实 验 报 告

学号：21160809
计算机学院计科八班
张 炎

实验五　重量测量

1.实验原理

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

2.实验内容

1）参考辅助材料，学习 C51 语言使用

2)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。

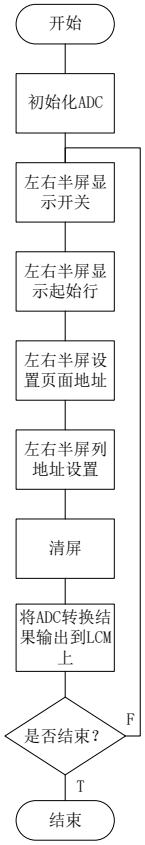
3.实验步骤

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；

I.流程图



II. 程序清单

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

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

sfr ADC_CONTR   = 0xBC;
sfr ADC_RES   = 0xBD;
sfr ADC_RESL2     = 0xBE;
sfr P1ASF        = 0x9D;
#define ADC_POWER       0x80
#define ADC_FLAG     0x10
#define ADC_START   0x08
#define ADC_SPEEDLL 0x00
#define ADC_SPEEDL    0x20
#define ADC_SPEEDH    0x40
#define ADC_SPEEDHH 0x60
void delay();
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;
sbit CS2    = P1^6;
sbit BUSY = P2^7;
```

```
void write_left_cmd   (UCHAR comd);
void write_left_data (UCHAR ldata);
void write_right_cmd (UCHAR comd);
void write_right_data(UCHAR rdata);

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,0x55,0x55,0x55,0x57,0x50,0x4
0,0x40,0x00,

0x20,0x20,0x20,0xBE,0xAA,0xAA,0xAA,0xAA,0xAA,0xAA,0xAA,0xBE,0x2
0,0x20,0x20,0x00,

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,0x00,0xF8,0x00,0x
FF,0x00,0x00,

0x04,0x04,0x7E,0x01,0x80,0x47,0x30,0x0F,0x10,0x27,0x00,0x47,0x80,0x7
F,0x00,0x00,

0x10,0x60,0x02,0x8C,0x00,0xFE,0x02,0xF2,0x02,0xFE,0x00,0xF8,0x00,0x
FF,0x00,0x00,

0x04,0x04,0x7E,0x01,0x80,0x47,0x30,0x0F,0x10,0x27,0x00,0x47,0x80,0x7
F,0x00,0x00,

0x20,0x20,0x20,0xBE,0xAA,0xAA,0xAA,0xAA,0xAA,0xAA,0xAA,0xBE,0x2
0,0x20,0x20,0x00,

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,0x48,0x48,0x4
8,0x08,0x00,

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

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

x20,0x20,0x20,0x00,

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, //"0",11

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,0x11,0x20,0x20,0x2
0,0x1F,0x00,    //"6",17

0x00,0x18,0x08,0x08,0x88,0x68,0x18,0x00,0x00,0x00,0x00,0x3E,0x01,0x0
0,0x00,0x00,    //"7",18

0x00,0x70,0x88,0x08,0x08,0x88,0x70,0x00,0x00,0x1C,0x22,0x21,0x21,0x2
2,0x1C,0x00,   //"8",19

0x00,0xF0,0x08,0x08,0x08,0x10,0xE0,0x00,0x00,0x01,0x12,0x22,0x22,0x1
1,0x0F,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<30;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);
        write_left_cmd(0x40);
```

```
        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(3,16,16,&charcode1[16]);
    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(3,0,16, &charcode2[16]);
    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_right_cmd(0x3F);
        write_left_cmd (0xC0);
        write_right_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_right_cmd(0x3F);
        write_left_cmd(0xC0) ;
        write_right_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;
          }
        i=(i-50)*3;
          x=i/100;
          i=i%100;
          y=i/10;
          z=i%10;
          show(x,y,z);
          delay();
          get_result();
    }
}
```

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。

4.思考题

1）调零的原理，软件调零和调零调零的区别。

答：**软件调零是采用软件进行补偿的方法，又称数字调零；调零调零是采用电路检测的方法对硬件进行机械调零。**

2）模/数和数/模的信号转换原理。

答：**数字电路处理的信号一般是多位二进制信息号是二进制数字量，输出模拟信号则是与输入数字量成正比的电压或电流数/模转换器的组成寄存器用来暂时存放数字量串行输入，但输出只能是并行输出。n 位寄存器的输出分别控制 N 个模拟开关的接通或断开数模和模数转换器数模和模数转换器数模和模数转换器数模和模数转换器。**

A/D 转换器是将模拟量转换成数字量的器件，模拟量可以是电压、电流等信号，也可以是声、光、压力、温度等随时间连续变化的非电的物理量。非电量的模拟量可以通过适当的传感器（如光电传感器、压力传感器、温度传感器）转换成电信号。

3）I2C 总线在信号通讯过程中的应用。

答：**目前有很多半导体集成电路上都集成了 I2C 接口。带有 I2C 接口的单片机有:CYGNAL 的 C8051F0XX 系列,PHILIPSP87LPC7XX 系列,MICROCHIP 的 PIC16C6XX 系列等。很多外围器件如存储器、监控芯片等也提供 I2C 接口。**

5.问题及分析

对于 LCM 的分页了解不够深入，一度出现显示错误的问题；在 A/D 转换的过程中对 A/D 转换器的工作过程出现误解，导致无法正确称重的问题。

实验六　直流电机脉宽调制调速

1.实验原理

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。

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。

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。

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。

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。

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。

2.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。

3.实验步骤

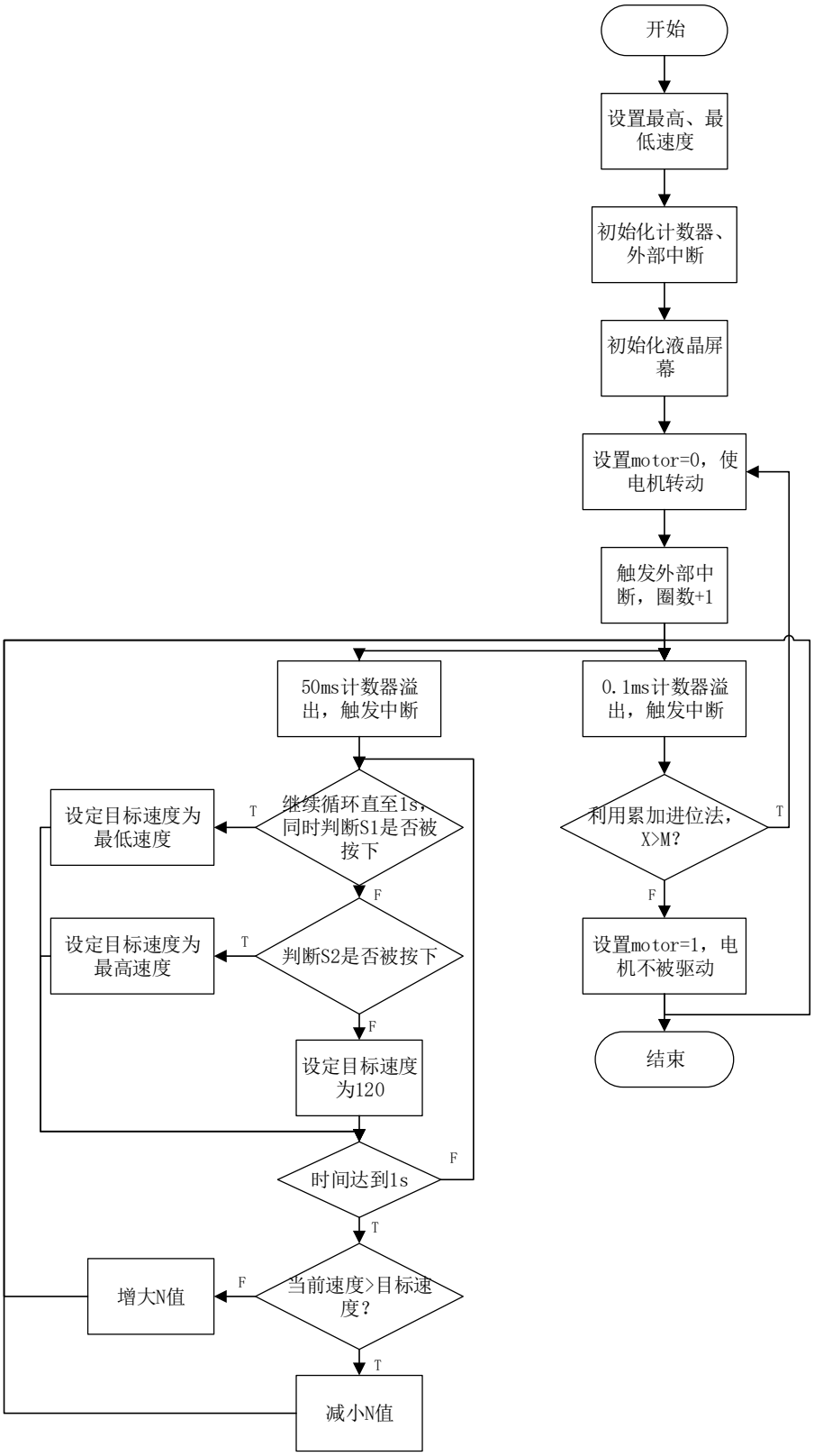
1）建立工程，实现实验内容 1。

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。

I.程序流程图



II. 程序清单

```
#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;
sbit swh2=P3^7;
sbit motor=P1^1;

uchar code zima[20][32]=
{
0x00,0x00,0xC0,0xE0,0x30,0x10,0x08,0x08,0x08,0x08,0x08,0x18,0x30,0xE0,0xC0,0x
00,
0x00,0x00,0x07,0x0F,0x18,0x10,0x20,0x20,0x20,0x20,0x20,0x10,0x18,0x0F,0x07,0x0
0,///*"0"*0/

0x00,0x00,0x00,0x10,0x10,0x10,0x10,0xF0,0xF8,0x00,0x00,0x00,0x00,0x00,0x00,0x0
0,
0x00,0x00,0x00,0x20,0x20,0x20,0x20,0x3F,0x3F,0x20,0x20,0x20,0x20,0x00,0x00,0x0
0,///*"1"*1/

0x00,0x00,0x60,0x50,0x10,0x08,0x08,0x08,0x08,0x08,0x08,0x98,0xF0,0x70,0x00,0x0
0,
0x00,0x00,0x20,0x30,0x28,0x28,0x24,0x24,0x22,0x22,0x21,0x20,0x30,0x18,0x00,0x0
0,///*"2"*2/

0x00,0x00,0x30,0x30,0x08,0x08,0x88,0x88,0x88,0x88,0x58,0x70,0x30,0x00,0x00,0x0
0,
0x00,0x00,0x18,0x18,0x20,0x20,0x20,0x20,0x20,0x20,0x31,0x11,0x1F,0x0E,0x00,0x0
```

0,///*"3"*3/

0x00,0x00,0x00,0x00,0x00,0x80,0x40,0x20,0x10,0xF0,0xF8,0xF8,0x00,0x00,0x00,0x0
0,
0x00,0x04,0x06,0x05,0x05,0x04,0x24,0x24,0x24,0x3F,0x3F,0x3F,0x24,0x24,0x24,0x0
0,///*"4"*4/

0x00,0x00,0x00,0xC0,0x38,0x88,0x88,0x88,0x88,0x88,0x88,0x88,0x08,0x08,0x00,0x0
0,
0x00,0x00,0x18,0x29,0x21,0x20,0x20,0x20,0x20,0x20,0x30,0x11,0x1F,0x0E,0x00,0x0
0,///*"5"*5/

0x00,0x00,0x80,0xE0,0x30,0x10,0x98,0x88,0x88,0x88,0x88,0x88,0x98,0x10,0x00,0x0
0,
0x00,0x00,0x07,0x0F,0x19,0x31,0x20,0x20,0x20,0x20,0x20,0x20,0x11,0x1F,0x0E,0x0
0,///*"6"*6/

0x00,0x00,0x30,0x18,0x08,0x08,0x08,0x08,0x08,0x88,0x48,0x28,0x18,0x08,0x00,0x0
0,
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x38,0x3E,0x01,0x00,0x00,0x00,0x00,0x00,0x00,0x0
0,///*"7"*7/

0x00,0x00,0x70,0x70,0xD8,0x88,0x88,0x08,0x08,0x08,0x08,0x98,0x70,0x70,0x00,0x0
0,
0x00,0x0C,0x1E,0x12,0x21,0x21,0x20,0x21,0x21,0x21,0x23,0x12,0x1E,0x0C,0x00,0x
00,///*"8"*8/

0x00,0xE0,0xF0,0x10,0x08,0x08,0x08,0x08,0x08,0x08,0x18,0x10,0xF0,0xC0,0x00,0x
00,
0x00,0x00,0x11,0x33,0x22,0x22,0x22,0x22,0x22,0x32,0x11,0x1D,0x0F,0x03,0x00,0x0
0,///*"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,0x48,0x40,0x40,0
x00,///*"?"*10/

0x40,0x40,0x40,0xDF,0x55,0x55,0x55,0xD5,0x55,0x55,0x55,0xDF,0x40,0x40,0x40,0x
00,
0x40,0x40,0x40,0x57,0x55,0x55,0x55,0x7F,0x55,0x55,0x55,0x57,0x50,0x40,0x40,0x0
0,///*"?"*11/

0x00,0x00,0x00,0x00,0x00,0x00,0xC0,0xC0,0xC0,0xC0,0x00,0x00,0x00,0x00,0x00,0x
00,

```
0x00,0x00,0x00,0x00,0x00,0x00,0x30,0x30,0x30,0x30,0x00,0x00,0x00,0x00,0x00,0x0
0,///*":"*12/

0x00,0x04,0x04,0xE4,0x24,0x24,0x24,0x3F,0x24,0x24,0x24,0xE4,0x04,0x04,0x00,0x0
0,
0x00,0x00,0x80,0x43,0x31,0x0F,0x01,0x01,0x01,0x3F,0x41,0x43,0x40,0x40,0x70,0x0
0,///*"?"*13/

};

uchar tspeed=0;
uchar cspeed=0;
uchar xspeed=120;
uchar speedUp = 140;
uchar speedLow =100;

uchar t1_cnt=0;
int N=40;
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 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);

                delay(50000);
        }
}

void init()
{       P4SW=0x30;

        IT0=1;//下降沿

        EA=1;//CPU 允许中断
        ET1=1;//允许计数器 T0 T1 中断
        ET0=1;
        EX0=1;//允许外部中断

        TMOD=0x11; //00010001B 16 位定时器
        TH1=0x3C;//50ms
        TL1=0xB0;
        TH0=0xFF;//0.1ms
        TL0=0x9C;

        TR0=1;//运行控制位
        TR1=1;
}

void ex_int0() interrupt 0
{
        tspeed++;
}

void t1_int() interrupt 3
{
        if(++t1_cnt<20)//50ms*20=1s
        {               TH1=0x3C;
                                TL1=0xB0;
                if(swh1==0)//按下 s1
                {
```

```
                xspeed = speedLow;
            }
            if(swh2==0){//按下 s2
                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
{
     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);
                }
    }
}
```

4. 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：调压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；PWM 是改变加到电枢上电源时间长短，也就是加一定时间的电，然后断开，过一定时间再加电，断续加电，相当于电机转一下，减速（停止），然后继续这个循环，微观看是非匀速的，宏观看是匀速的。对于脉宽调速，优点是效率高，调速范围还行，缺点是在最低转速时电机运行时脉动的，噪音变大，而且负载越大越严重；对于电压调速，优点是电机运行在整个调速范围内都平稳。调速范围也最大，从电机的始动电压可以调到额定电压。

2）说明程序原理中累加进位法的正确性。

答：从概率学的角度，在各个值设定合理的情况下，累加进位法可以最大程度

上实现每隔一定时间驱动电机转动一次，对实验的顺利进行有至关重要的作用。

3）计算转速测量的最大可能误差，讨论减少误差的办法。

答：**缩短累加进位法中断的计时间隔，以及在计秒中断中每个小循环都添加对于当前速度的监测并随之更改 N 的值。**

5.问题及分析

在实验过程中，开始时对直流电机如何触发中断理解有误差，对电信号如何控制电机转动认识不足，出现了编程错误；之后在计数器的中断内容上设计不够科学，会产生较大误差，后经过优化解决了问题。

实验八　温度测量与控制

1.实验原理

1）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 ℃。

2）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。

3）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。

4）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℃ 。

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。

6）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。

2.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。

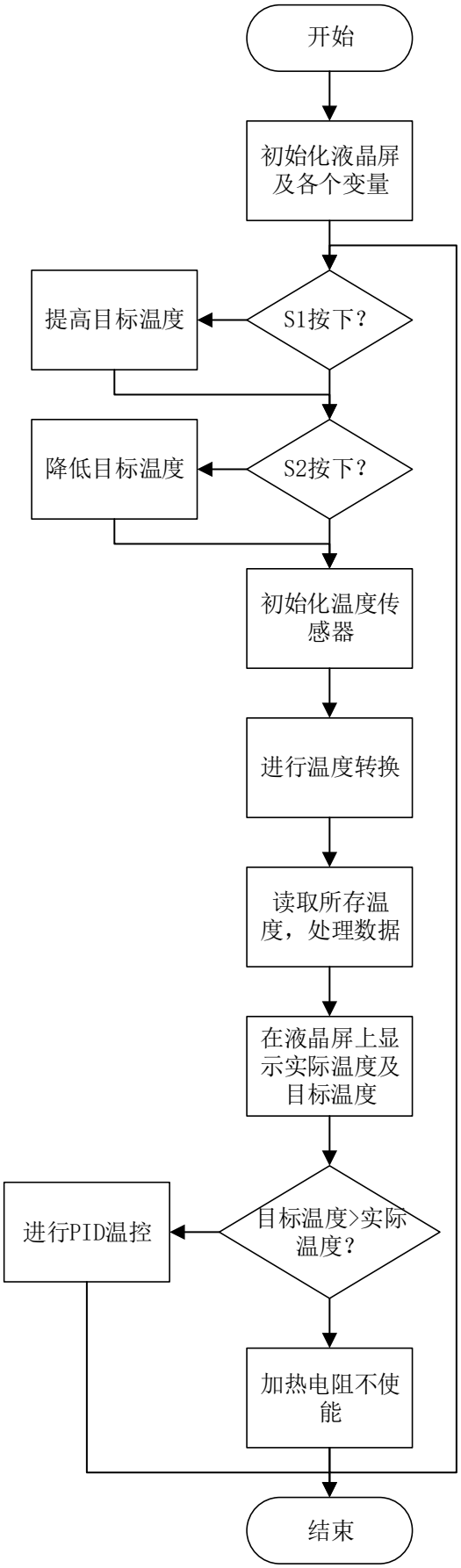
3.实验步骤

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

I.流程图


开始
初始化液晶屏及各个变量
S1按下？
提高目标温度
S2按下？
降低目标温度
初始化温度传感器
进行温度转换
读取所存温度，处理数据
在液晶屏上显示实际温度及目标温度
目标温度>实际温度？
进行PID温控
加热电阻不使能
结束

II. 程序清单

```
//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]={

0x00,0x00,0xC0,0xE0,0x30,0x10,0x08,0x08,0x08,0x08,0x08,0x18,0x30,0xE
0,0xC0,0x00,
0x00,0x00,0x07,0x0F,0x18,0x10,0x20,0x20,0x20,0x20,0x20,0x10,0x18,0x0
F,0x07,0x00,/*"0",0*/

0x00,0x00,0x00,0x10,0x10,0x10,0x10,0xF0,0xF8,0x00,0x00,0x00,0x00,0x0
0,0x00,0x00,
0x00,0x00,0x00,0x20,0x20,0x20,0x20,0x3F,0x3F,0x20,0x20,0x20,0x20,0x0
0,0x00,0x00,/*"1",1*/
```

```
0x00,0x00,0x60,0x50,0x10,0x08,0x08,0x08,0x08,0x08,0x08,0x98,0xF0,0x7
0,0x00,0x00,
0x00,0x00,0x20,0x30,0x28,0x28,0x24,0x24,0x22,0x22,0x21,0x20,0x30,0x1
8,0x00,0x00,/*"2",2*/

0x00,0x00,0x30,0x30,0x08,0x08,0x88,0x88,0x88,0x88,0x58,0x70,0x30,0x0
0,0x00,0x00,
0x00,0x00,0x18,0x18,0x20,0x20,0x20,0x20,0x20,0x20,0x31,0x11,0x1F,0x0
E,0x00,0x00,/*"3",3*/

0x00,0x00,0x00,0x00,0x00,0x80,0x40,0x20,0x10,0xF0,0xF8,0xF8,0x00,0x0
0,0x00,0x00,
0x00,0x04,0x06,0x05,0x05,0x04,0x24,0x24,0x24,0x3F,0x3F,0x3F,0x24,0x24
,0x24,0x00,/*"4",4*/

0x00,0x00,0x00,0xC0,0x38,0x88,0x88,0x88,0x88,0x88,0x88,0x88,0x08,0x0
8,0x00,0x00,
0x00,0x00,0x18,0x29,0x21,0x20,0x20,0x20,0x20,0x20,0x30,0x11,0x1F,0x0
E,0x00,0x00,/*"5",5*/

0x00,0x00,0x80,0xE0,0x30,0x10,0x98,0x88,0x88,0x88,0x88,0x88,0x98,0x1
0,0x00,0x00,
0x00,0x00,0x07,0x0F,0x19,0x31,0x20,0x20,0x20,0x20,0x20,0x20,0x11,0x1F
,0x0E,0x00,/*"6",6*/

0x00,0x00,0x30,0x18,0x08,0x08,0x08,0x08,0x08,0x88,0x48,0x28,0x18,0x0
8,0x00,0x00,
0x00,0x00,0x00,0x00,0x00,0x00,0x38,0x3E,0x01,0x00,0x00,0x00,0x00,0x0
0,0x00,0x00,/*"7",7*/

0x00,0x00,0x70,0x70,0xD8,0x88,0x88,0x08,0x08,0x08,0x08,0x98,0x70,0x7
0,0x00,0x00,
0x00,0x0C,0x1E,0x12,0x21,0x21,0x20,0x21,0x21,0x21,0x23,0x12,0x1E,0x0
C,0x00,0x00,/*"8",8*/

0x00,0xE0,0xF0,0x10,0x08,0x08,0x08,0x08,0x08,0x08,0x18,0x10,0xF0,0xC
0,0x00,0x00,
0x00,0x00,0x11,0x33,0x22,0x22,0x22,0x22,0x22,0x32,0x11,0x1D,0x0F,0x0
3,0x00,0x00,/*"9",9*/

}
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;
    }
}
```

4.思考题

1）进行精确的延时的程序有几种方法？各有什么优缺点？

答：使用_nop_()语句：简洁方便，但精确度不够且占用 CPU 资源，适用于短延时；使用 while 循环或 for 循环：延时长短易于调节，但精确度不够；使用定时器中断延时，精确度高，应用性强，能适用于各种情况，但使用方法较为复杂。

2）参考其他资料，了解 DS18B20 的其他命令用法。

答：

指令类型	指令	功能	详细描述
ROM 指令	[F0H]	搜索 ROM 指令	当系统初始化时，总线控制器通过此指令多次循环搜索 ROM 编码，以确认所有从机器件
	[33H]	读取 ROM 指令	当总线上只有一只 DS18B20 时才会使用此指令，允许总线控制器直接读取从机的序列码
	[55H]	匹配 ROM 指令	匹配 ROM 指令，使总线控制器在多点总线上定位一只特定的 DS18B20
	[CCH]	忽略 ROM 指令	忽略 ROM 指令，此指令允许总线控制器不必提供 64 位 ROM 编码就使用功能指令
	[ECH]	报警搜索指令	当总线上存在满足报警条件的从机时，该从机将响应此指令
功能指令	[44H]	温度转换指令	此条指令用来控制 DS18B20 启动一次温度转换，生成的温度数据以 2 字节的形式存储在高速暂存器中
	[4EH]	写暂存器指令	此指令向 DS18B20 的暂存器写入数据，开始位置在暂存器第 2 字节（TH 寄存器），以最低有效位开始传送
	[BEH]	读暂存器指令	此指令用来读取 DS18B20 暂存器数据，读取将从字节 0 开始，直到第 9 字节（CRC 校验位）读完
	[48H]	拷贝暂存器指令	此指令将 TH、TL 和配置寄存器的数据拷贝到 EEPROM 中得以保存
	[B8H]	召回 EEPROM 指令	将 TH、TL 以及配置寄存器中的数据从 EEPROM 拷贝到暂存器
	[B4H]	读电源模式指令	总线控制器在发出此指令后启动读时隙，若为寄生电源模式，DS18B20 将拉低总线，若为外部电源模式，则将总线拉高，用以判断 DS18B20 的电源模式