单片机控制与应用实验报告

班级：2016 级 8 班

学号：21160811

姓名：王京宇

实验五　重力测量实验

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 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

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

1. 液晶显示屏的控制方法:液晶显示屏驱动芯片 YM12864C 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。

2. 使用 PCtoLCD2002 字模软件提取自定义图形和文字的字模对应的字节。

3.字符点阵等数据，需要定义在 code 数据段中。

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）

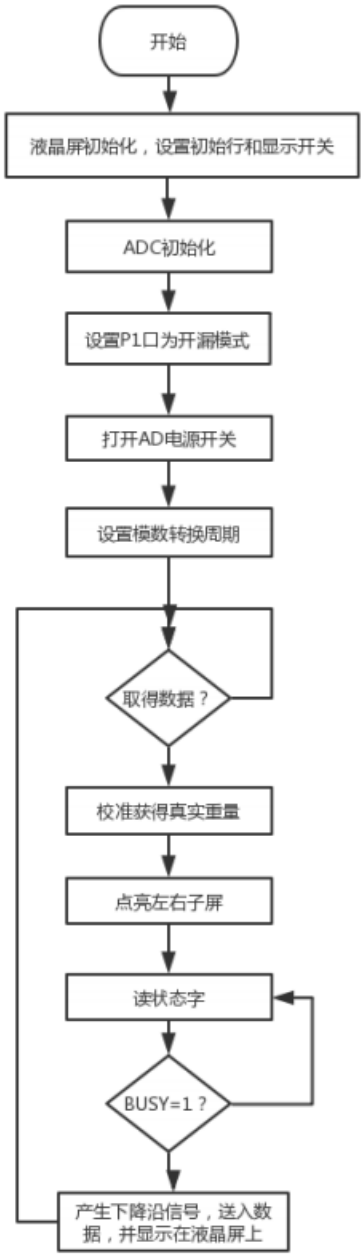
EA：CPU 的中断开放标志，EA=1，CPU 开放中断，EA=0，CPU 屏蔽所有中断申请。

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）

六、流程图



七、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
开始
初始化液晶屏
设置起始行并打开开关
初始化模/数
转换器
取得空载值/校准
使用拟和函数计算
实际重量
取值
送数据并
显示
延时
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]=
{
0x00,0x00,0xC0,0xE0,0x30,0x10,0x08,0x08,0x08,0x08,0x08,0x18,0
x30,0xE0,0xC0,0x00,
0x00,0x00,0x07,0x0F,0x18,0x10,0x20,0x20,0x20,0x20,0x20,0x10,0
x18,0x0F,0x07,0x00,///*"0"*0/
0x00,0x00,0x00,0x10,0x10,0x10,0x10,0xF0,0xF8,0x00,0x00,0x00,0
x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x20,0x20,0x20,0x20,0x3F,0x3F,0x20,0x20,0x20,0
x20,0x00,0x00,0x00,///*"1"*1/
0x00,0x00,0x60,0x50,0x10,0x08,0x08,0x08,0x08,0x08,0x08,0x98,0
xF0,0x70,0x00,0x00,
0x00,0x00,0x20,0x30,0x28,0x28,0x24,0x24,0x22,0x22,0x21,0x20,0
x30,0x18,0x00,0x00,///*"2"*2/
0x00,0x00,0x30,0x30,0x08,0x08,0x88,0x88,0x88,0x88,0x58,0x70,0
x30,0x00,0x00,0x00,
```

```
0x00,0x00,0x18,0x18,0x20,0x20,0x20,0x20,0x20,0x20,0x31,0x11,0
x1F,0x0E,0x00,0x00,///*"3"*3/
0x00,0x00,0x00,0x00,0x00,0x80,0x40,0x20,0x10,0xF0,0xF8,0xF8,0
x00,0x00,0x00,0x00,
0x00,0x04,0x06,0x05,0x05,0x04,0x24,0x24,0x24,0x3F,0x3F,0x3F,0
x24,0x24,0x24,0x00,///*"4"*4/
0x00,0x00,0x00,0xC0,0x38,0x88,0x88,0x88,0x88,0x88,0x88,0x88,0
x08,0x08,0x00,0x00,
0x00,0x00,0x18,0x29,0x21,0x20,0x20,0x20,0x20,0x20,0x30,0x11,0
x1F,0x0E,0x00,0x00,///*"5"*5/
0x00,0x00,0x80,0xE0,0x30,0x10,0x98,0x88,0x88,0x88,0x88,0x88,0
x98,0x10,0x00,0x00,
0x00,0x00,0x07,0x0F,0x19,0x31,0x20,0x20,0x20,0x20,0x20,0x20,0
x11,0x1F,0x0E,0x00,///*"6"*6/
0x00,0x00,0x30,0x18,0x08,0x08,0x08,0x08,0x08,0x88,0x48,0x28,0
x18,0x08,0x00,0x00,
0x00,0x00,0x00,0x00,0x00,0x00,0x38,0x3E,0x01,0x00,0x00,0x00,0
x00,0x00,0x00,0x00,///*"7"*7/
0x00,0x00,0x70,0x70,0xD8,0x88,0x88,0x08,0x08,0x08,0x08,0x98,0
x70,0x70,0x00,0x00,
0x00,0x0C,0x1E,0x12,0x21,0x21,0x20,0x21,0x21,0x21,0x23,0x12,0
x1E,0x0C,0x00,0x00,///*"8"*8/
0x00,0xE0,0xF0,0x10,0x08,0x08,0x08,0x08,0x08,0x08,0x18,0x10,0
xF0,0xC0,0x00,0x00,
0x00,0x00,0x11,0x33,0x22,0x22,0x22,0x22,0x22,0x32,0x11,0x1D,0
x0F,0x03,0x00,0x00,///*"9"*9/
0x08,0x08,0x0A,0xEA,0xAA,0xAA,0xAA,0xFF,0xA9,0xA9,0xA9,0xE9,0
x08,0x08,0x08,0x00,
0x40,0x40,0x48,0x4B,0x4A,0x4A,0x4A,0x7F,0x4A,0x4A,0x4A,0x4B,0
x48,0x40,0x40,0x00,///*"砝"*10/
0x40,0x40,0x40,0xDF,0x55,0x55,0x55,0xD5,0x55,0x55,0x55,0xDF,0
x40,0x40,0x40,0x00,
0x40,0x40,0x40,0x57,0x55,0x55,0x55,0x7F,0x55,0x55,0x55,0x57,0
x50,0x40,0x40,0x00,///*"码"*11/
0x00,0x00,0x00,0x00,0x00,0x00,0xC0,0xC0,0xC0,0xC0,0x00,0x00,0
x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,0x00,0x00,0x30,0x30,0x30,0x30,0x00,0x00,0
x00,0x00,0x00,0x00,///*"重"*12/
0x00,0x04,0x04,0xE4,0x24,0x24,0x24,0x3F,0x24,0x24,0x24,0xE4,0
x04,0x04,0x00,0x00,
0x00,0x00,0x80,0x43,0x31,0x0F,0x01,0x01,0x01,0x3F,0x41,0x43,0
x40,0x40,0x70,0x00,///*"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;01000
000|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);
}
}
```

八、思考题

1. 调零的原理，软件调零和硬件调零的区别。

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。

1）A/D 转换：模数转换器即 A/D 转换器，或简称 ADC，通常是指一个将模拟信号转变为数字信号的电子元件。通常的模数转换器是将一个输入电压信号转换为一个输出的数字信号。模数转换一般要经过采样（采样定理：当采样频率大于模拟信号中最高频率成分的两倍时，采样值才能不失真的反映原来模拟信号。）、保持和量化、编码这几个步骤。A/D 转换器的电路主要由时钟脉冲发生器、逻辑电路、移位寄存器电路及其开关指令数字寄存器构成。

2）D/A 转换：DAC 主要由数字寄存器、模拟电子开关、位权网络、求和运算放大器和基准电压源（或恒流源）组成。用存于数字寄存器的数字量的各位数码，分别控制对应位的模拟电子开关，使数码为 1 的位在位权网络上产生与其位权成正比的电流值，再由运算放大器对各电流值求和，并转换成电压值。可由三种方法实现：逐次逼近法、双积分法、电压频率转换法。

3. I2C 总线在信号通讯过程中的应用。

I 2 C 总线是一种两线式串行总线，用于连接微控制器及其外围设备。目前在视频处理、移动通信等领域采用 12C 总线接口器件已经比较普遍。另外，通用的 I 2 C 总线接口器件，如带 I 2 C 总线的单片机、RAM、ROM、A/D、D/A、LCD 驱动器等器件，也越来越多地应用于计算机及自动控制系统中。I 2 C 总线通过 SDA（串行数据线）及 SCL（串行时钟线）两根线在连到总线上的器件之间传送信息，并根据地址识别每个器件。目前在仪器仪表、移动通信、密码控制等领域采用 I 2 C 总线接口器件已经比较普遍。另外，通用的 I 2 C 总线接口器件，如带 I 2 C 总线的单片机、RAM、ROM、A/D、D/A、LCD 驱动器等器件，也越来越多地应用于计算机及自动控制系统中。

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. 根据输入修改电机的目标转速值，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

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

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。

五、实验原理

1.对于直流电机来说，其转速由输入电压决定，因此具有平滑调速的效果；相比而言，交流电机的转速由交流电频率和电机结构决定，难以改变速度。当然，交流电机构造简单，没有换向器，所以容易制造高转速、高电压、大电流、大容量的电机；而直流电机一般用在负荷小，但要求转速连续可调的场合，如伺服电机。

2. 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。使用 PWM 可以实现频率调制、电压调制等效果，并且需要的外围器件较少，特别适合于单片机控制领域。这里只关心通过 PWM 实现电压调制，从而控制直流电机转速的效果。也称作脉宽调制调速。

3. 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。

4. 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。

5. 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。

6. 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。

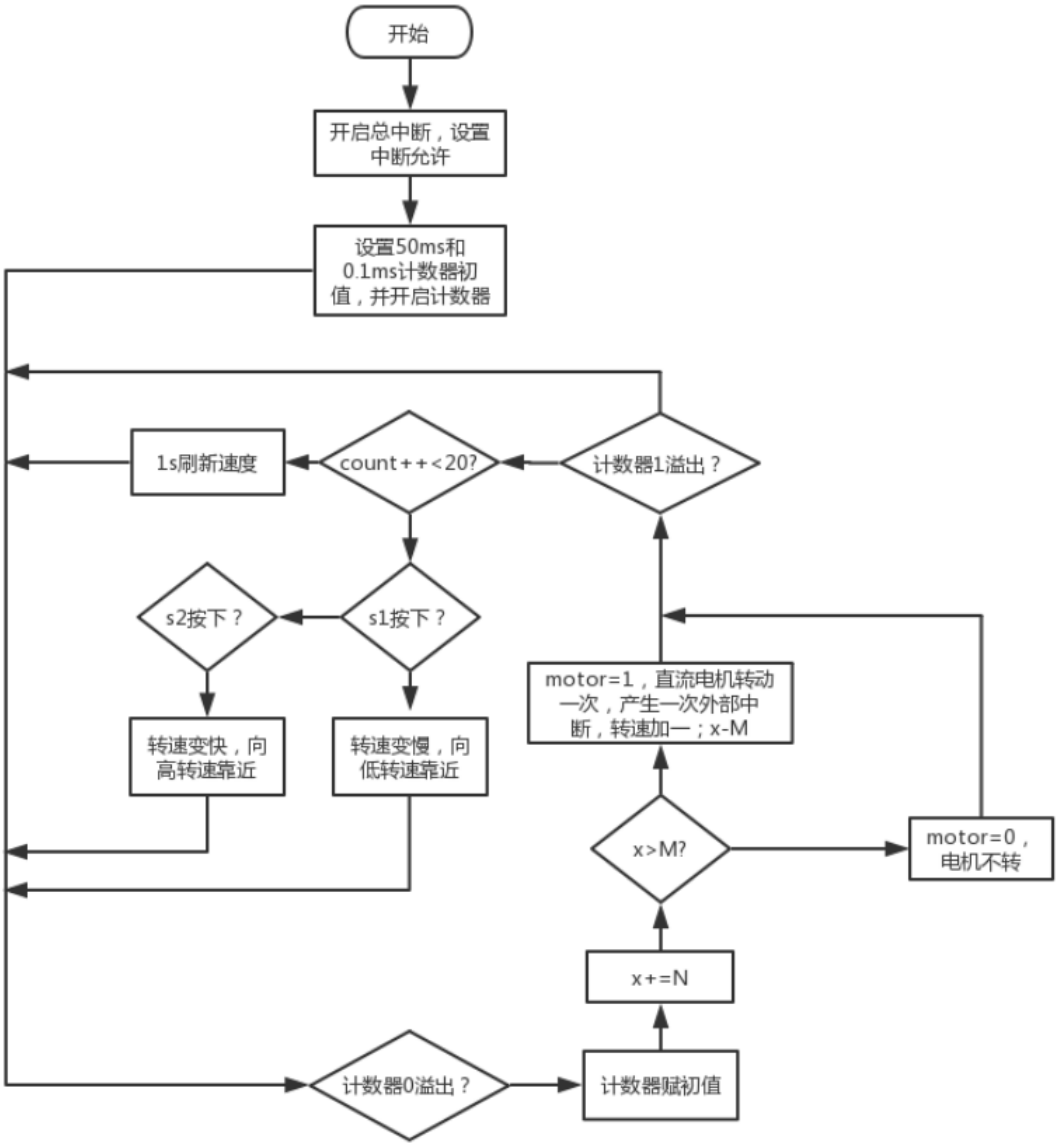
7. 进行转速控制时，涉及到三个变量：预期转速，实际转速和控制变量。这里控制变量就是占空比。我们并不能够预先精确知道某个控制变量的值会导致多少的实际转速，因为这里有很多内部和外部因素起作用（如摩擦力，惯性等），但可以确定就是随着控制变量的增加，实际转速会增加。

8. 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。

9. 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。

六、流程图



七、程序代码

#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]=
{
0x00,0x00,0xC0,0xE0,0x30,0x10,0x08,0x08,0x08,0x08,0x08,0x08,0x
18,0x30,0xE0,0xC0,0x00,
0x00,0x00,0x07,0x0F,0x18,0x10,0x20,0x20,0x20,0x20,0x20,0x20,0x
10,0x18,0x0F,0x07,0x00,///*"0"*0/
0x00,0x00,0x00,0x10,0x10,0x10,0x10,0xF0,0xF8,0x00,0x00,0x00,0x
00,0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x20,0x20,0x20,0x20,0x3F,0x3F,0x20,0x20,0x20,0x
20,0x00,0x00,0x00,///*"1"*1/
0x00,0x00,0x60,0x50,0x10,0x08,0x08,0x08,0x08,0x08,0x08,0x98,0x
F0,0x70,0x00,0x00,
0x00,0x00,0x20,0x30,0x28,0x28,0x24,0x24,0x22,0x22,0x21,0x20,0x
30,0x18,0x00,0x00,///*"2"*2/
0x00,0x00,0x30,0x30,0x08,0x08,0x88,0x88,0x88,0x88,0x58,0x70,0x
30,0x00,0x00,0x00,
0x00,0x00,0x18,0x18,0x20,0x20,0x20,0x20,0x20,0x20,0x31,0x11,0x
1F,0x0E,0x00,0x00,///*"3"*3/
0x00,0x00,0x00,0x00,0x00,0x80,0x40,0x20,0x10,0xF0,0xF8,0xF8,0x
00,0x00,0x00,0x00,
0x00,0x04,0x06,0x05,0x05,0x04,0x24,0x24,0x24,0x3F,0x3F,0x3F,0x
24,0x24,0x24,0x00,///*"4"*4/
0x00,0x00,0x00,0xC0,0x38,0x88,0x88,0x88,0x88,0x88,0x88,0x88,0x
08,0x08,0x00,0x00,
```

```
0x00,0x00,0x18,0x29,0x21,0x20,0x20,0x20,0x20,0x20,0x30,0x
11,0x1F,0x0E,0x00,0x00,///*"5"*5/
0x00,0x00,0x80,0xE0,0x30,0x10,0x98,0x88,0x88,0x88,0x88,0x
88,0x98,0x10,0x00,0x00,
0x00,0x00,0x07,0x0F,0x19,0x31,0x20,0x20,0x20,0x20,0x20,0x
20,0x11,0x1F,0x0E,0x00,///*"6"*6/
0x00,0x00,0x30,0x18,0x08,0x08,0x08,0x08,0x08,0x88,0x48,0x
28,0x18,0x08,0x00,0x00,
0x00,0x00,0x00,0x00,0x00,0x00,0x38,0x3E,0x01,0x00,0x00,0x
00,0x00,0x00,0x00,0x00,///*"7"*7/
0x00,0x00,0x70,0x70,0xD8,0x88,0x88,0x08,0x08,0x08,0x08,0x
98,0x70,0x70,0x00,0x00,
0x00,0x0C,0x1E,0x12,0x21,0x21,0x20,0x21,0x21,0x21,0x23,0x
12,0x1E,0x0C,0x00,0x00,///*"8"*8/
0x00,0xE0,0xF0,0x10,0x08,0x08,0x08,0x08,0x08,0x08,0x18,0x
10,0xF0,0xC0,0x00,0x00,
0x00,0x00,0x11,0x33,0x22,0x22,0x22,0x22,0x22,0x32,0x11,0x
1D,0x0F,0x03,0x00,0x00,///*"9"*9/
0x08,0x08,0x0A,0xEA,0xAA,0xAA,0xAA,0xFF,0xA9,0xA9,0xA9,0x
E9,0x08,0x08,0x08,0x00,
0x40,0x40,0x48,0x4B,0x4A,0x4A,0x4A,0x7F,0x4A,0x4A,0x4A,0x
4B,0x48,0x40,0x40,0x00,///*"?"*10/
0x40,0x40,0x40,0xDF,0x55,0x55,0x55,0xD5,0x55,0x55,0x55,0x
DF,0x40,0x40,0x40,0x00,
0x40,0x40,0x40,0x57,0x55,0x55,0x55,0x7F,0x55,0x55,0x55,0x
57,0x50,0x40,0x40,0x00,///*"?"*11/
0x00,0x00,0x00,0x00,0x00,0x00,0xC0,0xC0,0xC0,0xC0,0x00,0x
00,0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,0x00,0x00,0x30,0x30,0x30,0x30,0x00,0x
00,0x00,0x00,0x00,0x00,///*":"*12/
0x00,0x04,0x04,0xE4,0x24,0x24,0x24,0x3F,0x24,0x24,0x24,0x
E4,0x04,0x04,0x00,0x00,
0x00,0x00,0x80,0x43,0x31,0x0F,0x01,0x01,0x01,0x3F,0x41,0x
43,0x40,0x40,0x70,0x00,///*"?"*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（16 位计数），两个内部
中断
 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++);
}
```

}

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。

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

五、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—55℃到

+125℃，增量值为 0.5 ℃。

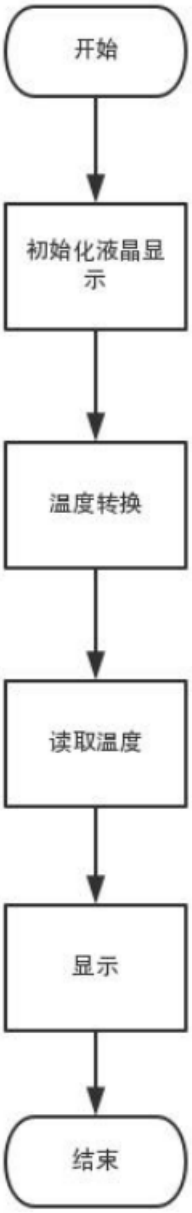
用于贮存测得的温度值的两个 8 位存贮器 RAM 编号为 0 号和 1 号。

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。

六、流程图



七、程序代码

```
#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/
0x00,0x00,0x00,0xC0,0x38,0x88,0x88,0x88,0x88,0x88,0x88,0x08,0x08,0x00,0x00
,
0x00,0x00,0x18,0x29,0x21,0x20,0x20,0x20,0x20,0x20,0x30,0x11,0x1F,0x0E,0x00,0x00,
///*"5"*5/
0x00,0x00,0x80,0xE0,0x30,0x10,0x98,0x88,0x88,0x88,0x88,0x98,0x10,0x00,0x00
,
0x00,0x00,0x07,0x0F,0x19,0x31,0x20,0x20,0x20,0x20,0x20,0x11,0x1F,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。