学号：53160828　　姓名：周福来　　班级：8 班

实验三　步进电机原理及应用

一、实验目的和要求

初步学习和掌握 MCS-51 的体系结构和汇编语言，了解 Keil 编程环境和程序下载工具的使用方法。

了解步进电机的工作原理，学习用单片机的步进电机控制系统的硬件设计方法，掌握定时器和中断系统的应用，熟悉单片机应用系统的设计与调试方法。了解数码管输出的原理及编程方式。

二、实验环境和开发环境

（1）单片机测控实验系统
（2）步进电机控制实验模块
（3）Keil 开发环境
（4）STC-ISP 程序下载工具

三、实验内容

编制 MCS-51 程序使步进电机按照规定的转速和方向进行旋转，并将已转动的步数显示在数码管上。

步进电机的转速分为两档，当按下 S1 开关时，进行快速旋转，速度为 60 转/分。当松开开关时，进行慢速旋转，速度为 10 转/分。当按下 S2 开关时，按照顺时针旋转；当松开时，按照逆时针旋转。

本程序要求使用定时器中断来实现，不准使用程序延时的方式。

四、实验步骤

4.1 预习

参考附录二、附录三和 expr/资料/原理的辅助材料，学习 MCS-51 汇编语言使用和步进电机原理，阅读数码显示器的电路图，重点理解步进电机的工作方式和数码管显示方式。

4.2 简单程序录入和调试

MCS51 单片机汇编语言的基本格式比较简单，程序中可以使用通用寄存器或者内存单元进行计算。另外，单片机的程序没有退出到操作系统的概念，一般都是死循环程序。

一个简单程序举例如下：

```
ORG 0000H ;复位起始地址
LJMP START ;中间地址保留给中断向量表
ORG 0040H ;程序实际起始地址
START:  ; 实际程序
   MOV 40H, #0H
```

```
NEXT:
   MOV A, 4OH
   INC A
   MOV PO, A ;板上的 PO 口连接到 8 个 LED，可以监视运行状态
   MOV 4OH, A
   MOV R6, #OFFH
L2:MOV R7, #OFFH
L1:DJNZ R7, L1
   DJNZ R2, L2 ;延迟一段时间
   LJMP NEXT
END
```

4.3 定时器中断

使用定时器时，首先应由外部条件得到要定时的时间长度 t，如本实验中，就是根据要求的速度计算出的每一步之间的间隔。然后选择适当的定时器工作方式，去计算想要设定的计数器初值 s，使用如下方程。

（2 定时器最大位数 － s）× 定时周期 =t

定时周期 = 12/CPU 晶振频率

（2 定时器最大位数 － s）× 定时周期 =t

得到的 s 需要分成高 8 位和低 8 位，分别放入计数器 THx 和 TLx 中（x 为 0 或 1）。如果 s 为负数，说明需要的定时时间太长，即使定时器的最大时间也无法满足要求。这种情况下，需要加入软件循环才能实现。我们可以将需要的定时时间分成 n 份，利用定时器达到 t/n 的时间长度，然后在定时器处理程序中，累计某一变量，如果到达 n，说明总的时间 t 已经达到。

要想使用定时器中断，除了上面的定时器初值设定外，还需要将其他相关的特殊功能寄存器也都设置好。如果使用方式 0 和方式 1，不要忘记在计数结束后重新恢复计数器初值。

4.4 程序调试及现象观测

用单步、断点、连续方式调试程序，观察状态指示灯及电机状态，检查运行结果。如果需要，可以将四个输出信号的状态同时输出到 P0 口的某些位上，便于观察。

五、实验原理

关于步进电机转动：本实验使用简单的双四拍工作模式即可，这也是 FAN8200 比较方便的工作方式。只要将 CE1 和 CE2 分别置为高，然后 IN1 和 IN2 按照预定的脉冲输出，即 01->11->10->00->01 这个循环构成一个方向旋转的输出脉冲，将此序列翻转，就是相反方向的输出脉冲。

关于数码管显示：本开发平台有 3 个数码管，使用串行方式连接在一起，具体电路参见实验原理。要想输出一个字形码，就需要从高位到低位依次向移位寄存器输出 8 个比特。移位寄存器的数据线和时钟线分别接到单片机的 P4.5 和 P4.4 管脚，可以使用 MCS-51 里面的位操作指令进行输出。连续输出 3 个

字形，24 个 bit 之后，欲显示的字形将稳定地显示在数码管上，程序可以转而执行其他工作。

七段字形的编码方式需要通过实验获得。这些编码作为程序中的常数，使用 DB 命令存放。在程序中，需要将数值转换为相应的字形编码，可以使用 MOVC 指令来完成。

六、实验实现步骤

准备工作

本程序需要使用定时器定时，并使用中断来同步。

中断程序的典型例子如下：

格式：void 函数名()interrupt 中断号 using 工作组

```
{
        中断服务程序内容
}
```

注意：中断不能返回任何值，所以前面是 void 后面是函数名，名字可以自己起，但不要与 c 语言的关键字相同；中断函数不带任何参数，所以 函数名后面的（）内是 空的，中 断号是指单片机的几个中断源的序号。这个序号是单片机识别不同中断的唯一标志。所以一定要写正确。 后面的 using 工作组 是指这个这个中断使用单片机内存中 4 个工作寄 存器的哪一组，c51 编译后会自动分配工作组，因此最后这句话我们通常省略不写。

c51 中断写法实例

```
void T1-time() interrupt 3
 {
        TH1=(65536-50000)/256;
      TL1=（65536-50000）%256;
 }
```

上面的意思是定时器 1 的中断服务程序，定时器 1 的中断服务序号是 3 ，因此我们要写成 interrupt 3 ，服务程序的内容是给 两个初值寄存器装入新值。。

写中断前的准备:

2.1. TMOD 赋值 确定工作方式。T0 还是 T1 的工作方式。

2.2.计算初值 装入 TH0 TL0 或者 TH1 TL1

2.3.中断方式时 ，对 IE 赋值，开放中断。

2.4. 使 TR0 和 TR1 置位，启动定时器/计数器 定时/计数。

汇编程序

```
ORG 0000H
LJMP    START
ORG 000BH
LJMP    T0_INT
ORG 0040H

START:
```

```
    P4  EQU 0C0H          ;新增的 P4 口
    P4SW    EQU 0BBH          ;PSW 程序状态字
    ;MOV    P4,#0FFH
    CLK EQU P4.4          ;P4.4 模拟串口时钟
    DAT EQU P4.5          ;P4.5 作为模拟串口数据
    MOV P4SW,#30H

    SWH1    EQU P3.6          ;P3.6 外部数据存储器写
    SWH2    EQU P3.7          ;P3.7 外部数据存储器读
    IN1 EQU P3.2          ;P3.2 外部中断 0 输入
    IN2 EQU P1.0
    CE1 EQU P1.3
    CE2 EQU P1.4          ;双四拍工作模式,只要将 CE1 和 CE2 分别置为高，然后
IN1 和 IN2 按照预定的脉冲输出

    ;MOV    SP,#60H
    MOV DPTR,#TABLE           ;断码表

    MOV R0,#0
    MOV R1,#0
    MOV R2,#0                 ;R0-R2 显示数码管
    MOV R3,#50                ;R3 控制转速，数字越大转速越慢
    MOV R5,#0
    MOV R6,#1                 ;从 01 开始，R5,6 为当前状态

    SETB    CE1
    SETB    CE2
    SETB    EA                    ;EA 是整个 CPU 的中断允许标志。当 EA＝1
时，CPU 可以响应中断
    SETB    ET0                   ;ET1 和 ET0 是 T1 和 T0 的中断允许位,两个
16 位定时器/计数器 T0，T1

    MOV TMOD,#01H             ;T0 计数器，方式 1。TMOD 高 4 位和低 4 位分别
控制 T1 和 T0
    MOV TL0,#3EH              ;TH0,TL0 为 T0 的 16 位计数的高 8 位和低 8 位
    MOV TH0,#5DH              ;计数初值
    SETB    TR0                   ;运行控制位 TR0 和 TR1 分别控制两个定时器
是否允许计数。高电平允许。

LL1:    LJMP    LL1                   ;长跳转指令 LJMP

;.............中断服务程序.....................
T0_INT:
    PUSH    ACC                   ;累加器
```

```
    ;PUSH   PSW
    ;PUSH   DPL
    ;PUSH   DPH
    CLR TR0                 ;清零
    MOV TL0,#3EH
    MOV TH0,#5DH            ;计数初值
    SETB    TR0                 ;允许计数
    DJNZ    R3,IEND             ;把源操作数减 1，结果回送到源操作数中；如果结果不为 0 则转移

    JNB SWH1,V1             ;为 0 跳转（SWH1 按下），如果直接寻址单元的值为 0，则转移
    MOV R3,#5               ;慢速
    JMP V2                  ;基址变址寻址 JMP
V1: MOV R3,#1;快速

V2: LCALL   DISPLAY;显示步数
    LCALL   STEP;电机转动

IEND:
    ;POP    DPH
    ;POP    DLH
    ;POP    PSW
    POP ACC
    RETI

;..............取段码 显示数字.............................
DISPLAY:
    MOV A,R0
    MOVC    A,@A+DPTR
    LCALL   SENDNUM

    MOV A,R1
    MOVC    A,@A+DPTR
    LCALL   SENDNUM

    MOV A,R2
    MOVC    A,@A+DPTR
    LCALL   SENDNUM

    RET
;.....................按位送数...........................
SENDNUM:
    MOV R4,#8
```

```
SE1:    CLR CLK                 ;清零
    RLC A                   ;将 A 和进位标志一起向左循环移位 1 位
    MOV DAT,C
    SETB    CLK
    DJNZ    R4,SE1
    RET
STEP:
    JNB SWH2,SHUN;按下，跳转，顺时针
;.....................逆时针.........................
    CJNE    R5,#1,N1;R5 不为 1 转移(R5==0)
    CJNE    R6,#1,N3;R6 不为 1 转移(R6==0)
    CLR IN1;(R5==1,R6==1)
    SETB    IN2;送 01
    MOV R5,#0
    MOV R6,#1
    LJMP    ST0
N1: CJNE    R6,#1,N2;R6 不为 1 转移(R6==0)
    CLR IN1;(R5==0,R6==1)
    CLR IN2;送 00
    MOV R5,#0
    MOV R6,#0
    LJMP    ST0
N2: SETB    IN1;(R5==0,R6==0)
    CLR IN2;送 10
    MOV R5,#1
    MOV R6,#0
    LJMP    ST0
N3: SETB    IN1;(R5==1,R6==0)
    SETB    IN2;送 11
    MOV R5,#1
    MOV R6,#1
    LJMP    ST0
;.....................顺时针.........................
SHUN:
    CJNE    R5,#1,SH1;R5 不为 1 转移(R5==0)
    CJNE    R6,#1,SH3;R6 不为 1 转移(R6==0)
    SETB    IN1;(R5==1,R6==1)
    CLR IN2;送 10
    MOV R5,#1
    MOV R6,#0
    LJMP    ST0
SH1:    CJNE    R6,#1,SH2;R6 不为 1 转移(R6==0)
    SETB    IN1;(R5==0,R6==1)
    SETB    IN2;送 11
```

```
    MOV R5,#1
    MOV R6,#1
    LJMP    ST0
SH2:    CLR IN1;(R5==0,R6==0)
    SETB    IN2;送 01
    MOV R5,#0
    MOV R6,#1
    LJMP    ST0
SH3:    CLR IN1;(R5==1,R6==0)
    CLR IN2;送 00
    MOV R5,#0
    MOV R6,#0
    LJMP    ST0
;.....................增加步数.............................
ST0:    INC R0
    CJNE    R0,#10,ST1          ;比较两个操作数是否相等，如果不相等则转移
    MOV R0,#0
    INC R1
ST1:    CJNE    R1,#10,ST2
    MOV R1,#0
    INC R2
ST2:    CJNE    R2,#10,ST3
    MOV R2,#0
ST3:    RET
;.....................段码表.....................................
TABLE:
    DB  0C0H,0F9H,0A4H,0B0H,99H,92H,82H,0F8H,80H,90H
END
```

七、思考题

1. 如采用单四拍工作模式，每次步进角度是多少，程序要如何修改？

答：每次步进角度是 15 度。

设 A=in1B=in2，(!A)表示 in1=0，(!B)表示 in2=0

输出脉冲修改为：A->B-(!A)->(!B)->A

2. 如采用单双八拍工作模式，每次步进角度是多少，程序要如何修改？

答：每次步进角度是 7.5 度。

输出脉冲修改为： A->AB->B->B(!A)->(!A)->!A!B->!B->(!B)A

3. 步进电机的转速取决于那些因素？有没有上、下限？

答：步进电机的转速主要由时钟的周期控制，通过改变输入脉冲的个数决定转过的角度；转速有上限，通过加大控制电压和降低线圈的时间常数可以提高上限；转速无下限。

4. 如何改变步进电机的转向？

答：通过反向 IN1 和 IN2 的输入即可，如将 01->11->10->00->01 改为：00->10->11->01->00

5. 步进电机有那些规格参数，如何根据需要选择型号？

答：步进电机的主要参数有最大工作电压、最小启动电压、最大允许功耗和工作频率等。

5. MCS51 中有哪些可存取的单元，存取方式如何？它们之间的区别和联系有哪些？

答：（1）工作寄存器组（00H——1FH）

（2）可位寻址 RAM 区（20H——2FH）

（3）通用的 RAM 区（30H——7FH）

6. 说明 MOVC 指令的使用方法。

答：MOVC 用来读取程序存储器；以 16 位的程序计数器 PC 或数据指针 DPTR 作为基寄存器，以 8 位 的累加器 A 作为变址寄存器，基址寄存器和变址寄存器的内容相加作为 16 位的 地址访问程序存储器。如：

```
MOVC A，@A+PC
MOVC A, @A+DPTR
```

7. MCS51 的指令时序是什么样的，哪类指令的执行时间较长？

答：一个机器周期包含 6 个状态（S1-S4），每个状态分为两个节拍 P1 和 P2，通常，一个机器周期会出现两次高电平 S1P2 和 S4P2，每次持续一个状态 S。乘法及除法指令占 4 个周期，三字节指令均为双周期指令 。

8. 在本实验环境下，能否控制显示数码的亮度？如何实现？

答：能，通过修改刷新频率

八．体会与收获

经过这次试验后，我对中断有了更深刻的理解。在代课老师举了“让车通行“的例子后，恍然大悟，当计时器计数到时，则会自动发生中断，而不是人为的去控制中断。在这个过程中，我也遇到过很多挫折，比如 R3 值的计算，计数器初值的设定等等。通过这次试验，我对步进电机的原理和应用有了进一步的理解。

实验四：LED 点阵实验

一、实验目的和要求

了解 LED 点阵显示的基本原理和实现方法。掌握点阵汉字库的编码和从标准字库中提取汉字编码的方法。

二、实验设备

1.单片机测控实验系统
2.LED 点阵显示器实验模块
3.Keil 开发环境
4.STC-ISP 程序下载工具

三、实验内容

1.了解 16*16 点阵电路的原理，编写汇编语言程序。
2.编写一行汉字字符（至少三个字）的显示程序。
3.能够从左到右（或从右到左）循环显示（要求显示过程中字的大小与屏幕尺寸相适应）。

四、实验步骤

1. 掌握点阵式 LED 显示屏的控制方法；
2. 使用 MCS-51 汇编语言，使用 LED 点阵显示器显示出正确的汉字字符及动态效果；
3. 将编译后的程序下载到 51 单片机，观察 LED 显示屏的显示结果。

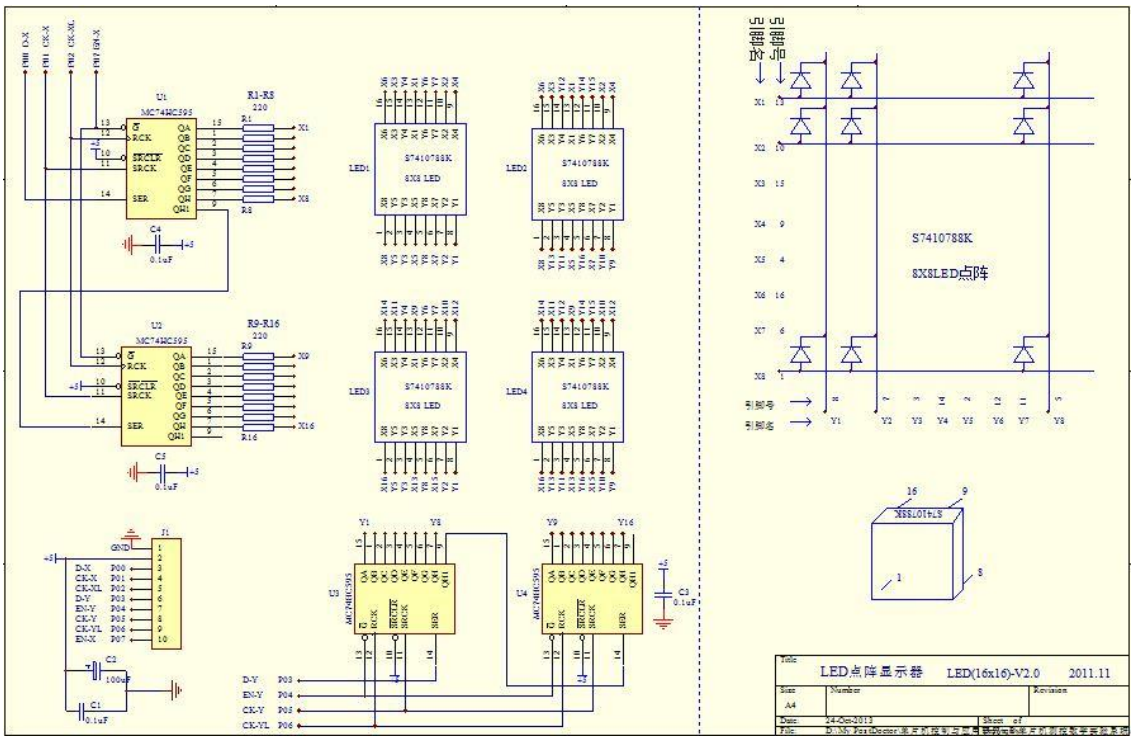
五、实验原理

高亮度 LED 发光管构成点阵，通过编程控制可以显示中英文字符、图形及视频动态图形。所显示字符的点阵数据可以自行编写（即直接点阵画图），也可从标准字库（如 ASC16、HZ16）中提取。后者需要正确掌握字库的编码方法和字符定位的计算。

实验用的 LED 点阵显示屏为 16*16 点阵。行和列分别使用两个移位寄存器作为输出。当移位寄存器输出的第 i 行为 0，第 j 列为 1 时点亮点(i,j)。为了能够显出一个点阵字型，需要进行循环扫描，也就是每一次只点亮一行，然后在列上输出该列对应的 16 个点阵值。输出一行后暂停一段时间，输出下一行。为了达到较好的显示效果，整屏总的扫描时间不高于 40ms。上述过程中行列可以互换。
实验中使用的移位寄存器是 74HC595，它是一个同时具有串行移位和输出锁存驱动功能的器件。74HC595 是具有 8 位移位寄存器和一个存储器，三态输出功能。移位寄存器和存储器是分别的时钟。数据在 SRCK（移位寄存器时钟输入）的上升沿输入到移位寄存器中，在 RCK（存储器时钟输入）的上升沿输入到存储寄存器中去。移位寄存器有一个串行移位输入（行 Dx（P00）、列 Dy(P03)），和一个串行输出（QH）,和一个异步的低电平复位，存储寄存器有一个并行 8 位的，具备三态的总线输出，当使能　（P02 和 P07 为低电平）时，存储寄存器的数据输出到总线。

在控制 74HC595 时，首先将数据放到串行输入的 SI 端，然后在串行时钟 SRCK

上产生一个脉冲，即可输出一个 bit，重复以上步骤 16 次，输出所有列值。然后给存储器时钟 RCK 一个脉冲，将串行数据锁存起来。将使能端输出低电平，驱动到 LED 点阵上。行的输出每次只移位一次，并重新锁存即可。其他信息见给定的参考资料。



六、实验代码

```
ORG 000H
LJMP START
ORG 0040H

START:
      DX EQU P0.0 ;行数据口
      DY EQU P0.3 ;列数据口
      CLKYWX EQU P0.1 ;行移位寄存器时钟
      CLKYWY EQU P0.5 ;列移位寄存器时钟
     CLKCCX EQU P0.2 ;行存储器时钟
      CLKCCY EQU P0.6 ;列存储器时钟

      OUTX EQU P0.7     ;行输出使能
      OUTY EQU P0.4     ;列输出使能

;.....................................................
```

```
SM:       ;无限循环

      MOV R0,#0
      MOV R1,#1

      MOV R4, #1 ;table 高 8 位指针
      MOV R5, #0 ;table 低 8 位指针

;.....逐行扫描.....

      MOV R3,#16    ;扫描 16 次
SM16:   SETB OUTX    ;行输出使能置高电平
      SETB OUTY    ;列输出使能置高电平


;.....送行扫描码.....
      CLR    CLKCCX      ;列存储器时钟置低电平

      MOV DPTR,#TABLE1

      MOV A,R0
      MOVC A,@A+DPTR
      MOV R6,#8
YW1:        CLR CLKYWX
      RLC A
      MOV DX,C
      SETB CLKYWX    ;将高 8 位列选码按位送入到移位寄存器中
      DJNZ R6,YW1

      MOV A,R1
      MOVC A,@A+DPTR
      MOV R6,#8
YW0:     CLR CLKYWX
      RLC A
      MOV DX,C
      SETB CLKYWX      ;将低 8 位列选码按位送入到移位寄存器中
      DJNZ R6,YW0

      SETB CLKCCX      ;将移位寄存器中的数据送到存储器中

      CLR OUTX          ;将行输出使能置低电平
```

```
;.....送列扫描码.....

    CLR   CLKCCY     ;列存储器时钟置低电平

    MOV DPTR,#TABLE

    MOV A,R4
    MOVC A,@A+DPTR
    MOV R6,#8
YW3:      CLR CLKYWY
    RRC A
    MOV DY,C
    SETB CLKYWY   ;将高 8 位列选码按位送入到移位寄存器中
    DJNZ R6,YW3

    MOV A,R5
    MOVC A,@A+DPTR
    MOV R6,#8
YW2:    CLR CLKYWY
    RRC A
    MOV DY,C
    SETB CLKYWY    ;将低 8 位列选码按位送入到移位寄存器中
    DJNZ R6,YW2

    SETB CLKCCY    ;将移位寄存器中的数据送到存储器中

    CLR OUTY        ;将列输出使能置低电平


    LCALL DELAY1



;.....更新扫描码.....

    INC R0
    INC R0
    INC R1
    INC R1

    INC R4        ;TABLE 指针 R4，R5 分别加 2
    INC R4
    INC R5
    INC R5
```

```
    DJNZ R3,SM16    ;进行下一次扫描

    LJMP SM             ;重新扫描
;.....延时函数.....

DELAY1:
    MOV R6,#20
DEL1:   MOV R2,#20
DEL2:   DJNZ R2,DEL2
    DJNZ R6,DEL1
RET


;.....扫描码表.....

TABLE:
DB
0FFH,0FFH,0F7H,0BFH,0CFH,0BFH,0DBH,0BFH,0DBH,0BFH,0DBH,0BDH,5BH,0BEH,9B
H,01H;
DB
0DAH,0BFH,0D9H,0BFH,0DBH,0BFH,0DFH,0BFH,0D7H,0BFH,0CFH,0BFH,0FFH,0BFH,0
FFH,0FFH;

TABLE1:
DB 80H,00H
DB  40H,00H
DB  20H,00H
DB  10H,00H
DB  08H,00H
DB  04H,00H
DB  02H,00H
DB  01H,00H

DB  00H,80H
DB  00H,40H
DB  00H,20H
DB  00H,10H
DB  00H,08H
DB  00H,04H
DB  00H,02H
DB  00H,01H

END
```

七、思考题

1. 如何使用软件调整和控制 LED 点阵的亮度

在显示后，利用代码对代码进行循环反复刷频，可以加强 LED 的亮度。

2. 如何尽量避免显示过程中的闪烁

显示过程中的闪烁原因是两次行扫描之间间隔时间过长，或是两次屏显示之间间隔时间过长。将延时调节到一个适当的值，就可以避免闪烁。

3. 如何将本实验的软硬件推广到多行多列的 LED 显示屏（如 64*1280）

硬件方面可以通过添加新的 led 以及 74hc59 来实现，软件方面将控制行扫描的 16 位数字 0fffeH 改为 64 位的 0ffffffffffffffeH 将读入列值的 2 字节改为 160 字节，及重复输出 1280bit,结束后令行的输出移位一次