实验五 重量测量（液晶显示）

一、实验内容

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。

二、实验设备

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

三、实验步骤

（1）简单程序录入和调试：使用 Keil 对测试用例《重量测量.hex》进行测试，出现现象“称重砝码时，显示屏上出现对应重量”现象。
（2）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；程序调试：编写简单的汇编程序，完成程序正常执行
（4）调试程序（调零，满量程校准；将编译后的程序下载到 51 单片机；在托盘中放上相应重量的法码，使显示值为正确重量。）

四、实验原理

一．液晶显示屏

数据送到LED显示屏：芯片开始转换工作时，由处理器向芯片时钟输入端CLK 输入时钟脉冲，DO/DI 端则使用DI 端输入通道功能选择的数据信号。片选信号CS1，CS2控制左右两屏，当为高电平禁用，低电平开启；（在第1 个时钟脉冲的下沉之前DI端必须是高电平.形成两次下降沿，确定取值，形成第三次下降沿）

8位寄存器屏页列。

1.本实验平台使用一个集成的液晶显示屏驱动芯片 YM12864C。它主要采用动态驱动原理由行驱动控制器和列驱动器两部分组成了 128(列)×64(行)的全点阵液晶显示。YM12864C 是全屏幕点阵,点阵数为 128(列)×64(行),可显示 8(每行)×4(行)个(16×16 点阵)汉字， 也可完成图形，字符的显示。与 CPU 接口采用 5 条位控制总线和 8 位并行数据总线输入输出，适配 M6800 系列时序。内部有显示数据锁存器，自带上电复位电路。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。

2.命令：

1）读状态字： 状态字是单片机了解 LCM（液晶显示模块）当前状态，或 LCM 向单片机提供其内部状态的唯一的信息渠道。BUSY 表示当前 LCM 接口控制电路运行状态。BUSY=1 表示 LCM 正在处理单片机发过来的指令或数据。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。ON/OFF 表示当前的显示状态。RESET 表示当前 LCM 的工作状态， 即反映/RES 端的电平状态。

2)显示开关设置：该指令设置显示开/关触发器的状态，由此控制显示数据锁存器的工作方式，从而控制显示屏上的显示状态。D 位为显示开/关的控制位。

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。

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。

二．A/D 转化

①A/D 转化原理：

1)如何转化：重量传感器采用压敏电阻。利用压敏电阻采集应变,产生变化的阻值。利用放大电路将其转化为电压值，通过数模转换将电压值转化成 CPU 处理的数字信号。传感器根据编制的程序将数字信号转换为砝码重量显示输出。

2）工作方式：逐次逼近：对于我们得到的模拟量，首先取产生的8位数字量的一半，如果得到的模拟量在低位，则清空高位寄存器，以此类推，类似二分法。

②A\D 具体构造

1.A/D 转换口在 P1 口((P1.7-P1.0) ，有 8 路 10 位高速 A/D 转换器 ，由多路选择开关、比较器、逐比较寄存器、10 位 DAC、转换结果寄存器(ADC_RES 和 ADC_RESL)以及 ADC_CONTR 构成。

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

2.寄存器：

1）ADC 控制寄存器 ADC_CONTR

2）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，当 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。

3.**注：**由于是 2 套时钟,所以,设置 ADC_CONTR 控制寄存器后,要加 4 个空操作延时才可以正确读到 ADC_CONTR 寄存器的值,原因是设置 ADC_CONTR 控制寄存器的语句执行后,

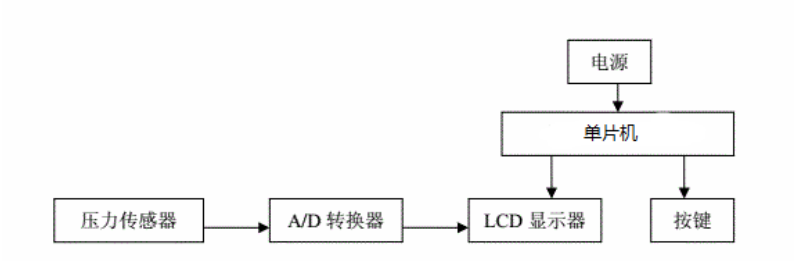
要经过 4 个 CPU 时钟的延时,其值才能够保证被设置进 ADC_CONTR 控制寄存器.

五、实验过程分析（附源代码）

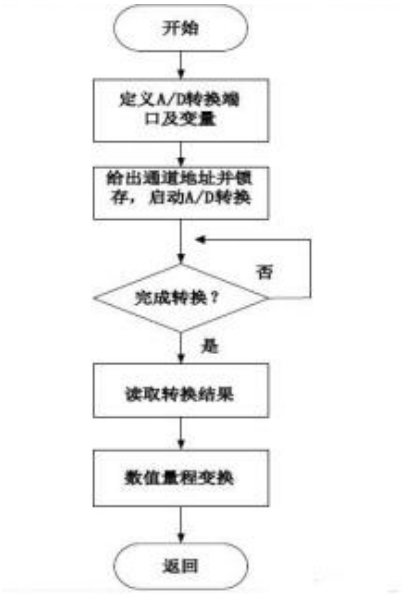
1. 设计思路：

初始化液晶显示屏和 AD 转化，并进行软硬件调零。显示器通过 send-byte 写入命令，send-all 写入字符，ADC 读入数据后，根据 AD 工作原理转化为十进制，根据每台设备的不同，进行测试，并在液晶显示屏上显示相应的砝码重量。

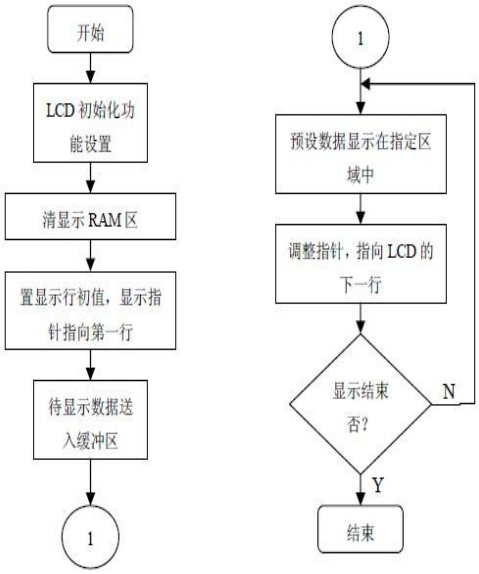
2. 程序框图：



A/D 转换部分：



LCD 液晶显示：



3. 源代码

```
#include <reg52.h>
#include <intrins.h>
#define uchar unsigned char
#define uint unsigned int
//液晶屏相关设置
sbit CS1=P1^7;//选屏左半部
sbit CS2=P1^6;//选屏右半部
sbit E=P3^3;//使能
sbit RW=P3^4;//读写选择
sbit RS=P3^5;//寄存器选择
sbit RES=P1^5;//复位
```

```
sbit BUSY=P2^7;//数据总线
//ADC 寄存器选择
sfr ADC_CONTR = 0xBC;    ///ADC control registerAD
sfr ADC_RES = 0xBD;    ///ADC hight 8-bit result registerAD
sfr ADC_LOW2 = 0xBE;    ///ADC low 2-bit result register
sfr P1ASF = 0x9D;    ///P1 secondary function control
sfr AURX1 = 0xA2; ///AURX1 与 ADRJ
#define ADC_POWER 0x80    ///ADC power control bit
#define ADC_FLAG 0x10    ///ADC complete flag
#define ADC_START 0x08    ///ADC start control bit
#define ADC_SPEEDLL 0x00    ///540 clocks
#define ADC_SPEEDL 0x20    ///360 clocks
#define ADC_SPEEDH 0x40    ///180 clocks
#define ADC_SPEEDHH 0x60    ///90 clocks
uchar ch = 0;    ///ADC channel NO.0
uchar code zima[20][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,
0x00,0x00,0x00,0x00,0x00,0x00,0x38,0x3E,0x01,0x00,0x00,0x00,0x00,0x00,0x00,0x00,///*"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/
0x08,0x08,0x0A,0xEA,0xAA,0xAA,0xAA,0xFF,0xA9,0xA9,0xA9,0xE9,0x08,0x08,0x08,0x00,
0x40,0x40,0x48,0x4B,0x4A,0x4A,0x4A,0x7F,0x4A,0x4A,0x4A,0x4B,0x48,0x40,0x40,0x00,///*"砝"*10/
0x40,0x40,0x40,0xDF,0x55,0x55,0x55,0xD5,0x55,0x55,0x55,0xDF,0x40,0x40,0x40,0x00,
0x40,0x40,0x40,0x57,0x55,0x55,0x55,0x7F,0x55,0x55,0x55,0x57,0x50,0x40,0x40,0x00,///*"码"*11/
0x00,0x00,0x00,0x00,0x00,0x00,0xC0,0xC0,0xC0,0xC0,0x00,0x00,0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,0x00,0x00,0x30,0x30,0x30,0x30,0x00,0x00,0x00,0x00,0x00,0x00,///*"重"*12/
0x00,0x04,0x04,0xE4,0x24,0x24,0x24,0x3F,0x24,0x24,0x24,0xE4,0x04,0x04,0x00,0x00,
0x00,0x00,0x80,0x43,0x31,0x0F,0x01,0x01,0x01,0x3F,0x41,0x43,0x40,0x40,0x70,0x00,///*"克"*13/
};
void send_byte(uchar dat ,uchar cs1,uchar cs2);
void send_all(uint page,uint lie,uint offset);
void delay(uint x);
void init_adc();
void init_yejing();
void calibrate();
int get_ad_result();
void clearscreen();
int cweight;//校准量
int weight;//测量结果
void main()
{
    init_yejing();//液晶屏初始化
    init_adc();//ADC 初始化
    calibrate();//初始校准
```

```
    while(1)
    {
     weight=(get_ad_result()-cweight)/2.05;//测量结果调整
        clearscreen();//清屏
        send_all(1,1,10);//输出重
        send_all(1,2,11);//输出量
        send_all(1,3,12);//输出：
        send_all(4,3,weight/100);//输出百位
        send_all(4,4,(weight/10)%10);//输出十位
        send_all(4,5,weight%10);//输出个位
        send_all(4,6,13);//输出克
        delay(50000);
    }
}
void init_yejing()
{
    send_byte(192,1,1);//设置起始行为 0
    send_byte(63,1,1);//设置开关为 1
}
void send_byte(uchar dat,uchar cs1,uchar cs2)
{
    P2=0xff;
    CS1=cs1; CS2=cs2;
    RS=0; RW=1; E=1;
    while(BUSY) ;//busy 为忙时不读入数据
    E=0;
    RS=!(cs1&&cs2),RW=0;
    P2=dat;
    E=1; delay(3); E=0;
    CS1=CS2=0;
}
void send_all(uint page,uint lie,uint offset)
{
    uint i,j,k=0;
    for(i=0;i<2;++i)
    {
        send_byte(184+i+page,1,1);//page=0xb8|page;//10111000|page,
        send_byte(64+lie*16-(lie>3)*64,1,1);//column=column&0x3f;column=0x40|column;01000000|column
        for(j=0;j<16;++j)
            send_byte(zima[offset][k++],lie<4,lie>=4);//写入数据
    }
}
void init_adc()
{
    P1ASF = 1;//选取通道
    AURX1 |= 0X04;//设置存储数据方式
    ADC_RES = ADC_LOW2 = 0; //存储数据寄存器清零
    ADC_CONTR = ADC_POWER | ADC_SPEEDLL | ADC_START | ch;//寄存器设置
    delay(4);
}
int get_ad_result()
{
    int ADC_result;
    ADC_RES = ADC_LOW2 = 0;
    ADC_CONTR = ADC_POWER | ADC_SPEEDLL | ch | ADC_START;
    _nop_(); _nop_(); _nop_(); _nop_(); _nop_(); _nop_();//延迟读入
    while (!(ADC_CONTR & ADC_FLAG)); //保证数据读入完成
    ADC_result = (ADC_RES & 0x03) *256 + ADC_LOW2;//将数据转化为十进制
    ADC_CONTR &= ~ADC_FLAG;
    return ADC_result;
}
void calibrate()
```

```
{
    cweight=get_ad_result();
}
void delay(uint x)
{
    while(x--) ;
}
void clearscreen()
{
     int i,j;
     for(i=0;i<8;++i)
     {
          send_byte(184+i,1,1);///10111000|page
          send_byte(64,1,1);///01000000|lie
               for(j=0;j<64;++j)
                    {
                         send_byte(0x00,0,1);
                         send_byte(0x00,1,0);
                    }
     }
}
```

六、思考题

1. 调零的原理，软件调零和调零调零的区别。

答：**调零的原理**：在未放上砝码之前，使液晶显示屏显示的重量为 000g,有软件调零和硬件调零两种。软件调零和硬件调零的区别：硬件调零是指在未放砝码时，为了使液晶显示屏初始现实为 000g,通过实验设备配套的工具，调节旋钮实现；而软件调零是指，在不通过硬件调节，而是通过程序实现，使未放置砝码时，液晶显示屏显示 000g。

2. 模/数和数/模的信号转换原理。

答：**模/数和数/模的信号转换原理**：ADC 的转换原理根据的电路形式有所不同。ADC 电路通常由两部分组成，它们是：采样、保持电路和量化、编码电路。其中量化、编码电路是最核心的部件，任何 AD 转换电路都必须包含这种电路。ADC 电路的形式很多，通常可以并为两类：间接法：它是将采样保持的模拟信号先转换成与模拟量成正比的时间或频率，然后再把它转换为数字量。这种通常是采用时钟脉冲计数器，它又被称为计数器式。它的工作特点是：工作速度低，转换精度高，抗干扰能力强。直接法：通过基准电压与采样保持信号进行比较，从而转换为数字量。它的工作特点是：工作速度高，转换精度容易保证。
DA 转换器是由数码寄存器、模拟电子开关电路、解码电路、求和电路及基准电压及部分组成。数字量是以串行或并行方式输入并存储在数码寄存器中，寄存器输出的每位数码驱动对应数位上的电子开关将电阻解码网络中获得的相应数位权值送入数位求和电路中 2，求和电路将各位权值相加就得到与数字量相应的模拟量。

3. I2C 总线在信号通讯过程中的应用。

答：**I2C 总线**在信号通讯过程中的应用：I2C 总线是 Phihps 公司开发的一种双向两线多主机总线，利用两根信号线来实现设备之间的信息传递，一根为数据线 SDA，一根为时钟线 SCL。它能方便地实现芯片间的数据传输与控制。通过两线缓冲接口和内部控制与状态寄存器，可

方便地完成多机间的非主从通信或主从通信。基于 I2C 总线的多机通信电路结构简单、程序编写方便，易于实现系统软硬件的模块化和标准化，被广泛用于系统内部微控制器和外部设备之间的串行通讯。

实验六 直流电机脉宽调制调速

一、 实验内容

编写 C51 程序，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。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；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。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。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。

二、 实验设备

（1）单片机测控实验系统
（2）直流电机调速实验模块
（3）Keil 开发环境
（4）STC-ISP 程序下载工具

三、 实验步骤

（1）简单程序录入和调试：使用 Keil 对测试用例《直流电机调速.hex》进行测试，出现“直流电机偏心轮转动，LCD 屏上显示对应的当前直流电机转速”现象。

（2）使用 C51 语言编写直流电机中断（外部中断）程序，测量电机转速；

（3）完成控制转速程序：编写两个定时器中断程序：定时器 0 采用累加进位法（目的为根据脉宽调制的原理----动态改变 P1.1 的输出，宏观上输出有效（0）的比例，根据调节占空比来改变直流电机转速）；定时器 1 的作用为：每 20ms 检测开关 S1/S2 状态，判断有无调速操作。

（4）调试程序（将编译后的程序下载到 51 单片机；显示屏上出现当前转速、最低转速、最高转速；当按下 S1 时，当前转速向最低转速靠近，并稳定在最低转速区间；当按下 S2S 时，当前转速向最高转速靠近，并稳定。）

四、实验原理

一. 液晶显示屏

数据送到LED显示屏：芯片开始转换工作时，由处理器向芯片时钟输入端CLK 输入时钟

脉冲，DO/DI 端则使用DI 端输入通道功能选择的数据信号。片选信号CS1，CS2控制左右两屏，当为高电平禁用，低电平开启；（在第1 个时钟脉冲的下沉之前DI端必须是高电平.形成两次下降沿，确定取值，形成第三次下降沿）

8位寄存器屏页列。

1.本实验平台使用一个集成的液晶显示屏驱动芯片 YM12864C。它主要采用动态驱动原理由行驱动控制器和列驱动器两部分组成了 128(列)×64(行)的全点阵液晶显示。YM12864C 是全屏幕点阵,点阵数为 128(列)×64(行),可显示 8(每行)×4(行)个(16×16 点阵)汉字， 也可完成图形，字符的显示。与 CPU 接口采用 5 条位控制总线和 8 位并行数据总线输入输出，适配 M6800 系列时序。内部有显示数据锁存器，自带上电复位电路。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。

2.命令：

1）读状态字： 状态字是单片机了解 LCM（液晶显示模块）当前状态，或 LCM 向单片机提供其内部状态的唯一的信息渠道。BUSY 表示当前 LCM 接口控制电路运行状态。BUSY=1 表示 LCM 正在处理单片机发过来的指令或数据。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。 ON/OFF 表示当前的显示状态。RESET 表示当前 LCM 的工作状态， 即反映/RES 端的电平状态。

2)显示开关设置：该指令设置显示开/关触发器的状态，由此控制显示数据锁存器的工作方式，从而控制显示屏上的显示状态。D 位为显示开/关的控制位。

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。

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。

二、直流电机

1.脉宽调制调速的原理与方法：

1）原理：**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。

2）PWM 的应用：在本实验中主要应用脉宽调制调速控制电压调制，从而控制直流电机转速；此外，还可以应用在频率调制等，如应用在蜂鸣器等。

2.中断原理

1）外部中断：采用边沿触发方式（当管脚 INTO 有由高电平变为低电平的过程，便认为有中断请求，EX0 向置高电平，向 CPU 发出中断请求）电机转速就是一秒钟之内 INT0 的中断个数，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。

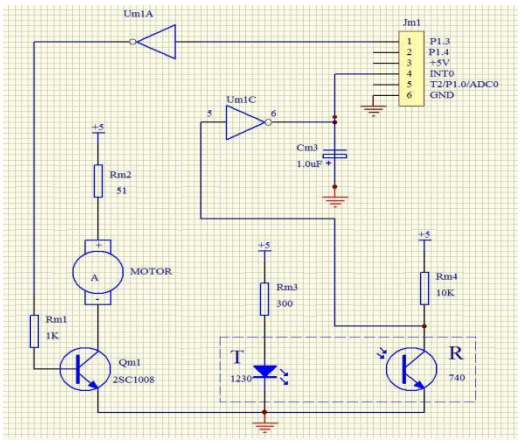
2）定时器中断 0：采用累加进位法（采用比例控制算法），在程序中，每 0.1ms 都会记录当前速度，当转速 S 大于预定值时，将输出 0 的个数减少；当转速小于预定值时，将输出 0 的个数增加。主要用于脉宽调制调速：将高频率的 0/1 输出，在外围积分元件作用下，输出δ ×A（A 为高电平电压）的电压，通过改变占空比来调节电压.

3）定时器中断 1：每 50ms 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。

3.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.电路原理图

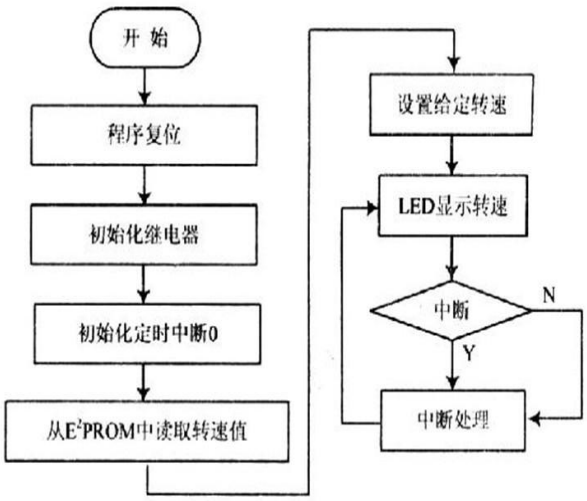


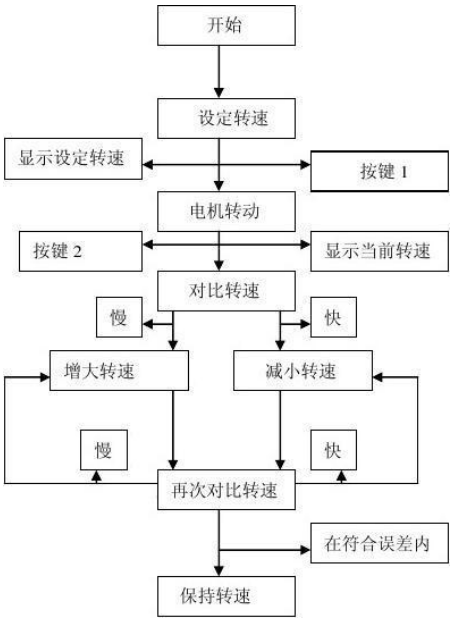
五、实验过程分析（附源代码）

1.设计思路：

程序共分三大部分：数码管显示，液晶屏显示，直流电机控制。前两个部分参考之前的实验。直流电机的控制设计三个中断：外部中断 0，计时器 0，计时器 1，采用累加法调节占空比。

2.程序框图：





3．源代码

```
#include <reg52.h>
#include <intrins.h>
#define uchar unsigned char
#define uint unsigned int
//数码管初始化
sfr P4=0xC0;
sfr P4SW=0xBB;
sbit sclk=P4^4;
sbit sdata=P4^5;
//液晶屏初始化
sbit CS1=P1^7;
sbit CS2=P1^6;
sbit E=P3^3;
sbit RW=P3^4;
sbit RS=P3^5;
sbit RES=P1^5;
sbit BUSY=P2^7;
//直流电机初始化
sbit swh1=P3^6;
sbit swh2=P3^7;
sbit motor=P1^1;
uchar code zima[20][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/
0x00,0x00,0x00,0x00,0x00,0x00,0x80,0x40,0x20,0x10,0xF0,0xF8,0xF8,0x00,0x00,0x00,
0x00,0x04,0x06,0x05,0x05,0x04,0x24,0x24,0x24,0x3F,0x3F,0x3F,0x24,0x24,0x24,0x00,//*"4"*4/
0x00,0x00,0x00,0xC0,0x38,0x88,0x88,0x88,0x88,0x88,0x88,0x08,0x08,0x00,0x00,
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,//*"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/
0x08,0x08,0x0A,0xEA,0xAA,0xAA,0xAA,0xFF,0xA9,0xA9,0xA9,0xE9,0x08,0x08,0x08,0x00,
0x40,0x40,0x48,0x4B,0x4A,0x4A,0x4A,0x7F,0x4A,0x4A,0x4A,0x4B,0x48,0x40,0x40,0x00,//*"?"*10/
```

```
0x40,0x40,0x40,0xDF,0x55,0x55,0x55,0xD5,0x55,0x55,0x55,0xDF,0x40,0x40,0x40,0x00,
0x40,0x40,0x40,0x57,0x55,0x55,0x55,0x7F,0x55,0x55,0x55,0x57,0x50,0x40,0x40,0x00,//*"?"*11/
0x00,0x00,0x00,0x00,0x00,0x00,0xC0,0xC0,0xC0,0xC0,0x00,0x00,0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,0x00,0x00,0x30,0x30,0x30,0x30,0x00,0x00,0x00,0x00,0x00,0x00,//*":"*12/
0x00,0x04,0x04,0xE4,0x24,0x24,0x24,0x3F,0x24,0x24,0x24,0xE4,0x04,0x04,0x00,0x00,
0x00,0x00,0x80,0x43,0x31,0x0F,0x01,0x01,0x01,0x3F,0x41,0x43,0x40,0x40,0x70,0x00,//*"?"*13/
};
uchar tab[15]= {0xC0,0xF9,0xA4,0xB0,0x99,0x92,0x82,0x0F8,0x80,0x90};//0-9
uchar tspeed=0;//脉冲计数
uchar cspeed=0;//当前转速
uchar xspeed=130;//预定转速
uchar speedUp = 160;//最高转速
uchar speedLow =100;//最低转速
uchar t1_cnt=0; ///1s=50ms*20
//占空比设置
int N=50;
int M=256;
int X=0;
void send_byte(uchar dat ,uchar cs1,uchar cs2);
void send_all(uint page,uint lie,uint offset);
void init();
void clearscreen();
void init_yejing();
void sendbyte(uchar ch);
void display(uchar n);
void delay1();
void delay2();
void delay(uint x)
{
    while(x--) ;
}
void main()
{
    init();
    init_yejing();
    motor=0;
    while(1)
    {
      clearscreen();
      send_all(1,3,speedLow/100);//最低值百位
          send_all(1,4,(speedLow/10)%10);//最低值十位
          send_all(1,5,speedLow%10);//最低值个位
      send_all(3,3,cspeed/100);//当前值百位
          send_all(3,4,(cspeed/10)%10);//当前值十位
          send_all(3,5,cspeed%10);//当前值个位
          send_all(5,3,speedUp/100);//最高值百位
          send_all(5,4,(speedUp/10)%10);//最高值十位
          send_all(5,5,speedUp%10);//最高值个位
      delay1();
          display(cspeed);//数码管显示
      delay(50000);
    }
}
//数码管和中断初始化
void init()
{    P4SW=0x30;
    IT0=1;
    EA=1;//中断使能
    ET1=1;//timer1
    ET0=1;//timer0
    EX0=1;//INT0
    TMOD=0x11; //16 位寄存器，模式 1
    TH1=0x3C;
    TL1=0xB0;   //50ms:65536-50000=15536
    TH0=0xFF;
    TL0=0x9C;   //0.1ms:65536-100=65436
    TR0=1;//0
    TR1=1;//1
}
//外部中断 0
```

```
void ex_int0() interrupt 0    ///????INT0
{
      tspeed++;
}
//计时器中断 0
void t0_int() interrupt 1    ///0.1ms
{
      TH0=0xFF;
      TL0=0x9C;
      //累加法
      X+=N;
         if(X>M)
         {
         motor=0;
             X-=M;
         }
         else
             motor=1;
}
//计时器中断 1
void t1_int() interrupt 3      ///50ms
{
      if(++t1_cnt<20)
      {           TH1=0x3C;
                  TL1=0xB0;
                  if(swh1==0)//S1 按下
                  {
                        xspeed = speedLow;
                  }
                  if(swh2==0)//S2 按下
                          {
                        xspeed = speedUp;
                     }
            return;
      }
      t1_cnt=0;
      cspeed=tspeed;
      tspeed=0;
      if(cspeed>xspeed) N--;//降低转速
      if(cspeed<xspeed) N++;//提高转速
}
//液晶屏初始化
void init_yejing()
{
      send_byte(192,1,1);
      send_byte(63,1,1);
}
//送 8 位数
void send_byte(uchar dat,uchar cs1,uchar cs2)
{
      P2=0xff;
      CS1=cs1; CS2=cs2;
      RS=0; RW=1; E=1;
      while(BUSY) ;
      E=0;
      RS=!(cs1&&cs2),RW=0;
      P2=dat;
      E=1; delay(3); E=0;
      CS1=CS2=0;
}
//显示相应字
void send_all(uint page,uint lie,uint offset)
{
      uint i,j,k=0;
      for(i=0;i<2;++i)
      {
            send_byte(184+i+page,1,1);
            send_byte(64+lie*16-(lie>3)*64,1,1);
            for(j=0;j<16;++j)
                  send_byte(zima[offset][k++],lie<4,lie>=4);
```

```
    }
}
//清屏
void clearscreen()
{
    int i,j;
    for(i=0;i<8;++i)
    {
        send_byte(184+i,1,1);
        send_byte(64,1,1);
            for(j=0;j<64;++j)
                {
                    send_byte(0x00,0,1);
                    send_byte(0x00,1,0);
                }
    }
}
//数码管显示 1 个数
void sendbyte(uchar ch)
{
    uchar shape,c;
    shape=tab[ch];
    for(c=0;c<8;c++)
    {
        sclk=0;
        sdata=shape & 0x80;
        sclk=1;
        shape <<= 1;
    }
}
//数码管显示
void display(uchar n)
{
    sendbyte(n%10);
    sendbyte((n/10)%10);
    sendbyte(n/100);
}
void delay1()
{
    int i,j;
    for(i=0;i<1000;i++)
        for(j=0;j<500;j++);
}
void delay2()
{
    int i,j;
    for(i=0;i<1000;i++)
        for(j=0;j<1000;j++);
}
```

六、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。

答：脉宽，其实就是指脉冲的宽度。开和关的时间比值就可以认为是脉冲的占空比，开的时间长，相应的关的时间就会缩短（每秒必须完成一次开和关，相当于脉冲的频率）。脉宽调速，实质上也是电压调速，因脉宽调制的输出，经滤波，续流，供给电机的也是连续的(可调)直流电压，所以也叫脉宽调压，对电机没有什么机械损伤，但要加滤波和续流电路。脉宽调速不需要在计算机接口中使用 D/A 转换器，基本原理是使用具有一定占空比的方波来模拟对应的电压值。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。脉宽调速可大大节省电量，具有很强的抗噪性，且节约空间、比较经济，适用于低频大功率控制。

2. 说明程序原理中累加进位法的正确性。
答：累加进位法相当于每输出 M/N-1 次 1，就输出一个 0，相当于输出的 0:1 为 N:（M-N），与占空比一致，所以累加进位法是正确的，并且实现了将总的周期内的 0 和 1 均匀分散开。

3. 计算转速测量的最大可能误差，讨论减少误差的办法。
答：转速变化为 M/(N-1)-M/N = M/(N*(N-1))，所以实际转速与预期转速之间存在误差为 M/(N*(N-1))。为了减小误差，可以增大 M 或减小。

实验七　温度测量与控制

一、实验内容

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。

二、实验设备

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

三、实验步骤

（1）简单程序录入和调试：使用 Keil 对测试用例《超声波测距-温度测控.hex》进行测试。
（2）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使用 C51 语言进行编程。
（3）将编译后的程序下载到 51 单片机，观察温度的测量结果。
（4）调试程序。

四、实验原理

一、DS18B20 温度传感器的编程结构

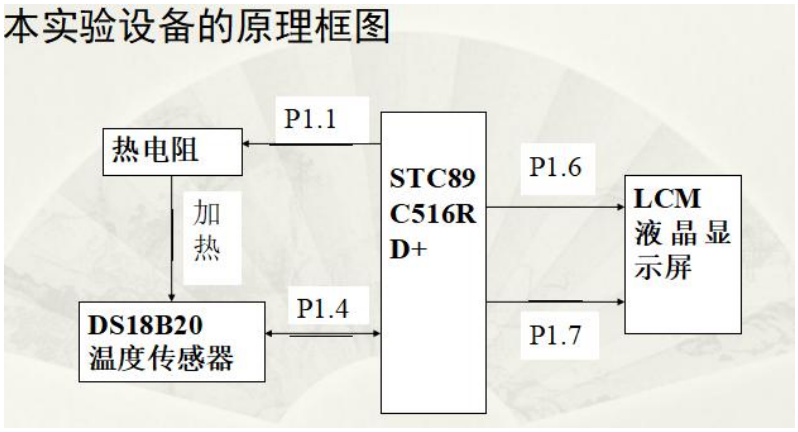
本实验使用的 DS18B20 是单总线数字温度计，测量范围从—55℃到+125℃，增量值为 0.5 ℃。

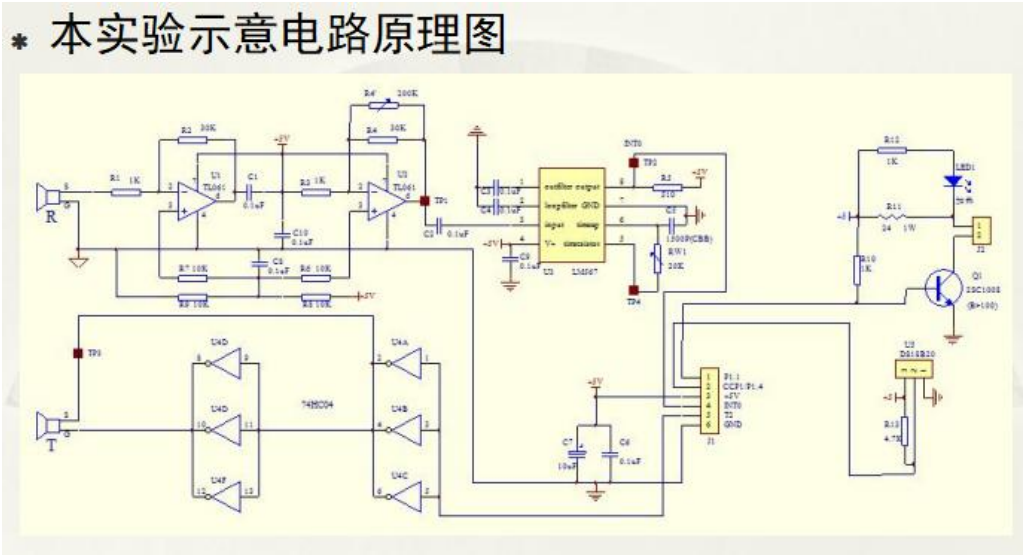
用于贮存测得的温度值的两个 8 位存贮器 RAM 编号为 0 号和 1 号。

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。





二、了解温度测量的原理

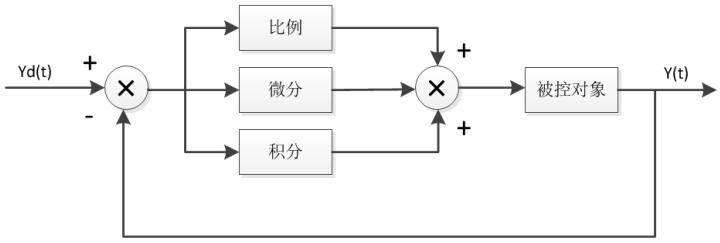
温度检测与控制系统由加热灯泡，温度二极管，温度检测电路，控制电路和继电器组成。温度二极管和加热灯泡封闭在一个塑料保温盒内，温度二极管监测保温盒内的温度，用温控实验板内部的 A/D 转换器 ADC7109 检测二极管两端的电压，通过电压和温度的关系，计算出盒内空气的实际温度。

DS 用一个高温度系数的振荡器确定一个门周期，内部计数器在这个门周期内对一个低温度系数的振荡器的脉冲进行计数来得到温度值。

三、PID 控制原理及实现方法

PID 是一种控制算法：PID 能够做到在温度快要达到设定值的时候降低加热功率，让温度上升速度变慢，最终稳定在设定值。

1. 比例部分：当前温度小于给定温度时，误差为正，增大电位器夹角，即增大加热的电流。当前温度大于给定温度时，误差为负，减小电位器夹角，即减小加热的电流。
2. 积分控制：主要用于采样周期的选择。误差为正，增大积分量；误差为负，减小积分量。
3. 微分部分：当温度上升过快，但尚未达到预定的目标值时候，就会采用超调，提前减小加热的电流，使得在快接近目标值时缓慢趋近。



模拟PID控制系统原理框图

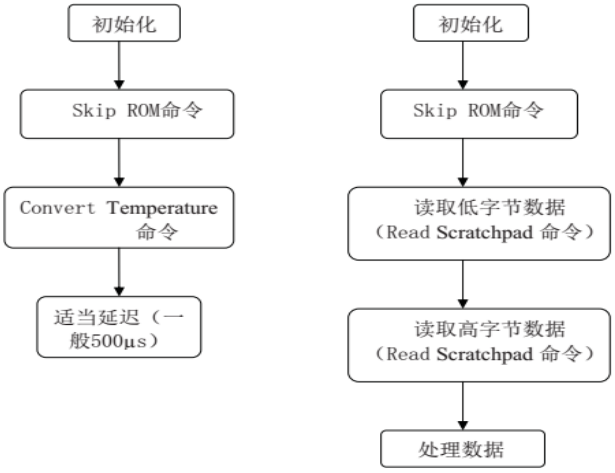
四. 数据处理部分

1. **存储数据**：DS18B20 高速暂存存储器共有 9 个字节。当温度转换命令发布后，经转换所得的温度以二字节补码形式存放在高速暂存存储器的第 0 和第 1 个字节。这是 12 位精度转换后得到的 16 位数据，其中前面 5 位为符号位。如果测得的温度大于等于 0 则为 0，只要将测得的数值乘于 0.0625 即可得到实际温度。单片机可通过单线接口读到该数据，低位在前，高位在后。2 个八位的 RAM，0-3 小数，4-10 整数，11-15 温度符号

2.**写操作**：通过单总线采取移位的方式来向DS18B20写入数据，按照从低到高位的顺序每次一位的方式写进去，需要满足写时间间隙的要求。整个位的发送时间应保持在60~120us。（当主机把数据线从逻辑高电平拉到逻辑低电平的时候，写时间隙开始。有两种写时间隙：写1 时间隙和写0 时间隙。所有写时间隙必须最少持续60μ s，包括两个写周期间至少1μ s 的恢复时间。

I/O 线电平变低后，DS1820 在一个15μ s 到60μ s 的窗口内对I/O 线采样。如果线上是高电平，就是写1，如果线上是低电平，就是写0）

3.**读操作**：同写操作。必须在读间隙开始的 15us 内读取数据位才可以保证通信的正确。主机在读时间隙开始后必须停止把 I/O 脚驱动为低电平 15μ s。

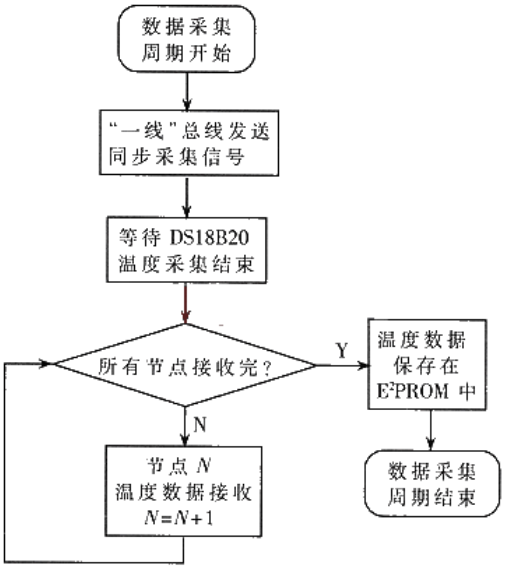


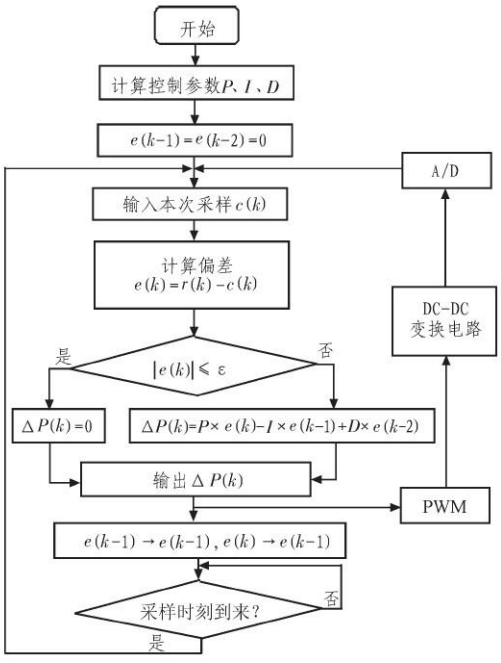
五、 实验过程分析（附源代码）

一．设计思路：

LCM 部分参照以前的程序，温度控制方面参考附录编写读写命令，将读入数据处理后转为温度，与目标值比较，并做相应处理。

二．程序框图：





三． 源代码

```
#include<reg52.h>
#include<intrins.h>
#include<math.h>
typedef unsigned char uchar;
typedef unsigned int uint;
//开关
sbit s1 = P3^6;
sbit s2 = P3^7;
//LCM
```

```
sbit RS=P3^5;
sbit RW=P3^4;
sbit EN=P3^3;
sbit CS1=P1^7;
sbit CS2=P1^6;
//温度
sbit DQ=P1^4;//数据和命令传输
sbit up=P1^1;//连接热电阻
uchar Ek,Ek1,Ek2;//Ek=e(k),Ek1=e(k-1),Ek2=e(k-2)
uchar Kp,Ki,Kd;//Kp 为比例系数，Ki 为积分系数，Kd 为微分系数
uint res,Pmax;
void delay_us(uchar n)
{
  while (n--)
  {
  _nop_();
  _nop_();
  }
}
unsigned    char code shu[10][32]={

{0x00,0x00,0x00,0xF8,0xFC,0x06,0x02,0x02,0x02,0x02,0x06,0xFC,0xF8,0x00,0x00,0x00,0x00,0x00,0x00,0x1F,0x3F,0x60,0x40,0x
40,0x40,0x40,0x60,0x3F,0x1F,0x00,0x00,0x00},/*"0",0*/

{0x00,0x00,0x00,0x00,0x00,0x08,0x0C,0xFE,0xFE,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x40,0x40,0x
7F,0x7F,0x40,0x40,0x00,0x00,0x00,0x00,0x00},/*"1",1*/

{0x00,0x00,0x00,0x18,0x1C,0x06,0x02,0x02,0x82,0x82,0x86,0xFC,0x78,0x00,0x00,0x00,0x00,0x00,0x00,0x78,0x7C,0x46,0x43,0
x41,0x41,0x40,0x40,0x70,0x70,0x00,0x00,0x00},/*"2",2*/

{0x00,0x00,0x00,0x0C,0x0E,0x02,0x02,0x82,0x82,0xC2,0x62,0x3E,0x1C,0x00,0x00,0x00,0x00,0x00,0x00,0x30,0x70,0x40,0x40,0
x40,0x40,0x41,0x63,0x3E,0x1C,0x00,0x00,0x00},/*"3",3*/

{0x00,0x00,0x00,0x00,0x80,0xE0,0x7C,0x1E,0xFE,0xFE,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x0C,0x0E,0x0B,0x09,0x48,0x
48,0x7F,0x7F,0x48,0x48,0x08,0x00,0x00,0x00},/*"4",4*/

{0x00,0x00,0x00,0xFE,0xFE,0xC2,0x42,0x42,0x42,0x42,0xC2,0x82,0x02,0x00,0x00,0x00,0x00,0x00,0x00,0x31,0x71,0x40,0x40,0x
40,0x40,0x40,0x60,0x3F,0x1F,0x00,0x00,0x00},/*"5",5*/

{0x00,0x00,0x00,0xF8,0xFC,0x86,0x82,0x82,0x82,0x82,0x86,0x1C,0x18,0x00,0x00,0x00,0x00,0x00,0x00,0x1F,0x3F,0x61,0x40,0x
40,0x40,0x40,0x61,0x3F,0x1E,0x00,0x00,0x00},/*"6",6*/

{0x00,0x00,0x00,0x0E,0x0E,0x02,0x02,0x02,0x82,0xC2,0x72,0x3E,0x0E,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x70,0x
7E,0x0F,0x01,0x00,0x00,0x00,0x00,0x00,0x00},/*"7",7*/

{0x00,0x00,0x00,0x38,0x7C,0xC6,0x82,0x82,0x82,0x82,0xC6,0x7C,0x38,0x00,0x00,0x00,0x00,0x00,0x00,0x1E,0x3F,0x61,0x40,0x
40,0x40,0x40,0x61,0x3F,0x1E,0x00,0x00,0x00},/*"8",8*/

{0x00,0x00,0x00,0x78,0xFC,0x86,0x02,0x02,0x02,0x02,0x86,0xFC,0xF8,0x00,0x00,0x00,0x00,0x00,0x00,0x18,0x38,0x61,0x41,0x
41,0x41,0x41,0x61,0x3F,0x1F,0x00,0x00,0x00}/*"9",9*/
};
uchar code shiji[2][32]={

{0x00,0x10,0x0C,0x04,0x4C,0xB4,0x94,0x05,0xF6,0x04,0x04,0x04,0x14,0x0C,0x04,0x00,0x00,0x82,0x82,0x42,0x42,0x23,0x12,0
x0A,0x07,0x0A,0x12,0xE2,0x42,0x02,0x02,0x00},/*"?",0*/
	{0xFE,0x02,0x22,0x5A,0x86,0x20,0x20,0x22,0x22,0xE2,0x22,0x22,0x22,0x22,0x20,0x00,0xFF,0x00,0x02,0x04,0x13,0x0C,0
x03,0x40,0x80,0x7F,0x00,0x01,0x02,0x1C,0x08,0x00},/*"?",1*/
};
uchar                                                                                                          code
du[]={0x06,0x09,0x09,0xE6,0xF8,0x0C,0x04,0x02,0x02,0x02,0x02,0x02,0x04,0x1E,0x00,0x00,0x00,0x00,0x00,0x07,0x1F,0x30,0x
20,0x40,0x40,0x40,0x40,0x40,0x20,0x10,0x00,0x00};/*?C*/
uchar code mubiao[2][32]={

{0x00,0x00,0x00,0xFE,0x22,0x22,0x22,0x22,0x22,0x22,0x22,0x22,0xFE,0x00,0x00,0x00,0x00,0x00,0x00,0x7F,0x22,0x22,0x22,0x
22,0x22,0x22,0x22,0x22,0x7F,0x00,0x00,0x00},/*"?",0*/
	{0x10,0x10,0xD0,0xFF,0x50,0x90,0x20,0x22,0x22,0x22,0xE2,0x22,0x22,0x22,0x20,0x00,0x04,0x03,0x00,0xFF,0x00,0x09,0
x04,0x03,0x40,0x80,0x7F,0x00,0x01,0x06,0x1C,0x00},/*"?",1*/
};
void delay(uint i)
{
	while(--i);
```

```
}
//LCM
void Read_busy()
{//busy p2^7
        P2=0xff;
        RS=0;
        RW=1;
        EN=1;
        while(P2&0x80);
        EN=0;
}
void write_command(uchar value)
{
      P2=0xff;
      Read_busy();
        RS=0;
        RW=0;
        P2=value;
        EN=1;
        delay(100);
        EN=0;
}
void write_data(uchar value)
{
      P2=0xff;
      Read_busy();
        RS=1;
        RW=0;
        P2=value;
        EN=1;
        delay(100);
        EN=0;
}
void Set_column(uchar column)
{
      column=column&0x3f;
      column=0x40|column;//01000000|column
      write_command(column);
}
void Set_line(uchar startline)
{
      startline=0xC0|startline;//11000000|startline
      write_command(startline);
}
void Set_page(uchar page)
{
      page=0xb8|page;//10111000|page
      write_command(page);
}
void display(uchar ss,uchar page,uchar column,uchar *p)
{
        uchar i;
        switch(ss)
      {
            case 0:CS1=1;CS2=1;break;
              case 1:CS1=1;CS2=0;break;
              case 2:CS1=0;CS2=1;break;
              default:break;
      }
      page=0xb8|page;//10111000|page
      write_command(page);
      column=column&0x3f;
      column=0x40|column;//01000000|column
      write_command(column);
      for(i=0;i<16;i++)
      {
            write_data(p[i]);
      }
      page++;
      write_command(page);
      write_command(column);
```

```
    for(i=0;i<16;i++)
    {
        write_data(p[i+16]);
    }
}
void SetOnOff(uchar onoff)
{
    onoff=0x3e|onoff;//00111110|onoff
    write_command(onoff);
}
void ClearScreen()
{
    uchar i,j;
    CS2=1;
    CS1=1;
    for(i=0;i<8;i++)
    {
        Set_page(i);
        Set_column(0);
        for(j=0;j<64;j++)
        {
            write_data(0x00);
        }
    }
}
void InitLCD()
{
    Read_busy();
    CS1=1;CS2=1;
    SetOnOff(0);
    CS1=1;CS2=1;
    SetOnOff(1);
    CS1=1;CS2=1;
    ClearScreen();
    Set_line(0);
}
//温度初始化
bit DS_init()
{
    bit flag;
    DQ=0;//总线拉低
    delay_us(255);//500us 延时
    DQ=1;//释放
    delay_us(40);//16~60us
    flag=DQ;//存在低脉冲
    delay_us(150);//60~240us
    return flag;
}
//读操作
uchar read()//byte
{
    uchar i;
    uchar val=0;
    for(i=0;i<8;i++)//移位读入，从低到高
    {
        val>>=1;
        DQ=0;//
    delay_us(1);//1us 低电平
        DQ=1;//释放
        delay_us(1);
        if(DQ)
            val|=0x80;
        delay_us(15);//必须 15us
    }
    return val;
}
//写操作
void write(char val)
{
    uchar i;
```

```
    for(i=0;i<8;i++) //移位，从低到高
    {
        DQ=0;//拉低电平
        delay_us(8); //15us
        val>>=1;
        DQ=CY;
        delay_us(35);
        DQ=1;
        delay_us(10);
    }
}
//数字 PID 增量
void PID()
{
    uchar Px,Pp,Pi,Pd;//Px=Δ u(k)
    uint count;
    Pp=Kp*(Ek-Ek1);
    Pi=Ki*Ek;
    Pd=Kd*(Ek-2*Ek1+Ek2);
    Px=Pp+Pi+Pd;//Px=Kp[e(k)-e(k-1)]+Ki*e(k)+Kd[e(k)-2e(k-1)+e(k-2)]
    res=Px;
    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=27;
    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);//skip rom
        write(0x44);//温度变换
        delay(600);
        while(DS_init());
        write(0xcc);
        write(0xBE);//读暂存存储器
        low=read();//读低位
        high=read();//读高位
        delay(255);
        result=high;
```

```
            result<<=8;
            result|=low;
            result>>=4;//result/=16;数值*0.625=实际温度
            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();
                        //up=1;
            else
                  up=0;
      }
}
```

六、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？

答：（1）定时器延时：通过设置定时器处置可以实现以 us 为单位的精确定时；

优点：定时精确，程序移植性好；

缺点：设置定时器本身需要一定的时间，要求延时较短的情况下不满足要求，实现复杂。

（2）软件定时：使用 while 循环，每执行一次循环需要 3-5us。

优点：实现简单；

缺点：不精确，严重依赖机器。