实验 4 熟悉和掌握并口的使用以及 ISP 工具

姓名：朱勇椿　学号：201411213004

2016 年 11 月 15 日

1　实验题目

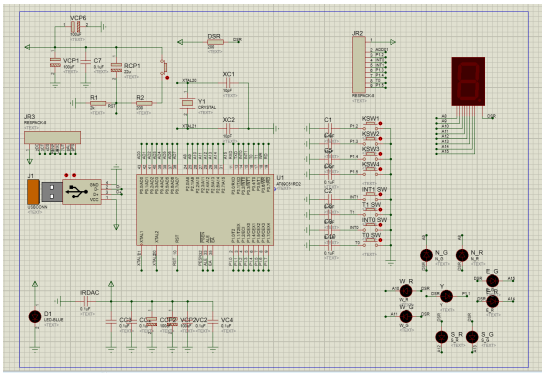
1. 学习并口操作。
2. 学习硬、软件仿真操作。
3. 学习 Proteus 与 Keil 联合仿真操作。
4. 熟悉 ISP 工具软件的使用。

2　实验内容

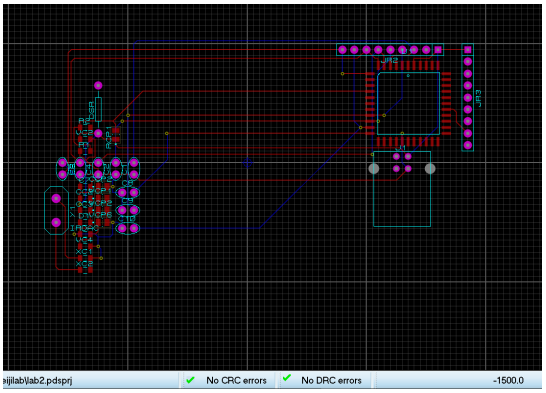
1. 设计并口控制电路图，参考原理图（主要添加元件：红、绿、黄 LED 灯、8 段 LED、开关、电阻、电容等）：
2. 练习硬、软件仿真操作，以及安装 vdmagdi 程序练习 Proteus 与 Keil 联合仿真操作。
3. 安装 Flash Magic 工具软件，练习使用 Flash Magic 软件写入应用程序。
4. 设计控制程序（优先选择汇编语言），要求如下：
   (a) 固定 4 个循环状态：东绿灯，其他红灯 2 秒，西绿灯，其他红灯 2 秒，南绿灯，其他红灯 2 秒，北绿灯，其他红灯 2 秒。仿真或写入验证是否正确。
   (b) 按 1 号键在 8 段数码管上显示 1（3 秒），然后实现东西绿灯，南北红灯，按 2 号键在 8 段数码管上显示 2（3 秒），然后实现南北绿灯，东西红灯，按 3 号键在 8 段数码管上显示 3（3 秒），然后实现全绿灯亮，按 4 号键在 8 段数码管上显示 4（5 秒），然后实现全红灯和黄灯亮。仿真或写入验证是否正确。

3 实验过程

3.1 设计电路图



(a) 原理图



(b) PCD 图

原理图是在第二次实验的基础上改进的，添加了 LED 灯、排阻、数码管等组件。关于 PCD 板的制作，在软件中有自动布局和布线，并且布局出来的图也没有任何错误，那还有必要人为地布局吗

3.2 固定 4 个循环状态，循环显示绿灯

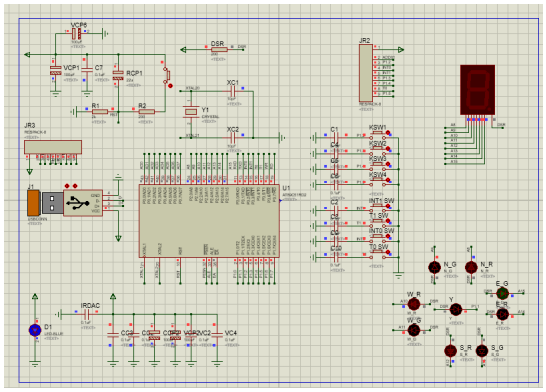
```
E_G BIT P2.7
E_R BIT P2.6
S_G BIT P2.5
S_R BIT P2.4
W_G BIT P2.3
W_R BIT P2.2
N_G BIT P2.1
N_R BIT P2.0
A_LAMP  EQU P2
LEN_GR  EQU 10H
    ORG 0000H
    LJMP    START
    ORG 0300H
START:  LCALL   INIT
MLOOP:
    CLR     E_G
    SETB    E_R
    SETB    W_G
    CLR     N_R
    MOV R5,#LEN_GR
    LCALL   DELAY
    SETB    E_G
    CLR     E_R
    SETB    W_R
    CLR     W_G
```

(c) 汇编代码

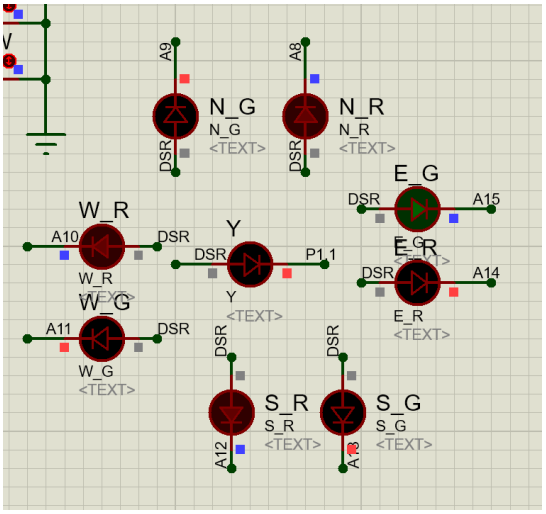
```
    LCALL   DELAY
    SETB    W_G
    CLR     W_R
    SETB    S_R
    CLR     S_G
    MOV R5,#LEN_GR
    LCALL   DELAY
    SETB    S_G
    CLR     S_R
    SETB    N_R
    CLR     N_G
    MOV R5,#LEN_GR
    LCALL   DELAY
    SJMP    MLOOP
INIT:   MOV A_LAMP,#0FFH
    CLR W_R
    CLR N_R
    CLR S_R
    RET
DELAY:  MOV R7,#00H
    MOV R6,#00H
LOOP:   DJNZ    R7,LOOP
    DJNZ    R6,LOOP
    DJNZ    R5,LOOP
    RET
    END
```

(d) 汇编代码

程序从书上的交通灯控制应用设计中的代码改编，主要改动就是把主循环中的代码换成四个状态，然后这四个状态不停地循环。

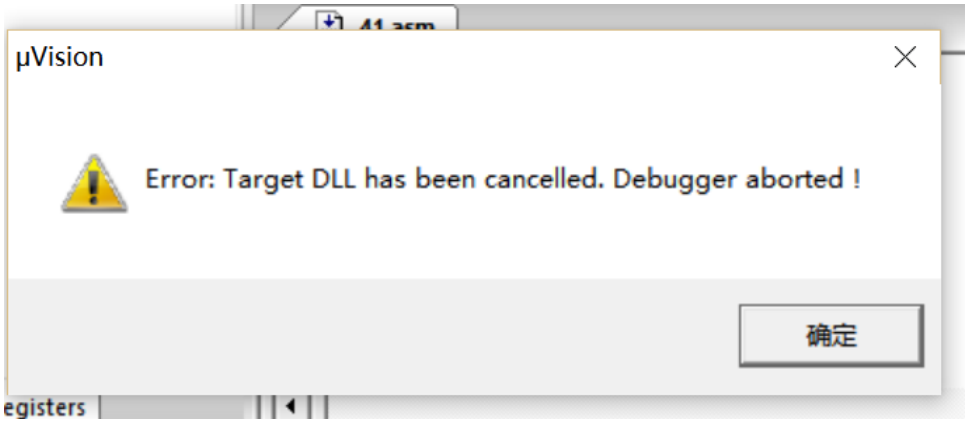


(e) 原理图



(f) 运行代码的结果

用 keil 生成.hex 文件，把.hex 文件导入 51 单片机中，运行结果如右图，每次都有 1 个绿灯和非相同路口的 3 个红灯亮。



(g) 问题

联合仿真出现如图的错误信息，不知道怎么处理

3.3 通过按键控制灯的亮灭和数码管的显示

```
SW1     BIT     P1.2
SW2     BIT     P1.3
SW3     BIT     P1.4
SW4     BIT     P1.5
A_LAMP  EQU     P2
C_LED   EQU     P2
THREE   EQU 15H
FIVE    EQU 25H
EW_G    EQU 01100110B
NS_G    EQU 10011001B
AL_G    EQU 01010101B
AL_R    EQU 10101010B
    ORG     0000H
    LJMP    START
    ORG     0030H
START:  JB  SW1,NEXT1
    MOV     A,#1
    LCALL   DISP
    MOV R5,#THREE
    LCALL   DELAY
    MOV     A_LAMP,#EW_G
    SJMP    START
NEXT1:  JB  SW2,NEXT2
    MOV     A,#2
    LCALL   DISP
```

(h) 原理图

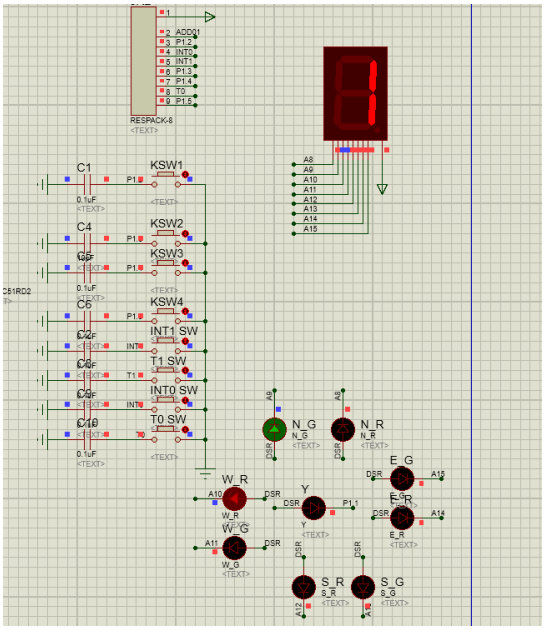
```
    LCALL   DELAY
    MOV     A_LAMP,#NS_G
    SJMP    START
NEXT2:  JB  SW3,NEXT3
    MOV     A,#3
    LCALL   DISP
    MOV R5,#THREE
    LCALL   DELAY
    MOV     A_LAMP,#AL_G
    SJMP    START
NEXT3:  JB  SW4,START
    MOV     A,#4
    LCALL   DISP
    MOV R5,#FIVE
    LCALL   DELAY
    MOV     A_LAMP,#AL_R
    SJMP    START
DELAY:  MOV     R7,#00H
        MOV     R6,#00H
LOOP:   DJNZ    R7,LOOP
        DJNZ    R6,LOOP
        DJNZ    R5,LOOP
        RET
DISP:   MOV     DPTR,#WTAB
        MOVC    A,@A+DPTR
```

(i) 运行代码的结果

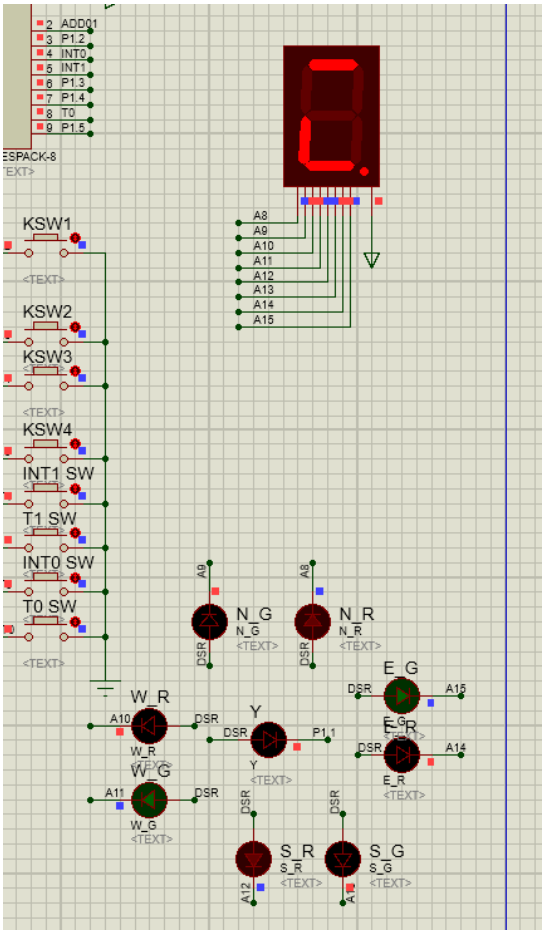
```
DISP:   MOV     DPTR,#WTAB
        MOVC    A,@A+DPTR
        MOV     C_LED,A
        RET
WTAB:   DB  0C0H,0F9H,0A4H,0B0H
        DB  099H,092H,082H,0F8H
        DB  080H,090H,088H,083H
        DB  0C6H,0A1H,086H,08EH
    END
```

(j) 运行代码的结果

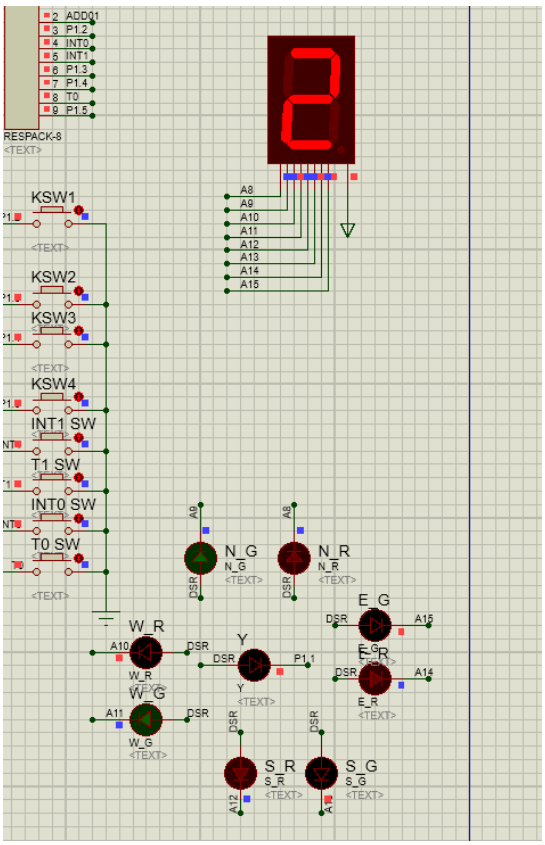
代码结合书上 9.2.4 和 9.2.5 改编，按下按键先显示数码管的数据，延时相应时间，再显示红绿灯。问题：老师给的原理图中数码管和交通灯用的端口是相同的，那么这样他们的结果就会互相干扰，改编数码管同时也会改变交通灯，结果就会如下图，数码管显示混乱



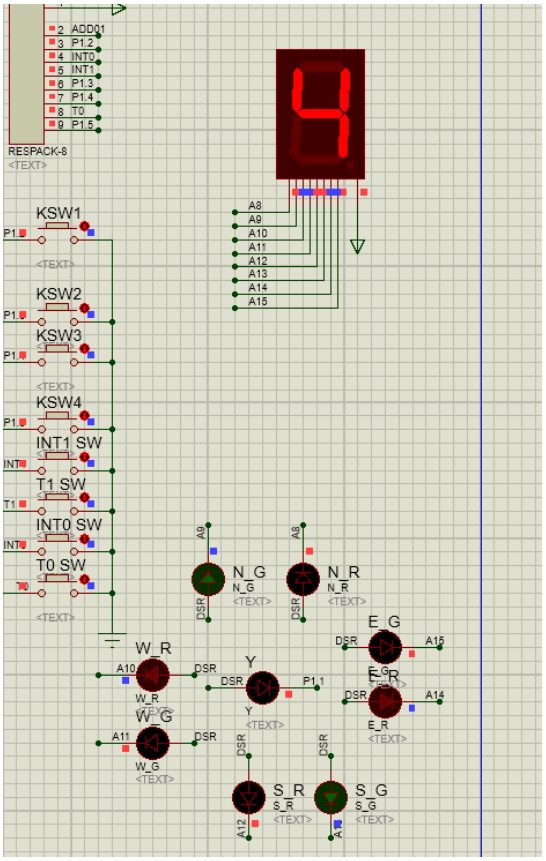
(k) 按下第一个按钮



(l) 延时之后



(m) 按下第一个按钮



(n) 延时之后

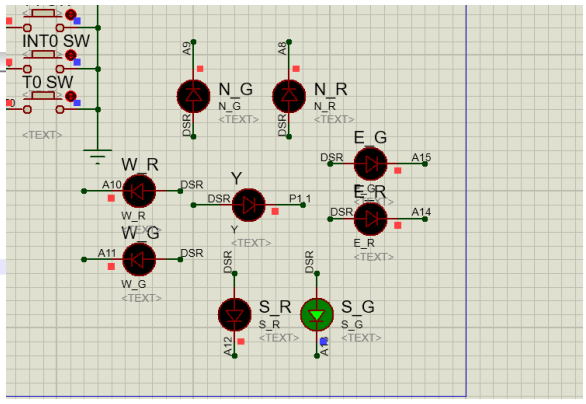
如果把按钮放近一些，在仿真时就容易报错 simulation must be paused whilst measuring，拉开一些就没错。另外前面的交通灯也是，如果原件放的近，就会有警告。

4 选做题

4.1 跑马灯程序

```
paoma.asm
01  LAMP    EQU     P2
02  INIT    EQU     0FEH
03  LEN     EQU     10H
04      ORG     0000H
05      LJMP    START
06      ORG     0030H
07  START:  MOV     LAMP,#INIT
08  MLOOP:  MOV     R5,#LEN
09          LCALL   DELAY
10          MOV     A,LAMP
11          RL      A
12          MOV     LAMP,A
13          SJMP    MLOOP
14  DELAY:  MOV     R7,#00H
15          MOV     R6,#00H
16  LOOP:   DJNZ    R7,LOOP
17          DJNZ    R6,LOOP
18          DJNZ    R5,LOOP
19          RET
20      END
```

(o) 代码



(p) 运行

跑马灯逆时针旋转程序可以看到灯依次逆时针亮，并且只亮一个灯亮度更大。

4.2 跑马灯程序加强版

```
SW1     BIT     P1.2
SW2     BIT     P1.3
SW3     BIT     P1.4
SW4     BIT     P1.5
A_LAMP  EQU     P2
INITG   EQU     01111111B
INITR   EQU     10111111B
LEN     EQU     3h
    ORG     0000H
    LJMP    START
    ORG     0030H
START:  MOV R1,#00H
        MOV R2,#00H
        MOV A_LAMP,#0FEH
NEXT:   JB  SW1,NEXT1
    MOV R1,#00H
    SJMP    NEXT
NEXT1:  JB  SW2,NEXT2
    MOV R1,#01H
    SJMP    NEXT
NEXT2:  JB  SW3,NEXT3
    MOV R2,#01H
```

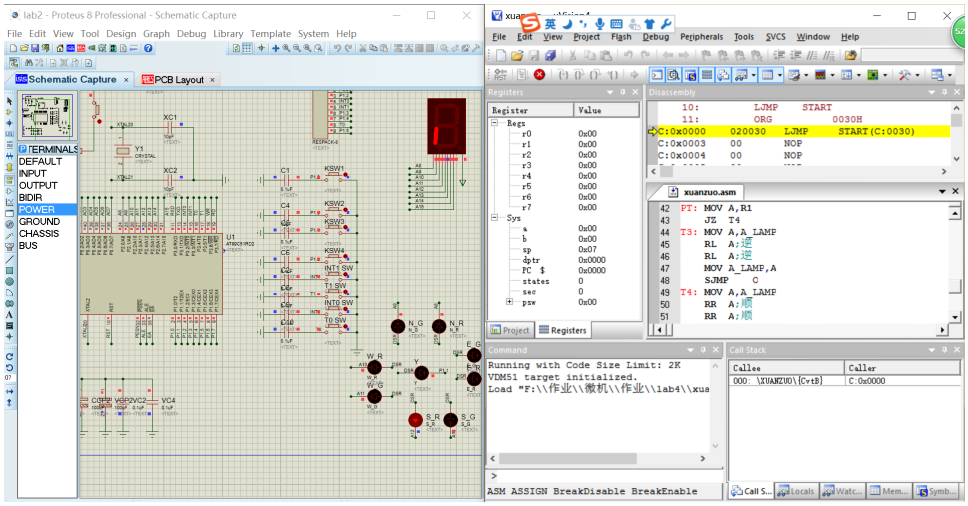
(q) 代码

```
    MOV A_LAMP,#INITG
    SJMP    NEXT
NEXT3:  JB  SW4,CAL
    MOV R2,#01H
    MOV A_LAMP,#INITR
    SJMP    NEXT
CAL:
    MOV A,R2
    JNZ PT
    MOV A,R1
    JZ  T2
T11:    MOV A,A_LAMP
    RL  A;逆
    MOV A_LAMP,A
    SJMP    O
T2: MOV A,A_LAMP
    RR  A;顺
    MOV A_LAMP,A
    SJMP    O
PT: MOV A,R1
    JZ  T4
T3: MOV A,A_LAMP
```

(r) 代码

```
    RL  A;逆
    RL  A;逆
    MOV A_LAMP,A
    SJMP    O
T4: MOV A,A_LAMP
    RR  A;顺
    RR  A;顺
    MOV A_LAMP,A
O:
    MOV R5,#LEN
    LCALL   DELAY
    SJMP    NEXT
DELAY:  MOV     R7,#00H
        MOV     R6,#00H
LOOP:   DJNZ    R7,LOOP
        DJNZ    R6,LOOP
        DJNZ    R5,LOOP
        RET
    END
```

(s) 代码



(t) 联合仿真

程序设计思路：和前面的按键程序类似，按下相应的键进行相应的操作，但是这里的操作是动态的，不是静止的，所以这里把动态的部分单独提取出来，放在判断按钮按下动作的后面，这里我用 R1 寄存器和 R2 寄存器来进行标识，R1=0 时顺时针，R=1 时逆时针，R2=0 时绿灯转，R2=1 时红灯转，在后面用分支跳转进行控制。

几个程序中只有这里联合仿真成功了，前面的都不行，不知道是为什么。

5 问题反思

在前面的红字处已经提出了问题。

6 经验感想

感觉这两个软件有的时候不是特别好用，比如前面提到的元件放的过紧会出现的一些问题，另外就是联合仿真那里不明白出错原因。不过这次调用了 CPU 的端口，了解了程序和硬件之间怎么建立连接。