姓名：许媛媛 学号：21160802

实验五　重力测量实验

一．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）

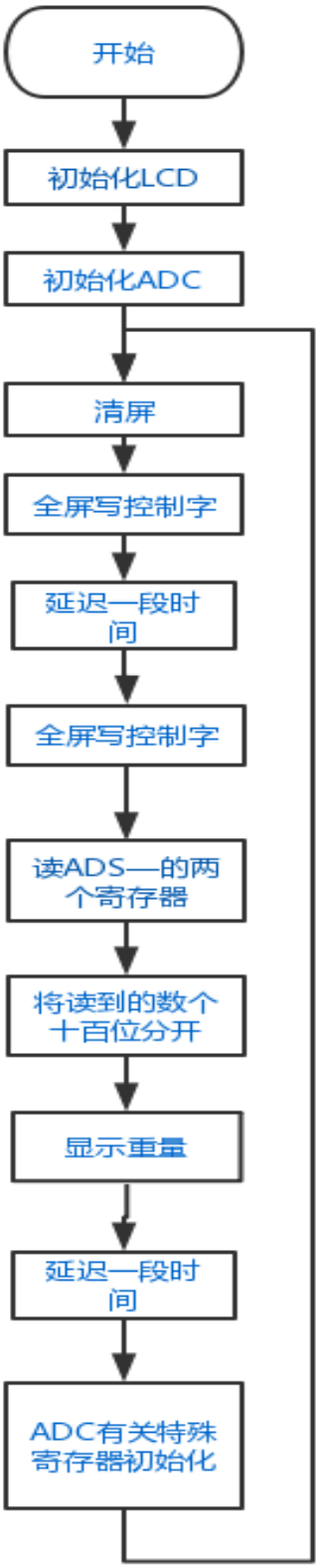
EA：CPU 的中断开放标志，EA=1，CPU 开放中断，EA=0，CPU 屏蔽所有中断申请。

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）

二．实验流程图：


开始
初始化LCD
初始化ADC
清屏
全屏写控制字
延迟一段时间
全屏写控制字
读ADS—的两个寄存器
将读到的数个十百位分开
显示重量
延迟一段时间
ADC有关特殊寄存器初始化

三．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。

四．实验代码：

```
#include <reg52.H>
#include <intrins.H>
typedef unsigned char UCHAR;
typedef unsigned int    UINT;

/***********ADC**********/


sfr ADC_CONTR      = 0xBC;
sfr ADC_RESL2      = 0xBE;
sfr P1ASF          = 0x9D;
#define ADC_POWER      0x80
#define ADC_START        0x08
#define ADC_SPEEDL    0x20          //360 clocks
#define ADC_SPEEDH    0x40          //180 clocks
#define ADC_SPEEDHH 0x60          // 90 clocks

void get_result()
{
      ADC_CONTR &= !ADC_FLAG;
      ADC_CONTR    = ADC_POWER | ADC_SPEEDLL | ADC_START;
      ADC_RES       =0;
}

void init_ADC( )
{
      P1ASF          = 0x01;
      ADC_RES      = 0;
      ADC_CONTR = ADC_POWER | ADC_SPEEDLL | ADC_START;
      delay();
}

/***********LCM*****************/
```

```
sbit RS      = P3^5;
sbit RW      = P3^4;
sbit E       = P3^3;
sbit CS1     = P1^7;
void write_left_cmd    (UCHAR comd);

UCHAR                                code                                charcode1[]
={0x10,0x10,0x14,0xD4,0x54,0x54,0x54,0xFC,0x52,0x52,0x52,0xD3,0x12,0x10,0x10,0x00,

0x40,0x40,0x50,0x57,0x55,0x55,0x55,0x7F,0x55,0x55,0x55,0x57,0x50,0x40,0x40,0x00,

0x20,0x20,0x20,0xBE,0xAA,0xAA,0xAA,0xAA,0xAA,0xAA,0xAA,0xBE,0x20,0x20,0x20,0x00,

0x00,0x80,0x80,0xAF,0xAA,0xAA,0xAA,0xFF,0xAA,0xAA,0xAA,0xAF,0x80,0x80,0x00,0x00,

0x10,0x60,0x02,0x8C,0x00,0xFE,0x02,0xF2,0x02,0xFE,0x00,0xF8,0x00,0xFF,0x00,0x00,

0x04,0x04,0x7E,0x01,0x80,0x47,0x30,0x0F,0x10,0x27,0x00,0x47,0x80,0x7F,0x00,0x00,

0x10,0x60,0x02,0x8C,0x00,0xFE,0x02,0xF2,0x02,0xFE,0x00,0xF8,0x00,0xFF,0x00,0x00,

0x04,0x04,0x7E,0x01,0x80,0x47,0x30,0x0F,0x10,0x27,0x00,0x47,0x80,0x7F,0x00,0x00,

0x20,0x20,0x20,0xBE,0xAA,0xAA,0xAA,0xAA,0xAA,0xAA,0xAA,0xBE,0x20,0x20,0x20,0x00,

0x00,0x80,0x80,0xAF,0xAA,0xAA,0xAA,0xFF,0xAA,0xAA,0xAA,0xAF,0x80,0x80,0x00,0x00,

0x20,0x30,0xAC,0x63,0x20,0x18,0x08,0x48,0x48,0x48,0x7F,0x48,0x48,0x48,0x08,0x00,

0x22,0x67,0x22,0x12,0x12,0x12,0x00,0xFE,0x42,0x42,0x42,0x42,0x42,0xFE,0x00,0x00};

UCHAR                                code                                charcode2[]
={0x20,0x20,0x20,0xBE,0xAA,0xAA,0xAA,0xAA,0xAA,0xAA,0xAA,0xBE,0x20,0x20,0x20,0x00,

0x00,0x80,0x80,0xAF,0xAA,0xAA,0xAA,0xFF,0xAA,0xAA,0xAA,0xAF,0x80,0x80,0x00,0x00,

0x10,0x0C,0x04,0x84,0x14,0x64,0x05,0x06,0xF4,0x04,0x04,0x04,0x04,0x14,0x0C,0x00,

0x04,0x84,0x84,0x44,0x47,0x24,0x14,0x0C,0x07,0x0C,0x14,0x24,0x44,0x84,0x04,0x00,

0x02,0xFA,0x82,0x82,0xFE,0x80,0x40,0x20,0x50,0x4C,0x43,0x4C,0x50,0x20,0x40,0x00,

0x08,0x18,0x48,0x84,0x44,0x3F,0x40,0x44,0x58,0x41,0x4E,0x60,0x58,0x47,0x40,0x00,
```

```
0x00,0x00,0x00,0xFE,0x92,0x92,0x92,0xFE,0x92,0x92,0x92,0xFE,0x00,0x00,0x00,0x00,

0x44,0x44,0x24,0x25,0x14,0x0C,0x04,0xFF,0x04,0x0C,0x14,0x25,0x24,0x44,0x44,0x00,

0x00,0x00,0x80,0x80,0x80,0x80,0x80,0x00,0x00,0x6B,0x94,0x94,0x94,0x93,0x60,0x00,

0x00,0xE0,0x10,0x08,0x08,0x10,0xE0,0x00,0x00,0x0F,0x10,0x20,0x20,0x10,0x0F,0x00,

0x00,0x00,0x10,0x10,0xF8,0x00,0x00,0x00,0x00,0x00,0x20,0x20,0x3F,0x20,0x20,0x00,   //"1",12

0x00,0x70,0x08,0x08,0x08,0x08,0xF0,0x00,0x00,0x30,0x28,0x24,0x22,0x21,0x30,0x00,   //"2",13

0x00,0x30,0x08,0x08,0x08,0x88,0x70,0x00,0x00,0x18,0x20,0x21,0x21,0x22,0x1C,0x00,   //"3",14

0x00,0x00,0x80,0x40,0x30,0xF8,0x00,0x00,0x00,0x06,0x05,0x24,0x24,0x3F,0x24,0x24,   //"4",15

0x00,0xF8,0x88,0x88,0x88,0x08,0x08,0x00,0x00,0x19,0x20,0x20,0x20,0x11,0x0E,0x00,   //"5",16

0x00,0xE0,0x10,0x88,0x88,0x90,0x00,0x00,0x00,0x0F,0x11,0x20,0x20,0x20,0x1F,0x00,   //"6",17

0x00,0x18,0x08,0x08,0x88,0x68,0x18,0x00,0x00,0x00,0x00,0x3E,0x01,0x00,0x00,0x00,   //"7",18

0x00,0x70,0x88,0x08,0x08,0x88,0x70,0x00,0x00,0x1C,0x22,0x21,0x21,0x22,0x1C,0x00,   //"8",19

0x00,0xF0,0x08,0x08,0x08,0x10,0xE0,0x00,0x00,0x01,0x12,0x22,0x22,0x11,0x0F,0x00};  //"9",20


void judge_lbusy()
{
    P2 = 0xFF;
    CS1=1;
    CS2=0;
    RS=     0;
    RW=     1;
    E=      1;
    while(BUSY);
    E=0;
}

void delay()
{
    UINT m;
    for (m=0;m<10;m++);
```

```
}

void write_left_cmd(UCHAR comd)
{
	judge_lbusy();
	CS1=1;
	CS2=0;
	RS=0;
	RW=0;
	E=1;
	P2 = comd;
	delay();
	E=0;
}

void write_left_data(UCHAR ldata)
{
	judge_lbusy();
	CS1=1;
	CS2=0;
	RS=1;
	RW=0;
	E=1;
	P2=ldata;
	delay();
	E=0;
}

void wl_datablock(UCHAR page,UCHAR cow,UCHAR bs,UCHAR* start)
{
	UCHAR setpage = 0xB8+page;
	UCHAR setcow    = 0x40+cow;
	UCHAR i;
	write_left_cmd(setpage);
	write_left_cmd(setcow);
	for(i=0;i<bs;i++)
	{
		write_left_data(*start);
		start++;
	}
}

void judge_rbusy()
```

```
{
    P2 = 0xFF;
    CS1=0;
    CS2=1;
    RS=0;
    RW=1;
    E=1;
    while(BUSY);
    E=0;
}

void write_right_cmd(UCHAR comd)
{
    judge_rbusy();
    CS1=0;
    CS2=1;
    RS=0;
    RW=0;
    E=1;
    P2 = comd;
    delay();
    E=0;
}

void write_right_data(UCHAR rdata)
{
    judge_rbusy();
    CS1=0;
    CS2=1;
    RS=1;
    RW=0;
    E=1;
    P2=rdata;
    delay();
    E=0;
}

void wr_datablock(UCHAR page,UCHAR cow,UCHAR bs,UCHAR* start)
{
    UCHAR setpage = 0xB8+page;
    UCHAR setcow = 0x40+cow;
    UCHAR i,n=bs;

    write_right_cmd(setpage);
```

```
        write_right_cmd(setcow);
        for(i=0;i<n;i++)
        {
            write_right_data(*start);
            start++;
        }
}

void clear_screen()
{
        UCHAR i,j,page=0xB8;
        for(i=0;i<8;i++)
        {
            write_left_cmd(page);
            for(j=0;j<64;j++)
                write_left_data(0x00);
            write_right_cmd(page);
            write_right_cmd(0x40);
            for(j=0;j<64;j++)
                write_right_data(0x00);
            page++;
        }
}

void show(UCHAR m1,UCHAR m2,UCHAR m3)
{
        wl_datablock(2,16,16,&charcode1[0]);
        wl_datablock(2,32,16,&charcode1[32]);
        wl_datablock(3,32,16,&charcode1[48]);
        wl_datablock(2,48,16,&charcode1[64]);
        wl_datablock(3,48,16,&charcode1[80]);
        wl_datablock(4,16,16,&charcode1[96]);
        wl_datablock(5,16,16,&charcode1[112]);
        wl_datablock(4,32,16,&charcode1[128]);
        wl_datablock(5,32,16,&charcode1[144]);
        wl_datablock(4,48,16,&charcode1[160]);
        wl_datablock(5,48,16,&charcode1[176]);

        wr_datablock(2,0,16, &charcode2[0]);
        wr_datablock(2,16,16,&charcode2[32]);
        wr_datablock(3,16,16,&charcode2[48]);
        wr_datablock(2,32,16,&charcode2[64]);
        wr_datablock(3,32,16,&charcode2[80]);
        wr_datablock(4,0,16, &charcode2[96]);
```

```
        wr_datablock(5,0,16, &charcode2[112]);
        wr_datablock(4,16,8, &charcode2[144+16*m1]);
        wr_datablock(5,16,8, &charcode2[144+16*m1+8]);
        wr_datablock(4,24,8, &charcode2[144+16*m2]);
        wr_datablock(5,24,8, &charcode2[144+16*m2+8]);
        wr_datablock(4,32,8, &charcode2[144+16*m3]);
        wr_datablock(5,32,8, &charcode2[144+16*m3+8]);
        wr_datablock(4,40,8, &charcode2[128]);
        wr_datablock(5,40,8, &charcode2[136]);
}

void main()
{
        init_ADC();
        while(1)
        {
                UCHAR x,y,z;
                UINT i,j;
                write_left_cmd (0x3F);
                write_left_cmd (0xC0);
                write_left_cmd (0xB8);
                write_right_cmd(0xB8);
                write_left_cmd (0x40);
                write_right_cmd(0x40);

                clear_screen();
                delay();

                write_left_cmd(0x3F) ;
                write_left_cmd(0xC0) ;
                write_left_cmd(0xB8) ;
                write_right_cmd(0xB8);
                write_left_cmd(0x40) ;
                write_right_cmd(0x40);

                i=ADC_RES;
                j=ADC_RESL2 & 0x03;
                switch(j)
                {
                case 0:
                        i*=4;
                        break;
                case 1:
                        i*=4;
```

```
            i++;
            break;
        case 2:
            i*=4;
            i+=2;
            break;
        case 3:
            i*=4;
            i+=3;
            break;
        }

        x=i/100;
        i=i%100;
        y=i/10;
        z=i%10;
        show(x,y,z);
        delay();
        get_result();           }
}
```

五．实验中的问题及分析：

此次实验需要软件调零，在未放砝码之前，液晶屏显示不为 0 时，就减去比显示屏小一的数，不减去整好的数是为了使其不出现溢出等问题。

六、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 I2C 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 I2C 总线接口器件已经比较普遍。另外，通用的 I2C 总线接口器件，如带 I2C 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。

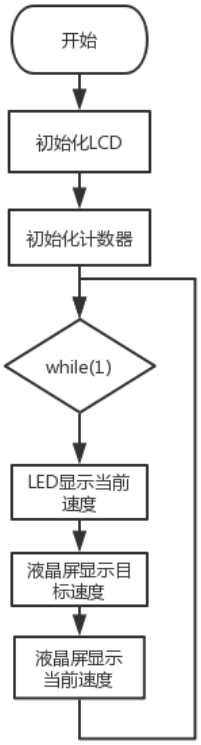
7. 进行转速控制时，涉及到三个变量：预期转速，实际转速和控制变量。这里控制变量就是占空比。我们并不能够预先精确知道某个控制变量的值会导致多少的实际转速，因为这里有很多内部和外部因素起作用（如摩擦力，惯性等），但可以确定就是随着控制变量的增加，实际转速会增加。

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。

二．实验流程图：



三．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。

四．实验代码：

```
#include <reg52.h>
#include <stdio.h>
#include <intrins.h>

#define TIMER 20
#define MAXSPEED 180
#define MIDSPEED 90
#define MINSPEED 50

 //sbit 定?特殊功能寄存器的值
sbit CS1=P1^7;      //液晶片?，左，高有效 64*64 一??刻只能有一??示屏操作
sbit CS2=P1^6;      //右

sbit RST=P1^5;
sbit E=P3^3;      //使能端
sbit RW=P3^4;     //读写操作
sbit RS=P3^5;     //寄存器选择
sbit BUSY=P2^7;   //BUSY=0 LCM “准备好“等待单片机访问
sbit RESET=P2^4; //正常时 reset=0 低电平有效
//定义开关
sbit KEY1=P3^6;     //开关 S1
sbit KEY2=P3^7;



sbit OUT=P1^1;
 //那 y??1 邦
sfr P4=0xc0;      //初始化     P4 端口地址
sfr P4SW=0xbb;      //P4SW 默认值   驱动?
sbit DCLK=P4^4;   //模拟串口时钟
sbit LED=P4^5;     //模拟串口数据

int i,j,k,a,b,c,d,temp;
int timer=TIMER,count=0,countS=0,currentSpeed=0,objSpeed=0;
int SUM=0,N=30,M=256;     //N M 为了求控制变量即占空比 控制转速

// char code 声明什么的意思是将数据放在 ROM（程序存储区中）不能更改
char code ledCode[10]={0xC0,0xF9,0xA4,0xB0,0x99,0x92,0x82,0xF8,0x80,0x90};   //0-9 的 16 进
```

制数表示 可以用数码管显示

```
char code mubiao[4][16]={

    {0x00,0x00,0xFE,0x22,0x22,0x22,0x22,0x22,0x22,0x22,0x22,0x22,0xFE,0x00,0x00,0x00},

    {0x00,0x00,0xFF,0x42,0x42,0x42,0x42,0x42,0x42,0x42,0x42,0x42,0xFF,0x00,0x00,0x00},/*"?
",0*/

    {0x10,0x10,0xD0,0xFF,0x90,0x10,0x20,0x22,0x22,0x22,0xE2,0x22,0x22,0x22,0x20,0x00},

    {0x04,0x03,0x00,0xFF,0x00,0x13,0x0C,0x03,0x40,0x80,0x7F,0x00,0x01,0x06,0x18,0x00},/*"?
",1*/
                                };
char code dangqian[4][16]={

    {0x00,0x40,0x42,0x44,0x58,0x40,0x40,0x7F,0x40,0x40,0x50,0x48,0xC6,0x00,0x00,0x00},

    {0x00,0x40,0x44,0x44,0x44,0x44,0x44,0x44,0x44,0x44,0x44,0x44,0xFF,0x00,0x00,0x00},/*"?
",0*/



    {0x08,0x08,0xE8,0x29,0x2E,0x28,0xE8,0x08,0x08,0xC8,0x0C,0x0B,0xE8,0x08,0x08,0x00},

    {0x00,0x00,0xFF,0x09,0x49,0x89,0x7F,0x00,0x00,0x0F,0x40,0x80,0x7F,0x00,0x00,0x00},/*"?"
,1*/
                                };
        //LCM 显示屏需要的 0-9 数字
char                                                                    code
number[10][16]={{0x00,0xE0,0x10,0x08,0x08,0x10,0xE0,0x00,0x00,0x0F,0x10,0x20,0x20,0x10,0x
0F,0x00},/*"0",0*/

    {0x00,0x10,0x10,0xF8,0x00,0x00,0x00,0x00,0x00,0x20,0x20,0x3F,0x20,0x20,0x00,0x00},/*"1
",1*/

    {0x00,0x70,0x08,0x08,0x08,0x88,0x70,0x00,0x00,0x30,0x28,0x24,0x22,0x21,0x30,0x00},/*"
2",2*/

    {0x00,0x30,0x08,0x88,0x88,0x48,0x30,0x00,0x00,0x18,0x20,0x20,0x20,0x11,0x0E,0x00},/*"
3",3*/

    {0x00,0x00,0xC0,0x20,0x10,0xF8,0x00,0x00,0x00,0x07,0x04,0x24,0x24,0x3F,0x24,0x00},/*"4
",4*/
```

```
    {0x00,0xF8,0x08,0x88,0x88,0x08,0x08,0x00,0x00,0x19,0x21,0x20,0x20,0x11,0x0E,0x00},/*"5",5*/

    {0x00,0xE0,0x10,0x88,0x88,0x18,0x00,0x00,0x00,0x0F,0x11,0x20,0x20,0x11,0x0E,0x00},/*"6",6*/

    {0x00,0x38,0x08,0x08,0xC8,0x38,0x08,0x00,0x00,0x00,0x00,0x3F,0x00,0x00,0x00,0x00},/*"7",7*/

    {0x00,0x70,0x88,0x08,0x08,0x88,0x70,0x00,0x00,0x1C,0x22,0x21,0x21,0x22,0x1C,0x00},/*"8",8*/

    {0x00,0xE0,0x10,0x08,0x08,0x10,0xE0,0x00,0x00,0x00,0x31,0x22,0x22,0x11,0x0F,0x00},/*"9",9*/};

void wait(unsigned int count){        //延时程序
    while(count--){
        _nop_();                          //程序什么都不执行
    }
}
   //将数据传递到数码管 led
void send2LED(char temp){
    for(d=0;d<8;d++){
        DCLK=0;         //P4.4 串口时针    当有上升沿，会将数据传过去
        LED=temp & 0x80;    //P4.5 串口数据      串口的要求是数据一位一位的传   所以在这里还是通过移位   做逻辑与 取一位数据
        DCLK=1;         //制造上升沿，先传一位数据
        temp<<=1;   //左移一
    }
}
    //输出数码管
void outLed(int num){
    a=num/100;                //百位
    b=(num-a*100)/10;     //十位
    c=num-a*100-b*10;   //个位
    send2LED(ledCode[c]);      //去表格中找对应数据传递 先传个位
    send2LED(ledCode[b]);
    send2LED(ledCode[a]);
}

     //检查显示屏 是否可用
void checkReady(bit cs){
    CS1= cs==0 ? 1:0;
    CS2= cs==1 ? 1:0;
```

```
    P2=0xff;   //通过 P2 口把数据输出到屏幕

    E=1;
    RS=0;
    RW=1;    //读状态 高电平有效

    while(BUSY==1);
    E=0;//置低电平 读取结束
    //每次读取 写之后屏幕控制都得清零
    CS2=0;
    CS1=0;
}
//D∩?邦芍?log?谷豕??????
//写指令
void write_command(bit cs,char com){
    checkReady(cs);
    CS1= cs==0 ? 1:0;
    CS2= cs==1 ? 1:0;
    //输出对变量的要求
    RS=0;
    RW=0;
    E=1;        //使能电平   下降沿 写指令代码
    P2=com;   //P2 是 P2.0-P2.7 表示 DB0-DB7 输入线
    wait(20);
    E=0;
    //每次读取 写之后屏幕控制都得清零
    CS1=0;
    CS2=0;
}
//清屏
void cls(bit cs){
    for(i=0;i<8;i++){
        write_command(cs,0xb8+i);
        for(j=0;j<64;j++){
            checkReady(cs);
            CS1= cs==0 ? 1:0;
            CS2= cs==1 ? 1:0;
            //写显示数据，下降沿时写
            E=1;
            RS=1;
            RW=0;
            P2=0x00;
            wait(20);   //PPT 上要求先延时一会在输出
            E=0;        //制造下降沿
```

```
            }
        }
}
    //输入字符
void write_bytes(bit cs,char * buffer,int len){
    for(i=0;i<len;i++){
        checkReady(cs);      //检查屏幕是否正在读取数据   cs 的值判断哪一半的屏幕
        CS1= cs==0 ? 1:0;
        CS2= cs==1 ? 1:0;
        E=1;
        RS=1;
        RW=0;
        P2=*(buffer+i);
        wait(20);
        E=0;
    }
}
//展示文字
void outChina(bit cs,char x,char y,char china[][16]){
    //字的上半部分
    write_command(cs,x+0xb8);   //设置行八组第一    0xb8 是选页
    write_command(cs,y+0x40); //设置列    0x40 是选列 默认值
    write_bytes(cs,*china,16);
    //字的下半部分
    write_command(cs,x+1+0xb8);
    write_command(cs,y+0x40);
    write_bytes(cs,*(china+1),16);
}
//展示数字
void outNumber(bit cs,char x,char y,int num){
    //上半部分的字
    write_command(cs,x+0xb8);
    write_command(cs,y+0x40);
    write_bytes(cs,number[num],8);   //8 列是因为数字的宽度是文字宽度的 1/2
    //下半部分的字
    write_command(cs,x+1+0xb8);
    write_command(cs,y+0x40);
    write_bytes(cs,number[num]+8,8);
}

//展示到 LCD 屏上 现在的数据
void showNow(int num){
    a=num/1000;        //千位
    b=(num-a*1000)/100;
```

```
    c=(num-a*1000-b*100)/10;
    d=num-a*1000-b*100-c*10;      //个位
    outNumber(1,4,43,d);   //左半屏 第四页（垂直方向分八页？） 43 列开始个位
    outNumber(1,4,33,c);
    outNumber(1,4,23,b);
    outNumber(1,4,13,a);
}

//显示目标数据到 LCD 上
void showTarget(int num){
    a=num/1000; //千位
    b=(num-a*1000)/100;
    c=(num-a*1000-b*100)/10;
    d=num-a*1000-b*100-c*10;
    outNumber(0,4,43,d);   //右半屏 第四页 43 列开始
    outNumber(0,4,33,c);
    outNumber(0,4,23,b);
    outNumber(0,4,13,a);
}
//初始化 LCD
void initLCD(){
    cls(1);
    write_command(0,0x3f);      // 3f 为啥？？
    cls(0);
    write_command(1,0x3f);
    outChina(0,1,5,mubiao);    //右半屏 第一页第五列 目字
    outChina(0,1,35,mubiao+2);   //标字 加 2 是因为是数组形式存储的

    outChina(1,1,5,dangqian);
    outChina(1,1,35,dangqian+2);
}

//中断程序 INT0   转一圈加一
void ex_int0() interrupt 0      //interrupt 表明当前是一个中断函数,不需要被主函数直接或间接调用.interrupt 后的数字表明是中断号几,
                                //单片机中 51 系列的有 0 1 2 3 4 等几个中断, 中断号与中断事件是绑定的，不能随便设置,对应的中断向量会指向这个函数入口地址
                                //可选的 using N 表示中断中使用第几个寄存器组
{
    count=count+1;
}
//当前值与目标值的差距转换成占空比
int getStepLen(int num){
    if(num<6){          //小于 6 的时候占空比不改变
```

```
        return 0;
    }
    if(num<10){          //6--10 的时候加 1
        return 1;
    }
    temp=num/100;     //百位
    temp=(num-temp*100)/10;     //十位
    temp=temp/3*2+1;
    if((N-temp)<1){     //N<=temp
        return 1;
    }
    return temp;    //N>temp
}
//定时器 0 中断   为了测一秒钟有多少个中断
void t0_int0() interrupt 1
{
    TR0=0;   //计数器 0 停止计数

    timer=timer-1;   //timer 是干啥的？、
    if(timer==0){
    /*   if(countS==0){
            countS=count;
            currentSpeed=count*2;
        }else{
            currentSpeed=count+countS;
            countS=0;
        }*/
        currentSpeed=count;
        timer=TIMER;
        count=0;
        //?迄?Y?迄?豕米﹣?迪 N
        if(currentSpeed<objSpeed){
            N=N+getStepLen(objSpeed-currentSpeed);
        }
        if(currentSpeed>objSpeed){
            N=N-getStepLen(currentSpeed-objSpeed);
        }
    }
    //计时器 0     t=0.05ms
    TH0=0x3c;
    TL0=0xb0;
    TR0=1; //计时器开始计数
}
//定时器 1 中断
```

```
void t1_int0() interrupt 3
{
    TR1=0;

    SUM=SUM+N;    //累加进位 每次加 N
    if((!KEY1)&&(!KEY2)){        //两个开关都按下时
        objSpeed=MIDSPEED;
    }else{
        if(!KEY1){
            objSpeed=MAXSPEED;          //S1 按
        }
        if(!KEY2){                      //S2 按
            objSpeed=MINSPEED;
        }
        if(KEY1 && KEY2){               //都没按
            objSpeed=MIDSPEED;
        }
    }
        if(SUM>M){
            OUT=0;          //调整占空比，本实验要求是 0 和 1 反过来使用 0 占比
            SUM=SUM-M;    //输出一个 0
        }else{
            OUT=1;
        }

    //计数初值  0.1MS=t
    TH1=0xff;
    TL1=0x9c;
    TR1=1;
}
//初始化计时器的初始值
void initTimer(){
    P4SW=0x30;

    TMOD=0x11;  //工作方式选择方式 1          //求一下这个值怎么求？
    TH0=0x3c;    //T0 计数器  16 位
    TL0=0xb0;

    TH1=0xff;          //T1 计数器 16 位 //求一下？？
    TL1=0x9c;



    IT0=1;
    EA=1;  //整个 CPU 的中断标志  为 1 时可以响应中断
```

```
    ET0=1;   //T0 中断允许位   为 1 时允许响应
    ET1=1;   //T1 中断允许
    EX0=1;    //INT0 的终中断允许位 为 1  允许

    TR0=1;   //计时器 0 是否允许计数
    TR1=1;
}
//长延时
void waitL(int count){
    k=0xffff;
    while(count--){
        while(k--);
    }
}
void main(){
    initLCD();
    initTimer();
    while(1){
            outLed(currentSpeed);
            showTarget(objSpeed);
            showNow(currentSpeed);
        }
        waitL(0x05);
}
```

五．实验中的问题及分析：

由直流电机旁的光电装置记录直流电机转动的圈数，再设置一个一秒钟的中断，实现每秒钟读一次直流电机转动的圈数既得直流电机的当前转速；又设置一个中断，用来判断是 S1 S2 的按下请况，用来修改目标转速，由目标转速以及当前转速的差值，根据一个固定的正比函数来调整占空比，逐步实现靠近目标转速。

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 N 或减小 M。

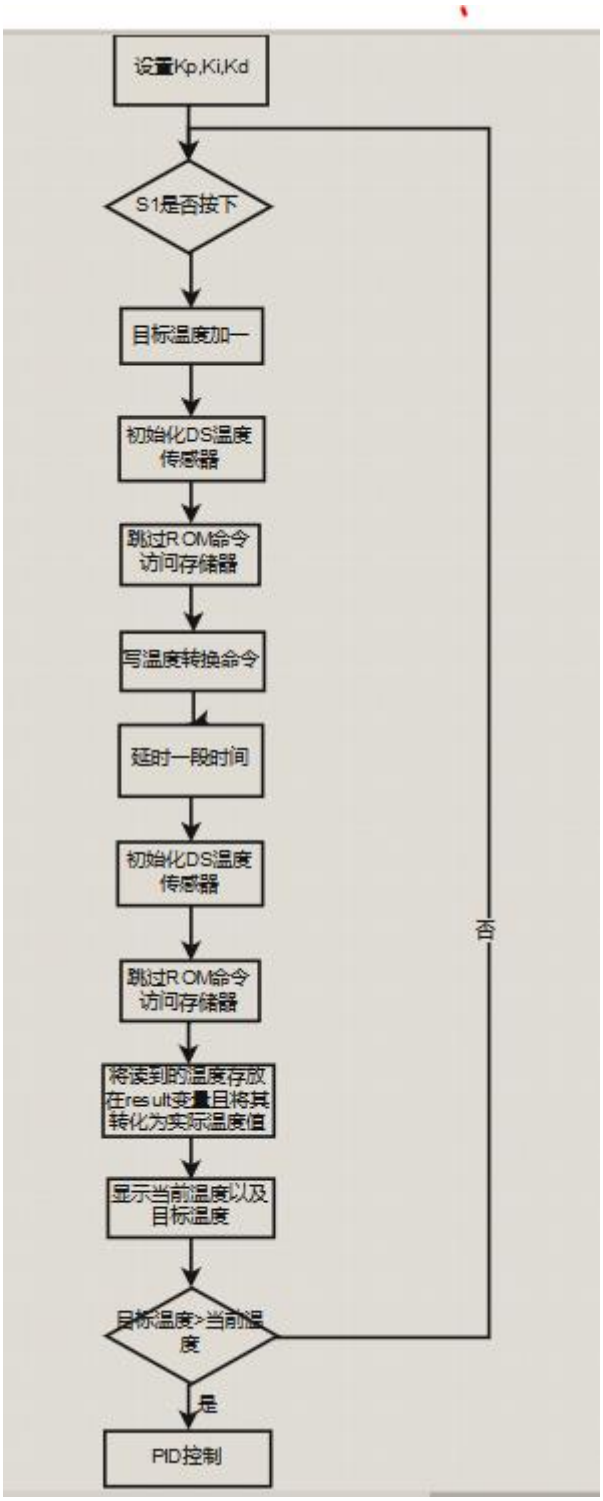
实验八　温度测量与控制

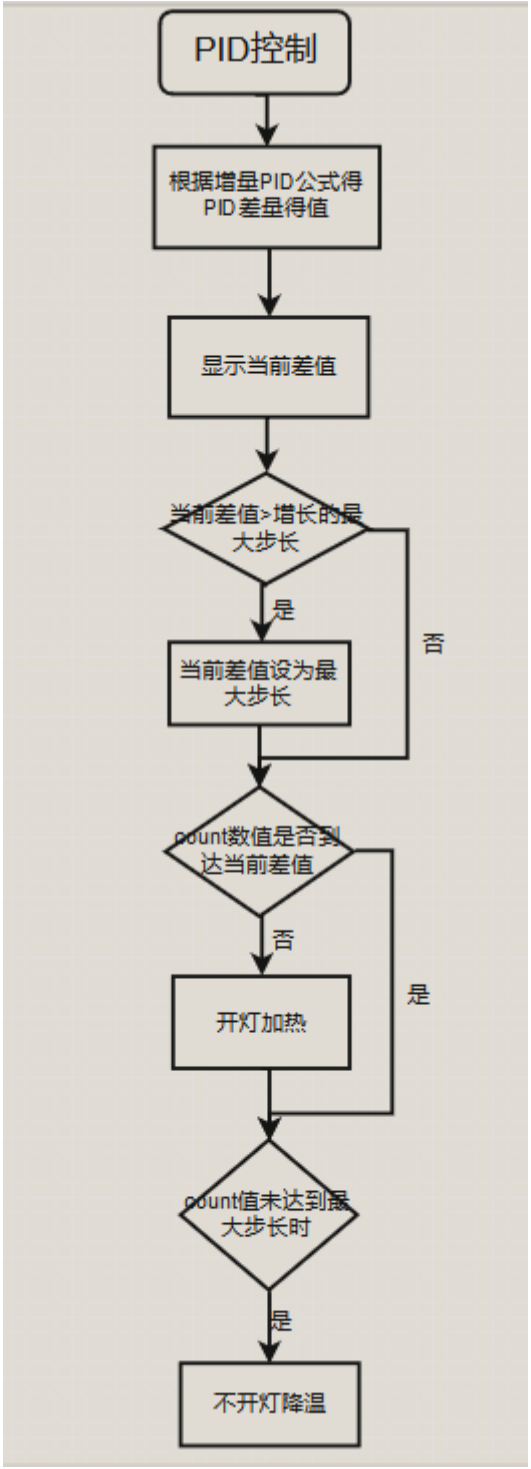
一．实验原理：

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。

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。

二．实验流程图：


设置Kp,Ki,Kd
S1是否按下
目标温度加一
初始化DS温度传感器
跳过ROM命令访问存储器
写温度转换命令
延时一段时间
初始化DS温度传感器
跳过ROM命令访问存储器
将读到的温度存放在result变量且将其转化为实际温度值
显示当前温度以及目标温度
目标温度>当前温度
否
是
PID控制



三．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

四．实验代码：

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

typedef unsigned char uchar;
typedef unsigned int uint;

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

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


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

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

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

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

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

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

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

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

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

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

uchar code shiji[2][32]={

{0x00,0x10,0x0C,0x04,0x4C,0xB4,0x94,0x05,0xF6,0x04,0x04,0x04,0x14,0x0C,0x04,0x00,0x00,0x8
2,0x82,0x42,0x42,0x23,0x12,0x0A,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,0x03,0x40,0x80,0x7F,0x00,0x01,0x02,0x1C,0x08,0x00},/*"?",1*/
};
uchar                                                                    code
du[]={0x06,0x09,0x09,0xE6,0xF8,0x0C,0x04,0x02,0x02,0x02,0x02,0x02,0x04,0x1E,0x00,0x00,0x0
0,0x00,0x00,0x07,0x1F,0x30,0x20,0x40,0x40,0x40,0x40,0x40,0x20,0x10,0x00,0x00};/*?C*/
uchar code mubiao[2][32]={

{0x00,0x00,0x00,0xFE,0x22,0x22,0x22,0x22,0x22,0x22,0x22,0x22,0xFE,0x00,0x00,0x00,0x00,0x0
0,0x00,0x7F,0x22,0x22,0x22,0x22,0x22,0x22,0x22,0x22,0x7F,0x00,0x00,0x00},/*"?",0*/
    {0x10,0x10,0xD0,0xFF,0x50,0x90,0x20,0x22,0x22,0x22,0xE2,0x22,0x22,0x22,0x20,0x00,0x04
,0x03,0x00,0xFF,0x00,0x09,0x04,0x03,0x40,0x80,0x7F,0x00,0x01,0x06,0x1C,0x00},/*"?",1*/
};
void delay(uint i)//延时子程序,i 最大 256，超过 256 部分无效
{
    while(--i);
}
void Read_busy()   //等待 BUSY=0
```

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

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

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

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

void Set_line(uchar startline)//显示起始行设置
{
  startline=0xC0|startline;// 11000000|startline，根据 startline 后六位选择起始行
  write_command(startline);
```

```
}

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

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

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

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

bit DS_init()
{
       bit flag;
       DQ = 0;
       delay_us(255);      //500us 以上
       DQ = 1;               //释放
       delay_us(40);       //等待 16~60us
       flag = DQ;
       delay_us(150);
       return flag;        //成功返回 0
}
uchar read()      //byte
{
        uchar i;
        uchar val = 0;
        for (i=0; i<8; i++)
        {
              val >>= 1;
```

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

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

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

            display(1,4,2*16,shu[a]);delay(255);
            display(1,4,3*16,shu[b]);delay(255);
            display(2,4,0*16,shu[c]);delay(255);
      Ek2 = Ek1;
      Ek1 = Ek;
      count = 0;
```

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

}

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

        while(1)
        {
            if(s1 == 0)
                aim++;
            if(s2 == 0)
                aim--;
            while(DS_init());
            write(0xcc);    //跳过 ROM 命令
            write(0x44);    //温度转换命令
            delay(600);
            while(DS_init());
            write(0xcc);
```

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

            Ek = aim - result;

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

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

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

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

五．实验中的问题及分析：

实验中需要调试比较多的地方在于使用 PID 的时候所需要的 Kp,Ki,Kd 的数值的确定，需要在多次测试后，观察到其值能够成比例的拟合期望的数值。

六、思考题：

1.精确延时的种类，以及其优缺点：

a.循环方法:让单片机使用 while 等循环语句进行循环延时程序,实现简单但是浪费 CPU 资源,而且精确度较低。

b.定时器方法:通过定时器来进行延时,通常可以规定 2-3 个计时器同时进行，好处是复用性好，效率和精确度比较高,缺点是计时时间有上限而且会浪费一个计时器。

2.参考其他资料，了解 DS18820 的其他命令的用法，

1) Read ROM[33H]

2) Match ROM[55H]

这个是匹配 ROM 命令,后限 64 位 ROM 序列,让总线控制器在多点总线上定位一只特定的 DS18B20。

3) Skip ROM[OCCH]

这条命令允许总线控制器不用提供 64 位 ROM 编码就使用存储器操作命令在单点总线情况下,可以节省时间。

4) Search ROM[OFOH]

当一个系统初次启动时,总线控制器可能并不知道单线总线上有多个器件或它们的 64 位编码，搜索 ROM 命令允许总线控制器用排除法识别总线上的所有从机的 64 位编码。

5) Alarm SearchfOECH]

这条命令的流程和 Search ROM 相同，然而，只有在最近一次测温后遇到符合报警条件的情况, DS18B20 才会响应这条命令。

6) Write Scratchpad[4EH]

这个命令向 DS18B20 的暂存器 TH 和 TL 中写入数据。可以在任何时刻发出复位命令来中止写入。

7) Read Scrachpad(BEH]

这个命令读取暂存器的内容。读取将从第 1 个字节开始，一直进行下去， 直到第 9(CRC)字节读完。如果不想读完所有字节，控制器可以在任何时间发出复位命令来中止读取。

8) Copy Scratchpad[48H]

这个命令把暂存器的内容拷贝到 DS1820 的 E2ROM 存储器里，即把温度报警触发字节存入非易失性存储器里。

9) Convert T44H]

这条命令启动一次温度转换而无需其他数据。

10) Recall E2

这条命令把报警触发器里的值拷贝回暂存器。这种拷贝操作在 DS18B20 上电时自动执行，这样器件一上电 ，暂存器里马上就存在有效的数据了。若在这条命令发出之后发出读数据隙，器件会输出温度转换忙的标识 :0 为忙, 1 为完成。

11) Read Power Supply[0B4H]

若把这条命令发给 DS18B20 后发出读时间隙,器件会返回它的电源模式:0 为寄生电源 I 为外部电源。