实验五 重量测量

53160821 2016 级计科 8 班 刘洋洋

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 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 。

2.1P1 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。

2.2ADC 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

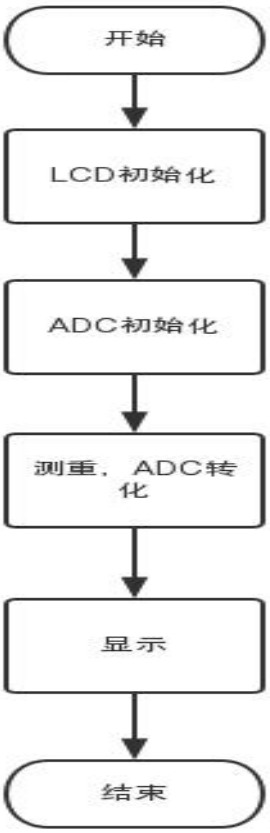
三、实验器材

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

四、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。

五、程序流程图



六、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。

七、实验代码

```
sfr P1ASF = 0x9D;//选择 P1.7~P1.0 的某一个口作为转换通道后，响应的位置 1
sfr ADC_CONTR = 0xBC;//控制 AD 电源位、转换口、转换时间和转换标志
sfr ADC_RES = 0xBD;//保存 AD 转换结果的高位
sfr ADC_RESL = 0xBE;//保存 AD 转换结果的低位
sfr AUXR1 = 0xA2;//控制 AD 转换结果在 ADC_RES 和 ADC_RESL 中存储方式
//位寻址地址
#define ADC_POWER 0x80
#define ADC_FLAG 0x10
#define ADC_START 0x08
#define ADC_SPEEDLL 0x00
//LCD 显示屏的接口定义
sbit RST = P1^5;//复位端，低电平有效
sbit CS1 = P1^7;//左半屏开关
sbit CS2 = P1^6;//右半屏开关
sbit EN= P3^3;//使能开关
sbit RW = P3^4;//读、写操作选择信号
sbit RS = P3^5;//寄存器选择信号
sbit BUSY = P2^7;//表示 LCD 是否准备好
sbit ON = P2^5;//表示当前的显示状态
uchar code
zhong[]={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};/*"?",2*/
uchar code
liang[]={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};/*"?",3*/
uchar code
wei[]={0x00,0x10,0x10,0x12,0x14,0x1C,0x10,0xF0,0x9F,0x10,0x10,0x10,0x10,0xF8,0x
10,0x00,0x00,0x00,0x40,0x20,0x10,0x08,0x06,0x01,0x00,0x11,0x26,0x40,0x20,0x1F,0
x00,0x00};/*"?",4*/
uchar code
```

```
maohao[]={0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,0x36,0x36,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x00,0x00};/*":",5*/
uchar code number[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
,
0x00,0x00,0x00,0x00,0x00,0x00,0x38,0x3E,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/
}
//LCD 的函数定义
void InitLCD();//LCD 初始化函数
void ClearScreen();//清屏
void delay(uint i);//延时函数
void Read_busy();//等待 LCD 准备好，即 busy 位为 0
void write_command(uchar value);//写命令
void write_data(uchar value);//写数据
void Set_column(uchar column);//设置列
void display(uchar ss,uchar page,uchar column,uchar *p);//显示在屏幕上
void delay(uint i){
    while(--i);
}
void Read_busy()  {
    P2=0xff;//busy 位置 1
    RS=0;
    RW=1;
    EN=1;//读状态字
    while(P2&0x80);//busy 位为 1 时，循环等待
    EN=0;//呈高阻状态
}
void write_command(uchar value){
    P2=0xff;
    Read_busy();//等待 busy 位为 0
    RS=0;
    RW=0;//准备写指令
    P2=value;
    EN=1;
    delay(100);
    EN=0;//EN 下降沿时写指令
}
void write_data(uchar value){
     P2=0xff;
     Read_busy();//等待 busy 位为 0
```

```
    RS=1;//准备写数据
    RW=0;
    P2=value;
    EN=1;
    delay(100);
    EN=0;//EN 下降沿时写数据
}
void Set_column(uchar column){
  column=column&0x3f;//将 column 最高位和次高位置 0
  column=0x40|column;//将 column 最高位置 0，次高位值 1
  write_command(column);//写指令设置起始列 01+D5~D0
}
void display(uchar ss,uchar page,uchar column,uchar *p){
    uchar i;
    if(ss==1){//显示左半屏
        CS1=1;
        CS2=0;
    }
    else if(ss==2){//显示右半屏
        CS1=0;
        CS2=1;
    }
    page=0xb8|page;//设置起始页的指令为 1011 1+D2~D0
    write_command(page);
    Set_column(column);
    for(i=0;i<16;i++){
        write_data(p[i]);//将要显示字符的字模码的前 16 位写入 LCD
    }
    page++;//显示下一页
    write_command(page);
    write_command(column);
    for(i=0;i<16;i++){
        write_data(p[i+16]);//将要显示字符的字模码的后 16 位写入 LCD
    }
}
void ClearScreen(){
  uchar i,j;
  CS2=1;
  CS1=1;
  for(i=0;i<8;i++){//循环 8 个页面
    i=0xb8|i;//设置起始页的指令为 1011 1+D2~D0
    write_command(i);
    Set_column(0);//从每页的第 0 列开始，列数自动加 1
    for(j=0;j<64;j++){
```

```
            write_data(0x00);//清屏
        }
      }
}
void InitLCD(){
    Read_busy();//等待 busy 位为 0
    CS1=1;CS2=1;
    write_command(0x3E);//关屏幕
    CS1=1;CS2=1;
    write_command(0x3F);//打开显示开关
    CS1=1;CS2=1;//开屏幕
    ClearScreen();//清屏
    write_command(0xC0);//设置显示起始行为第 0 行
}
//ADC 转换的函数定义
void InitADC();//AD 转换器初始化函数
void InitADC_n(uchar n);//初始化 AD 转换器的通道
uint ADC_GET(uchar n);//得到 AD 转换器的结果
void DelayMs(uint n);//延时函数
//ADC 转换的函数内容
void DelayMs(uint n){
    uint x;
    while(n--){
        x = 5000;
        while(x--);
    }
}
void InitADC(){
    P1ASF=0xff;
    ADC_RES=0;//结果寄存器清零
    ADC_RESL=0;//结果寄存器清零
    ADC_CONTR=0xE0;//给 ADC 上电，设置转换速度，且目前禁止转换
    _nop_();
    _nop_();
    _nop_();
    _nop_();
}
void InitADC_n(uchar n){
    n &= 0x07;//保留 n 的低三位
    AUXR1 |= 0x04;//设置 AD 转换结果存储方式
    P1ASF = 1<<n;//将 AD 通道对应位置 1
}
uint ADC_GET(uchar n){
    uint adc_data;
```

```
    ADC_RES = 0;
    ADC_RESL = 0;
    ADC_CONTR=0xE8;//允许开始 AD 转换
    _nop_();
    _nop_();
    _nop_();
    _nop_();
    _nop_();
    _nop_();
    while(!((ADC_CONTR & ADC_FLAG) == 0x10))
//当 ADC_FLAG 位为 1 时不能进行 AD 转换
    adc_data = (ADC_RES&0x03)*256 + ADC_RESL;
//将转换结果放入 adc_data 中
    adc_data=(adc_data-23)/1.8;//重量测量的一次函数
    return adc_data;
}
void main(){
    InitLcd();
    InitADC();
    while(1){
         uint ad = 0;
        uint ge=0;//个位
        uint shi=0;//十位
        uint bai=0;//百位
         InitADC_n (0);//选择 P1.0 为转换通道
         ad=ADC_GET(0);
        bai=ad/100;
        shi=(bai%100)/10;
        ge=(bai%100)%10;
       display(1,2,0*16,zhong); //在左半屏第 2 页第 0 列开始显示“重”
        delay(255);
        display(1,2,1*16,liang); //在左半屏第 2 页第 16 列开始显示“量”
        delay(255);
        display(1,2,2*16,wei); //在左半屏第 2 页第 32 列开始显示“为”
        delay(255);
        display(1,2,3*16,maohao);//在左半屏第 2 页第 48 列开始显示“：”
        delay(255);
        display(2,2,0*16,number[bai]);//在右半屏第 2 页第 0 列开始显示百位
        delay(255);
        display(2,2,1*16,number[shi]);//在右半屏第 2 页第 16 列开始显示十位
        delay(255);
        display(2,2,2*16,number[ge]);//在右半屏第 2 页第 32 列开始显示个位
        delay(255);
    }
```

}

八、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。

D/A 的定义：将数字信号转换为模拟信号的电路称为数模转换器(简称 d/a 转换器或 dac,digital to analog converter)。

DA 转换器的内部电路构成无太大差异，一般按输出是电流还是电压、能否作乘法运算等进行分类。大多数 DA 转换器由电阻阵列和 n 个电流开关(或电压开关)构成。按数字输入值切换开关，产生比例于输入的电流(或电压)。此外，也有为了改善精度而把恒流源放入器件内部的。一般说来，由于电流开关的切换误差小，大多采用电流开关型电路，电流开关型电路如果直接输出生成的电流，则为电流输出型 DA 转换器，如果经电流椀缪棺?缓第涞觯?蛭?缪故涞鲈?/FONT> DA 转换器。此外，电压开关型电路为直接输出电压型 DA 转换器。

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。

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

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。）

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。

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。

三、实验器材

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

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、程序流程图


开始
初始化，设置中断允许
初始化液晶显示
通过外部中断INT0测速
通过定时器中断，调整转速
显示
结束

六、实验步骤

1 建立工程，实现实验内容 1
2 编写中断程序，测量电机转速
3 完成控制转速程序
4 完成整体实验内容

七、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;
sbit CS2=P1^6;
sbit E=P3^3;
sbit RW=P3^4;
sbit RS=P3^5;
sbit RES=P1^5;
sbit BUSY=P2^7;


sbit swh1=P3^6;//S1
sbit swh2=P3^7;//S2
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, 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,
///*"?"*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;//累加转数
uchar cspeed=0;//当前速度
uchar xspeed=120;//期望速度
uchar speedUp = 140;//转速上限
uchar speedLow =100;//转速下限

uchar t1_cnt=0; ///1s 控制延时 50ms*20?
int N=100;//占空比
int M=256;
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 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);
        send_all(1,4,(speedLow/10)%10);
        send_all(1,5,speedLow%10);
    send_all(3,3,cspeed/100);
        send_all(3,4,(cspeed/10)%10);
        send_all(3,5,cspeed%10);
      send_all(5,3,speedUp/100);
        send_all(5,4,(speedUp/10)%10);
        send_all(5,5,speedUp%10);

      delay1();
          display(cspeed);
    delay(50000);
    }
}

void init()
{    P4SW=0x30;

    IT0=1; ///设置 INT0 为边沿触发

    EA=1; //系统中断允许
    ET1=1;//定时器中断允许
    ET0=1;
    EX0=1;  //外部中断允许
/*
50ms  ==    50000
 ??  ==    65536-50000  D=  15536  D=  0x3CB0
 0.1ms ==    100
??  ==    65536-100    D=  65436  D=  0xFF9C
*/
```

```
    TMOD=0x11; ///设置定时器 0 和 1 的工作方式
    TH1=0x3C;
    TL1=0xB0;  ///50ms 计数值
    TH0=0xFF;
    TL0=0x9C;  ///0.1ms 计数值

    TR0=1;
    TR1=1;   ///启动定时器
}


void ex_int0() interrupt 0  //外部中断 INT0
{
    tspeed++;
}




void t1_int() interrupt 3   ///50ms 定时器中断 T1
{
    if(++t1_cnt<20)
    {        TH1=0x3C;
                    TL1=0xB0;
          if(swh1==0)
          {

            xspeed = speedLow;

          }


          if(swh2==0){

                xspeed = speedUp;

            }
            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
{
     TH0=0xFF;
         TL0=0x9C;
         X+=N;
     if(X>M)
     {
         motor=0;
         X-=M;
     }
     else
         motor=1;
}
void init_yejing()
{
    send_byte(192,1,1);
    send_byte(63,1,1);
}
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;
}

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);
    }
}

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);
                }
    }
}

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;
    }
}
//LED??
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++);
}
```

八、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。

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。

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。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。

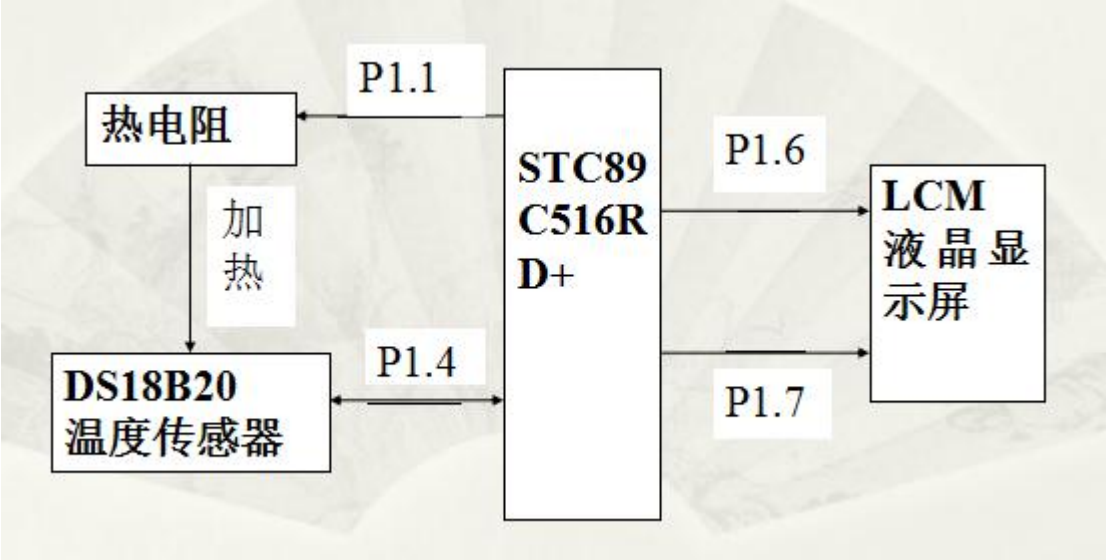
实验八 温度测量与控制

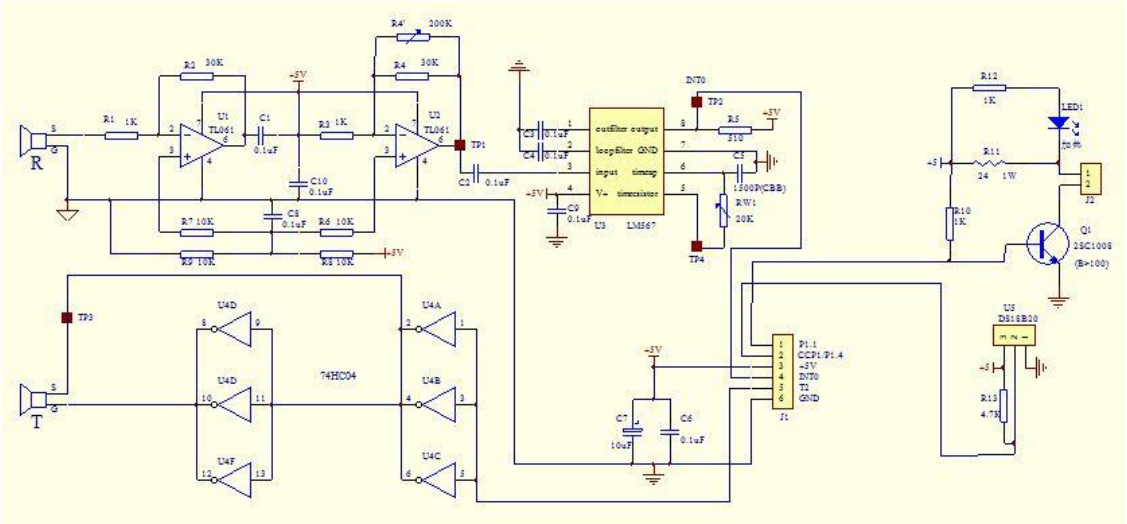
一、实验目的和要求

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。
3. 加深 C51 编程语言的理解和学习。

二、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：

如何获取温度值？——通过 DS18B20

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。

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。

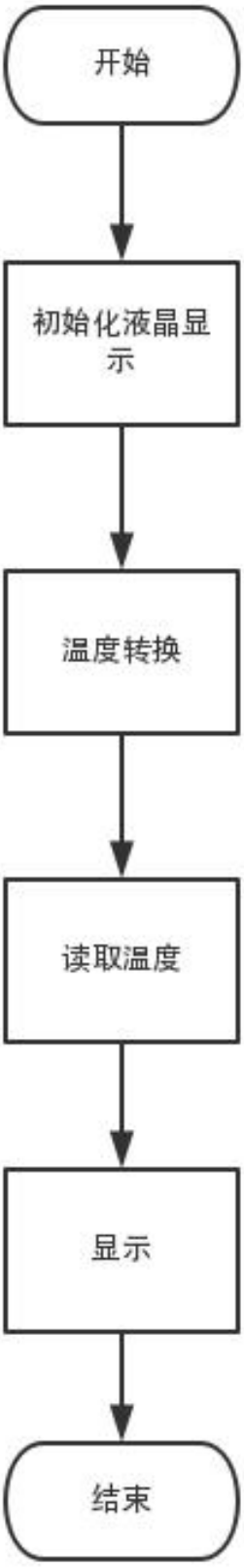
三、实验器材

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

四、实验内容

1、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。
2、编程实现测量当前教室的温度，显示在 LCM 液晶显示屏上。
3、通过 S1 设定一个高于当前室温的目标温度值。
4、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、程序流程图


开始
初始化液晶显示
温度转换
读取温度
显示
结束

六、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

七、实验代码

```
#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
,
0x00,0x04,0x06,0x05,0x05,0x04,0x24,0x24,0x24,0x3F,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/

0x10,0x21,0x86,0x70,0x00,0x7E,0x4A,0x4A,0x4A,0x4A,0x4A,0x7E,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);
                }
    }
}
```

八、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 Delay80μs( )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。

2. 参考其他资料，了解 DS18B20 的其他命令用法。