東北大學

生物医学工程专业

实验报告

实验课程： 单片机原理

班级： 1502 班 **姓名：** 尚麟静

学号： 20155467 **同组人：** 乌日娜

指导教师： 蒋芳芳

实验成绩（教师签字）：

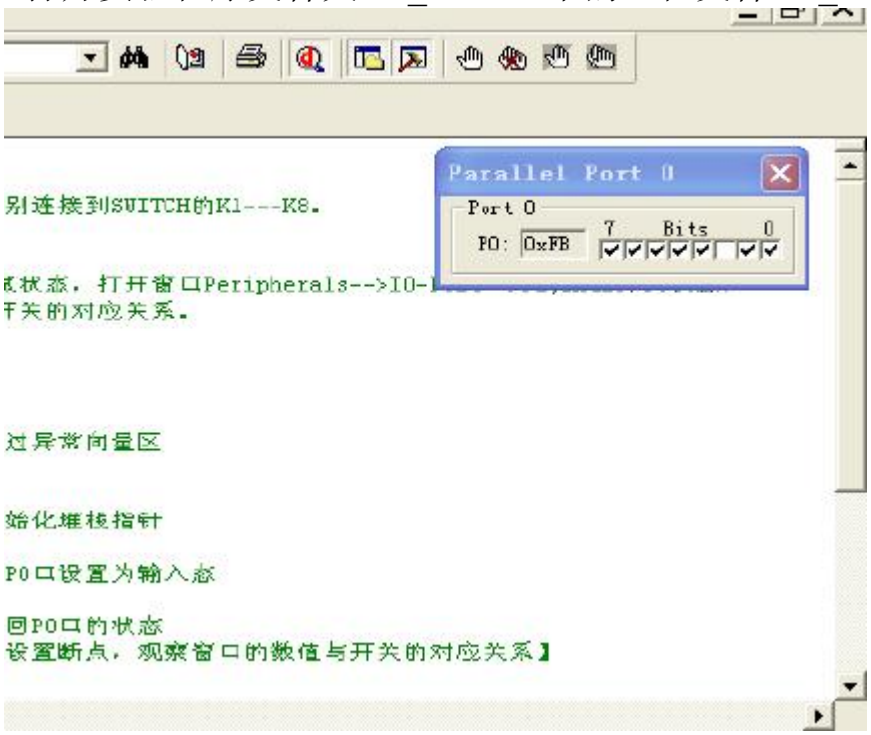
中荷学院教学实验中心制

实验一 单片机的 IO 教程

实验 1 IO 开关量输入实验

1、实验目的：学习单片机读取 IO 引脚状态的的方法。
2、实验内容：编程读取 IO 引脚状态。
3、实验设备：EL-EMCU-I 试验箱、EXP-89S51/52/53 CPU 板。
4、编程：首先要把相关的引脚设置在 IO 的输入状态，然后写一个循环，不停地检测引脚的状态。

打开实验程序文件夹 IO_INPUT 下的工程文件 IO_INPUT.Uv2 编译程序。




```
/*
接线：
1、用导线将 MCU 的 IO1---IO8 分别连接到 SWITCH 的 K1---K8。

过程：
上电，在程序注释处设置断点，进入调试状态，打开窗口 Peripherals-->IO-Port-->P0,改变开关状态，
运行程序到断点处，观察窗口的数值与开关的对应关系。
*/

    ORG     0000H
    LJMP    MAIN
```

```
    ORG     0030H          ;跳过异常向量区
;------------主程序------------
MAIN:
    MOV     SP,#53H        ;初始化堆栈指针
START:
    MOV      P0,#0FFH      ;将 P0 口设置为输入态
LOOP:
    MOV      A,P0          ;读回 P0 口的状态
    LCALL    DELAY10       ;【设置断点，观察窗口的数值与开关的对应关系】
    LJMP     LOOP

;--------延时约 10ms 程序--------
DELAY10:
    MOV     R6,#20
D1:
    MOV     R7,#248
        DJNZ    R7,$
        DJNZ    R6,D1
        RET

    END
```

温度传感器试验

```
/*
```

接线：

用导线将 MCU 的 IO1 连接到 TEMP SENSOR DS18B20 的 DQ。

过程：

上电，编译、下载程序，按注释说明设置断点，运行程序到断点处，观察寄存器 R7 中的数据，用手摸住传感器 DS18B20 芯片，再运行到断点处，比较 R7 的变化。

```
*/
   A_BIT      EQU      20H        ;存放个位数变量
   B_BIT      EQU      21H        ;存放十位数变量
   FLAG       EQU      38H        ;DS18B20 是否存在标志

   DQ         EQU      P0.0       ;DQ 引脚由 P0.0 控制

    ORG     0000H
    AJMP    MAIN
    ORG     0030H          ;跳过异常向量区
;------------主程序------------
MAIN:
    MOV     SP,#53H        ;初始化堆栈指针
LOOP:
```

```
          LCALL   RE_TEMP          ;调用读取温度子程序
     LCALL   TURN          ;数据转化子程序
     MOV     R7,29H
     LJMP     LOOP         ;【设置断点，比较 R7 单元中的变量变化】

;---初始化及读取温度值子程序---
RE_TEMP:
     SETB    DQ        ;拉高单总线
     LCALL   RESET_1820 ;调用复位子程序
     JB   FLAG,ST           ;判断 DS18B20 是否存在
 HOME:
     RET
    ST:                     ;DS18B20 存在
     MOV     A,#0CCH           ;跳过 ROM 匹配
     LCALL   WRITE_1820 ;调用写入数据子程序
     MOV     A,#44H        ;发出温度转换命令
     LCALL   WRITE_1820 ;调用写入数据子程序
     LCALL   RESET_1820 ;调用读温度前先复位
     MOV     A,#0CCH           ;跳过 ROM 匹配
     LCALL   WRITE_1820 ;调用写入数据子程序
     MOV     A,#0BEH       ;发出读温度命令
     LCALL   WRITE_1820 ;调用写入数据子程序
     LCALL   READ_1820  ;调用读取数据子程序
     LJMP    HOME
;---------复位子程序-----------
RESET_1820:
     SETB    DQ        ;拉高单总线
     NOP
     CLR     DQ        ;拉低单总线
;-----主机发出复位低脉冲-------
     MOV     R1,#3          ;延时，模拟时序
   DLY:
     MOV     R0,#53
     DJNZ    R0,$
     DJNZ    R1,DLY
;-------然后拉高数据线---------
     SETB    DQ        ;拉高单总线
     NOP
     NOP
     NOP
;------等待 DS18B20 回应---------
     MOV     R0,#13H       ;延时，模拟时序
    T2:
     JNB DQ,T3          ;等待 DS18B20 回应
```

```
    DJNZ    R0,T2
    LJMP    T4
;-标志位 FLAG=1，表示 DS18B20 存在-
  T3:
    SETB    FLAG        ;DS18B20 存在
    LJMP    T5
;-标志位 FLAG=0，表示 DS18B20 不存在-
  T4:
    CLRFLAG         ;DS18B20 不存在
    LJMP    T7
;-----时序要求延时一段时间-----
  T5:                   ;延时，模拟时序
    MOV     R0,#55
  T6:
    DJNZ    R0,T6
  T7:
    SETB    DQ
    RET
;---------写入子程序-----------
WRITE_1820:
    MOV     R2,#8           ;一共 8 位数据
    CLRC            ;C=0
 WR1:
    CLRDQ           ;总线低位，开始写入
    MOV     R3,#4
    DJNZ    R3,$            ;保持 16us 以上
    RRC     A           ;把字节 DATA 分成 8 个位，环移给 C
    MOV     DQ,C            ;写入一个位
    MOV     R3,#12
    DJNZ    R3,$        ;等待
    SETB    DQ          ;重新释放总线
    NOP
    DJNZ    R2,WR1          ;写入下一个位
    SETB    DQ          ;拉高单总线
    RET
;----------读子程序------------
READ_1820:
    MOV     R4,#2           ;读出两个字节的数据
    MOV     R1,#29H         ;低位存入 29H，高位存入 28H
 RE0:
    MOV     R2,#8           ;数据位一共有 8 位
 RE1:
    CLRC            ;清零进位
    SETB    DQ          ;拉高单总线
```

```
    NOP
    NOP
    CLRDQ       ;读前总线保持为低
    NOP
    NOP
    NOP
    SETB    DQ      ;开始读总线释放
    MOV     R3,#4
  RE2:
    DJNZ    R3,RE2      ;延时 18us
    MOV     C,DQ        ;从总线读到一个位
    MOV     R3,#12
  RE3:
    DJNZ    R3,RE3      ;等待 50us
    RRC     A           ;把读得的位值环移给 A
    DJNZ    R2,RE1      ;读下一个位
    MOV     @R1,A
    DEC     R1      ;存入下一个地址单元
    DJNZ    R4,RE0
    RET
;--------数据转化子程序--------
 TURN:              ;只保留读回的高 8 位，存入 29H 单元中
    MOV     A,29H
    MOV     C,40H           ;28H.0，将 28 中的最低位移入 C(40~43H 对应的是 28H 的位地址，也可以改用 28H.0~28H.3)
    RRC     A           ;将 A 中内容和进位位一起循环右移一位
    MOV     C,41H       ;28H.1
    RRC     A
    MOV     C,42H       ;28H.2
    RRC     A
    MOV     C,43H       ;28H.3
    RRC     A
    MOV     29H,A       ;最后的温度值存入 29H 单元中
    RET

    END
```

实验 2 IO 输出驱动继电器（或光电隔离器）实验

1、实验目的：学习 IO 输出控制方法。

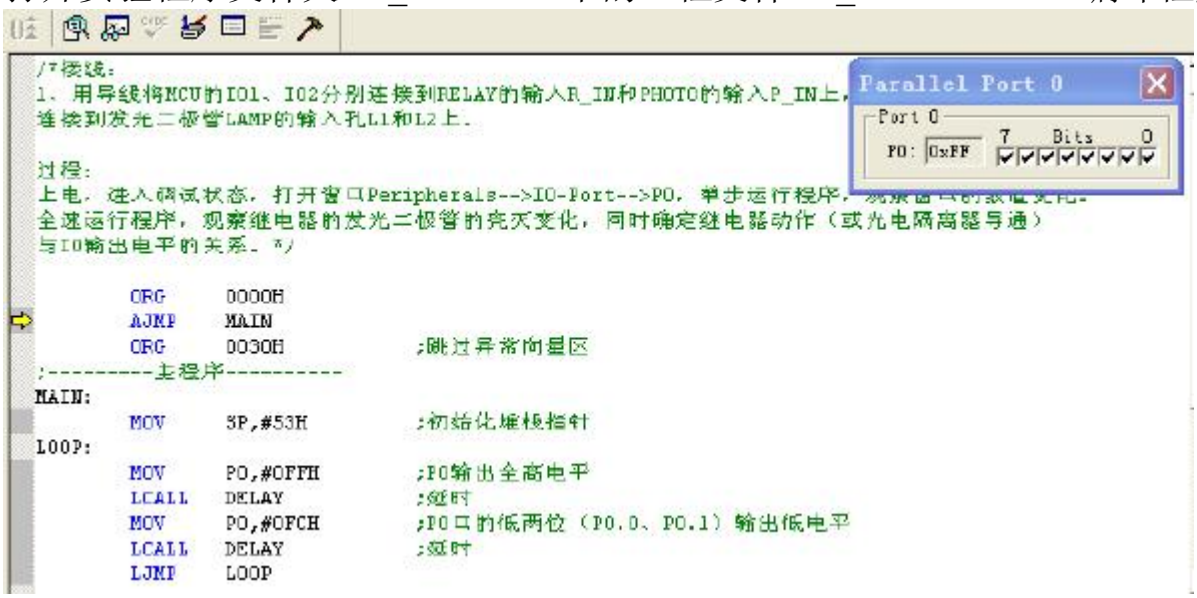
2、实验内容：通过单片机的 IO 引脚驱动继电器（或光电隔离器）动作。

3、实验设备：EL-EMCU-I 试验箱、EXP-89S51/52/53 CPU 板。

4、编程：首先要把相关的引脚设置在 IO 的输出状态，然后写一个循环，依次输出高低电平。

连线：用导线将 MCU 的 IO1、IO2 分别连接到 RELAY 的输入 R_IN 和 PHOTO 的输入 P_IN 上，R_OUT 和 P_OUT 分别连接到发光二极管 LAMP 的输入孔 L1 和 L2 上。

打开实验程序文件夹 IO_OUTPUT 下的工程文件 IO_OUTPUT.Uv2 编译程序



按 5 次灯亮

/*

接线：

用导线将 MCU 的 INT0 连接到单脉冲输出孔 P-。

过程：

按照程序注释说明设置断点，全速运行程序，

每按下一次单脉冲开关 Paulse，程序运行到断点处一次，观察寄存器 R1 的变化。

*/

```
    ORG     0000H
    LJMP    MAIN
    ORG     0003H       ;外部中断 0 中断入口地址
        LJMP    INT_INT0
    ORG     0030H       ;跳过异常向量区
;----------主程序----------
MAIN:
    MOV     SP,#53H     ;初始化堆栈指针
START:
    LCALL   INIT    ;调用初始化子程序
        LJMP    $       ;等待中断
```

```
;-----初始化子程序-----
INIT:
    MOV     R1,#0           ;初始化计数寄存器值
    SETB    IT0         ;设置中断触发为跳变触发方式
        SETB    EX0         ;允许外部中断 0
    SETB    EA          ;开总中断
    RET             ;子程序返回
;----INT0 中断服务子程序-----
INT_INT0:
    INC R1          ;计数值加 1
    CJNE    R1,#100,AA  ;限制计数寄存器 R1 的值
    MOV     R1,#0           ;重新清零
AA:
    NOP                 ;【设置断点，观察寄存器 R1 的变化】
    RETI                ;中断返回

    END
```

灯一直闪亮

```
/*
接线：
1、用导线将 MCU 的 T0 连接到单脉冲输出孔 P-。
2、用导线连接 MCU 的 IO1 到发光二极管 L1 孔。
过程：
按下 5 次单脉冲按键后发光二极管点亮，再按 5 次后发光二极管熄灭，如此重复。
*/

    ORG     0000H
    AJMP    MAIN
    ORG     000BH           ;定时/计数器 0 中断入口地址
    LJMP    INT_T0
    ORG     0030H           ;跳过异常向量区
;--------主程序--------
MAIN:
    MOV     SP,#53H         ;初始化堆栈指针
START:
    LCALL   INIT        ;调用初始化子程序
        LJMP    $           ;等待中断
;-----初始化子程序-----
INIT:
    SETB    P0.0        ;初始化口线状态
    MOV     TMOD,#06H   ;T0 方式 2，计数模式
    SETB     EA         ;开启总中断允许
```

```
    SETB    ET0       ;允许中断
    MOV     TH0,#251  ;设置计数器初值
    MOV     TL0,#251
    SETB    TR0       ;启动计数器
    RET               ;子程序返回
;----定时器 T0 中断服务子程序-----
INT_T0:
    CPL P0.0          ;状态取反，输出方波
    RETI              ;中断返回

    END
```

实验 3 IO 输入/输出---半导体温度传感器 DS18B20 实验

1、实验目的：学习 IO 引脚编程实现交替输入、输出的方法。

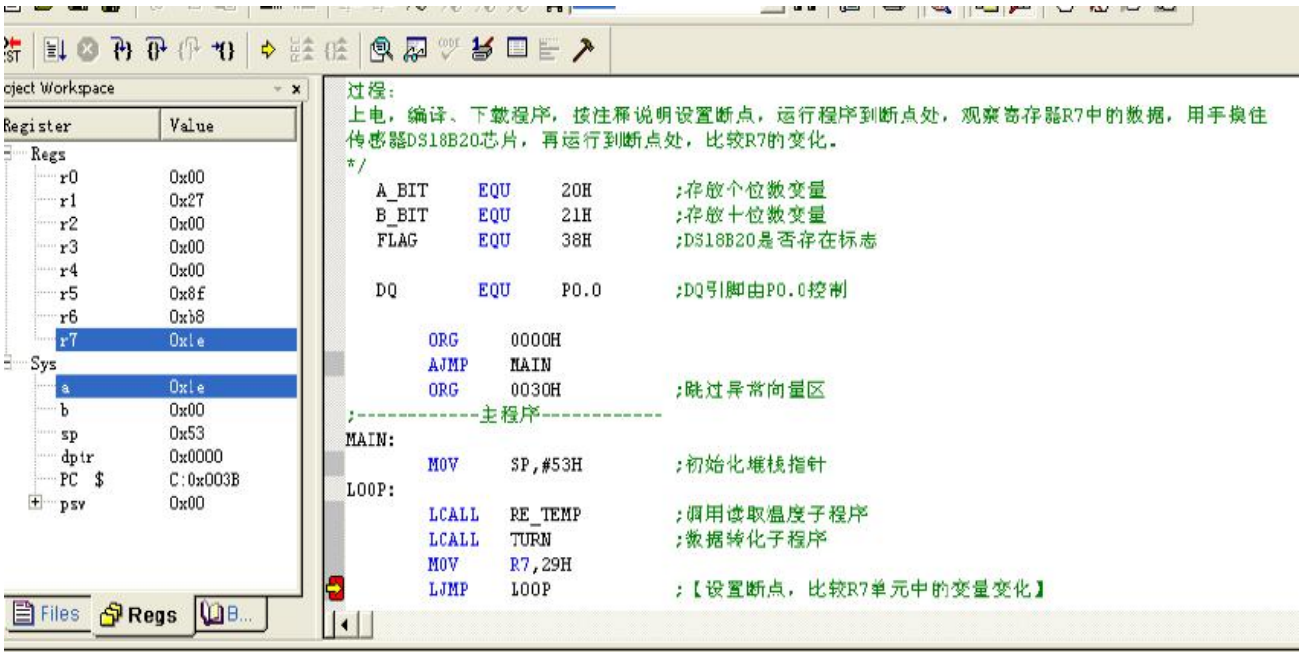
2、实验内容：通过单片机的 IO 引脚与半导体温度传感器实现单线通讯。

3、实验设备：EL-EMCU-I 试验箱、EXP-89S51/52/53 CPU 板。

4、编程：根据 18B20 的资料（见备注），将 IO 引脚设置在输出状态，分别模拟出不同的命令时序，例如复位、读寄存器等；再改变 IO 引脚的为输入状态，接收传感器输出的数据。

连线：用导线将 MCU 的 IO1 连接到 TEMP SENSOR DS18B20 的 DQ。

打开实验程序文件夹 IO_INOUTPUT 下的工程文件 IO_INOUTPUT.Uv2 编译程序，上电，进入调试状态，按照程序注释说明设置断点，全速运行程序到断点处，观察寄存器 R7 中的数据，用手摸住传感器 DS18B20 芯片，再运行到断点处，比较 R7 的变化。



拓展实验 流水灯实验

1、实验目的：学习 IO 引脚编程实现交替输出的方法。

2、实验内容：通过单片机的 IO 引脚与 LED 小灯实现多线交替循环通讯。

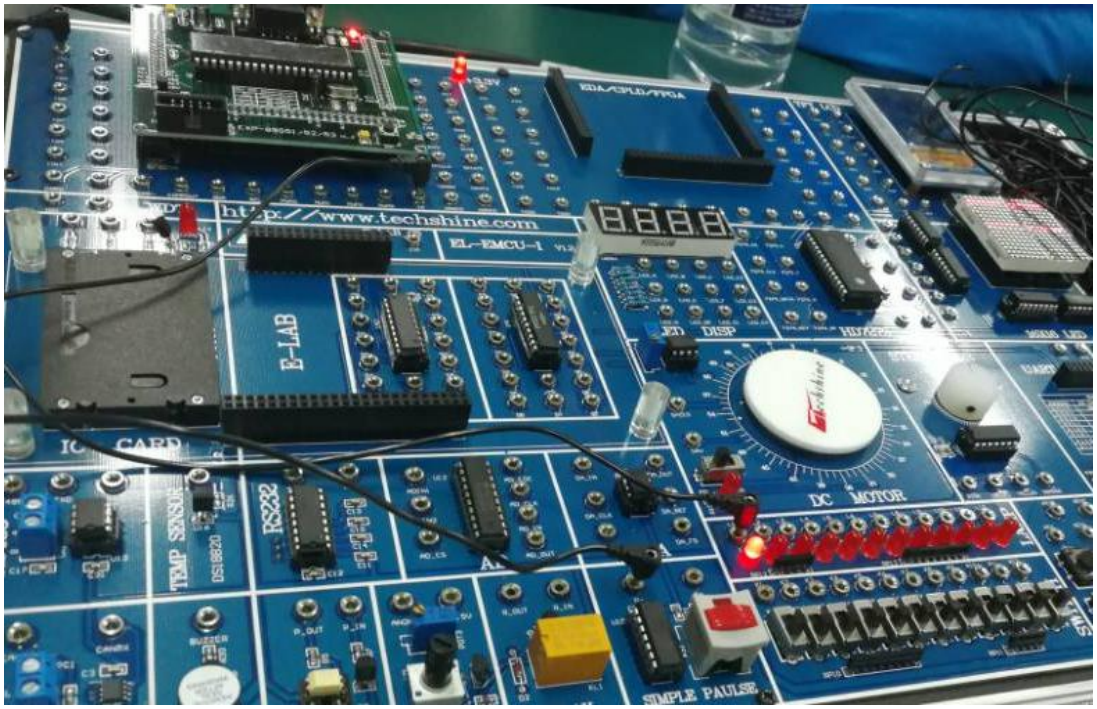
3、实验设备：EL-EMCU-I 试验箱、EXP-89S51/52/53 CPU 板。

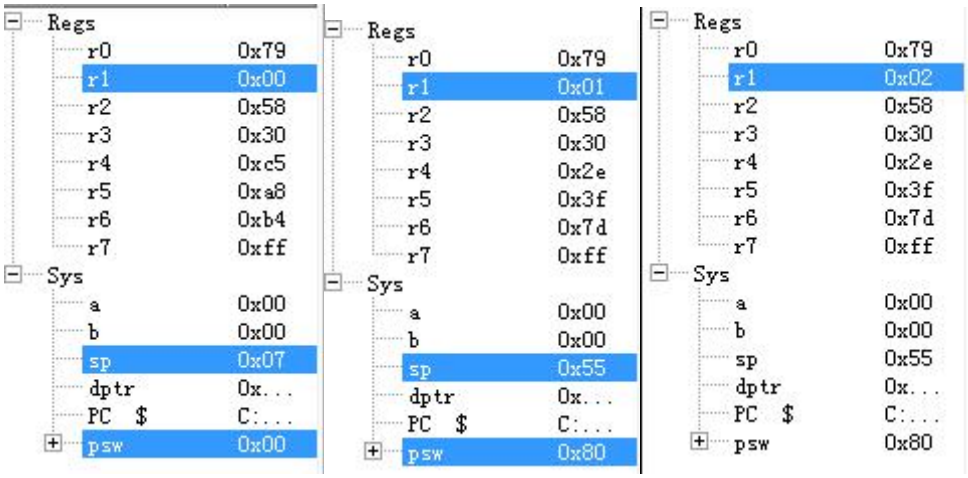
4、编程：独立编程，完成 LED 小灯按顺序交替循环点亮的功能。
连线：用导线将 MCU 的 IO1、IO2 分别连接到 RELAY 的输入 R_IN 和 PHOTO 的输入 P_IN 上，R_OUT 和 P_OUT 分别连接到发光二极管 LAMP 的输入孔 L1 和 L2 上。
代码：

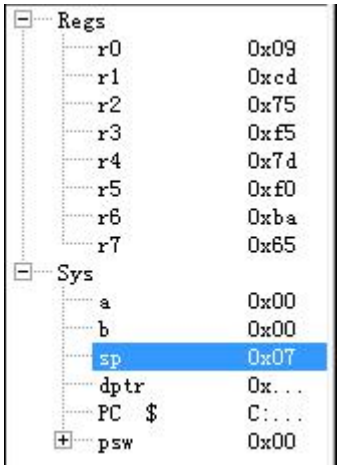
```
        ORG         0000H
        AJMP        MAIN
        ORG         0030H
MAIN:MOV        A,0FEH
CYCLE:
        MOV         P0，A
        LCALL       DELAY1
        RL          A
        SJMP        CYCLE

DELAY1：
        MOV         R7，#200
```

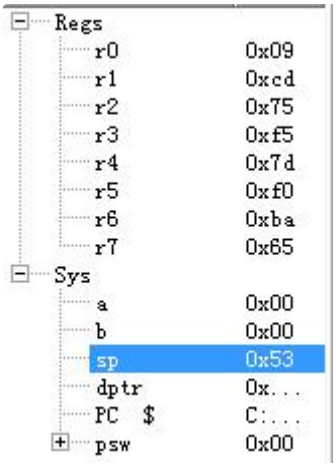
```
DELAY2：
        MOV       R6，#250
        DJNZ      R6，$
        DJNZ      R7，DELAY2

        END
```

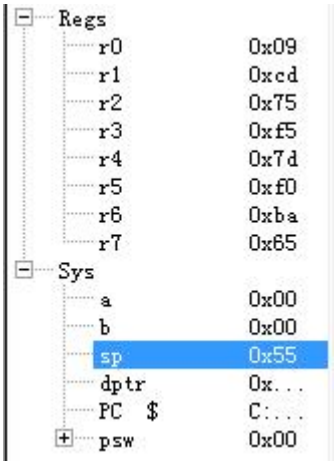
5、实验现象：小灯从左到右依次点亮直至最右端小灯点亮后，再次循环点亮。

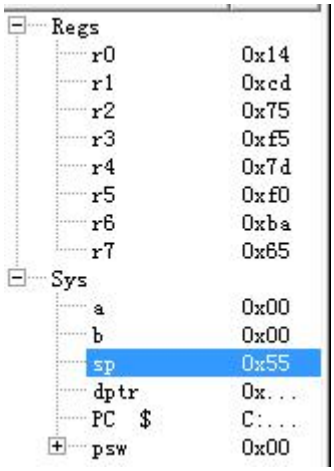












/*

接线：

用导线将 MCU 的 IO1 连接到发光二极管 L1 的输入 L1 上。

过程：

上电，全速运行程序，观察发光二极管的亮灭变化。

*/

```
        ORG     0000H
        AJMP    MAIN
        ORG     000BH           ;定时器 T0 中断入口地址
        LJMP    INT_T0
        ORG     0030H           ;跳过异常向量区
;--------主程序--------
MAIN:
        MOV     SP,#53H         ;初始化堆栈指针
START:
        LCALL   INIT        ;调用初始化子程序                          断点 1
            LJMP    $           ;等待中断
;-----初始化子程序-----
```

```
INIT:
    MOV     R0,#20      ;置指数值，实现长时间的定时                    断点 2
    MOV     TMOD,#01H   ;T0 方式 1
    MOV     TH0,#4BH  ;定时 50ms(11.0592MHz)                          断点 3
    MOV   TL0,#0FFH
    SETB   EA       ;总中断允许位开启
    SETB   ET0      ;允许定时器 T0 中断
    SETB   TR0      ;开 T0 定时
    RET          ;子程序返回
;----定时器 T0 中断服务子程序-----
INT_T0:
    MOV     TH0,#4BH  ;重装初值
    MOV   TL0,#0FFH
    DJNZ   R0,GO_OUT  ;判断计数值状态                                  断点 4  数值同断点 3
    MOV     R0,#20      ;重新初始化计数值
    CPL P0.0      ;口线状态取反，输出方波
GO_OUT:
    RETI            ;中断返回

    END




    ORG    0000H
    AJMP   MAIN
    ORG    000BH        ;定时/计数器 0 中断入口地址
    LJMP   INT_T0
    ORG    0030H        ;跳过异常向量区
;--------主程序--------
MAIN:
    MOV   SP,#53H       ;初始化堆栈指针
START:
    LCALL  INIT1        ;调用初始化子程序
        LJMP    $       ;等待中断          ;等待中断
;-----初始化子程序-----
INIT1:
    SETB   P0.0      ;初始化口线状态
    MOV   TMOD,#06H  ;T0 方式 2，计数模式
    SETB    EA      ;开启总中断允许
    SETB   ET0      ;允许中断
    MOV     TH0,#255   ;设置计数器初值
    MOV     TL0,#255
```

```
    SETB    TR0     ;启动计数器
LOOP1:  JBC     TF0，INIT2
        SJMP    LOOP1
INIT2:
    MOV     R0,#20      ;置指数值，实现长时间的定时                    断点 2
    MOV     TMOD,#01H   ;T0 方式 1
    MOV     TH0,#4BH  ;定时 50ms(11.0592MHz)                          断点 3
    MOV     TL0,#0FFH
    SETB    EA      ;总中断允许位开启
    SETB    ET0     ;允许定时器 T0 中断
    SETB    TR0     ;开 T0 定时
    RET         ;子程序返回

;----定时器 T0 中断服务子程序-----
INT_T0：CPL     P0.0
        RETI
INT_T1:
    MOV     TH0,#4BH  ;重装初值
    MOV     TL0,#0FFH
    DJNZ    R0,GO_OUT  ;判断计数值状态                                 断点 4  数值同断点 3
    MOV     R0,#20      ;重新初始化计数值
    CPL P0.0    ;口线状态取反，输出方波
GO_OUT:
    RETI            ;中断返回

    END
```

实验二 数码管与键盘显示实验

实验 1 HD7279LED 数码管显示实验

1、实验目的：学习 HD7279 的通讯方法。

2、实验内容：利用 IO 向 HD7279 写入控制命令和数据。

3、实验设备：EL-EMCU-I 试验箱、EXP-89S51/52/53 CPU 板。

4、编程：配置 IO，向 HD7279 写入控制命令，控制数码管的显示。

连线：用导线将 MCU 的 IO1----IO3 分别连接到 HD7279 的 7279_CS、7279_CLK、7279_DATA。用导线将 HD7279 的 7279_A、7279_B、7279_C、7279_D、7279_E、7279_F、7279_G、7279_DP 分别连接到 LED DISP 的 LED_A、LED_B、

LED_C、LED_D、LED_E、LED_F、LED_G、LED_DP；用导线将 HD7279 的 7279_C1、7279_C2、7279_C3、7279_C4 分别连接到 LED DISP 的 LED_C1、LED_C2、LED_C3、LED_C4。

打开实验程序文件夹 HD7279DISP 下的工程文件 HD7279DISP.Uv2，编译程序

5、实验现象：0-9 十个数字在数码管上循环显示。

实验 2 HD7279 键盘实验

1、实验目的：学习 HD7279 的通讯方法。

2、实验内容：利用总线向 HD7279 写入控制命令并显示键值。

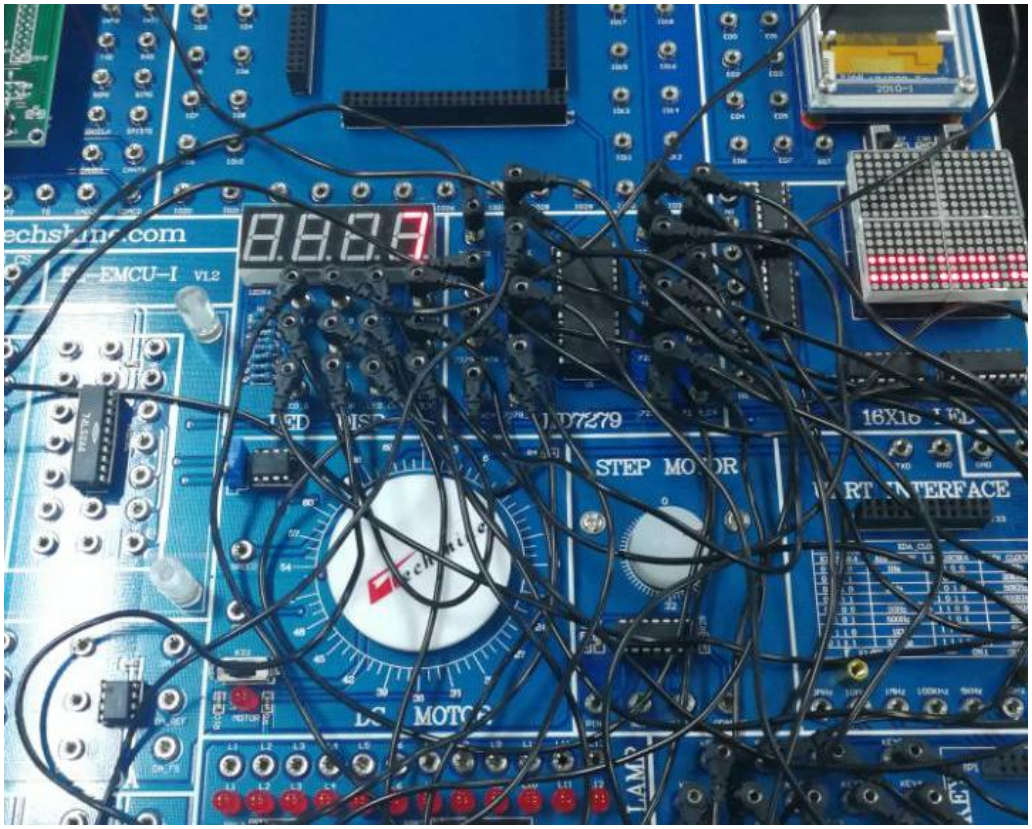
3、实验设备：EL-EMCU-I 试验箱、EXP-89S51/52/53 CPU 板。

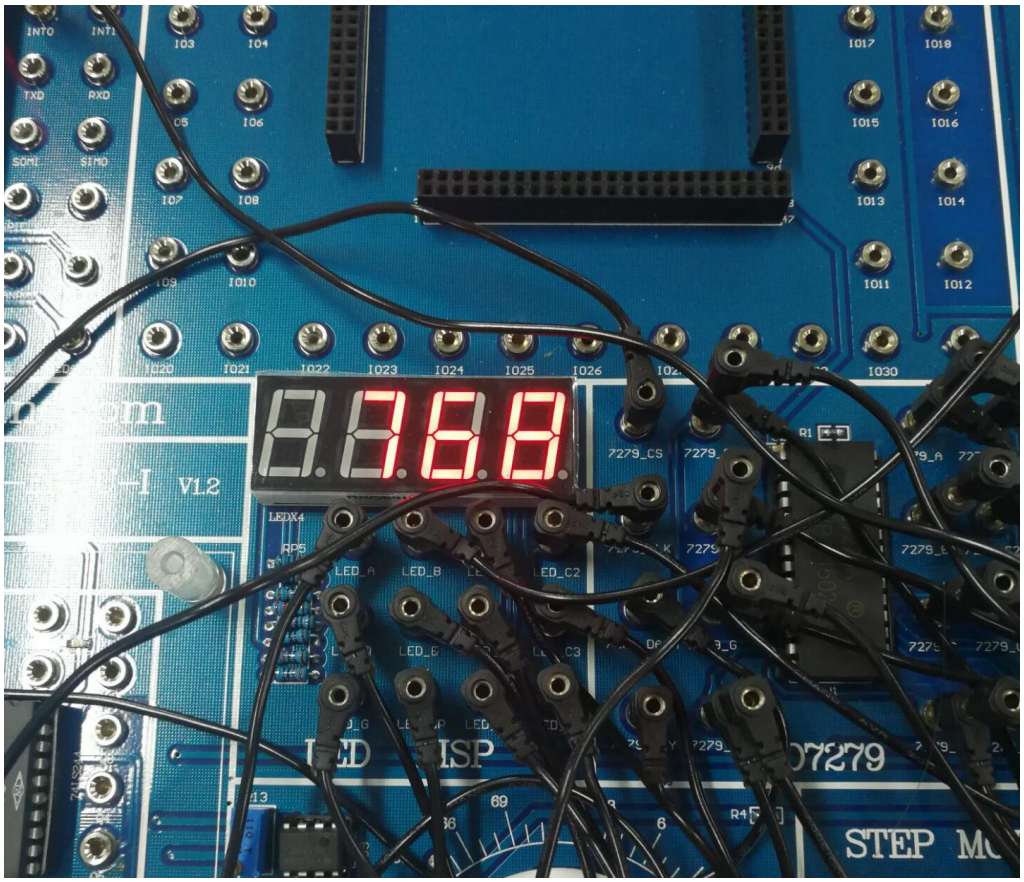
4、编程：配置 2 个 IO 输出作为 SPI 的 CS 和 CLK，配置 1 个 IO 做 I2C 的 LDA。使能外部中断，并将中断引脚连接到 HD7279 的中断输出 INT 上。

连线：用导线将 MCU 的 IO1----IO3、INT0 分别连接到 HD7279 的 7279_CS、7279_CLK、7279_DATA、7279_KEY；用导线将 HD7279 的 7279_A、7279_B、7279_C、7279_D、7279_E、7279_F、7279_G、7279_DP 分别连接到 LED DISP 的 LED_A、LED_B、LED_C、LED_D、LED_E、LED_F、LED_G、LED_DP；用导线将 HD7279 的 7279_C1、7279_C2、7279_C3、7279_C4 分别连接到 LED DISP 的 LED_C1、LED_C2、LED_C3、LED_C4；用导线将 HD7279 的 7279_A、7279_B、7279_C、7279_D、7279_E、7279_F、7279_G、7279_DP 分别连接到 KEY 的 KEY7、KEY6、KEY5、KEY4、KEY3、KEY2、KEY1、KEY8；用导线将 HD7279 的 7279_C1 连接到 KEY 的插孔 KEY。

打开实验程序文件夹 HD7279 下的工程文件 HD7279.Uv2，编译程序。

5、实验现象：按键盘可以在 LED 数码管中显示对应的数字


STEP MOTOR


INT0
INT1
TXD
RXD
SCMI
SIMO
IO3
IO4
IO5
IO6
IO7
IO8
IO9
IO10
IO11
IO12
IO13
IO14
IO15
IO16
IO17
IO18
IO20
IO21
IO22
IO23
IO24
IO25
IO26
IO30
7279_CS
LEDX4
LED_A
LED_B
LED_C2
V1.2
STEP

拓展实验 循环显示学生个人学号

1、实验目的：学习 HD7279 的通讯方法。

2、实验内容：利用 IO 向 HD7279 写入控制命令和数据。

3、实验设备：EL-EMCU-I 试验箱、EXP-89S51/52/53 CPU 板。

4、编程：配置 IO，向 HD7279 写入控制命令，控制数码管的显示。

连线：用导线将 MCU 的 IO1----IO3 分别连接到 HD7279 的 7279_CS、7279_CLK、7279_DATA。用导线将 HD7279 的 7279_A、7279_B、7279_C、7279_D、7279_E、7279_F、7279_G、7279_DP 分别连接到 LED DISP 的 LED_A、LED_B、LED_C、LED_D、LED_E、LED_F、LED_G、LED_DP；用导线将 HD7279 的 7279_C1、7279_C2、7279_C3、7279_C4 分别连接到 LED DISP 的 LED_C1、LED_C2、LED_C3、LED_C4。

打开实验程序文件夹 HD7279DISP 下的工程文件 HD7279DISP.Uv2，编译程序

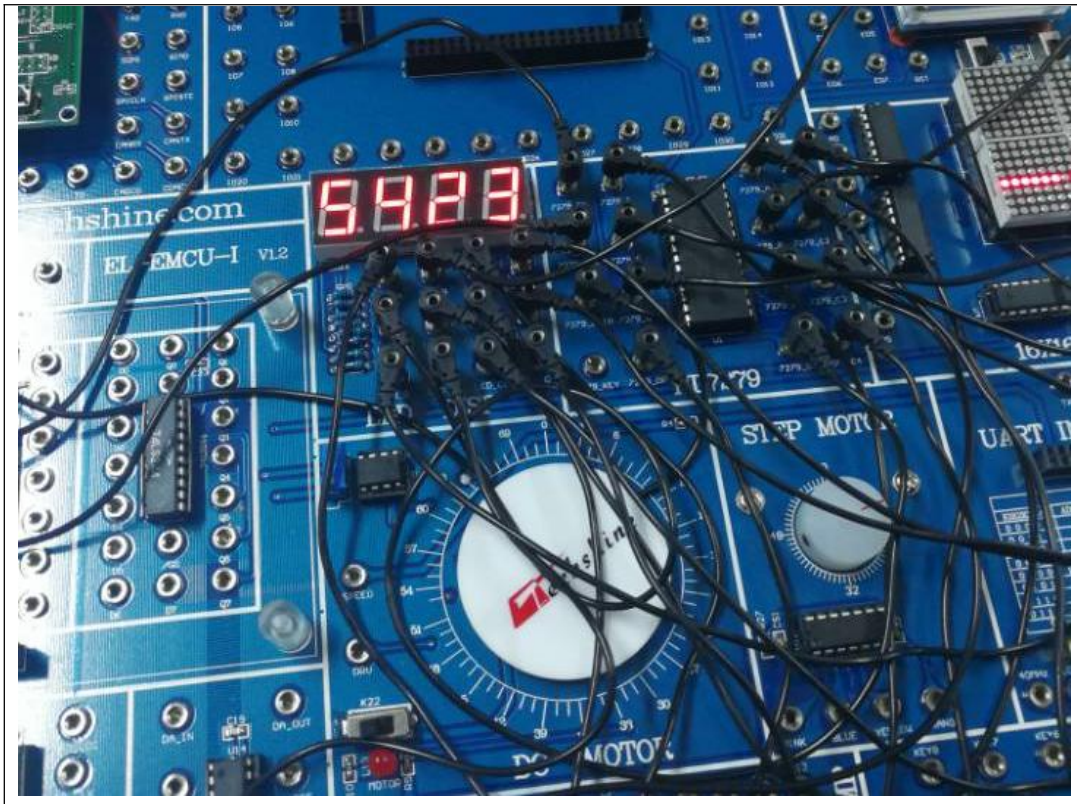
5、实验现象：循环显示学生个人学号

```
;*****************************************
;                  变量定义
;*****************************************
  BIT_COUNT        DATA    03FH   ;要发送的数据位数
  TIMER            DATA    03EH   ;延时子程序中用作临时存储计数值

  DATT             DATA    039H   ;LED 要显示的数据
  CMD              DATA    038H ;要向 7279 发送的命令
  DATA_OUT         DATA    021H   ;向 7279 发出的数据

;*****************************************
;               输入输出引脚定义
;*****************************************
  CS               BIT     P0.0   ;7279 的片选信号
  CLK              BIT     P0.1   ;7279 的时钟信号
  DAT              BIT     P0.2   ;7279 的数据信号

    ORG    0000H
    LJMP   MAIN
    ORG    000BH        ;定时器 T0 中断入口地址
    LJMP   INT_T0
    ORG    0030H        ;跳过异常向量区
```

```
;--------主程序--------
MAIN:
    MOV     SP,#53H         ;初始化堆栈指针
START:
    LCALL   INIT        ;调用初始化子程序

    MOV     DATA_OUT,#0A4H      ;复位 7279
    LCALL   SEND
    LCALL   TEST7279    ;7279 运行测试程序
    SETB    TR0         ;启动定时器

    LJMP    $           ;等待中断
;-----初始化子程序-----
INIT:
    MOV     R1,#20          ;设置定时器的计数次数，达到长定时目的
    MOV     CMD,#0C8H  ;初始化显示
    MOV     DATT,#0
    MOV     R6,#4           ;计数值初始化
    MOV     R7,#10
        MOV     R4,#0
    MOV     TMOD,#01H       ;T0 方式 1
        MOV     TH0,#4CH    ;定时 50 毫秒(6MHz)
        MOV     TL0,#00H
        SETB    ET0         ;开定时器 T0 中断
    SETB    EA          ;开总中断
    RET             ;子程序返回
;------T0 中断服务子程序-----
INT_T0:
    CLRET0      ;禁止 T0 中断
        MOV     TH0,#4CH    ;重装定时 50 毫秒的参数
        MOV     TL0,#00H
        DJNZ    R1,GORET    ;达到定时 50ms*20=1s 的目的
        MOV       A,R4
        MOV       DPTR,#STU_NUM
        MOVC      A,@A+DPTR
    MOV       DATT,A
    LCALL   WRITE7279   ;向 LED 写显示数据和命令
    MOV     A,CMD
    INC A           ;显示超过 4 位后回到右边第一位显示
    MOV     CMD,A
    DJNZ    R6,GO1
    MOV     R6,#4           ;计数值重新初始化
    MOV     CMD,#0C8H
GO1:    MOV     A,DATT
```

```
    INC R4          ;显示数字超过 9 后回到 0 显示
    MOV     DATT,A
    DJNZ    R7,GO2
    MOV     R7,#10          ;计数值重新初始化
    MOV     R4,#0
GO2:    MOV     R1,#20          ;重装定时器的计数次数
GORET:
    SETB    ET0         ;重新开启 T0 中断允许
    RETI                ;中断返回
;----向 LED 写显示数据和命令-----
WRITE7279:
    MOV     A,CMD           ;写命令
    CJNE    A,#255,TT1
    LJMP    END_OUT
TT1:
    ;MOV    A,CMD
    MOV     DATA_OUT,A
    LCALL   SEND

    MOV     A,DATT          ;写显示数据
    CJNE    A,#255,TT2
    LJMP    END_OUT
TT2:
    ;MOV    A,DATT
    ANL     A,#15
    MOV     DATA_OUT,A
    LCALL   SEND
END_OUT:
    SETB        CS              ;置高片选位
    RET
;------显示测试子程序-----
TEST7279:
    MOV     DATA_OUT,#0BFH      ;发送测试命令
    LCALL   SEND
    CALL        LONG_DELAY   ;等待以便观察
    MOV     DATA_OUT,#0A4H      ;发送复位命令
    LCALL   SEND
    RET
;------发送字节子程序-----
SEND:
    MOV         BIT_COUNT,#8 ;要发送的数据长度
    CLR         CS          ;置低片选信号
    LCALL       LONG_DELAY   ;延时等待稳定
SEND_LOOP:
```

```
    MOV     C,DATA_OUT.7 ;准备发送最高位
    MOV     DAT,C        ;置数据线状态
    SETB    CLK          ;置高时钟位
    MOV     A,DATA_OUT
    RL      A            ;输出参数变量左移一位
    MOV     DATA_OUT,A
    LCALL   SHORT_DELAY ;延时至稳定
    CLR     CLK          ;置低时钟位
    LCALL   SHORT_DELAY ;延时至稳定
    DJNZ    BIT_COUNT,SEND_LOOP;循环发送 8 位数据
    CLR     DAT
    RET
;------延时子程序 1 部分-----
LONG_DELAY:
    MOV     TIMER,#25 ;置延时参数
DELAY_LOOP:
    DJNZ    TIMER,DELAY_LOOP;计数延时
    RET

;------延时子程序 2 部分-----
SHORT_DELAY:
    MOV     TIMER,#4   ;置延时参数
DELAY_LOOP_S:
    DJNZ    TIMER,DELAY_LOOP_S;计数延时
    RET
STU_NUM:
        DB      05H
        DB      01H
        DB      00H
        DB      02H
        DB      03H
        DB      02H
        DB      04H
        DB      05H

        END
```




```
        ORG     0000H
        AJMP    MAIN
        ORG     000BH           ;定时器T0中断入口地址
        LJMP    INT_T0
        ORG     0030H           ;跳过异常向量区
;--------主程序--------
MAIN:
        MOV     SP,#53H         ;初始化堆栈指针
START:
        LCALL   INIT            ;调用初始化子程序                    断点1
        LJMP    $               ;等待中断
;-----初始化子程序-----
INIT:
        MOV     R0,#20          ;置指数值，实现长时间的定时          断点2
        MOV     TMOD,#01H       ;T0方式1
        MOV     TH0,#4BH        ;定时50ms(11.0592MHz)                断点3
        MOV     TL0,#0FFH
        SETB    EA              ;总中断允许位开启
        SETB    ET0             ;允许定时器T0中断
        SETB    TR0             ;开T0定时
        RET                     ;子程序返回
;----定时器T0中断服务子程序-----
INT_T0:
        MOV     TH0,#4BH        ;重装初值
        MOV     TL0,#0FFH
        DJNZ    R0,GO_OUT       ;判断计数值状态                      断点4 数值同断点3
        MOV     R0,#20          ;重新初始化计数值
        CPL     P0.0            ;口线状态取反，输出方波
GO_OUT:
        RETI                    ;中断返回

        END
```

/*

接线：

用导线将 MCU 的 INT0 连接到单脉冲输出孔 P-。

过程：

```
按照程序注释说明设置断点，全速运行程序，
每按下一次单脉冲开关 Paulse，程序运行到断点处一次，观察寄存器 R1 的变化。
*/
    ORG     0000H
    LJMP    MAIN
    ORG     0003H       ;外部中断 0 中断入口地址
    LJMP    INT_INT0
    ORG     000BH       ;定时器 T0 中断入口地址
    LJMP    INT_T0
    ORG     0030H       ;跳过异常向量区
;------------主程序------------
MAIN:
    MOV     SP,#53H     ;初始化堆栈指针
START:
    LCALL   INIT        ;调用初始化子程序
    LJMP    $           ;等待中断
;-----初始化子程序-----
INIT:
    MOV     R1,#0       ;初始化计数寄存器值
    SETB    IT0         ;设置中断触发为跳变触发方式
    SETB    EX0         ;允许外部中断 0

    MOV     R0,#20      ;置指数值，实现长时间的定时
    MOV     TMOD,#01H   ;T0 方式 1
    MOV     TH0,#4BH    ;定时 50ms(11.0592MHz)
    MOV     TL0,#0FFH
    SETB    EA          ;总中断允许位开启
    SETB    ET0         ;允许定时器 T0 中断

    RET                 ;子程序返回
;----定时器 T0 中断服务子程序-----
INT_T0:
    MOV     TH0,#4BH    ;重装初值
    MOV     TL0,#0FFH
    DJNZ    R0,GO_OUT   ;判断计数值状态
    MOV     R0,#20      ;重新初始化计数值
    CPL P0.0            ;口线状态取反，输出方波
GO_OUT:
    RETI                ;中断返回
;----INT0 中断服务子程序-----
INT_INT0:
    INC R1              ;计数值加 1
    CPL     TR0         ;开 T0 定时
    CJNE    R1,#100,AA  ;限制计数寄存器 R1 的值
```

```
    MOV     R1,#0       ;重新清零
AA:
    NOP                 ;【设置断点，观察寄存器 R1 的变化】
    RETI                ;中断返回

    END

/*
接线：
用导线将 MCU 的 IO1--IO4 分别连接到 STEP MOTOR 的 ORANGE、YELLOW、PINK、BLUE。

过程：
全速运行程序,步进电机应顺时针转动，说明该模块正常。交换 YELLOW 与 BLUE 两根线，观察电机转动
方向的变化。
*/

    ORG     0000H
    LJMP    MAIN
    ORG     0030H       ;跳过异常向量区
;--------主程序--------
MAIN:
    MOV     SP,#53H     ;初始化堆栈指针
START:
    MOV     P0, #06H
    LCALL   DEL0

    MOV     P0, #07H
    LCALL   DEL0

    MOV     P0, #03H
    LCALL   DEL0

    MOV     P0, #0BH
    LCALL   DEL0

    MOV     P0, #09H
    LCALL   DEL0

    MOV     P0, #0DH
    LCALL   DEL0

    MOV     P0, #0CH
    LCALL   DEL0
```

```
        MOV         P0, #0EH
        LCALL       DEL0
        AJMP        START

DEL0:   MOV     R2, #06H
DEL1:   MOV     R3, #07FH
        DJNZ R3, $
        DJNZ R2, DEL1
        RET
        END
```