单片机实验报告

（5、6、8）

计科 8 班
学号：21160830
姓名：汪正涛

实验五　重量测量

一、实验目的和要求

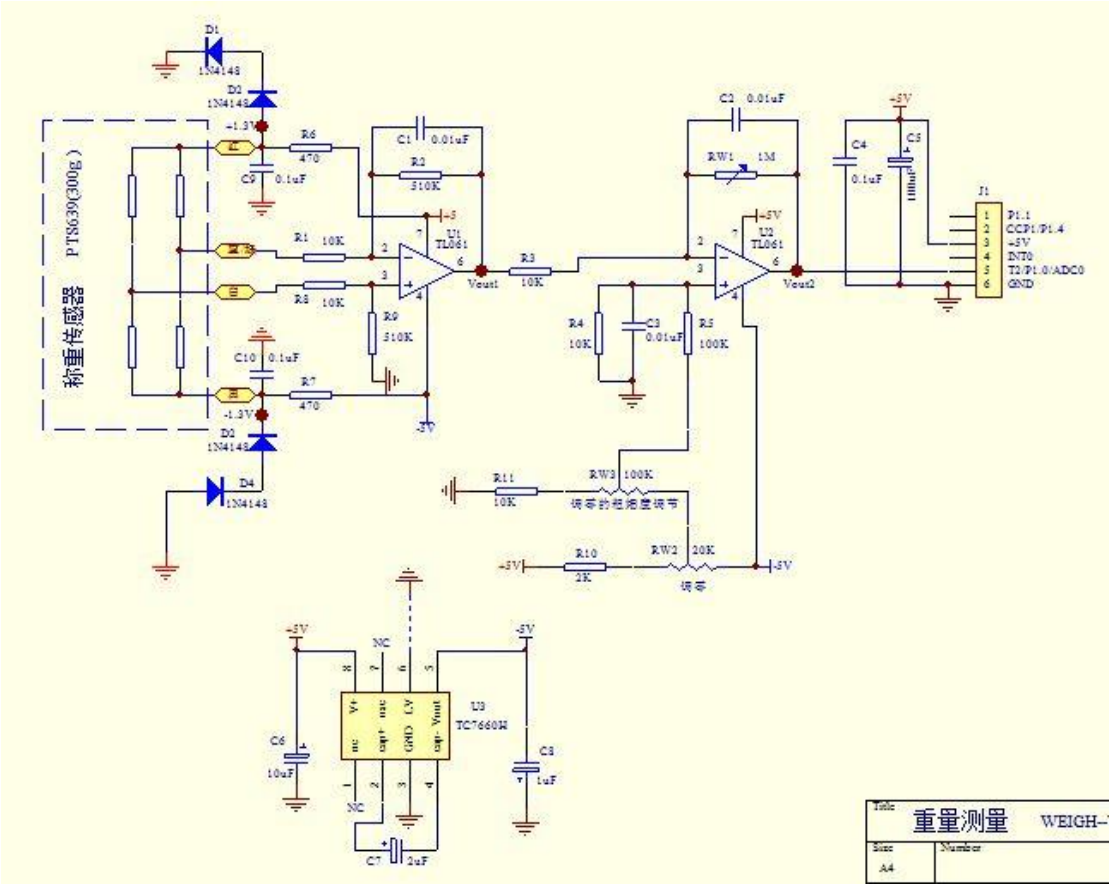
掌握点阵式液晶显示屏的原理和控制方法，掌握点阵字符的显示方法。

掌握模拟/数字（A/D）转换方式。

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、实验原理

本实验示意电路原理图：



1. 参考附录六，学习点阵式液晶显示屏的控制方法。

2. 在液晶显示中，自定义图形和文字的字模对应的字节表需要使用专门的字模软件来生成。可以使用 PCtoLCD2002 字模软件提取。

3. 字符点阵等数据，需要定义在 code 数据段中，具体原理参见示例程序设计部分。

4. 向 LCM 输出一个命令或数据时，应当在选通信号为高时准备好数据，然后延迟若干指令周期，再将选通信号置为低。

5. 与 A/D 转换相关的寄存器

ADC_POWER：ADC 电源控制位，0 关 1 开。

SPEED1,SPEED0：模数转换器速度控制位，控制 A/D 转换所需时间。

ADC_FLAG：模数转换结束标志位，AD 转换完后，ADC_FLAG=1，一定要软件清 0。

ADC_START：模数转换器（ADC）转换启动控制位，1 开始转换，转换结束后为 0。

CHS2/CHS1/CHS0 :模拟输入通道选择, 选择使用 P1.0~P1.7 作为 A/D 输入。

ADC_RES、ADC_RESL： A/D 转换结果寄存器，是特殊功能寄存器，用于保存 A/D 转换结果。

IE：中断允许寄存器（可位寻址）

EA：CPU 的中断开放标志，EA=1，CPU 开放中断，EA=0，CPU 屏蔽所有中断申请。

EADC：A/D 转换中断允许位。1 允许 0 禁止。

IPH：中断优先级控制寄存器高（不可位寻址）。

IP：中断优先级控制寄存器低（可位寻址）。

6. 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。

三、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

四、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。

五、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。

六、实验程序

```
#include <reg52.h>
#include <intrins.h>
#define uchar unsigned char
#define uint unsigned int
```

```
    sbit CS1=P1^7;///左半边
    sbit CS2=P1^6;///右半边
    sbit E=P3^3;///使能信号
    sbit RW=P3^4;///读写操作
    sbit RS=P3^5;///寄存器选择(数据/指令)
    sbit RES=P1^5;///复位 低电平有效
    sbit BUSY=P2^7;
    /**Declare SFR(特殊功能寄存器) associated with the ADC(模数转换)
*/
    sfr ADC_CONTR = 0xBC;  ///ADC control register
    sfr ADC_RES = 0xBD;  ///ADC hight 8-bit result register
    sfr ADC_LOW2 = 0xBE;  ///ADC low 2-bit result register
    sfr P1ASF = 0x9D;  ///P1 secondary function control register
    sfr AURX1 = 0xA2; ///AURX1 中的 ADRJ 位用于转换结果寄存器的数据格式
调整控制
    /**Define ADC operation const for ADC_CONTR*/
    #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,///*"重"*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,///*"量"*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,///*"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;
int weight;
int temp;
int initW;
void main()
{
    init_yejing();
    init_adc();
    calibrate();///取得空载值
    while(1){
              initW = get_ad_result();
             weight=(initW-cweight)/3.3+(initW-cweight)/20/3;
              temp = weight;
        clearscreen();
        send_all(1,1,10);///重
        send_all(1,2,11);///量
        send_all(1,3,12);///:
        send_all(4,3,weight/100);///百
        send_all(4,4,(weight/10)%10);///十
        send_all(4,5,weight%10);///个
        send_all(4,6,13);///克
        delay(50000);
    }
}
void init_yejing()
{
    send_byte(192,1,1);///设置起始行
    send_byte(63,1,1);///打开显示开关
}
void send_byte(uchar dat,uchar cs1,uchar cs2)
{
    P2=0xff;
    CS1=cs1; CS2=cs2;
    RS=0; RW=1; E=1;///输出，读取状态字
    while(BUSY) ;
    ///送数据或控制字
    E=0;
    RS=!(cs1&&cs2),RW=0;
    P2=dat;
    E=1; ///输入，写显示数据
      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);///选择页面
        send_byte(64+lie*16-(lie>3)*64,1,1);///选择列号
        for(j=0;j<16;++j)
            send_byte(zima[offset][k++],lie<4,lie>=4);///送数
    }
}
void init_adc()
{
    P1ASF = 1; ///Set  P1.0 as analog input port 选择 P1.0 口
    AURX1 |= 0X04; ///AURX1 中的 ADRJ 位用于转换结果寄存器的数据格式调整控制
    ADC_RES = ADC_LOW2 = 0; ///Clear previous result
    ADC_CONTR = ADC_POWER | ADC_SPEEDLL | ADC_START | ch;  ///ch=0 ADC channel NO.0
    delay(4);  ///ADC power-on delay and Start A/D conversion
}
int get_ad_result()
{
    int ADC_result;
    ADC_RES = ADC_LOW2 = 0; ///Clear previous result
    ADC_CONTR = ADC_POWER | ADC_SPEEDLL | ch | ADC_START;
    _nop_(); _nop_(); _nop_(); _nop_(); _nop_(); _nop_(); ///Must wait before inquiry
    while (!(ADC_CONTR & ADC_FLAG)); ///Wait complete flag
    ADC_result = (ADC_RES & 0x03) *256 + ADC_LOW2;///ADC_RES 中存高 2 位于其低两位中
    ADC_CONTR &= ~ADC_FLAG; ///Close ADC flag 位置 0
    return ADC_result;  ///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);///页
      send_byte(64,1,1);///列
         for(j=0;j<64;++j)
            {
               send_byte(0x00,0,1);
               send_byte(0x00,1,0);
            }
   }
}
```

七、思考题

1. 调零的原理，软件调零和调零调零的区别。

原理：以测量仪器真实值为零的时候为标准，通过软件或硬件的方法调整读数，使读数为零，从而减小误差。软件调零是指在程序内部获得测量值后，通过一定的算法或公式来计算得到一个准确值再进行输出，硬件调零是指在读取模拟量时更改电阻的阻值从而使测量值接近真实值，二者的区别主要在于硬件调零的操作发生在 A/D 转换之前，因此程序读取的测量值就是真实值，而软件调零发生在转换之后，因此程序读取到测量值并不是真实值。硬件调零通过调零电阻的精确度保障调零的精度，软件调零通过数据结构和算法来保证调零的精度。

2. 模/数和数/模的信号转换原理。

模拟信号转换为数字信号经历了采样→保持→量化→编码的过程。首先，按照一定的采样率，将一段连续的模拟信号分成许多离散的节点；采样的信号需要在一段时间内保持不变，这样才能保持转换的正确性与采样精度。这样采样后的信号便会成为一个连续的阶梯梯状函数。量化是指按照精度将信号经过舍入或截尾的方法将数值固定在某一个量化步长内。如 8 位精度的 A/D 转换器，其量化

步长为最大电压幅值的 1/256。最后将这些量化后的信号转换为 0/1 信号的过程就称为编码。数/模转换的过程为模/数转换过程的逆过程。

3. I2C 总线在信号通讯过程中的应用。

I2C 是一种双向两线制的串行数据传输标准总线。当 I2C 空闲时，SDA 和 SCL 线都处于高电平。SDA 由高电平到低电平的转变能够产生启动条件；当 SCL 在高电平保持稳定时，SDA 由低电平到高电平的转变能够产生停止条件。启动和停止条件一般由主设备产生。当接收到 ACK 时钟脉冲时，发送器应通过使 SDA 线变成高电平来释放 SDA 线。接收器也需在 ACK 时钟脉冲期间使 SDA 线变为低电平。I2C 还具有总线仲裁程序以及异常中断条件。

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。

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。最简单的办法就是以某个时间单位（如 0.1ms，相当于 10kHz）为基准，在前 N 段输出 1，后 M-N 段输出 0，总体的占空比就是 N/M。这种方法由于 0 和 1 分布不均匀，所以要求基准频率要足够高，否则会出现颠簸现象。

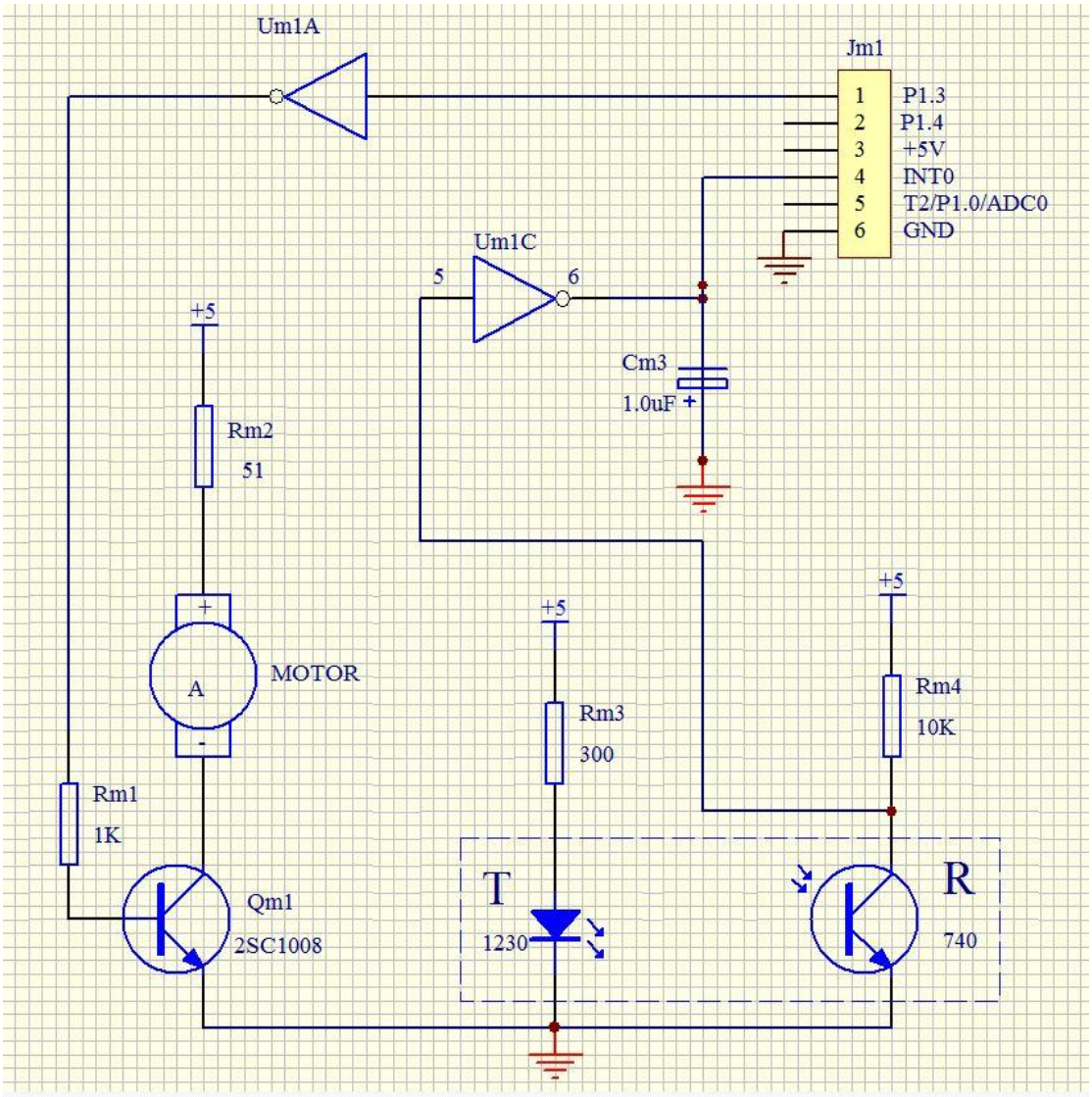
要达到更稳定的效果，可以采用累加进位法如果将总的周期内的 0 和 1 均匀分散开。设置一个累加变量 x，每次加 N，若结果大于 M，则输出 1，并减去 M；否则输出 0。这样整体的占空比也是 N/M。在实验中取 M=256 可以使程序更加简单。

另外，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。

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。也可自行使用其他更加复杂的算法。

实验中采用的电机最大转速在 200 转/s 左右，转速小于 40 转/s 左右将不稳定，可能会停转。

本实验示意电路原理图：



三、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

四、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。

五、实验步骤

1. 建立工程，实现实验内容 1

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。

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。

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 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、实验代码

```
#include <reg52.h>
#include <intrins.h>
#define uchar unsigned char
#define uint unsigned int
sfr P4=0xC0;
sfr P4SW=0xBB;
sbit sclk=P4^4;
sbit sdata=P4^5;
sbit CS1=P1^7;///片选，左
sbit CS2=P1^6;///右半屏选择
sbit E=P3^3;///使能信号
sbit RW=P3^4;///读写操作选择
sbit RS=P3^5;///寄存器选择(数据/指令)
sbit RES=P1^5;///复位 低电平有效
sbit BUSY=P2^7;
///sfr IE=0xA8; ///中断允许寄存器
sbit swh1=P3^6;///开关
sbit swh2=P3^7;
```

```
sbit motor=P1^1;///电机
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,///*"重"*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,///*"量"*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,///*"克"*13/
    };
    uchar tab[15]=                    ///数码管字码
        {0xC0,0xF9,0xA4,0xB0,0x99,
            0x92,0x82,0x0F8,0x80,0x90};
    uchar tspeed=0;///累加转数
    uchar cspeed=0;///当前速度
    uchar xspeed=120;///期望速度
    uchar speedUp = 150;///高速
    uchar speedLow = 90;///低速
    uchar t1_cnt=0; ///1s 延时控制变量 50ms*20 次
    int N=100;///占空比
    int M=256;///分母
    int X=0;///起始变量
    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 display(uchar n);
    void sendbyte(uchar ch)
    void delay1();
    void delay2();
    void delay(uint x)
    {
        while(x--) ;
    }
    uchar flag1=0;
    uchar flag2=0;
    void main()
```

```
{
    init();
init_yejing();
    motor=0;
    while(1)
    {
     display(cspeed);
     delay2();
     display(xspeed);
     delay1();
     clearscreen();
          send_all(1,3,speedUp/100);///百
        send_all(1,4,(speedUp/10)%10);///十
        send_all(1,5,speedUp%10);///个
          send_all(3,3,cspeed/100);///百
        send_all(3,4,(cspeed/10)%10);///十
        send_all(3,5,cspeed%10);///个
          send_all(5,3,speedLow/100);///百
        send_all(5,4,(speedLow/10)%10);///十
        send_all(5,5,speedLow%10);///个
       delay1();
          delay(50000);
    }
}
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;
    }
}
void display(uchar n)
{
    sendbyte(n%10);        ///个
    sendbyte((n/10)%10);   ///十
    sendbyte(n/100);       ///百
}
void init()
{
```

```
    P4SW=0x30;
     IT0=1; ///设置 INT0 为边沿触发(零时为电平触发)
     EA=1;///CPU 可以响应中断
     ET1=1;///T1 中断允许位
     ET0=1;///T0 中断允许位
     EX0=1;///外部中断 INT0 中断允许
    /// GATE 为 0 定时器不受外部控制;C/T＝0 为定时方式;M1M0 为 01
即方式 1, 16 位的定时/计数器,由 TH 和 TL 联合构成
     ///当计数溢出时置位 TF
    TMOD=0x11; ///设置方式寄存器,定时器 0 和定时器 1 的工作方式
     TH1=0x3C;
     TL1=0xB0;  ///50ms 计数值,15536
     TH0=0xFF;
     TL0=0x9C;  ///0.1ms 计数值,65436
     TR0=1;
     TR1=1;   ///启动定时器
}
void ex_int0() interrupt 0  ///外部中断 INT0，记录转动次数
{
    tspeed++;
}
void t1_int() interrupt 3   ///50ms 定时器中断 T1
{
    if(++t1_cnt<20)
    {
        if(swh1==0)
        {
            flag1=1;
                xspeed = speedLow;
            flag1 = 0;
        }
        if(swh2==0){
            flag2=1;
            xspeed = speedUp;
            flag2 = 0;
         }
         if(swh1==1 &&swh2==1 ){
            xspeed = 120;
         }
        return;
    }
    t1_cnt=0;
    cspeed=tspeed;
    tspeed=0;
```

```
    if(cspeed>xspeed) N++;
    if(cspeed<xspeed) N--;
}
void t0_int() interrupt 1  ///0.1ms 定时器中断 T0
{
     X+=N;
    if(X>M)
    {
        motor=1; ///不转
         X-=M;
    }
    else
         motor=0; ///转
}
void init_yejing()
{
    send_byte(192,1,1);///设置起始行
    send_byte(63,1,1);///打开显示开关
}
void send_byte(uchar dat,uchar cs1,uchar cs2)
{
    P2=0xff;
    CS1=cs1; CS2=cs2;
    RS=0; RW=1; E=1;///输出，读取状态字
    while(BUSY) ;
    ///送数据或控制字
    E=0;
    RS=!(cs1&&cs2),RW=0;
    P2=dat;
    E=1; delay(3); E=0;///输入，写显示数据
    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);///选择页面
        send_byte(64+lie*16-(lie>3)*64,1,1);///选择列号
        for(j=0;j<16;++j)
            send_byte(zima[offset][k++],lie<4,lie>=4);///送数
    }
}
void clearscreen()
```

```
{
   int i,j;
   for(i=0;i<8;++i)
   {
      send_byte(184+i,1,1);///页
      send_byte(64,1,1);///列
         for(j=0;j<64;++j)
             {
                 send_byte(0x00,0,1);
                 send_byte(0x00,1,0);
             }
   }
}
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++);
}
```

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。

区别 :脉宽调速和电压调速的不同之处在于电压调速指的是调节直流电机端电压来改变转速，供电是连续的，可以做到无级调速，而脉宽调速指的是通过开关量输出达到模拟量输出效果，宏观看是一个匀速的过程，而微观看是非匀速的过程，脉宽调速通过改变信号的占空比来调节宏观上的输出电压进而改变转速。

优缺点 :电压调速优点在于可以实现从使动电压到额定电压范围内的无级调速，调速范围大，点击运行平稳，缺点是需要外围器件多（如调压器等）且调压过程中转差功率损耗大，效率较低。脉宽调速的优点在于需要外围器件较少，效率高，缺点是调速范围没有电压调速大，当直流电机转速低时噪音较大。

应用范围：电压调速适用于对调速范围和平稳运行有特殊要求，调速精度要求不高，且对效率影响不大的情况下。脉宽调速适用于对噪音要求不高且采用电压调速对效率影响较大的情况下。此外，在交流电机中，PWM 也具有调节频率的作用。

2. 说明程序原理中累加进位法的正确性。

我们将整个周期分为 M 份，那么累加进位法在一个周期中需要累加并判断 M 次。又因为每一次判断前累加变量 X 都需要加 N，因此如果不进行减法操作，一周期后 X 的值为 N×M。因为累加进位法判断的逻辑为：X 大于等于 M 则减去 M，否则不减，所以我们可以将该过程转换为除法，即在一个周期中，X 减去 M 的操作发生了 N×M÷M=N 次。又因为每次 X 的值大于等于 M 时都会使电机停转，所以电机停转的次数为一周期中 X 减去 M 的次数，即 N 次，因此整体的占空比为 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。

设置的默认转速为 120。单片机启动时，电机转速的最大可能误差会达到 ±30 转/秒，随后会缓慢地靠近默认转速，在经过一段时间后，电机的转速会稳定在 120±5 转/秒的范围内。电机的转速产生误差主要来源于因为电机本身转速改变时的惯性，当采用二位式控制算法时，闭环控制只能根据当前反馈回来的电机转速来进行调节，因此总是存在调节的滞后性，具体表现就是转速会在某一个区间波动。因为电机的惯性是由其特性决定的无法改变，因此只能进行软件校正。可以采用增量式 PID 算法来替代二位式控制算法调节电机转速。PID 算法的优点在于可以根据电机之前几个时刻的转速，通过比例、积分和微分算法来预测接下来的转速，从而调节当前输出的控制信号，可以较好的减小波动，使速度变化

的得更加平稳。增量式 PID 算法的离散公式为 :Δu(k)=u(k)-u(k-1)=Kp*(e(k)-e(k-1))+Ki*e(k)+Kd*(e(k)-2*e(k-1)+e(k-2))，其中积分系数 Ki=Kp*T/Ti,微分系数 Kd=Kp*Td/T，T 为采样间隔，Ti 为积分时间常数，Td 为微分时间常数。当前转速 u(k)=Δu(k)+u(k-1)。

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。

3. 加深 C51 编程语言的理解和学习。

二、实验原理

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

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（C°），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 0.5C° 。

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。

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。

相关背景知识参见 DS18B20 中文资料。

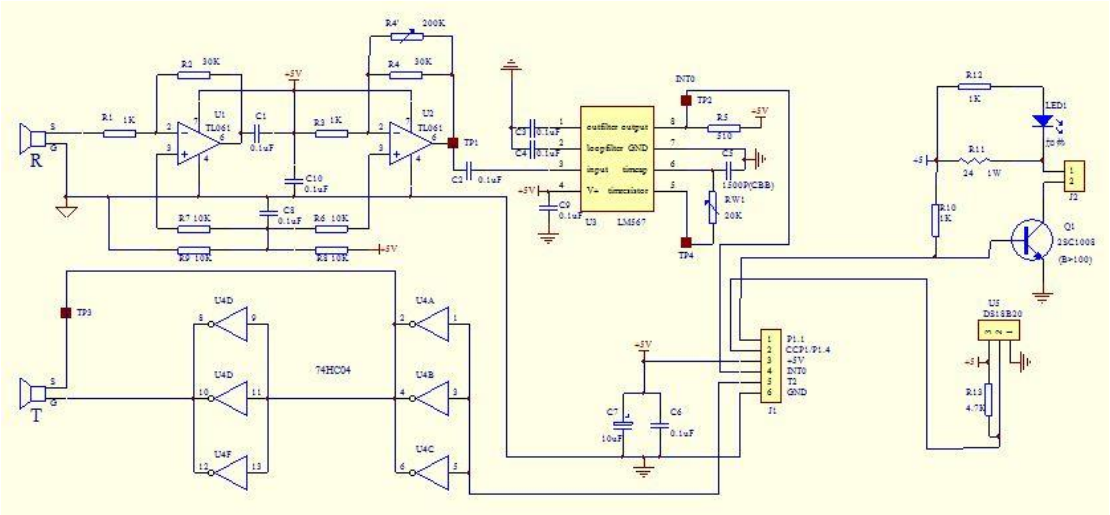
实验原理见附录七。

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。

本实验示意电路原理图：



三、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

四、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。

五、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。

六、实验程序

```
#include <reg52.h>
#include <intrins.h>
#define uchar unsigned char
#define uint unsigned int
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,
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/
    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/
    0x10,0x21,0x86,0x70,0x00,0x7E,0x4A,0x4A,0x4A,0x4A,0x4A,0x7E,0x00,
0x00,0x00,0x00,0x00,
    0x02,0xFE,0x01,0x40,0x7F,0x41,0x41,0x7F,0x41,0x41,0x7F,0x41,0x41,
0x7F,0x40,0x00,///*"温",14*/
    0x00,0x00,0xFC,0x04,0x24,0x24,0xFC,0xA5,0xA6,0xA4,0xFC,0x24,0x24,
0x24,0x04,0x00,
    0x80,0x60,0x1F,0x80,0x80,0x42,0x46,0x2A,0x12,0x12,0x2A,0x26,0x42,
0xC0,0x40,0x00,///*"度",15*/
    };
    sbit CS1=P1^7;///左半边
    sbit CS2=P1^6;///右半边
    sbit E=P3^3;///使能信号
    sbit RW=P3^4;///读写操作选择
    sbit RS=P3^5;///寄存器选择(数据/指令)
    sbit RES=P1^5;///复位 低电平有效
    sbit BUSY=P2^7;
    sbit De=P1^1; ///加热
    sbit DQ=P1^4; ///DS18B20 单数据总线
    uchar TPH,TPL; ///温度值高位 低位
    unsigned int t; ///温度值
    unsigned int t1=30; ///目标温度值
    sbit swh1=P3^6;
    sbit swh2=P3^7;
    uchar flag1=0;
    uchar flag2=0;
    void send_byte(uchar dat ,uchar cs1,uchar cs2);
```

```
void send_all(uint page,uint lie,uint offset);
void delay(uint x);
void init_yejing();
void clearscreen();
void DelayXus(uchar n); ///微秒级延时
void ow_rest(); ///复位
void write_byte(char dat);
unsigned char read_bit(void);
void main(void)
{
   init_yejing();
   t=0;
   while(1)
   {
        if(swh1==0)
        {
             flag1=1;
        }
        if(swh1==1 && flag1==1)
        {
             t1++;
             flag1=0;
        }
        if(swh2==0)
             flag2=1;
        if(swh2==1 && flag2==1)
        {
             t1--;
             flag2=0;
        }
    if(t<t1)
    De=1;
    else De=0;
    ow_rest(); ///设备复位
    write_byte(0xCC); ///跳过 ROM 命令
    write_byte(0x44); ///开始转换命令
    while (!DQ); ///等待转换完成
    ow_rest(); ///设备复位
    write_byte(0xCC); ///跳过 ROM 命令
    write_byte(0xBE); ///读暂存存储器命令
    TPL = read_bit(); ///读温度低字节
    TPH = read_bit(); ///读温度高字节
    t=TPH; ///取温度高位
    t<<=8; ///高位 8 位
```

```
    t|=TPL; ///加上温度低位
    t*=0.625; ///实际温度 可直接显示
    t=t/10;
   send_all(1,1,14);///温
   send_all(1,2,15);///度
   send_all(1,3,12);///:
   send_all(4,2,t1/10);///十
   send_all(4,3,t1%10);///个
   send_all(4,5,t/10);///十
   send_all(4,6,t%10);///个
   delay(50000);
   clearscreen();
   }
}
void DelayXus(uchar n)
{
    while (n--)
    {
        _nop_();
        _nop_();
    }
}
unsigned char read_bit(void)///读位
{
    uchar i;
    uchar dat = 0;
    for (i=0; i<8; i++) ///8 位计数器
    {
        dat >>= 1;
        DQ = 0; ///开始时间片
        DelayXus(1); ///延时等待
        DQ = 1; ///准备接收
        DelayXus(1); ///接收延时
        if (DQ) dat |= 0x80; ///读取数据
        DelayXus(60); ///等待时间片结束
    }
    return dat;
}
void ow_rest()///复位
{
    CY = 1;
    while (CY)
    {
        DQ = 0; ///送出低电平复位信号
```

```
        DelayXus(240); ///延时至少 480us
        DelayXus(240);
        DQ = 1; ///释放数据线
        DelayXus(60); ///等待 60us
        CY = DQ; ///检测存在脉冲,DQ 为 0 转换完成
        DelayXus(240); ///等待设备释放数据线
        DelayXus(180);
    }
}
void write_byte(char dat)///写字节
{
    uchar i;
    for (i=0; i<8; i++) ///8 位计数器
    {
        DQ = 0; ///开始时间片
        DelayXus(1); ///延时等待
        dat >>= 1; ///送出数据
        DQ = CY;
        DelayXus(60); ///等待时间片结束
        DQ = 1; ///恢复数据线
        DelayXus(1); ///恢复延时
    }
}
void init_yejing()
{
    send_byte(192,1,1);///设置起始行
    send_byte(63,1,1);///打开显示开关
}
void send_byte(uchar dat,uchar cs1,uchar cs2)
{
    P2=0xff;
    CS1=cs1; CS2=cs2;
    RS=0; RW=1; E=1;
    while(BUSY) ;
    ///送数据或控制字
    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);///选择页面
        send_byte(64+lie*16-(lie>3)*64,1,1);///选择列号
        for(j=0;j<16;++j)
            send_byte(zima[offset][k++],lie<4,lie>=4);///送数
    }
}
void delay(uint x)
{
    while(x--) ;
}
void clearscreen()
{
   int i,j;
   for(i=0;i<8;++i)
   {
      send_byte(184+i,1,1);///页
      send_byte(64,1,1);///列
         for(j=0;j<64;++j)
             {
                 send_byte(0x00,0,1);
                 send_byte(0x00,1,0);
             }
   }
}
```

七、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？。

实现延时有两种方法，硬件延时和软件延时。

硬件延时，即采用单片机内部的定时器来进行延时。这在前几次的实验中也有应用，计时初值可以通过公式得出，在晶振频率为 12MHz 时，计时长度可达65536μs。计时结束后通常采用中断的方式来进行响应。影响计时精度有两个因素，一个是计时器的工作方式，当工作在方式 2 时可实现短时间精确延时，但工作在方式 1 时，重装初值需要 2 个机器周期，这个需要从计数初值中减去。另一个是采用 C51 编译中断程序时会自动加上保护及回复现场的语句，会占用

4 个机器周期，同样需要从计数初值中减去。硬件延时的好处是计时较软件来说更为精准，因为采用独立部件，也可以提高 CPU 的运行效率；缺点是操作相比软件延时更为复杂，另外若计时器被用作其他用途时便只能采用软件延时。软件延时多以函数+函数内部调用_NOP()函数（使用_NOP()函数需要包含头文件“intrins.h”）的方式实现，STC12 单片机（1 时钟/机器周期，12MHz）中计算方法是延时时间=（6（LCALL 指令）+n（NOP 指令数）+4（RET 指令）/12us。软件延时的优点在于可以进行长时间的延时，且实现简单易理解，缺点是不如硬件延时精确，且当嵌套调用延时函数时需要注意调用过程对延时时间的影响。

2. 参考其他资料，了解 DS18B20 的其他命令用法。

DS18B20 的其他操作命令有：

Read ROM 命令（33H）：此命令允许总线主机读 DS18B20 的 8 位产品系列编码，唯一的 48 位序列号，以及 8 位的 CRC。

Match ROM 命令（55H）：自命令后继以 64 位的 ROM 数据序列，允许总线主机对多点总线上特定的 DS18B20 寻址。

Search ROM 命令（F0H）：此命令允许总线控制器用排除法书别总线上所有从机的 64 位编码。

Alarm Search 命令（ECH）：自命令的流程与搜索 ROM 命令相同。但是，仅在最近一次温度测量出现告警的情况下，DS18B20 才对此命令做出相应调用。

八、实验的体会与收获

用了几周的时间了解并学习了重力测量，直流电机以及温控实验，我基本上掌握了单片机测控实验系统以及 Keil 开发环境这个软件的一些基本操作，了解了如何运用 Keil 编写代码，如何在单片机测控实验系统中运行自己设计的程序，观

察程序的运行状态，分析代码的正确与否。

在完成重力测量的实验时，利用软件调零的办法，通过数据的拟合将测量的重量与实际重力进行对比，得出了较为准确的拟合曲线，由于二次及以上曲线太过复杂，于是我采用了一次。但是实际数据得出一次在保证数据尽量靠近曲线时，无法满足在重量为 0 时，显示数据为零，经反复调试无果后，觉得用条件判断语句使其强制为 0。

经过这两次实验的学习和训练，我明白了理论和实践确实是两种不一样的形式，只有将理论运用到了实践上，才能真正算掌握了知识。