单片机控制与应用实验报告

班级：8 班

姓名：杨钊

学号：21160834

实验三 步进电机原理及应用

一、实验目的和要求

1、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。

2、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。

3、了解数码管输出的原理及编程方式。

二、实验设备

单片机测控实验系统
步进电机控制实验模块
Keil 开发环境
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

1 预习
2 简单程序录入和调试
3 程序调试
4 编写程序，完成功能

五、实验原理

1、旋转方向

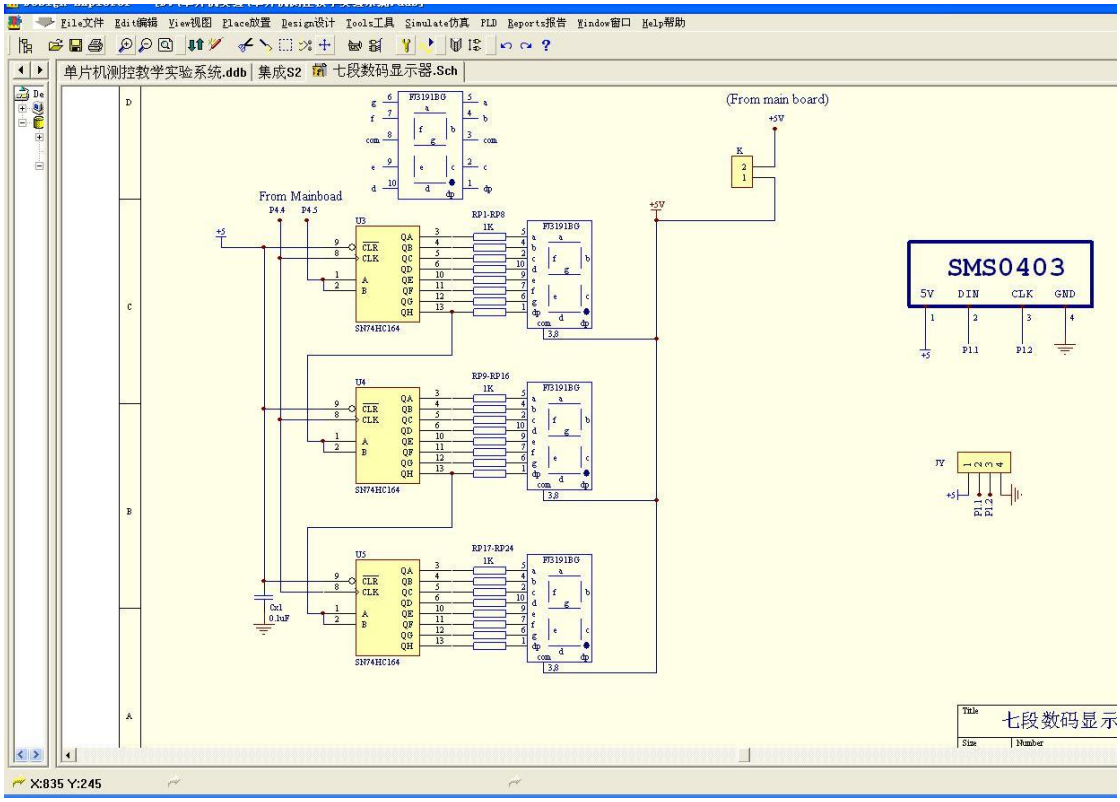
本实验使用简单的双四拍工作模式即可，这也是 FAN8200 比较方便的工作方式。只要将 CE1 和 CE2 分别置为高，然后 IN1 和 IN2 按照预定的脉冲输出，即 01->11->10->00->01 这个循环构成一个方向旋转的输出脉冲，将此序列翻转，就是相反方向的输出脉冲。

2、旋转速度

只需要给计时器设置不同的计数初值即可。

3、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。



74HC164 是高速 CMOS 器件。74HC164 是 8 位边沿触发式移位寄存器，**串行输入数据，然后并行输出**。数据通过两个输入端（A 或 B）之一串行输入；任一输入端可以用作高电平使能端，控制另一输入端的数据输入。两个输入端或者连接在一起，或者把不用的输入端接高电平，一定不要悬空。

时钟（CLK）每次由低变高时，数据右移一位，输入到 Q0，Q0 是两个数据输入端（A 和 B）的逻辑与，它将上升时钟沿之前保持一个建立时间的长度。

何时输入？

（本次实验中，由于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，因此，当 P4.4 从 0 变 1 时，即上升沿时，数据输入至 Q0。）

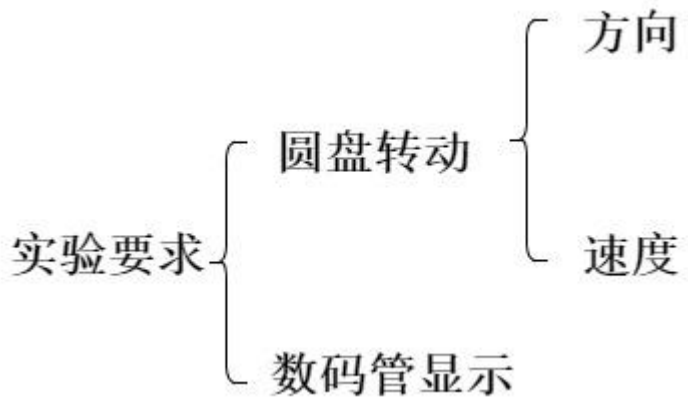
输入什么？

（由于输入信号是 A 和 B 的逻辑与，且 A 和 B 均连接的是上一个 74HC164

的 Q7（第一个连接的是 p4.5），因此，若上一个 74HC164 的 Q7 是 0，则输入 0；若是 1，则输入 1）

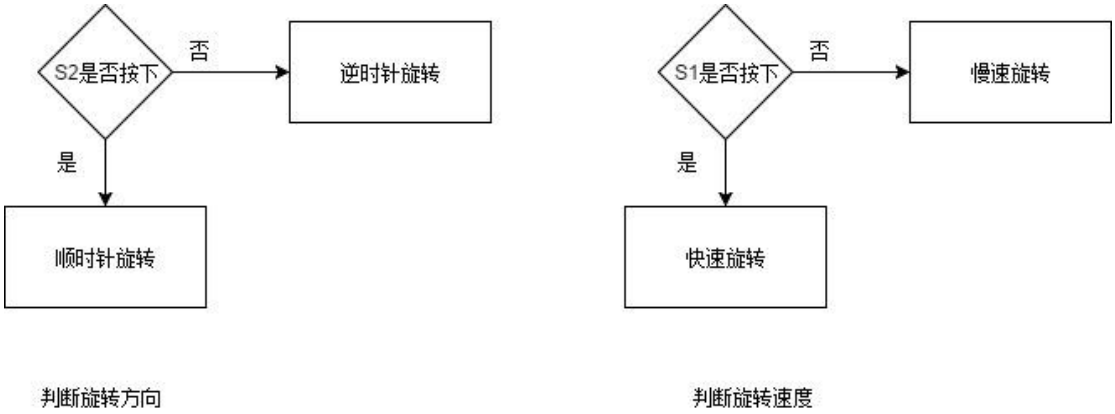
六、代码实现过程

实验要求分析如下：

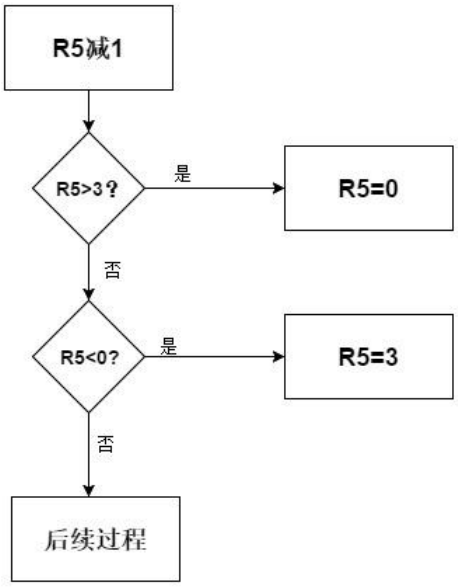


【功能 1、圆盘转动】

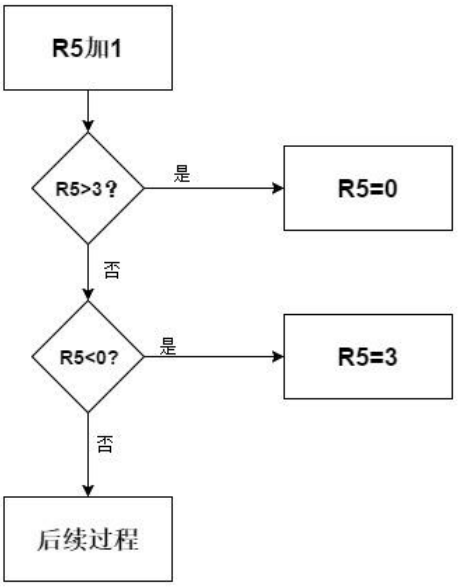
本次实验用计时器 T0 控制圆盘的转动。中断发生时需要扫描两个按键，分别用来控制旋转方向和旋转速度，流程图如下所示：



顺时针旋转和逆时针旋转实现过程流程图如下：

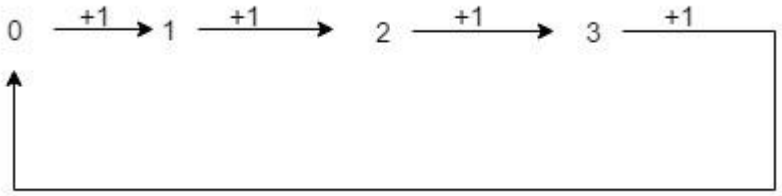


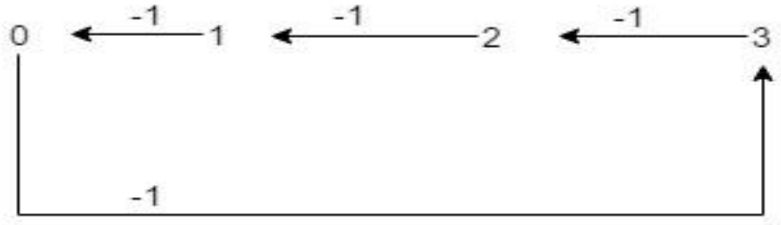
顺时针旋转



逆时针旋转

（注：顺时针和逆时针旋转只有第一步对 R5 的操作不同，如果是从前往后的顺序 00H→ 01H→ 03H→ 02H，则是逆时针，顺时针则正好相反。可见，R5 在这里是起到一个下标的作用，注意下标的合法范围是 0--3，因此如果下标范围不合法要进行相应的调整。具体调整方法为：





至此，我们可以给出顺时针旋转和逆时针旋转的全过程。

1、将 00H,01H,03H,02H（注意顺序）依次存放在 0040H 为起始地址的存储空间。（注意这些数据的低两位对应了控制旋转方向的脉冲序列）

```
ORG 0040H
```

```
DB 00H, 01H, 03H, 02H
```

2、以 R5 为下标，取出存储空间不同的数据。(如 R5 为 0 时取出第一个数据 00H，R5 为 3 时取出第四个数据 02H，可见 R5 增大时是从前往后取数据，对应逆时针)。然后按照 S2 是否按下，对 R5 进行相应的加或减操作。(如果 S2 没有按下，R5 加 1，取数的顺序是从前往后 00H→ 01H→ 03H→ 02H，对应逆时针，否则 R5 减 1，取数顺序是 00H←01H←03H←02H，对应顺时针。可见，旋转方向就是通过对 R5 执行加减操作控制的)

```
MOV A, R5
MOV DPTR, #0040H
MOVC A, @A+DPTR
```

3、将 CE1 和 CE2 分别置为高，然后 IN1 和 IN2 按照预定的脉冲输出，即将第 2 步取出的数据低两位送给 IN1 和 IN2。每次中断服务程序中都不断循环以上三步，即可完成指定方向的旋转。

```
CLR P1.1            ;CE1, CE2 至置高，才能按照指定脉冲显示
CLR P1.4
RRC A               ;右移
MOV P1.0, C         ;第八位给 P1.0 :IN2
RRC A
MOV P3.2, C         ;第七位给 P3.2 :IN1
SETB P1.1            ;CE1, CE2 置高，才能按照指定脉冲显示
SETB P1.4
```

以上是对旋转方向的说明。

旋转速度通过给计时器设置不同的计数初值即可实现，不再赘述。

【功能 2、数码管显示】

1、存储段码表。

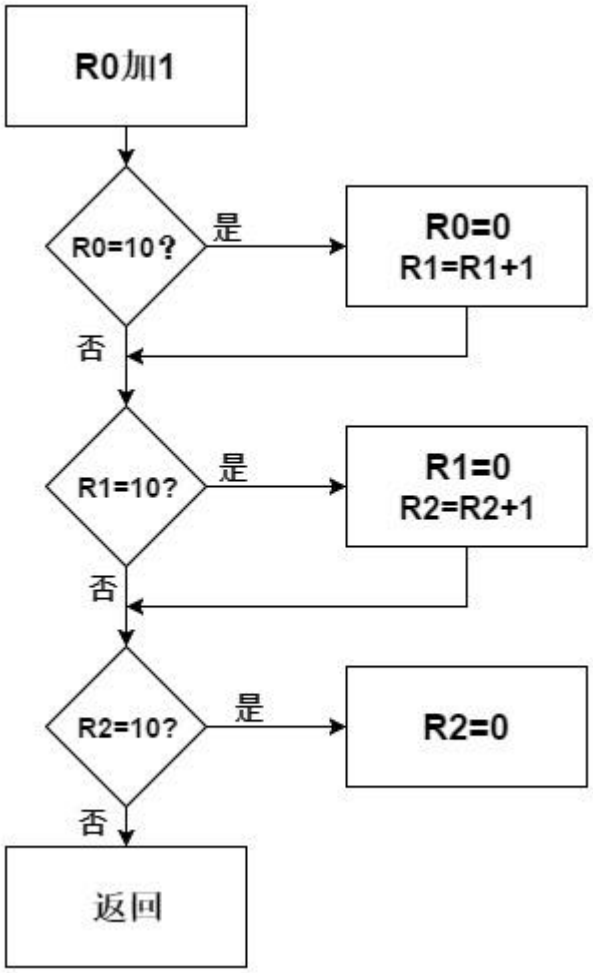
```
TABLE:
```

```
DB  0C0H,0F9H,0A4H,0B0H,99H,92H,82H,0F8H,80H,90H
```

2、每次步进电机前进一步（不论顺时针还是逆时针），计数值均加 1。

（R0、R1、R2 分别对应了个位、十位、百位）

（注意进位问题，例如 R0 原来值为 9，加一之后变成 10，应该 R9 置 0，R1 加 1）



3、显示数字

因为要想输出一个字形码，就需要从高位到低位（因此是循环左移）依次向移位寄存器输出 8 个比特。因此需要对 R0、R1、R2 分别向移位寄存器输出，在 24 个 bit 之后，欲显示的字形将稳定地显示在数码管上。

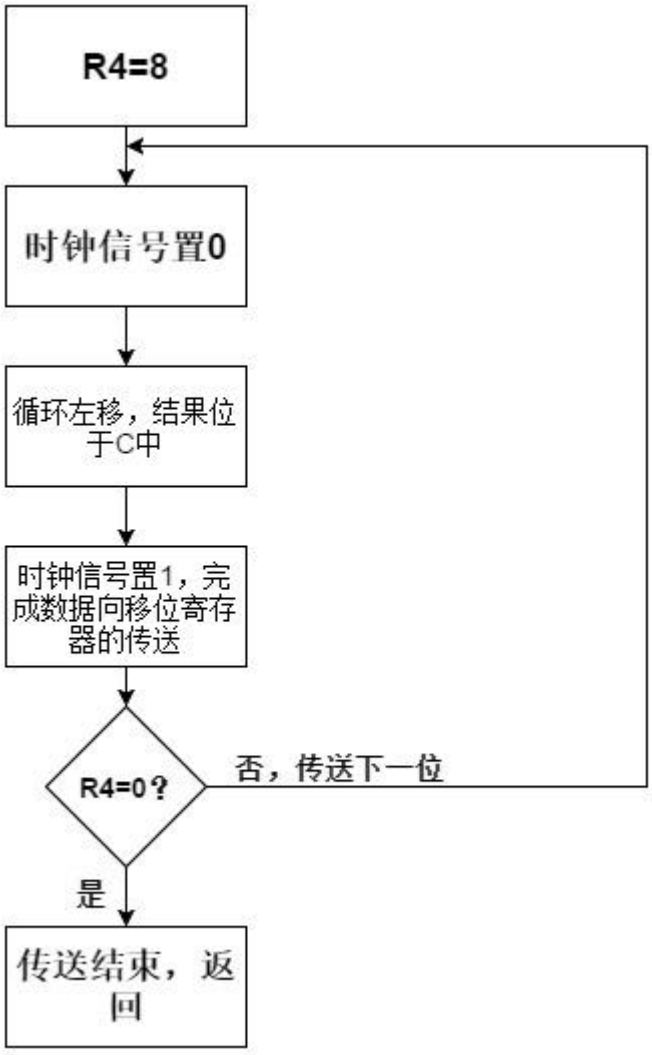
```
DISPLAY:
    MOV A,R0
    MOVC    A,@A+DPTR      ;DPTR 是 table
    LCALL   SENDNUM        ; 对 R0 调用 SENDNUM
```

```
MOV A,R1
MOVC    A,@A+DPTR
LCALL   SENDNUM     ；对 R0 调用 SENDNUM

MOV A,R2
MOVC    A,@A+DPTR
LCALL   SENDNUM     ；对 R0 调用 SENDNUM

RET
```

SENDNUM 流程图如下：



当对 R0、R1、R2 均完成数据传送后（8×3=24 位），计数值将稳定地显示在数码管上，程序可以转而执行其他工作。

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。

6. 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）

内部 RAM 的 0-1FH 为四组工作寄存器区,每个区有 8 个工作寄存器(R0－R7)。在同一时刻，只能使用一组工作寄存器，这是通过程序状态字 PSW 的地 3，4 位来控制的。例如当此两位为 00 时，使用第 0 组工作寄存器，对应于 00H 到 07H 的内部 RAM 空间。也就是说，这时指令中使用 R0 与直接使用 00 单元是 等价的，不过使用工作寄存器的指令简单，且执行快。

（2）可位寻址 RAM 区（20H——2FH）

内部 RAM 的 20H－2FH 为位寻址区域，这 16 个单元的每一位都对应一个位地址，占据位地址空间的 0－7FH，每一位都可以独立置位、清除、取反等操作。

（3）通用的 RAM 区（30H——7FH）

在中断和子程序调用中都需要堆栈。MCS－51 的堆栈理论上可以设置在内部 RAM 的任意区域，但由于 0－1FH 和 20－2FH 区域有上面说的特殊功能，因此一般设置在 30H 以后。

在内部 RAM 中，所有的单元都可以作为通用的数据存储器使用，存放输入 的数据或计算的中间结果等。

也可以作为条件转移的条件使用。

7. 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
```

8. 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 。

9. 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、老师布置思考题

比较已有的两份代码之间差异，解释为什么代码 1 能使圆盘旋转一周而代码 2 只能使圆盘旋转一部分？

答：代码 1 的实现过程在上面【六、代码实现过程】中已有详细解释。代码 1 利用 R5 完成旋转操作。

代码一实现如下：

```
CLR P1.1                ;CE1，CE2 至置高，才能按照指定脉冲显示
```

```
CLR P1.4
RRC A                ;右移
MOV P1.0, C          ;第八位给 P1.0 :IN2
RRC A
MOV P3.2, C          ;第七位给 P3.2 :IN1
SETB P1.1            ;CE1，CE2 至置高，才能按照指定脉冲显示
SETB P1.4
```

需要注意的是,程序并不是将CE1、CE2置高就能顺利完成对IN1、IN2的数据传送，而是每次都先置0、再置1，在上升沿完成对数据的传送！从这个角度考虑，之所以代码二不能旋转完整的一周，应该是时序有问题。经检查代码二发现：代码二只在实验起始位置对CE1、CE2置1，接下来的代码里没有涉及对CE1、CE2的操作，由于不是处于CE1、CE2上升沿，因此代码二在对IN1、IN2进行数据传送时出错。

修改：

每次数据传送都加上如下四行代码即可：

```
CLR CE1
CLR CE2      ;  CE1、CE2 置 0
SETB   IN1
SETB   IN2
MOV R5,#1
MOV R6,#1
```

SETB CE1

SETB CE2 ； CE1、CE2 置 1

实验四 LED 点阵显示屏

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

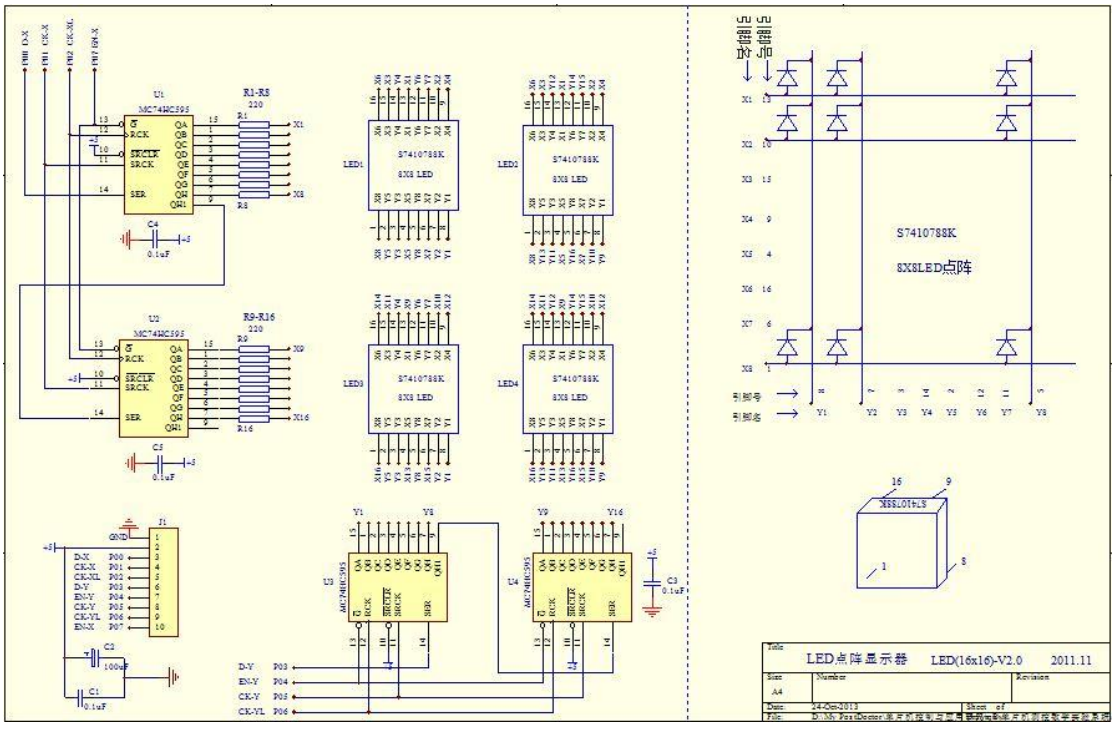
1、单片机测控实验系统
2、LED 点阵显示器实验模块
3、Keil 开发环境
4、STC-ISP 程序下载工具

三、实验内容

了解 16*16 点阵电路的原理，编写汇编语言程序。
编写一行汉字字符（至少三个字）的显示程序。
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、实验原理

本实验原理图如下：



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)。为了能够显示出一个点阵字型，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。

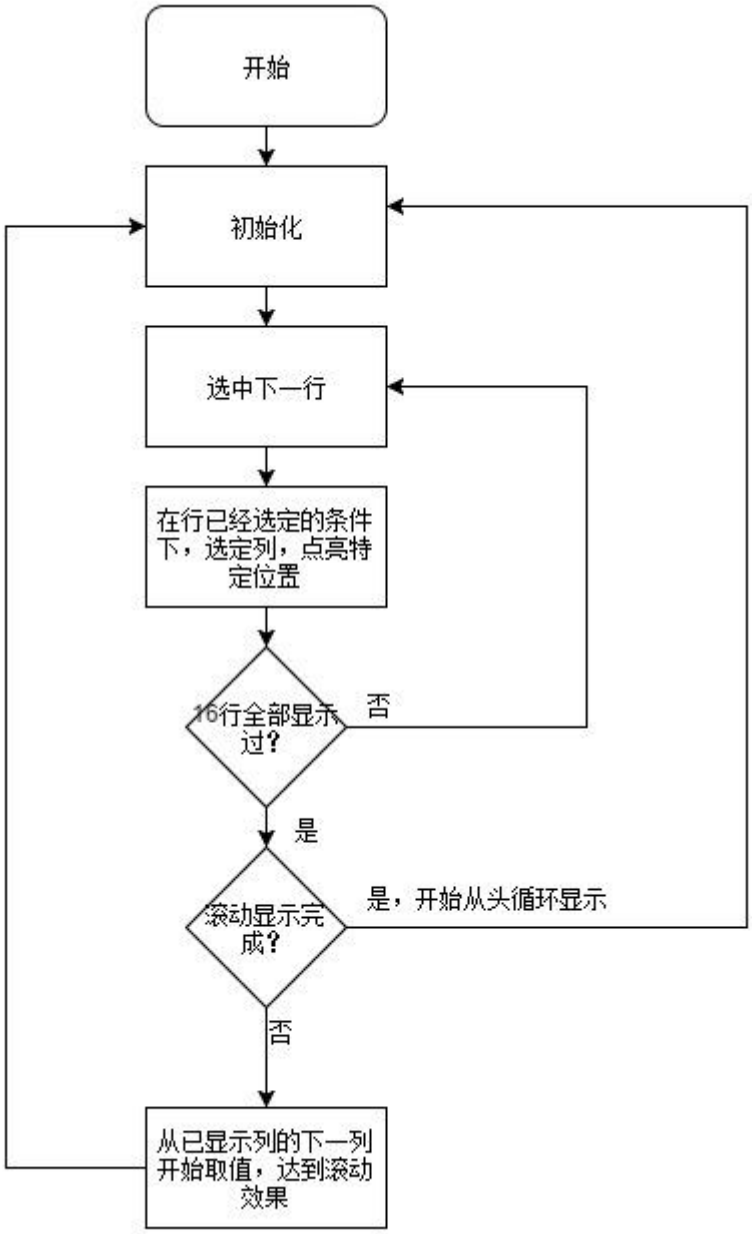
实验中使用的移位寄存器是 74HC595，它是一个同时具有串行移位和输出锁存驱动功能的器件。74HC595 是具有 8 位移位寄存器和一个存储器，三态输出功能。 移位寄存器和存储器是分别的时钟。数据在 SRCK（移位寄存器时钟输入）的上升沿输入到移位寄存器中，在 RCK（存储器时钟输入）的上升沿输入到存储寄存器中去。移位寄存器有一个串行移位输入（行 Dx（P00）、列 Dy(P03)），和一个串行输出（QH），和一个异步的低电平复位，存储寄存器有一个并行 8 位的，具备三态的总线输出，当使能 （P02 和 P07 为低电平）时，存储寄存器的数据输出到总线。

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。

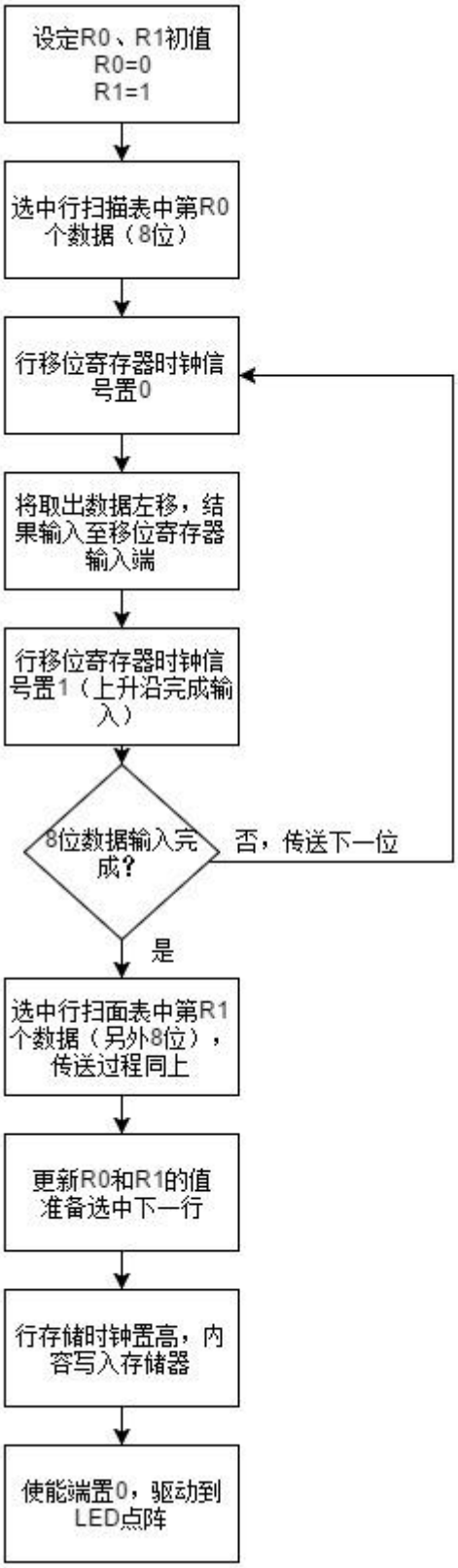
先送出第 1 行控制发光管亮灭的数据并锁存，然后选通第 1 行使其点亮一段时间；然后送出第 2 行控制发光管亮灭的数据并锁存，然后选通第 2 行使其点亮相同时间……第 16 行之后又重头点亮第 1 行，反复轮回，速度够快，由于人眼的暂留效果，就能看到屏上稳定的字。

每次循环显示完 16 行即一屏之后，再从第 1 行开始显示时，直接从上次开始的列数的下一列开始取值，比如上次从第 1 列开始显示，则循环完 16 行后，再从第 2 列开始显示…….这样就能形成字符滚动显示的现象。

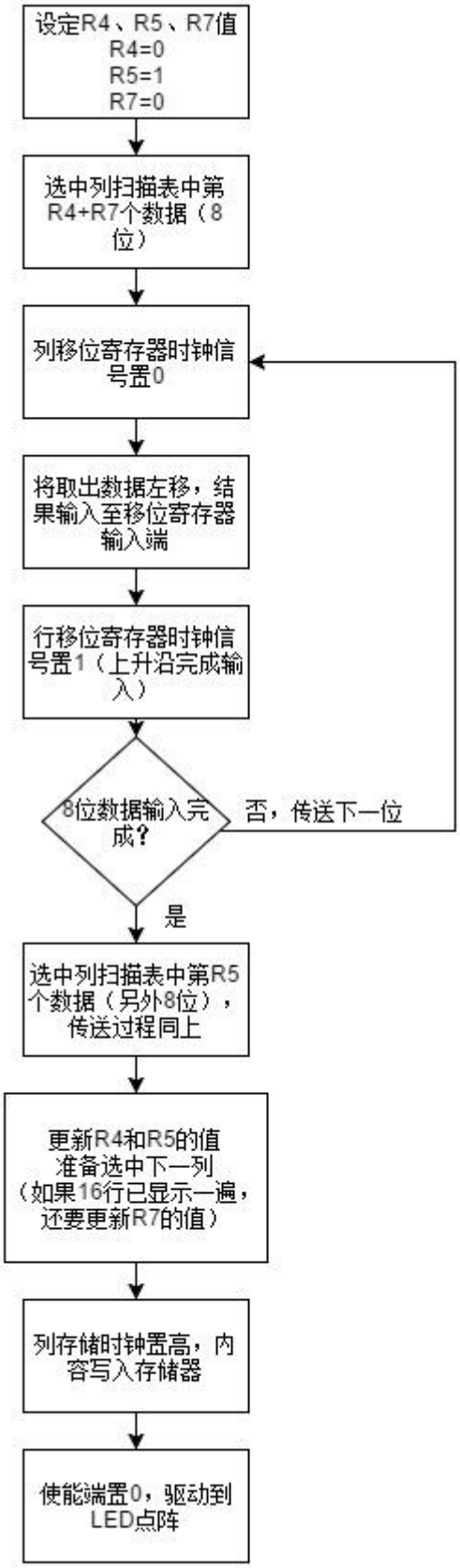
五、实验流程图



其中，选中某行的流程图如下：


设定R0、R1初值
R0=0
R1=1
选中行扫描表中第R0个数据（8位）
行移位寄存器时钟信号置0
将取出数据左移，结果输入至移位寄存器输入端
行移位寄存器时钟信号置1（上升沿完成输入）
8位数据输入完成？
否，传送下一位
是
选中行扫面表中第R1个数据（另外8位），传送过程同上
更新R0和R1的值准备选中下一行
行存储时钟置高，内容写入存储器
使能端置0，驱动到LED点阵

列选中流程图如下：


设定R4、R5、R7值
R4=0
R5=1
R7=0
选中列扫描表中第R4+R7个数据（8位）
列移位寄存器时钟信号置0
将取出数据左移，结果输入至移位寄存器输入端
行移位寄存器时钟信号置1（上升沿完成输入）
8位数据输入完成?
否，传送下一位
是
选中列扫描表中第R5个数据（另外8位），传送过程同上
更新R4和R5的值
准备选中下一列
（如果16行已显示一遍，还要更新R7的值）
列存储时钟置高，内容写入存储器
使能端置0，驱动到LED点阵

六、思考题

1. 如何使用软件调整和控制 LED 点阵的亮度
答：可以通过控制行显示延时调整亮度。延时越短，扫描频率越快，LED 点阵越亮。

2. 如何尽量避免显示过程中的闪烁
答：增加每一屏显示次数，一般刷新频率提高到 24hz 以上。

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）
答： 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 0ffffH 改为 64 位的 0ffffffffffffffffeH 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。

实验五 重量测量

一、实验目的和要求

1、掌握点阵式液晶显示屏的原理和控制方法，掌握点阵字符的显示方法。
2、掌握模拟/数字（A/D）转换方式，
3、进一步掌握使用 C51 语言编写程序的方法，使用 C51 语言编写实现重量测量的功能。

二、实验设备

单片机测控实验系统
重量测量实验板/砝码
Keil 开发环境
STC-ISP 程序下载工具

三、实验内容

参考辅助材料，学习 C51 语言使用
编写 C51 程序，使用重量测量实验板测量标准砝码的重量，将结果（以克计）显示到液晶屏上。误差可允许的范围之间。

四、实验步骤

1. 阅读实验原理，掌握 YM12864C 的控制方式，编写出基本的输出命令和数据的子程序；
2. 掌握点阵字模的构成方式。使用字模软件 PCtoLCD2002，设定正确的输出模式，生成点阵数据
3. 使用 C51 语言编写重量测量程序；
4. 调零，满量程校准；
5. 将编译后的程序下载到 51 单片机；
6. 在托盘中放上相应重量的法码，使显示值为正确重量。

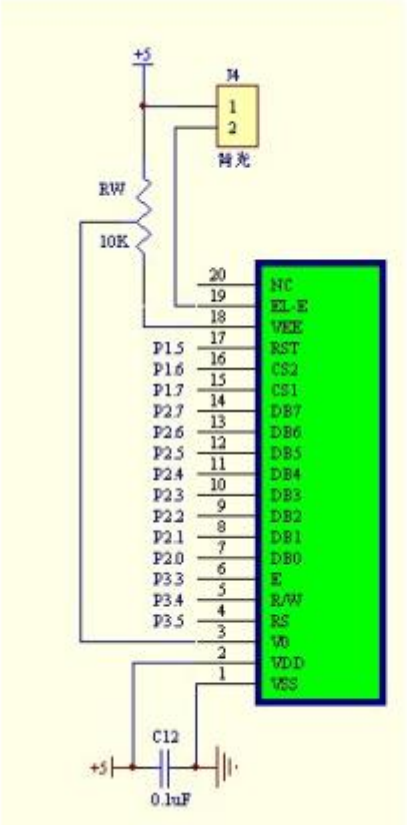
五、实验原理

实验分为**液晶显示**和**重力测量（主要用到的 A/D 转换器）**两个部分。

1、液晶显示

本实验平台使用一个集成的液晶显示屏驱动芯片 YM12864C。

1.1 硬件说明



引脚特性如下表所示：

引脚名称	级别	引脚功能描述
CS1	H/L	片选信号，当/CS1=L 时,液晶左半屏显示
CS2	H/L	片选信号，当/CS2=L 时,液晶右半屏显示
VSS	0V	电源地
VDD	+5V	电源电压
V0	0 至-10V	LCD 驱动负电压，要求 VDD-VLCD=10V
RS	H/L	寄存器选择信号
R/W	H/L	读/写操作选择信号
E	H/L	使能信号
DB0	H/L	八位三态并行数据总线
DB1		
DB2		
DB3		
DB4		
DB5		
DB6		
DB7		
RES	H/L	复位信号，低电平有效

VOUT	-10V	输出-10V 的负电压（单电源供电）
LED+(EL)	+5V	背光电源,Idd≤960mA
LED-(EL)	0V	

YM12864C 的液晶分为左边和右边两个 64×64 的子屏，分别通过 CS1 和 CS2 选通，每个子屏相应的内部寄存器是相互独立的。在一个时刻只能选择一个子屏操作。

1.2 寄存器功能说明

内部扫描时钟实时将 DDRAM 数据通过光学震荡显示在 LCM 液晶屏上， 微处理器将数据通过数据总线在时序电路的控制下写入 DDDRAM 某个单元， DDRAM 单元的选择通过 X， Y， Z 3 个地址寄存器决定。 CS1 和 CS2 决定片选子屏， X 地址寄存器决定子屏中显示单元页位置，从上至下，每 8 行，即一个字节为一页，范围从 D0h ～ D7h ；Y 地址寄存器决定子屏中显示单元列位置，范围从 0 ～3Fh ; Z 地址寄存器决定行滚动的首行地址，首行范围从 1～64 。

下面将详细介绍每个寄存器的功能和使用方法。

1）显示数据 RAM(DDRAM)

DDRAM（64×8×8 bits）是存储图形显示数据的。此 RAM 的每一位数据对应显示面板上一个点的显示（数据为 H）与不显示（数据为 L）。

2） I/O 缓冲器（DB0～DB7）

I/O 缓冲器作用是将两个不同时钟下工作的系统连接起来，实现通讯。 I/O 缓冲器在片选信号/CS 有效状态下,I/O 缓冲器开放，实现 LCM（液晶显示模块）与单片机之间的数据传递。当片选信号为无效状态时， I/O 缓冲器将中断 LCM（液晶显示模块）内部总线与单片机数据总线的联系，对外总线呈高阻状态，从而不影响单片机的其他数据操作功能。

3）输入寄存器

输入寄存器用于接收传送给 LCM（液晶显示模块）的数据并将其锁存在输入寄存器内，其输出将在 LCM（液晶显示模块）内部工作时钟的运作下将数据写入指令寄存器或显示存储器内。

4）输出寄存器

输出寄存器用于暂存从显示存储器读出的数据，在单片机读操作时，输出寄存器将当前锁存的数据通过 I/O 缓冲器送入单片机数据总线上。

5）指令寄存器

指令寄存器用于接收单片机发来的指令代码， 通过译码将指令代码置入相关的寄存器或触发器内。

6）状态字寄存器

状态字寄存器是 LCM（液晶显示模块）与单片机通讯时唯一的“握手”信号。状态字寄存器向单片机表示了 LCM（液晶显示模块）当前的工作状态。 尤其是状态字中的“忙”标志位是单片机在每次对 LCM（液晶显示模块）访问时必须要读出判别的状态位。当处于“忙”标志位时， I/O 缓冲器被封锁，此时单片机对 LCM（液晶显示模块）的任何操作（除读状态字操作外）都将是无效的。

7） X 地址寄存器

X 地址寄存器是一个三位页地址寄存器，其输出控制着 DDRAM 中 8 个页面的选 择，也是控制着数据传输通道的八选一选择器。 X 地址寄存器可以由单片机以指令形式设置。X 地址寄存器没有自动修改功能，所以要想转换页面需要重新设置 X 地址寄存器的内容。

8） Y 地址计数器

Y 地址计数器是一个 6 位循环加一计数器。它管理某一页面上的 64 个单元。 Y 地址计数器可以由单片机以指令形式设置，它和页地址指针结合唯一选通显示存储器的一个单元， Y 地址计数器具有自动加一功能。在显示存储器读/写操作后 Y 地址计数将自动加一。当计数器加至 3FH 后循环归零再继续加一。

9） Z 地址计数器

Z 地址计数器是一个 6 位地址计数器，用于确定当前显示行的扫描地址。 Z 地址计数器具有自动加一功能。它与行驱动器的行扫描输出同步，选择相应的列驱动的数据输出。

10）显示起始行寄存器

显示起始行寄存器是一个 6 位寄存器，它规定了显示存储器所对应显示屏上第一行的行号。该行的数据将作为显示屏上第一行显示状态的控制信号。

11）显示开/关触发器

显示开/关触发器的作用就是控制显示驱动输出的电平以控制显示屏的开关。

1.3 软件功能说明

1.3.1 指令表

指令名称	控制信号		控制代码							
	RS	R/W	D7	D6	D5	D4	D3	D2	D1	D0
显示开关设置	0	0	0	0	1	1	1	1	1	D
显示起始行设置	0	0	1	1	L5	L4	L3	L2	L1	L0
页面地址设置	0	0	1	0	1	1	1	P2	P1	P0

列地址设置	0	0	0	1	C5	C4	C3	C2	C1	C0
读取状态字	0	1	BUSY	0	ON/OFF	RESET	0	0	0	0
写显示数据	1	0	数据							
读显示数据	1	1	数据							

1）读状态字

BUSY=1 表示 LCM 正在处理单片机发过来的指令或数据。 此时接口电路被封锁，不能接受除读状态字以外的任何操作。 BUSY=0 表示 LCM 接口控制电路已外于“准备好”状态，等待单片机的访问。在指令设置和数据读写时要注意状态字中的 BUSY 标志。只有在 BUSY=0 时，单片机对 LCM 的操作才能有效。因此单片机在每次对 LCM 操作之前，都要读出状态字判断 BUSY 是否为“0”。若不为“0”，则单片机需要等待，直至 BUSY=0 为止。

ON/OFF 表示当前的显示状态。 ON/OFF 1 表示关显示状态， ON/OFF 0 表示开显示状态。

RES 为低电平状态时，LCM 处于复位工作状态，标志位 RESET=1。

2)显示开关设置

该指令设置显示开/关触发器的状态， D 位为显示开/关的控制位。当 D=1 为开显示设置，显示数据锁存器正常工作，此时在状态字中 ON/OFF=0。当 D=0 为关显示设置，显

示数据锁存器被置零，显示屏呈不显示状态，但显示存储器并没有被破坏，在状态字中ON/OFF=1。

3)显示起始行设置

该指令设置了显示起始行寄存器的内容。 LCM 通过/CS 的选择分别具有 64 行显示的管理能力，该指令中 L5～L0 为显示起始行的地址，取值在 0～3FH（1～64 行）范围内，它规定了显示屏上最顶一行所对应的显示存储器的行地址。

4)页面地址设置

该指令设置了页面地址 X 地址寄存器的内容。 LCM 将显示存储器分成 8 页，指令代码中 P2～P0 就是要确定当前所要选择的页面地址，取值范围为 0～7H ，代表 1～8 页。该指令规定了以后的读/写操作将在哪一个页面上进行。

5)列地址设置

该指令设置了 Y 地址数计数器的内容， LCM 通过/CS 的选择分别具有 64 列显示的管理能力， C5～C0=0～3FH（1～64）代表某一页面上的某一单元地址， 随后的一次读或写数据将在这个单元上进行。 Y 地址计数器具有自动加一功能，在每一次读/写数据后它将自动加一，所以在连续进行读/写数据时， Y 地址计数器不必每次都设置一次。页面地址的设置和列地址的设置将显示存储器单元唯一地确定下来，为后来的显示数据的读/写作了地址的选通。

6)写显示数据

该操作将 8 位数据写入先前已确定的显示存储器的单元内。操作完成后列地址计数器自动加一。

7)读显示数据

该操作将 LCM 接口部的输出寄存器内容读出，然后列地址计数器自动加一。

1.3.2 控制时序表

CS1	CS2	RS	R/W	E	DB7~DB0	功能
X	X	X	X	0	高阻	总线释放
1	1	0	0	下降沿	输入	写指令代码
1	1	0	1	1	输出	读状态字
1	1	1	0	下降沿	输入	写显示数据
1	1	1	1	1	输出	读显示数据

1.3.3 DDRAM 表

下图表示了内部显存的组织方式，也就是要显示的数据需要依照的格式。

	CS1=1					CS2=1					
Y=	0	1	...	62	63	0	1	...	62	63	行号
X=0 ↓ X=7	DB0 ↓ DB7	DB0 ↓ DB7	DB0 ↓ DB7	DB0 ↓ DB7	DB0 ↓ DB7	DB0 ↓ DB7	DB0 ↓ DB7	DB0 ↓ DB7	DB0 ↓ DB7	DB0 ↓ DB7	0 ↓ 7
	DB0 ↓ DB7	DB0 ↓ DB7	DB0 ↓ DB7	DB0 ↓ DB7	DB0 ↓ DB7	DB0 ↓ DB7	DB0 ↓ DB7	DB0 ↓ DB7	DB0 ↓ DB7	DB0 ↓ DB7	0 ↓ 7
	DB0 ↓ DB7	DB0 ↓ DB7	DB0 ↓ DB7	DB0 ↓ DB7	DB0 ↓ DB7	DB0 ↓ DB7	DB0 ↓ DB7	DB0 ↓ DB7	DB0 ↓ DB7	DB0 ↓ DB7	0 ↓ 7

2、ADC 转换器

2.1 P1 口模拟功能控制寄存器 P1ASF

用户可以通过软件设置将 8 路中的任何一路设置为 A/D 转换，不需作为 A/D 使用的 P1 口可继续作为 I/O 口使用(建议只作为输入)。需作为 A/D 使用的口需先将 P1ASF 特殊功能寄存器中的相应位置为‘1’，将相应的口设置为模拟功能。

2.2 ADC 控制寄存器 ADC_CONTR

ADC_CONTR : ADC 控制寄存器

SFR name	Address	bit	B7	B6	B5	B4	B3	B2	B1	B0
ADC_CONTR	BCH	name	ADC_POWER	SPEED1	SPEED0	ADC_FLAG	ADC_START	CHS2	CHS1	CHS0

ADC_POWER: ADC 电源控制位。

0：关闭 A/D 转换器电源；

1：打开 A/D 转换器电源.

SPEED1，SPEED0：模数转换器转换速度控制位

SPEED1	SPEED0	A/D 转换所需时间
1	1	90 个时钟周期转换一次,CPU 工作频率 21MHz 时，A/D 转换速度约 250KHz
1	0	180 个时钟周期转换一次
0	1	360 个时钟周期转换一次
0	0	540 个时钟周期转换一次

ADC_FLAG：模数转换器转换结束标志位，当 A/D 转换完成后，ADC_FLAG=1，要由软件清 0。

CHS2/CHS1/CHS0：模拟输入通道选择， CHS2/CHS1/CHS0

CHS2	CHS1	CHS0	Analog Channel Select（模拟输入通道选择）
0	0	0	选择 P1.0 作为 A/D 输入来用
0	0	1	选择 P1.1 作为 A/D 输入来用
0	1	0	选择 P1.2 作为 A/D 输入来用
0	1	1	选择 P1.3 作为 A/D 输入来用
1	0	0	选择 P1.4 作为 A/D 输入来用
1	0	1	选择 P1.5 作为 A/D 输入来用
1	1	0	选择 P1.6 作为 A/D 输入来用

2.3 A/D 转换结果寄存器 ADC_RES、 ADC_RESL

特殊功能寄存器 ADC_RES 和 ADC_RESL 寄存器用于保存 A/D 转换结果，其格式如下：

nemonic	Add	Name	B7	B6	B5	B4	B3	B2	B1	B0
DC_RES	BDh	A/D 转换结果寄存器高							ADC_RES9	ADC_RES8
DC_RESL	BEh	A/D 转换结果寄存器	ADC_RES7	ADC_RES6	ADC_RES5	ADC_RES4	ADC_RES3	ADC_RES2	ADC_RES1	ADC_RES0

		低								
AUXR1	A2H	Auxiliary register1	-					ADRJ=1	-	

nemonic	Add	Name	B7	B6	B5	B4	B3	B2	B1	B0
DC_RES	BDh	A/D 转换结果寄存器高 8 位	ADC_RES9	ADC_RES8	ADC_RES7	ADC_RES6	ADC_RES5	ADC_RES4	ADC_RES3	ADC_RES2
DC_RESL	BEh	A/D 转换结果寄存器低 2 位	-	-	-	-	-	-	ADC_RES1	ADC_RES0
AUXR1	A2H	Auxiliary register1	ADRJ = 0							

ADRJ=1 时， 10 位 A/D 转换结果的高 2 位存放在 ADC_RES 的低 2 位中，低 8 位存放在 ADC_RESL 中。

nemonic	Add	Name	B7	B6	B5	B4	B3	B2	B1	B0
DC_RES	BDh	A/D 转换结果寄存器高 2 位	-	-	-	-	-	-	ADC_RES9	ADC_RES8
DC_RESL	BEh	A/D 转换结果寄存器低 8 位	ADC_RES7	ADC_RES6	ADC_RES5	ADC_RES4	ADC_RES3	ADC_RES2	ADC_RES1	ADC_RES0
AUXR1	A2H	Auxiliary register1	ADRJ = 1							

六、思考题

1. 调零的原理，软件调零和调零调零的区别

调零是指在未放置砝码时，液晶屏示数应该为 0。
软件调零是在程序中通过减去空砝码时重力测量值 cweight，使得示数为 0；
硬件调零是通过调整压敏电阻的阻值进行调整，从而进行调零。

2. 模/数和数/模的信号转换原理

A/D 转换器是用来通过一定的电路将模拟量转变为数字量。模拟量可以是电压、电流等电信号，也可以是压力、温度、湿度、位移、声音等非电信号。但在 A/D 转换前，输入到 A/D 转换器的输入信号必须经各种传感器把各种物理量转换成电压信号。

A/D 转换器的工作原理方法：

逐次逼近法：

基本原理是从高位到低位逐位试探比较，好像用天平称物体，从重到轻逐级增减砝码进行试探。逐次逼近法转换过程是：初始化时将逐次逼近寄存器各位清零；转换开始时，先将逐次逼近寄存器最高，送入 D/A 转换器，经 D/A 转换后生成的模拟量送入比较器,称为 Vo,与送入比较器的待转换的模拟量 Vi 进行比较，若 Vo<Vi，该位 1 被保留，否则被清除。然后再置逐次逼近寄存器次高位为 1，将寄存器中新的数字量送 D/A 转换器，输出的 Vo 再与 Vi 比较，若 Vo<Vi，该位 1 被保留，否则被清除。重复此过程，直至逼近寄存器最低位。转换结束后，将逐次逼近寄存器中的数字量送入缓冲寄存器，得到数字量的输出。逐次逼近的操作过程是在一个控制电路的控制下进行的。

双积分法：

基本原理是将输入电压变换成与其平均值成正比的时间间隔,再把此时间间隔转换成数字量，属于间接转换。 双积分法 A/D 转换的过程是：先将开关接通待转换的模拟量 Vi，Vi 采样输入到积分器，积分器从零开始进行固定时间 T 的正向积分，时间 T 到后，开关再接通与 Vi 极性相反的基准电压 VREF，将 VREF 输入到积分器，进行反向积分，直到输出为 OV 时停止积分。Vi 越大，积分器输出电压越大，反向积分时间也越长。计数器在反向积分时间内所计的数 值，就是输入模拟电压 Vi 所对应的数字量，实现了 A/D 转换。 双积分式 AD 转换原理图

电压频率转换法

它的工作原理是 V/F 转换电路把输入的模拟电压转换成与模拟电压成正比的脉冲信号。

3. I2C 总线在信号通讯过程中的应用

I2C 总线是由 Philips 公司开发的一种简单、双向二线制同步串行总线。它只需要两根线即可在连接于总线上的器件之间传送信息，提供集成电路（ICs）之间的通信线路，广泛应用于电视，录像机和音频等设备。I2C 总线的意思:“完成集成电路或功能单元之间信息交换的规范或协议”。Philips 公司推出的 I2C 总线采用一条数据线（SDA），加一条时钟线（SCL）来完成数据的传输及外围器件的扩展；对各个节点的寻址是软寻址方式，节省了片选线，标准的寻址字节 SLAM 为 7 位，可以寻址 127 个单元。

七、老师布置思考题

问：如何实现重量值的稳定显示？

答：由于电路的电压是有波动的，因此测出来的重量值一定不是稳定的，要想实现在测量数值不断波动的情况下的稳定显示，可以考虑四舍五入的方法。具体做法为：

判断个位数是否<5，如果是，则个位数赋值为 0，十位数不变；

否则，个位数赋值为 0，十位数加 1。

实验六　直流电机脉宽调制调速

一、实验目的和要求

掌握脉宽调制调速的原理与方法，学习频率/周期测量的方法，了解闭环控制的原理。

二、实验设备

单片机测控实验系统

直流电机调速实验模块

Keil 开发环境

STC-ISP 程序下载工具

三、实验内容

1. 在液晶显示屏上显示出直流电机的：当前转速、低目标转速、高目标转速。

2. 固定向 P1.1 输出 0，然后测量每秒钟电机转动的转数，将其显示在数码管，每秒刷新一次即可。

3. 使用脉宽调制的方法，动态调整向 P1.1 输出的内容，使得电机转速能够稳定在一个预定值附近，同时实时显示当前转速。

4. 根据输入修改电机得目标转速值，设置两个转速目标值：低转速和高转速。

5. 每隔一秒钟读取两个开关的状态，如果 S1 按下，动态调整输出，使得电机转速能够稳定到低转速目标值附近，如果 S2 按下，动态调整输出，使得电机转速能够稳定到高转速目标值附近。交替显示目标值和当前转速值。

四、实验步骤

4.1 建立工程，实现实验内容

预习附录四，学习 C51 编程方法。设计实现一个进行显示的 C51 程序。

建立工程，实现实验内容 1。

将例子程序补充完整。建立一个新的工程，然后加入一个 C 语言文件，输入上述例子程序，编译并下载执行调试。

4.2 编写中断程序，测量电机转速

编写中断程序，测量直流电机的转速。

按照实验原理，电机转速就是一秒钟之内 INT0 的中断个数。编写带有中断的 C51 程序，包括一个能够实现 1 秒钟的定时器中断和一个

外部中断。注意外部中断要设置边沿触发方式。程序框架参考附录四。

4.3 完成控制转速程序

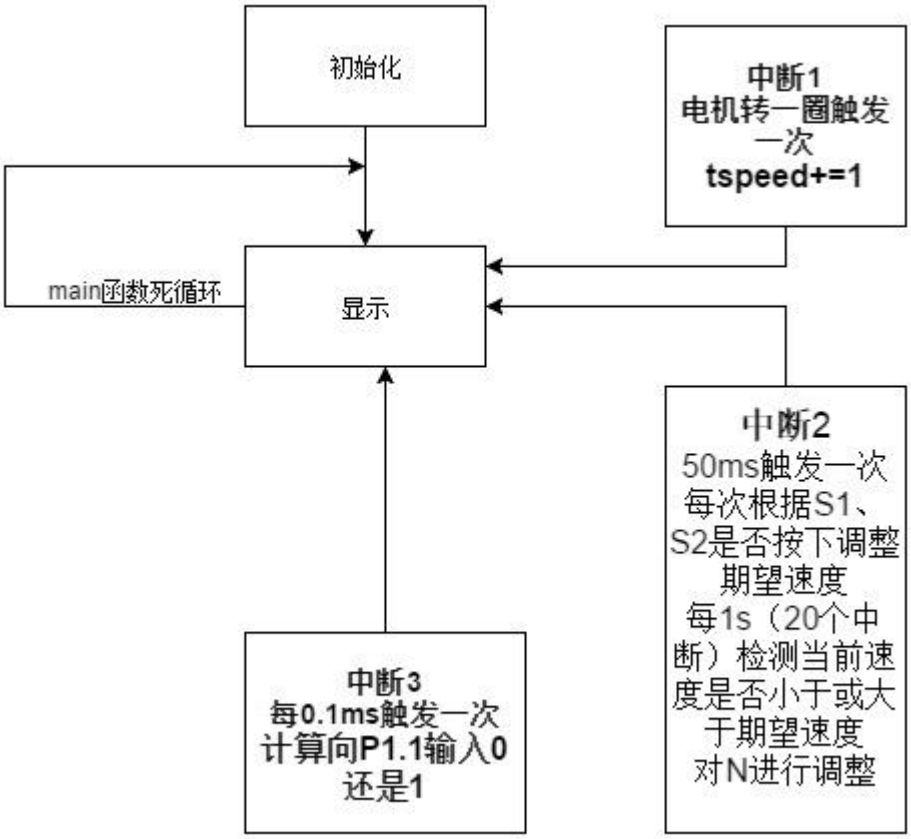
按照脉宽调制的原理，再添加一个快速的定时中断（0.1ms 左右），在这个中断里面动态改变 P1.1 的输出，宏观上输出有效（0）的比例就是预定的控制变量。这个控制变量增大，电机转速就应该提高，但由于各种内部和外部因素，它们之间不存在简单的函数关系，因此必须根据测量出来的实际转速进行动态调整。

首先将电机转速控制在一个预定数值附近，在每一个 1 秒钟中断测量出当前转速之后，将其与目标值相对比，如果不够则增加控制变量，否则减少之，这样就能逐步达到稳定转速的目的。同时将速度显示出来。

4.4 完成整体实验内容

在上面程序的基础上，再加上根据开关状态改变预定转速的代码。同时，在主程序中交替显示目标值和当前转速值，显示一个内容之后等待一段时间（可以由延时代码实现），然后再显示另一个并延时。要显示的内容都是在中断中被修改的。

六、实验原理



1、PWM

脉宽调制（Pulse Width Modulation，PWM）是一种能够通过开关量输出达到模拟量输出效果的方法。PWM 的基本原理是通过输出一个很高频率的 0/1 信号，其中 1 的比例为 δ （也叫做占空比），在外围积分元件的作用下，使得总的效果相当于输出 δ ×A（A 为高电平电压）的电压。通过改变占空比就可以调整输出电压，从而达到模拟输出并控制电机转速的效果。

使用单片机实现 PWM，就是根据预定的占空比 δ 来输出 0 和 1，这里 δ 就是控制变量。可以采用累加进位法如果将总的周期内的

0 和 1 均匀分散开。设置一个累加变量 x，每次加 N，若结果大于 M，则输出 1，并减去 M；否则输出 0。这样整体的占空比也是 N/M。在实验中取 M=256 可以使程序更加简单。（由于本实验板的设计，输出 0 使电机工作。因此对于本实验，上面所说的 0 和 1 要翻转过来用。）

2、转速测量

在本实验板中，电机每转动一次，与之相连的偏心轮将遮挡光电对管一次，因此会产生一个脉冲，送到 INT0。要测量转速，既可以测量相邻两次中断之间的时间；也可以测量一秒种之内发生的中断次数。显然，后一种方法更加简单。

进行转速控制时，涉及到三个变量：预期转速，实际转速和控制变量。这里控制变量就是占空比。我们并不能够预先精确知道某个控制变量的值会导致多少的实际转速，因为这里有很多内部和外部因素起作用（如摩擦力，惯性等），但可以确定就是随着控制变量的增加，实际转速会增加。

3、反馈控制

反馈控制的基本原理就是根据实际结果与预期结果之间的差值，来调节控制变量的值。当实际转速高于预期转速时，我们需要减少控制变量，以降低速度；反之则需要调高控制变量。

4、转速控制

本实验的转速控制可以使用简单的比例控制算法，也就是当转速

S 大于预定值时，将输出 0 的个数减少；当转速小于预定值时，将输出 0 的个数增加。改变值正比于测量出的差值。

七、思考题

（1）讨论脉宽调速和电压调速的区别、优缺点和应用范围。

脉宽调制是利用电力电子开关器件的导通与关断，将直流电压变成连续的直流脉冲序列，并通过控制脉冲的宽度或周期来达到变压的目的。PWM 的占空比决定输出到直流电机的平均电压，PWM 不是调节电流的，而是调节方波低电平和高电平的时间。占空比越大，高电平时间越长，则输出的脉冲幅度越高，电压越大，也即通过调节占空比可以达到调节电压的目的，而且输出电压可以无级连续调节。

PWM 不需要在计算机接口中使用 D/A 转换器，适用于低频大功率控制。脉宽调速可大大节省电量，具有很强的抗噪性，且节约空间、经济实惠。

电压调速是改变加大电枢上的电压大小，一般是连续的供电，电机低速连续转动。电压调速工作时不能超过特定电压，优点是机械特性较硬并且电压降低后硬度不变，稳定性好，适用于对稳定性要求较高的环境。

（2）说明程序原理中累加进位法的正确性

累加进位法：设置一个累加变量 x，每次加 N，若结果大于 M，则输出 1，并减去 M；否则输出 0。这样整体的占空比也是 N/M。在实验中取 M=256 可以使程序更加简单。

从上述说明可以看出，x 每次增加 N，当 x 的累加值大于 M 的某

个倍数时，输出 1，否则输出 0。在 x 从 0 递增至 kM 的期间，由于 x 每次递增 N，因此一共输出 kM/N 次，其中输出 1 是 k-1 次（在 x 从 0 到 kM 期间，x 分别大于 M、2M、3M、...（k-1）M，因此输出 1 是 k-1 次），占空比为（k-1）/（kM/N）≈ N/M。

（3）计算转速测量的最大可能误差，讨论减少误差的办法。

电机转动 1 周触发 1 次中断，本实验是通过对 1s 触发的中断进行计数来间接得到转速的，可知，当电机转速较高时，精度较高，当电机转速较低时，可能会产生较大误差（最极端的情况，如果电机 1s 还没有转 1 圈，按照此方法测出来速度是 0，但显然，即使速度非常慢，也不是 0）。

减少误差的方法：可以增加齿盘的齿轮数，使得转 1 圈的脉冲计数增大。如每转 1 圈发出 10 个脉冲，则测速精度可精确至 0.1 圈。

实验八　温度测量与控制

一、实验目的和要求

1.学习 DS18B20 温度传感器的编程结构。

2.了解温度测量的原理。

3. 掌握 PID 控制原理及实现方法。

4. 加深 C51 编程语言的理解和学习。

二、实验设备

单片机测控实验系统

温控实验模块

Keil 开发环境

STC-ISP 程序下载工具

三、实验内容

掌握使用传感器测量与控制温度的原理与方法，使用 C51 语言编写实现温度控制的功能，使用超声波/温度实验板测量温度，将温度测量的结果（单位为摄氏度）显示到液晶屏上。

编程实现测量当前教室的温度，显示在 LCM 液晶显示屏上。

通过 S1 设定一个高于当前室温的目标温度值。

编程实现温度的控制，将当前温度值控制到目标温度值并稳定的显示。

四、实验步骤

1.预习，参考附录三，预习 DS18B20 的编程结构，编程时注意 DS18B20 的时间要求，必须准确满足。根据实验原理附录中的流程图进行编程。

2. 将编译后的程序下载到 51 单片机，观察温度的测量结果。

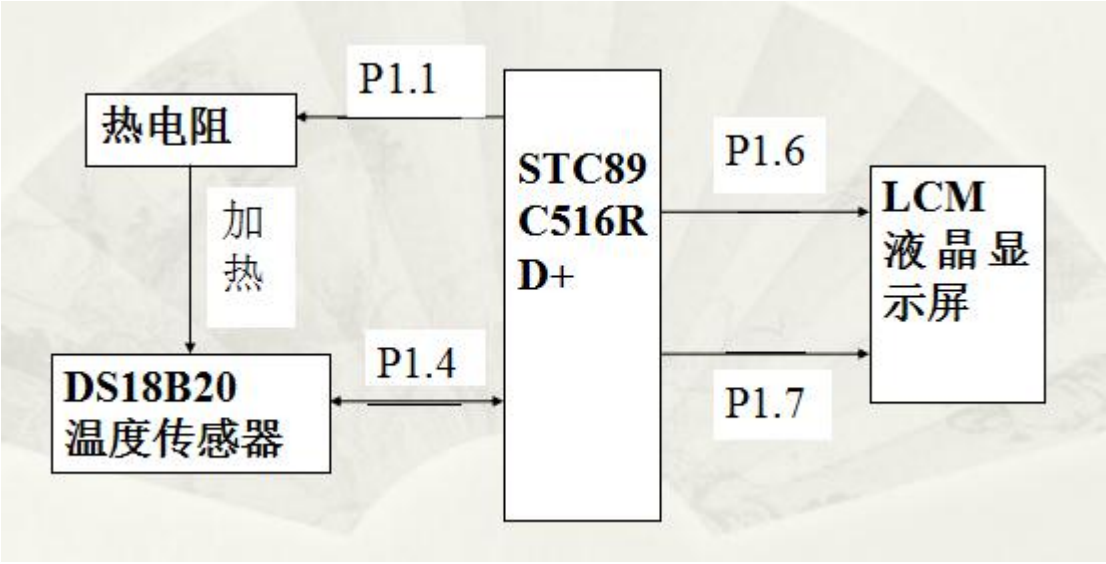
3. 程序调试

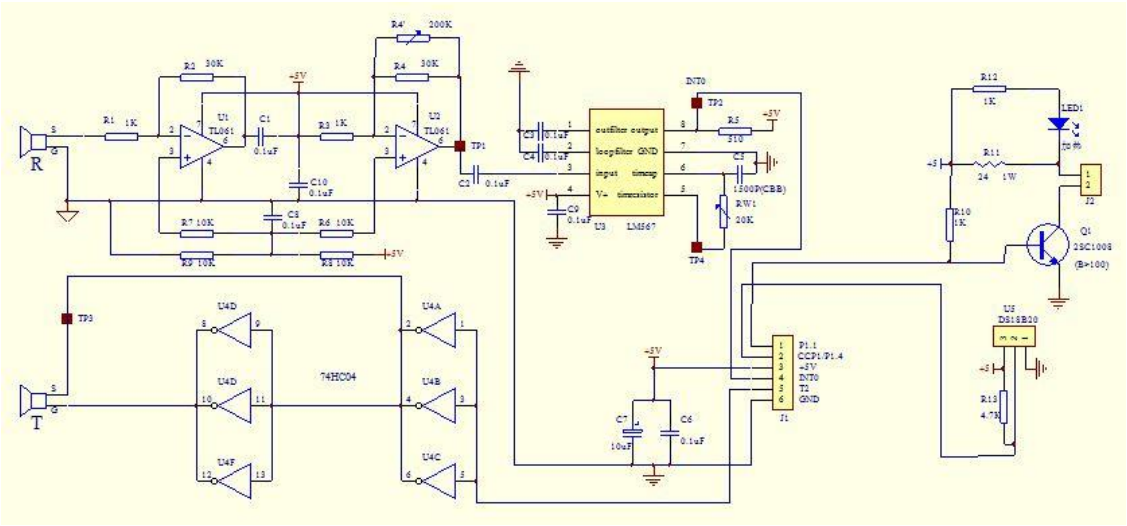
八、实验原理

本实验使用 STC89C516RD+单片机实验板。单片机的 P1.4 与 DS18B20 的 DQ 引脚相连，进行数据和命令的传输。

单片机的 P1.1 连接热电阻。当 P1.1 为高电平时，加热热电阻。

温度控制的方法采用 PID 控制实现。





本实验分为两个部分——获取温度值和液晶屏显示。

获取温度值的实验原理如下：

1、如何获取温度值？——通过 DS18B20

2、测量结果存储在哪？——用于贮存测得的温度值的两个 8 位存贮器 RAM 编号为 0 号和 1 号。1 号存贮器存放温度值的符号，如果温度为负（℃），则 1 号存贮器 8 位全为 1，否则全为 0。0 号存贮器用于存放温度值的补码 LSB(最低位)的 1 表示 0.5℃ 。将存贮器中的二进制数求补再转换成十进制数并除以 2，就得到被测温度值。

3、如何实现温度控制？——PID 控制

PID 控制器就是根据系统的误差，利用比例、积分、微分计算出控制量进行控制的。

比例 P 控制

比例控制是一种最简单的控制方式。其控制器的输出与输入误差信号成比例关系。当仅有比例控制时系统输出存在稳态误差。

积分 I 控制

在积分控制中，控制器的输出与输入误差信号的积分成正比关系。对一个自动控制系统，如果在进入稳态后存在稳态误差，则称这个控制系统是有稳态误差的或简称有差系统。为了消除稳态误差，在控制器中必须引入“积分项”。积分项对误差取决于时间的积分，随着时间的增加，积分项会增大。这样，即便误差很小，积分项也会随着时间的增加而加大，它推动控制器的输出增大使稳态误差进一步减小，直到等于零。因此，比例+积分（PI）控制器，可以使系统在进入稳态后无稳态误差。

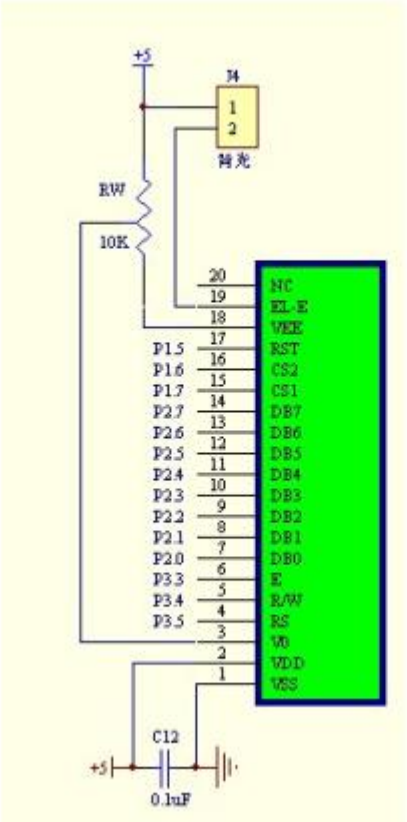
微分 D 控制

在微分控制中，控制器的输出与输入误差信号的微分（即误差的变化率）成正比关系。自动控制系统在克服误差的调节过程中可能会出现振荡甚至失稳。其原因是由于存在有较大惯性组件（环节）或有滞后（delay）组件，具有抑制误差的作用，其变化总是落后于误差的变化。解决的办法是使抑制误差的作用的变化“超前”，即在误差接近零时，抑制误差的作用就应该是零。这就是说，在控制器中仅引入“比例”项往往是不够的，比例项的作用仅是放大误差的幅值，而需要增加的是“微分项”，它能预测误差变化的趋势，这样，具有比例+微分的控制器，就能够提前使抑制误差的控制作用等于零，甚至为负值，从而避免了被控量的严重超调。所以对有较大惯性或滞后的被控对象，比例+微分（PD）控制器能改善系统在调节过程中的动态特性。

液晶屏显示的原理如下：

本实验平台使用一个集成的液晶显示屏驱动芯片 YM12864C。

硬件说明



引脚特性如下表所示：

引脚名称	级别	引脚功能描述
CS1	H/L	片选信号，当/CS1=L 时,液晶左半屏显示
CS2	H/L	片选信号，当/CS2=L 时,液晶右半屏显示
VSS	0V	电源地
VDD	+5V	电源电压
V0	0 至-10V	LCD 驱动负电压，要求 VDD-VLCD=10V
RS	H/L	寄存器选择信号
R/W	H/L	读/写操作选择信号
E	H/L	使能信号
DB0	H/L	八位三态并行数据总线
DB1		
DB2		
DB3		
DB4		
DB5		
DB6		
DB7		
RES	H/L	复位信号，低电平有效
VOUT	-10V	输出-10V 的负电压（单电源供电）
LED+(EL)	+5V	背光电源,Idd≤960mA
LED-(EL)	0V	

YM12864C 的液晶分为左边和右边两个 64×64 的子屏，分别通过 CS1 和 CS2 选通，每个子屏相应的内部寄存器是相互独立的。在一个时刻只能选择一个子屏操作。

1.2 寄存器功能说明

内部扫描时钟实时将 DDRAM 数据通过光学震荡显示在 LCM 液晶屏上， 微处理器将数据通过数据总线在时序电路的控制下写入 DDDRAM 某个单元， DDRAM 单元的选择通过 X， Y， Z 3 个地址寄存器决定。 CS1 和 CS2 决定片选子屏， X 地址寄存器决定子屏中显示单元页位置，从上至下，每 8 行，即一个字节为一页，范围从 D0h ～ D7h ；Y 地址寄存器决定子屏中显示单元列位置，范围从 0 ～3Fh ; Z 地址寄存器决定行滚动的首行地址，首行范围从 1～64 。

下面将详细介绍每个寄存器的功能和使用方法。

1）显示数据 RAM(DDRAM)

DDRAM（64×8×8 bits）是存储图形显示数据的。此 RAM 的每一位数据对应显示面板上一个点的显示（数据为 H）与不显示（数据为 L）。

2） I/O 缓冲器（DB0～DB7）

I/O 缓冲器作用是将两个不同时钟下工作的系统连接起来，实现通讯。 I/O 缓冲器在片选信号/CS 有效状态下,I/O 缓冲器开放，实现 LCM（液晶显示模块）与单片机之间的数据传递。当片选信号为无效状态时， I/O 缓冲器将中断 LCM（液晶显示模块）内部总线与单片机数据总线的联系，对外总线呈高阻状态，从而不影响单片机的其他数据操作功能。

3）输入寄存器

输入寄存器用于接收传送给 LCM（液晶显示模块）的数据并将其锁存在输入寄存器内，其输出将在 LCM（液晶显示模块）内部工作时钟的运作下将数据写入指令寄存器或显示存储器内。

4）输出寄存器

输出寄存器用于暂存从显示存储器读出的数据，在单片机读操作时，输出寄存器将当前锁存的数据通过 I/O 缓冲器送入单片机数据总线上。

5）指令寄存器

指令寄存器用于接收单片机发来的指令代码， 通过译码将指令代码置入相关的寄存器或触发器内。

7）状态字寄存器

状态字寄存器是 LCM（液晶显示模块）与单片机通讯时唯一的“握手”信号。状态字寄存器向单片机表示了 LCM（液晶显示模块）当前的工作状态。 尤其是状态字中的“忙”标志位是单片机在每次对 LCM（液晶显示模块）访问时必须要读出判别的状态位。当处于“忙”标志位时， I/O 缓冲器被封锁，此时单片机对 LCM（液晶显示模块）的任何操作（除读状态字操作外）都将是无效的。

7） X 地址寄存器

X 地址寄存器是一个三位页地址寄存器，其输出控制着 DDRAM 中 8 个页面的选 择，也是控制着数据传输通道的八选一选择器。 X 地址寄存器可以由单片机以指令形式设置。X 地址寄存器没有自动修改功能，所以要想转换页面需要重新设置 X 地址寄存器的内容。

8） Y 地址计数器

Y 地址计数器是一个 6 位循环加一计数器。它管理某一页面上的 64 个单元。 Y 地址计数器可以由单片机以指令形式设置，它和页地址指针结合唯一选通显示存储器的一个单元， Y 地址计数器具有自动加一功能。在显示存储器读/写操作后 Y 地址计数将自动加一。当计数器加至 3FH 后循环归零再继续加一。

9） Z 地址计数器

Z 地址计数器是一个 6 位地址计数器，用于确定当前显示行的扫描地址。 Z 地址计数器具有自动加一功能。它与行驱动器的行扫描输出同步，选择相应的列驱动的数据输出。

10）显示起始行寄存器

显示起始行寄存器是一个 6 位寄存器，它规定了显示存储器所对应显示屏上第一行的行号。该行的数据将作为显示屏上第一行显示状态的控制信号。

11）显示开/关触发器

显示开/关触发器的作用就是控制显示驱动输出的电平以控制显示屏的开关。

软件功能说明

1.3.1 指令表

指令名称	控制信号		控制代码							
	RS	R/W	D7	D6	D5	D4	D3	D2	D1	D0
显示开关设置	0	0	0	0	1	1	1	1	1	D
显示起始行设置	0	0	1	1	L5	L4	L3	L2	L1	L0
页面地址设置	0	0	1	0	1	1	1	P2	P1	P0

列地址设置	0	0	0	1	C5	C4	C3	C2	C1	C0
读取状态字	0	1	BUSY	0	ON/OFF	RESET	0	0	0	0
写显示数据	1	0	数据							
读显示数据	1	1	数据							

1）读状态字

BUSY=1 表示 LCM 正在处理单片机发过来的指令或数据。 此时接口电路被封锁，不能接受除读状态字以外的任何操作。 BUSY=0 表示 LCM 接口控制电路已外于“准备好”状态，等待单片机的访问。在指令设置和数据读写时要注意状态字中的 BUSY 标志。只有在 BUSY=0 时，单片机对 LCM 的操作才能有效。因此单片机在每次对 LCM 操作之前，都要读出状态字判断 BUSY 是否为“0”。若不为“0”，则单片机需要等待，直至 BUSY=0 为止。

ON/OFF 表示当前的显示状态。 ON/OFF 1 表示关显示状态， ON/OFF 0 表示开显示状态。

RES 为低电平状态时，LCM 处于复位工作状态，标志位 RESET=1。

2)显示开关设置

该指令设置显示开/关触发器的状态， D 位为显示开/关的控制位。当 D=1 为开显示设置，显示数据锁存器正常工作，此时在状态字中 ON/OFF=0。当 D=0 为关显示设置，显示数据锁存器被置零，显示屏呈不显示状态，但显示存储器并没有被破坏，在状态字中 ON/OFF=1。

3)显示起始行设置

该指令设置了显示起始行寄存器的内容。 LCM 通过/CS 的选择分别具有 64 行显示的管理能力，该指令中 L5～L0 为显示起始行的地址，取值在 0～3FH（1～64 行）范围内，它规定了显示屏上最顶一行所对应的显示存储器的行地址。

4)页面地址设置

该指令设置了页面地址 X 地址寄存器的内容。 LCM 将显示存储器分成 8 页，指令代码中 P2～P0 就是要确定当前所要选择的页面地址，取值范围为 0～7H ，代表 1～8 页。该指令规定了以后的读/写操作将在哪一个页面上进行。

5)列地址设置

该指令设置了 Y 地址数计数器的内容， LCM 通过/CS 的选择分别具有 64 列显示的管理能力， C5～C0=0～3FH（1～64）代表某一页面上的某一单元地址， 随后的一次读或

写数据将在这个单元上进行。 Y 地址计数器具有自动加一功能，在每一次读/写数据后它将自动加一，所以在连续进行读/写数据时， Y 地址计数器不必每次都设置一次。页面地址的设置和列地址的设置将显示存储器单元唯一地确定下来，为后来的显示数据的读/写作了地址的选通。

6)写显示数据

该操作将 8 位数据写入先前已确定的显示存储器的单元内。操作完成后列地址计数器自动加一。

7)读显示数据

该操作将 LCM 接口部的输出寄存器内容读出，然后列地址计数器自动加一。

控制时序表

CS1	CS2	RS	R/W	E	DB7~DB0	功能
X	X	X	X	0	高阻	总线释放
1	1	0	0	下降沿	输入	写指令代码
1	1	0	1	1	输出	读状态字
1	1	1	0	下降沿	输入	写显示数据
1	1	1	1	1	输出	读显示数据

1.3.3 DDRAM 表

下图表示了内部显存的组织方式，也就是要显示的数据需要依照的格式。

	CS1=1					CS2=1					
Y=	0	1	...	62	63	0	1	...	62	63	行号
X=0 ↓ X=7	DB0 ↓ DB7	DB0 ↓ DB7	DB0 ↓ DB7	DB0 ↓ DB7	DB0 ↓ DB7	DB0 ↓ DB7	DB0 ↓ DB7	DB0 ↓ DB7	DB0 ↓ DB7	DB0 ↓ DB7	0 ↓ 7
	DB0 ↓ DB7	DB0 ↓ DB7	DB0 ↓ DB7	DB0 ↓ DB7	DB0 ↓ DB7	DB0 ↓ DB7	DB0 ↓ DB7	DB0 ↓ DB7	DB0 ↓ DB7	DB0 ↓ DB7	0 ↓ 7
	DB0 ↓ DB7	DB0 ↓ DB7	DB0 ↓ DB7	DB0 ↓ DB7	DB0 ↓ DB7	DB0 ↓ DB7	DB0 ↓ DB7	DB0 ↓ DB7	DB0 ↓ DB7	DB0 ↓ DB7	0 ↓ 7

九、思考题

（1） 进行精确的延时的程序有几种方法？各有什么优缺点？

实现延时通常有两种方法：一种是硬件延时，要用到定时器/计数器，这种方法可以提高 CPU 的工作效率，也能做到精确延时；另一种是软件延时，这种方法主要采用循环体进行。

1　　使用定时器/计数器实现精确延时

单片机系统一般常选用 11.059　2　MHz、12　MHz 或 6　MHz 晶振。第一种更容易产生各种标准的波特率，后两种的一个机器周期分别为 1　μs 和 2　μs，便于精确延时。若定时器工作在方式 2，则可实现极短时间的精确延时；如使用其他定时方式，则要考虑重装定时初值的时间（重装定时器初值占用 2 个机器周期）。

在实际应用中，定时常采用中断方式，如进行适当的循环可实现几秒甚至更长时间的延时。使用定时器/计数器延时从程序的执行效率和稳定性两方面考虑都是最佳的方案。但应该注意，C51 编写的中断服务程序编译后会自动加上 PUSH　ACC、PUSH　PSW、POP　PSW 和 POP　ACC 语句，执行时占用了 4 个机器周期；如程序中还有计数值加 1 语句，则又会占用 1 个机器周期。这些语句所消耗的时间在计算定时初值时要考虑进去，从初值中减去以达到最小误差的目的。

2　软件延时与时间计算

在很多情况下，定时器/计数器经常被用作其他用途，这时候就只能用软件方法延时。下面介绍几种软件延时的方法。

2.1　短暂延时

可以在 C 文件中通过使用带_NOP_( )语句的函数实现，定义一系列不同的延时函数，如 Delay10us( )、Delay25us( )、Delay40us( )等存放在一个自定义的 C 文件中，需要时在主程序中直接调用。如延时 10 μs 的延时函数可编写如下：

```
void  Delay10us( )  {

_NOP_( );

_NOP_( );

_NOP_( );

_NOP_( );
```

```
    _NOP_(  );

    _NOP_(  );

    }
```

Delay10us( )函数中共用了 6 个_NOP_( )语句，每个语句执行时间为 1 μs。主函数调用 Delay10us( )时，先执行一个 LCALL 指令（2 μs），然后执行 6 个_NOP_( )语句（6 μs），最后执行了一个 RET 指令（2 μs），所以执行上述函数时共需要 10 μs。可以把这一函数当作基本延时函数，在其他函数中调用，即嵌套调用，以实现较长时间的延时；但需要注意，如在 Delay40us( )中直接调用 4 次 Delay10us( )函数，得到的延时时间将是 42 μs，而不是 40 μs。这是因为执行 Delay40us( )时，先执行了一次 LCALL 指令（2 μs），然后开始执行第一个 Delay10us( )，执行完最后一个 Delay10us( )时，直接返回到主程序。依此类推，如果是两层嵌套调用，如在 Delay80us( )中两次调用 Delay40us( )，则也要先执行一次 LCALL 指令（2 μs），然后执行两次 Delay40us( )函数（84 μs），所以，实际延时时间为 86 μs。简言之，只有最内层的函数执行 RET 指令。该指令直接返回到上级函数或主函数。如在 Delay80 μs( )中直接调用 8 次 Delay10us( )，此时的延时时间

为 82 μs。通过修改基本延时函数和适当的组合调用，上述方法可以实现不同时间的延时。

注意：计算时间时还应加上函数调用和函数返回各 2 个机器周期时间。

十、实验代码

```
//字模方式：列行式，逆向，16*16
#include<reg52.h>
#include<intrins.h> //声明本征函数库
#include<math.h>

typedef unsigned char uchar;
typedef unsigned int uint;

sbit s1 = P3^6;
sbit s2 = P3^7;
sbit RS=P3^5;//寄存器选择信号
sbit RW=P3^4;//读/写操作选择信号，高电平读，低电平写
sbit EN=P3^3;//使能信号
sbit CS1=P1^7;//左半屏显示信号，低电平有效
sbit CS2=P1^6;//右半屏显示信号，低电平有效
sbit DQ=P1^4;
sbit up=P1^1;

uchar Ek,Ek1,Ek2;
uchar Kp,Ki,Kd;
uint res,Pmax;
uint xx=0; //页面
uint times=0;//延时计数
void delay_us(uchar n)
{
  while (n--)
  {
  _nop_();
  _nop_();
  }
}
unsigned    char code shu[10][32]={
```

```
0x00,0x00,0xC0,0xE0,0x30,0x10,0x08,0x08,0x08,0x08,0x08,0x18,0x30,0xE0,0xC0,0x00,
0x00,0x00,0x07,0x0F,0x18,0x10,0x20,0x20,0x20,0x20,0x20,0x10,0x18,0x0F,0x07,0x00,/*"0",0*
/

0x00,0x00,0x00,0x10,0x10,0x10,0x10,0xF0,0xF8,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x20,0x20,0x20,0x20,0x3F,0x3F,0x20,0x20,0x20,0x20,0x00,0x00,0x00,/*"1",1*
/

0x00,0x00,0x60,0x50,0x10,0x08,0x08,0x08,0x08,0x08,0x08,0x98,0xF0,0x70,0x00,0x00,
0x00,0x00,0x20,0x30,0x28,0x28,0x24,0x24,0x22,0x22,0x21,0x20,0x30,0x18,0x00,0x00,/*"2",2*
/

0x00,0x00,0x30,0x30,0x08,0x08,0x88,0x88,0x88,0x88,0x58,0x70,0x30,0x00,0x00,0x00,
0x00,0x00,0x18,0x18,0x20,0x20,0x20,0x20,0x20,0x20,0x31,0x11,0x1F,0x0E,0x00,0x00,/*"3",3*
/

0x00,0x00,0x00,0x00,0x00,0x80,0x40,0x20,0x10,0xF0,0xF8,0xF8,0x00,0x00,0x00,0x00,
0x00,0x04,0x06,0x05,0x05,0x04,0x24,0x24,0x24,0x3F,0x3F,0x3F,0x24,0x24,0x24,0x00,/*"4",4*/

0x00,0x00,0x00,0xC0,0x38,0x88,0x88,0x88,0x88,0x88,0x88,0x88,0x08,0x08,0x00,0x00,
0x00,0x00,0x18,0x29,0x21,0x20,0x20,0x20,0x20,0x20,0x30,0x11,0x1F,0x0E,0x00,0x00,/*"5",5*
/

0x00,0x00,0x80,0xE0,0x30,0x10,0x98,0x88,0x88,0x88,0x88,0x88,0x98,0x10,0x00,0x00,
0x00,0x00,0x07,0x0F,0x19,0x31,0x20,0x20,0x20,0x20,0x20,0x20,0x11,0x1F,0x0E,0x00,/*"6",6*
/

0x00,0x00,0x30,0x18,0x08,0x08,0x08,0x08,0x08,0x88,0x48,0x28,0x18,0x08,0x00,0x00,
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x38,0x3E,0x01,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,/*"7",7*
/

0x00,0x00,0x70,0x70,0xD8,0x88,0x88,0x08,0x08,0x08,0x08,0x98,0x70,0x70,0x00,0x00,
0x00,0x0C,0x1E,0x12,0x21,0x21,0x20,0x21,0x21,0x21,0x23,0x12,0x1E,0x0C,0x00,0x00,/*"8",8
*/

0x00,0xE0,0xF0,0x10,0x08,0x08,0x08,0x08,0x08,0x08,0x18,0x10,0xF0,0xC0,0x00,0x00,
0x00,0x00,0x11,0x33,0x22,0x22,0x22,0x22,0x22,0x32,0x11,0x1D,0x0F,0x03,0x00,0x00,/*"9",9*
/



}
void delay(uint i)//延时子程序,i 最大 256，超过 256 部分无效
{
```

```
    while(--i);
}
void Read_busy()   //等待 BUSY=0
{          //busy p2^7
    P2=0xff;
    RS=0;//RS/RW=0/1,读取状态字指令
    RW=1;
    EN=1;//控制 LCM 开始读取
    while(P2&0x80);//判忙，循环等待 P2.7=0.
    EN=0;//控制 LCM 读取结束
}

void write_command(uchar value)//设置地址或状态
{
   P2=0xff;
   Read_busy();//等待 LCM 空闲
    RS=0;//RS/RW=00,设置 LCM 状态或选择地址指令
    RW=0;
    P2=value;//设置
    EN=1;//控制 LCM 开始读取
    delay(100);
    EN=0;//控制 LCM 读取结束
}

void write_data(uchar value)//写数据到显示存储器
{
   P2=0xff;
   Read_busy();
    RS=1;// RS/RW=10，写数据指令
    RW=0;
    P2=value;//写数据
    EN=1;
    delay(100);
    EN=0;
}

void Set_column(uchar column)//选择列地址（Y）
{
  column=column&0x3f;//高两位清 0，保留后六位的列地址
  column=0x40|column;//01000000|column,根据后六位选择列地址
  write_command(column);
}

void Set_line(uchar startline)//显示起始行设置
```

```
{
  startline=0xC0|startline;// 11000000|startline, 根据 startline 后六位选择起始行
  write_command(startline);
}

void Set_page(uchar page)//选择页面地址（X）
{
  page=0xb8|page;//10111000|page，根据 page 后三位确定所选择的页
  write_command(page);
}
void display(uchar ss,uchar page,uchar column,uchar *p)
{//ss 选择屏幕，page 选择页面，column 选择列，P 是要显示的数据数组的指针
     uchar i;
     switch(ss)
  {
   case 0: CS1=1;CS2=1;break; //全屏
     case 1: CS1=1;CS2=0;break; //左半屏
     case 2: CS1=0;CS2=1;break; //右半屏
     default: break;
  }
  page=0xb8|page;//10111000|page，根据 page 后三位确定所选择的页
  write_command(page);

  column=column&0x3f;//高两位清 0，保留后六位的列地址
  column=0x40|column;//01000000|column,根据后六位选择列地址
  write_command(column);
  for(i=0;i<16;i++)//列地址自动+1
  {
    write_data(p[i]);//写前 16 个长度数据
  }
  page++;
  write_command(page);
//   column--;
  write_command(column);
  for(i=0;i<16;i++)//列地址自动+1
  {
    write_data(p[i+16]);//写后 16 个长度数据
  }
}
void SetOnOff(uchar onoff)//显示开关设置
{
  onoff=0x3e|onoff;//00111110|onoff，根据最后一位设置开/关触发器状态，从而控制显示屏
的显示状态
  write_command(onoff);
```

```
}
void ClearScreen()//清屏
{
   uchar i,j;
   CS2=1;
   CS1=1;
   for(i=0;i<8;i++)
   {
      Set_page(i); //依次选择 8 个页面
       Set_column(0);//选择第 0 列
      for(j=0;j<64;j++)//列地址具有自动加 1 的功能，依次对页面的 64 列写入 0 从而清屏
      {
       write_data(0x00);
      }
   }
}

void InitLCD()//初始化
{
    Read_busy();
     CS1=1;CS2=1;
     SetOnOff(0);
    CS1=1;CS2=1;
     SetOnOff(1);//打开显示开关
     CS1=1;CS2=1;
    ClearScreen();//清屏
    Set_line(0);//设置显示起始行
}

bit DS_init()
{
      bit flag;
      DQ = 0;
      delay_us(255);     //500us 以上
      DQ = 1;                 //释放
      delay_us(40);      //等待 16~60us
      flag = DQ;
      delay_us(150);
      return flag;        //成功返回 0
}
uchar read()     //byte
{
       uchar i;
       uchar val = 0;
```

```
        for (i=0; i<8; i++)
        {
              val >>= 1;
              DQ = 0; //拉低总线产生读信号
              delay_us(1);
              DQ = 1; //释放总线准备读信号
              delay_us(1);
              if (DQ) val |= 0x80;
              delay_us(15);
       }
  return val;
}
void write(char val)      //byte
{
  uchar i;

  for (i=0; i<8; i++)
  {
  DQ = 0; //拉低总线,产生写信号
  delay_us(8);
  val >>= 1;
  DQ = CY;
  delay_us(35);
  DQ = 1;
  delay_us(10);
  }
}

void PID()
{
      uchar Px,Pp,Pi,Pd,a,b,c;
      uint count;
      Pp = Kp*(Ek-Ek1);
      Pi = Ki*Ek;
      Pd = Kd*(Ek-2*Ek1+Ek2);
      Px = Pp+Pi+Pd;
      res = Px;
            a=res/100;
            b=res%100/10;
            c=res%10;

            display(1,4,2*16,shu[a]);delay(255);
            display(1,4,3*16,shu[b]);delay(255);
            display(2,4,0*16,shu[c]);delay(255);
```

```
        Ek2 = Ek1;
        Ek1 = Ek;
        count = 0;
        if(res>Pmax)
            res =Pmax ;
        while((count++)<=res)
        {
            up = 1;
            delay_us(250);
            delay_us(250);
        }
        while((count++)<=Pmax)
        {
            up = 0;
            delay_us(250);
            delay_us(250);
        }

}

void main()
{
        uchar aim,low,high,b,c;
        uint result;
        InitLCD();
        Set_line(0);
        aim = 40;
        Kp = 4;
        Ki = 5;
        Kd = 2;
        Pmax = 5;
        Ek1 = 0;
        Ek2 = 0;
        res = 0;

        while(1)
        {
            if(s1 == 0)
                aim++;
            if(s2 == 0)
                aim--;
            while(DS_init());
            write(0xcc);    //跳过 ROM 命令
            write(0x44);    //温度转换命令
```

```
            delay(600);
            while(DS_init());
            write(0xcc);
          write(0xBE);        //读 DS  温度暂存器命令
            low = read(); //采集温度
            high = read();
            delay(255);
            result = high;
            result <<= 8;
            result |= low;
            result >>= 4 ; //result /= 16;

            Ek = aim - result;

            b=result/10;
            c=result%10;

            display(1,0,0*16,shiji[0]);delay(255);
            display(1,0,1*16,shiji[1]);delay(255);
            display(1,0,3*16,shu[b]);delay(255);
            display(2,0,0*16,shu[c]);delay(255);
            display(2,0,1*16,du);delay(100);

            b=aim/10;
            c=aim%10;

            display(1,2,0*16,mubiao[0]);delay(255);
            display(1,2,1*16,mubiao[1]);delay(255);
            display(1,2,3*16,shu[b]);delay(255);
            display(2,2,0*16,shu[c]);delay(255);
            display(2,2,1*16,du);delay(100);
            if(aim>=result)
                  PID();
            else
                  up = 0;
      }
}
```