k
#elif DELAY_TIMER == 2
  TCCR2 = 0x42;  //定时计数器 2，CTC 模式，8 分频
  OCR2 = 124;    //1M/8/125 = 1k
#endif
#asm("sei"); //开启全局中断允许
}

unsigned int _ms = 0;
```

```c
unsigned int msCnt = 0;
unsigned char delayMsFlag = 0;

void DelayMs(unsigned int ms) //毫秒级延时
{
  delayMsFlag = 0;
  msCnt = 0;
  _ms = ms;
#if DELAY_TIMER == 0
  TCNT0 = 0x00;  //从 0 开始计数
  TIMSK |= 0x02; //允许比较中断
  while (!delayMsFlag)
    ;
  TIMSK &= 0xfd; //失能比较中断
#elif DELAY_TIMER == 2
  TCNT2 = 0x00;  //从 0 开始计数
  TIMSK |= 0x80; //允许比较中断
  while (!delayMsFlag)
    ;
  TIMSK &= 0x7f; //失能比较中断
#endif
}

void DelayS(unsigned int s) //秒级延时
{
  while (s--)
  {
    DelayMs(1000); //延时 1000 毫秒
  }
}

unsigned int cnt = 0;

#if DELAY_TIMER == 0
interrupt[TIM0_COMP] void Timer0CompInt(void) //定时器 0 中断函数
{
  if (cnt >= CLOCK_FREQUENCY - 1)
  {
    msCnt++;
    cnt = 0;
  }
  else
  {
    cnt++;
```

```
  }
  if (msCnt >= _ms) //间隔_ms 毫秒
  {
    delayMsFlag = 1;
  }
  TCNT0 = 0x00; //计数器清零
}
#elif DELAY_TIMER == 2
interrupt[TIM2_COMP] void Timer2CompInt(void) //定时器 2 中断函数
{
  if (cnt >= CLOCK_FREQUENCY - 1)
  {
    msCnt++;
    cnt = 0;
  }
  else
  {
    cnt++;
  }
  if (msCnt >= _ms) //间隔_ms 毫秒
  {
    delayMsFlag = 1;
  }
  TCNT2 = 0x00; //计数器清零
}
#endif
```