第五次实验（重量测量）报告

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、实验内容

1. 参考辅助资料，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（以克记）显示到液晶屏上。误差可允许的范围之间。

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、实验原理

点阵式液晶显示屏

1. 特性

全屏幕点阵，点阵数为 128（列）×64（行），可显示 32 个（16×16 点阵）汉字，也可完成图形、字符的显示。

2. 引脚特性

引脚号	引脚名称	级　别	引　　脚　　功　　能　　描　　述
1	/CS1	H/L	片选信号，当/CS1=L 时，液晶左半屏显示
2	/CS2	H/L	片选信号，当/CS2=L 时，液晶右半屏显示
3	VSS	0V	电源地
4	VDD	+5V	电源电压
5	VLCD	0～-10V	LCD 驱动负电压，要求 VDD-VLCD=10V
6	RS	H/L	寄存器选择信号
7	R/W	H/L	读/写操作选择信号
8	E	H/L	使能信号
9	DB0	H/L	八位三态并行数据总线
10	DB1		
11	DB2		
12	DB3		
13	DB4		
14	DB5		
15	DB6		
16	DB7		
17	LED-		背光电源，$I_{dd} \leqslant 960mA$
18	LED+(5.0V)		

3. 主要各部分详解

1）显示数据 RAM（DDRAM）

DDRAM 是存储图形显示数据的。此 RAM 的每一位数据对应显示面板上一个点的显示。

2）指令寄存器

指令寄存器用于接收 MPU 发来的指令代码，通过译码将指令代码置入相关的寄存器或触发器内。

3）状态字寄存器

状态字寄存器是 LCM（液晶显示模块）与 MPU 通讯时唯一的“握手”信号。状态字寄存器向 MPU 表示了 LCM（液晶显示模块）当前的工作状态。***尤其是状态字中的“忙”标志位是 MPU 在每次对 LCM（液晶显示模块）访问时必须要读出判别的状态位。***当处于“忙”标志位时，I/O 缓冲器被封锁，此时 MPU 对 LCM（液晶显示模块）的任何操作（除读状态字操作外）都将是无效的。

4）X 地址寄存器

X 地址寄存器是一个三位页地址寄存器，其输出控制着 DDRAM 中 8 个页面的选择，也是控制着数据传输通道的八选一选择器。X 地址寄存器可以由 MPU 以指令形式设置。X 地址寄存器没有自动修改功能，所以要想转换页面需要重新设置 X 地址寄存器的内容。

5）Y 地址计数器

Y 地址计数器是一个 6 位循环加一计数器。它管理某一页面上的 64 个单元。Y 地址计数器可以由 MPU 以指令形式设置，它和页地址指针结合唯一选通显示存储器的一个单元，***Y 地址计数器具有自动加一功能。***在显示存储器读/写操作后，Y 地址计数将自动加一。当计数器加至 3FH 后循环归零再继续加一。

6）Z 地址计数器

Z 地址计数器是一个 6 位地址计数器，用于确定当前显示行的扫描地址。***Z 地址计数器具有自动加一功能。***它与行驱动器的行扫描输出同步，选择相应的列驱动的数据输出。

7）显示起始行寄存器

显示起始行寄存器是一个 6 位寄存器，它规定了显示存储器所对应显示屏上第一行的行号。该行的数据将作为显示屏上第一行显示状态的控制信号。

8）显示开/关触发器

显示开/关触发器的作用就是控制显示驱动输出的电平以控制显示屏的开关。在触发器输出为“关”电平时，显示数据锁存器的输入被封锁并将输出置“0”。从而使显示驱动输出全部为非选择波形，显示屏呈不显示状态。在触发器输出为“开”电平时，显示数据锁存器被控制，显示驱动输出受显示驱动数据总线上数据控制，显示屏将呈显示状态。

4. 软件说明

1）指令表

指令名称	控制信号		控制代码							
	RS	R/W	D7	D6	D5	D4	D3	D2	D1	D0
显示开关设置	0	0	0	0	1	1	1	1	1	D
显示起始行设置	0	0	1	1	L5	L4	L3	L2	L1	L0
页面地址设置	0	0	1	0	1	1	1	P2	P1	P0
列地址设置	0	0	0	1	C5	C4	C3	C2	C1	C0
读取状态字	0	1	BUSY	0	ON/OFF	RESET	0	0	0	0
写显示数据	1	0	数据							
读显示数据	1	1	数据							

2）读状态字

BUSY 表示当前 LCM 接口控制电路运行状态。BUSY=1 表示 LCM 正在处理 MPU 发过来的指令或数据。此时接口电路被封锁，不能接受除读状态字以外的任何操作。BUSY=0 表示 LCM 接口控制电路已外于“准备好”状态，等待 MPU 的访问。

ON/OFF 表示当前的显示状态。ON/OFF=1 表示关显示状态，ON/OFF=0 表示开显示状态。

RESET 表示当前 LCM 的工作状态，即反映/RES 端的电平状态。当/RES 为低电平状态时，LCM 处于复位工作状态，标志位 RESET=1。当/REST 为高电平状态时，LCM 为正常工作状态，标志位 RESET=0。

MPU 在每次对 LCM 操作之前，都要读出状态字判断 BUSY 是否为“0”。若不为“0”，则 MPU 需要等待，直至 BUSY=0 为止。

3）显示开关设置

设置显示开关触发器的状态。D 位为显示开/关控制位，D=1 为开显示设置，D=0 为关显示设置。

4）显示起始行设置

设置了显示起始行寄存器的内容，指令中 $L_5 \sim L_0$ 为显示起始行的地址，取值 1~64 行，它规定了显示屏上最顶一行所对应的的显示存储器的行地址。

5）页面地址设置

设置 X 的内容，指令中 $P_2 \sim P_0$ 就是要确定当前选择的页面地址，规定了以后的读写操作将在哪一个页面上进行。

6）列地址设置

$C_5 \sim C_0$ 代表某一页面上的某一个单元地址，随后的一次读或写数据将在这个单元上进行。

5. 控制时序表

/CS1	/CS2	RS	R/W	E	DB7～DB0	功能
X	X	X	X	0	高阻	总线释放
0	0	0	0	下降沿	输入	写指令代码

/CS1	/CS2	RS	R/W	E	DB7～DB0	功能
0	0	0	1	1	输出	读状态字
0	0	1	0	下降沿	输入	写显示数据
0	0	1	1	1	输出	读显示数据

6. DDRAM 地址表

	/CS1=0					/CS2=0					
Y=	0	1	···	62	63	0	1	···	62	63	行号
X=0	DB0↓DB7	DB0↓DB7	DB0↓DB7	DB0↓DB7	DB0↓DB7	DB0↓DB7	DB0↓DB7	DB0↓DB7	DB0↓DB7	DB0↓DB7	0↓7
↓	DB0↓DB7	DB0↓DB7	DB0↓DB7	DB0↓DB7	DB0↓DB7	DB0↓DB7	DB0↓DB7	DB0↓DB7	DB0↓DB7	DB0↓DB7	8↓55
X=7	DB0↓DB7	DB0↓DB7	DB0↓DB7	DB0↓DB7	DB0↓DB7	DB0↓DB7	DB0↓DB7	DB0↓DB7	DB0↓DB7	DB0↓DB7	56↓63

A/D 转换器

1. 结构

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，不需要作为 A/D 使用的口可继续作为 I/O 口使用。

通过模拟多路开关，将通过 ADC0~7 的模拟量输入送给比较器。用数/模转换器转换的模拟量与本次输入的模拟量通过比较器进行比较，将结果保存到逐次比较器，并通过逐次比较寄存器输出转换结果。

置位 ADC 控制寄存器 ADC_CONTR 中的 A/D 转换结束标志位 ADC_FLAG，以供程序查询或发出中断申请。模拟通道的选择控制由 ADC 控制寄存器 ADC_CONTR 中的 CHS2～CHS0 确定。ADC 的转换速度由 ADC 控制寄存器中的 SPEED1 和 SPEED0 确定。在使用 ADC 之前，应先给 ADC 上电，也就是置位 ADC 控制寄存器中的 ADC_POWER 位。

2. 与 A/D 转换相关的寄存器

1）P1 口模拟功能控制寄存器 P1ASF

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。

SFR name	Address	bit	B7	B6	B5	B4	B3	B2	B1	B0
P1ASF	9DH	name	P17ASF	P16ASF	P15ASF	P14ASF	P13ASF	P12ASF	P11ASF	P10ASF

2）ADC 控制寄存器 ADC_CONTR

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，要由软件清零。

ADC_START：模数转换器转换启动控制位，设置为“1”时，开始转换，转换结束后为 0。

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

3）中断允许寄存器 IE

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

EADC：A/D 转换中断允许位。EADC=1，允许 A/D 转换中断，EADC=0，禁止 A/D 转换中断 。

SFR name	Address	bit	B7	B6	B5	B4	B3	B2	B1	B0
IE	A8H	name	EA	ELVD	EADC	ES	ET1	EX1	ET0	EX0

4）AUXR1 寄存器

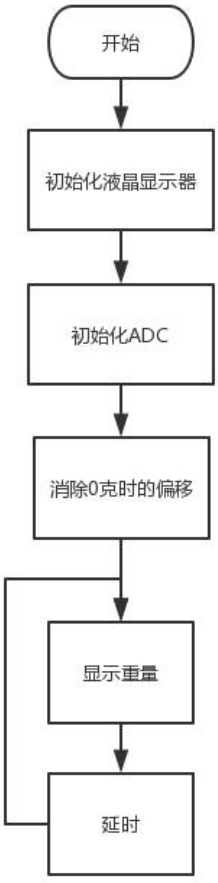
AUXR1 寄存器的 ADRJ 位是 A/D 转换结果寄存器（ADC_RES，ADC_RESL）的数据格式调整控制位。

当 ADRJ=0 时，10 位 A/D 转换结果的高 8 位存放在 ADC_RES 中，低 2 位存放在 ADC_RESL 的低 2 位中。

当 ADRJ=1 时，10 位 A/D 转换结果的高 2 位存放在 ADC_RES 的低 2 位中，低 8 位存放在 ADC_RESL 中。

AUXR1	A2H	Auxiliary register1	-	PCA_P4	SPI_P4	S2_P4	GF2	ADRJ	-	DPS

五、实验流程图



六、实验中遇到的问题以及总结

问题 1：***如何使用 C51 编写 A/D 转换？***

答：

```
#include “reg51.h”
#include “intrins.h”
sfr ADC_CONTR = 0xBC;
sfr ADC_RES = 0xBD;//ADC 高 8 位结果寄存器
sfr ADC_LOW2 = 0xBE;//ADC 低 2 位结果寄存器
sfr P1ASF = 0x9D;
sfr AUXR1 = 0xA2;
//定义 ADC_CON 操作常数
#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
int res = 0;//记录 AD 转换结果
void initADC()
{
    P1ASF = 0x01;//选择 P1.0 作为 A/D 输入来用
    ADC_RES = 0x00;
    ADC_CONTR = ADC_POWER|ADC_SPEEDLL|ADC_START|0;//开始进行 AD 转换
    AUXR1 = 0x04;
    IE = 0xA0;
}
void adc_isr() interrupt 5 using 1
{
    ADC_CONTR &= !ADC_FLAG;//将 ADC_FLAG 位软件清 0
    res = ADC_RESL + ADC_RES*256;
    ADC_CONTR = ADC_POWER|ADC_SPEEDLL|ADC_START|0;//开始进行 AD 转换
}
```

问题 2：***如何使用 C51 编写液晶显示程序？***

答：

```
#include “reg51.h”
#include “intrins.h”
char code zhong[2][16] = {{},{}};
char code liang[2][16] = {{},{}};
char code number[10][16] = {{},{},{},{},{},{},{},{},{},{}};
```

PS:其余各部分函数见代码

七、思考题

1. 调零的原理，软件调零和硬件调零的区别。

    硬件调零：使用外接电路或者改变压敏电阻的初始阻止等方式实现的调零。通过附加电

路或者对压敏电阻调整.

软件调零：在不适用任何外接电路的情况下，对采集的数据进行数学处理从而实现调零的过程。

使用软件对 A/D 采集值进行调整来达到在未放置物体的时候显示 0。

2. 模/数和数/模的信号转换原理。

A/D 逐次逼近法：由一个比较器、D/A 转换器、缓冲寄存器及控制逻辑电路组成。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位置 1，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。D/A 转换:将二进制数的每位按权大小转换为相对应的模拟量， 然后将代表的各位的模拟量想家，就得到对应数字量对应的模拟量。

3. I2C 总线在信号通讯过程中的应用。

主器件用于启动总线传送数据，并产生时钟以开放传送的器件，此时任何被寻址的器件均被认为是从器件。在总线上主和从、发和收的关系不是恒定的，而取决于此时数据传送方向。如果主机要发送数据给从器件，则主机首先寻址从器件，然后主动发送数据至从器件，最后由主机终止数据传送。

如果主机要接收从器件的数据，首先由主器件寻址从器件，然后主机接收从器件发送的数据，最后由主机终止接收过程，在这种情况下，主机负责产生定时时钟和终止数据传送。

八、代码

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

sbit CS1=P1^7;
sbit CS2=P1^6;
sbit RST=P1^5;
sbit E=P3^3;
sbit RW=P3^4;
sbit RS=P3^5;
sbit BUSY=P2^7;
sbit RESET=P2^4;
sbit KEY1=P3^6;
sbit KEY2=P3^7;

sfr     P1ASF=0x9d;
sfr    ADC_CONTR=0xbc;
sfr    ADC_RES=0xbd;
sfr    AUXR1=0xa2;
sfr    ADC_RESL=0xbe;

#define ADC_POWER 0x80
#define ADC_FLAG  0x10
```

```
#define ADC_START 0x08
#define ADC_SPEEDLL 0x00

int a,b,c,d,i,j,k,temp,currentRes=0,res=0,offset=0;
char code zhong[2][16]={{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}};
char code liang[2][16]={{0x20,0x20,0x20,0xBE,0xAA,0xAA,0xAA,0xAA,0xAA,0xAA,0xBE,0x20,0x20,0x2
0,0x00},

    {0x00,0x80,0x80,0xAF,0xAA,0xAA,0xAA,0xFF,0xAA,0xAA,0xAA,0xAF,0x80,0x80,0x00,0x00}};
char code ke[2][16]={{0x04,0x04,0xE4,0x24,0x24,0x24,0x24,0x3F,0x24,0x24,0x24,0x24,0xE4,0x04,0x04,0x0
0},

    {0x80,0x80,0x43,0x22,0x12,0x0E,0x02,0x02,0x02,0x7E,0x82,0x82,0x83,0x80,0xE0,0x00}};
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){
	while(count--){
		_nop_();
	}
}
void checkReady(bit cs){
	CS1= cs==0 ? 1:0;
	CS2= cs==1 ? 1:0;
	P2=0xff;
	E=1;
	RS=0;
	RW=1;
	while(BUSY==1);
	E=0;
	CS2=0;
	CS1=0;
}
void write_command(bit cs,char com){
	checkReady(cs);
	CS1= cs==0 ? 1:0;
	CS2= cs==1 ? 1:0;
	RS=0;
	RW=0;
	E=1;
	P2=com;
	wait(20);
	E=0;
	CS1=0;
	CS2=0;
}
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;
			E=1;
```

```
                RS=1;
                RW=0;
                P2=0x00;
                wait(20);
                E=0;
            }
        }
}
void write_bytes(bit cs,char * buffer,int len){
        for(i=0;i<len;i++){
            checkReady(cs);
            CS1= cs==0 ? 1:0;
            CS2= cs==1 ? 1:0;
            E=1;
            RS=1;
            RW=0;
            P2=*(buffer+i);
            wait(20);
            E=0;
        }
}
void outChina(bit cs,char x,char y,char china[][16]){
        write_command(cs,x+0xb8);
        write_command(cs,y+0x40);
        write_bytes(cs,*china,16);

        write_command(cs,x+1+0xb8);
        write_command(cs,y+0x40);
        write_bytes(cs,*(china+1),16);
}
void outNumber(bit cs,char x,char y,int num){
        write_command(cs,x+0xb8);
        write_command(cs,y+0x40);
        write_bytes(cs,number[num],8);
        write_command(cs,x+1+0xb8);
        write_command(cs,y+0x40);
        write_bytes(cs,number[num]+8,8);
}
void showResult(int num){
        a=num/1000;
        b=(num-a*1000)/100;
        c=(num-a*1000-b*100)/10;
        d=num-a*1000-b*100-c*10;
        outNumber(1,4,0,d);
```

```
    outNumber(0,4,53,c);
    outNumber(0,4,43,b);
    outNumber(0,4,33,a);
}
void initLCD(){
    cls(1);
    write_command(0,0x3f);
    cls(0);
    write_command(1,0x3f);
    outChina(0,1,40,zhong);
    outChina(1,1,7,liang);
    outChina(1,4,16,ke);
}
void initADC(){
    P1ASF=0x01;
    ADC_RES=0x00;
    ADC_CONTR=ADC_POWER | ADC_SPEEDLL | ADC_START | 0;
    wait(20);
    AUXR1=0x04;
    IE=0xa0;
}
void adc_isr() interrupt 5 using 1
{
    ADC_CONTR &=!ADC_FLAG;
    res=ADC_RESL+ADC_RES*256;
    ADC_CONTR=ADC_POWER | ADC_SPEEDLL | ADC_START | 0;
}
void waitL(int count){
    k=0xffff;
    while(count--){
        while(k--);
    }
}

void main(){
    initLCD();
    initADC();
    res=0;
    showResult(i);
    while(1){
        P0=0x03;
        if(!KEY1){
            showResult(9999);
            offset=res;
```

```
            }else{
                if(res!=currentRes){
                    currentRes=res;
                    temp=currentRes-offset;
                    if(temp<0){
                        showResult(0);
                    }else{
                        showResult(temp);
                    }
                }
                waitL(0x03);
            }
        }
}
```

第六次实验（直流电机脉宽调制调速）报告

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速，将其显示在数码管，每秒刷新一次。
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，同是实时显示当前转速。
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。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。

三、实验步骤

实现实验内容一---->编写中断程序，测量电机转速---->完成控制转速程序---->完成整体实验内容。

编写中断程序，测量电机转速：电机转速就是一秒钟之内 INT0（外部中断）的中断个数。包括一个能够实现 1 秒钟的定时器中断和一个外部中断。

完成控制转速程序：添加一个快速的定时中断，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。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。

四、实验原理

1.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。

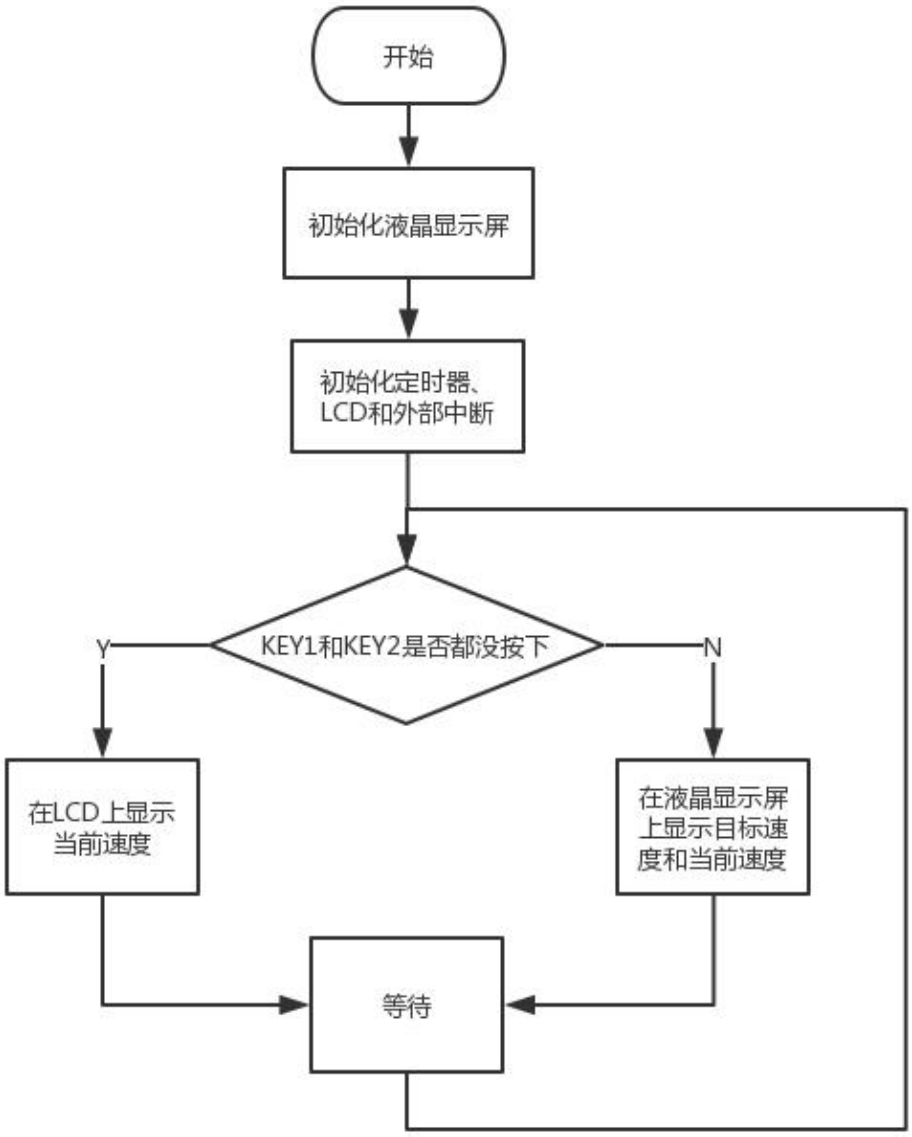
2.***PWM 的基本原理是通过输出一个很高频率的 0/1 信号，其中 0 的比例为θ（也叫作占空比）。***在外围积分元件的作用下，使得总的效果相当于输出θ×A（A 为高电平电压）的电压，通过改变占空比就可以调整输出电压。

3.***使用单片机实现 PWM，可以采用累加进位法。设置一个累加变量 x，每次加 N，若结果大于 M，则输出 0，并减去 M；否则输出 1。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。***

4.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。

5.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，随着控制变量的增加，实际转速会增加。

五、实验流程图



六、实验中遇到的问题以及总结

问题 1：C51 程序中，sbit 和 sfr 分别有什么作用？

答：bit 定义位变量，定义位变量时可以为变量赋值，但不能指定变量的地址。bit 变量名=变量值。

sbit(特殊功能位寄存器)此类型变量只能用于访问可位寻址的特殊功能寄存器的某个位。sbit 变量名=位地址。

Sfr(特殊功能寄存器)此类型变量可以访问指定的 8 位特殊功能寄存器，地址范围为 0x80~0xFF。sfr 变量名=变量地址。

sfr16 此类型变量可访问指定的 16 位特殊功能寄存器，地址范围为 0x80~0xFF。sfr16 变量名=变量地址，此处的变量地址为 16 位中的低 8 位地址。

问题 2：按钮 S1 和 S2 的输出和按下之间的关系。

答：按钮按下为输出 0，按钮不按下为输出 1。

问题 3：***如何使用 C51 编写定时器中断？***

答：

```
void t0_int0() interrupt 1//定时器 0 溢出中断
{
    TR0 =0;
    TH0 = ??;
    TH1 = ??;
    TR1 = 1;
}
void t1_int0() interrupt 3//定时器 1 溢出中断
{
    TR0 =0;
    TH0 = ??;
    TH1 = ??;
    TR1 = 1;
}
init()
{
    TMOD = 0x11;//16 位定时器 0、1，为模式 1
    TH0 = ??;
    TL0 = ??;
    TH1 = ??;
    TH1 = ??;
    EA = 1;
    TR0 = 0;//定时器 0 停止运行
    TR1 = 1;//定时器 1 停止工作
    ET0 = 1;//定时器 0 开中断
    ET1 = 1;//定时器 1 开中断
    TR0 = 1;
    TR1 = 1;
```

```
}
void main()
{
    init();
}
```

问题 4：***如何使用 C51 编写外部中断？***

```
EA = 1;
IT0 = 1;//使外部中断在下降沿时触发
EX0 = 1;//使能外部中断
```

问题 5：***在单片机中怎么实现 PWM？***

```
int t0_cnt = 0;//记录定时器 0 的中断次数，一次中断为 20ms，二十次即为 1s
//前者记录当前的电机速度。后者用于外部中断中，统计一秒钟内电机转动的圈数
sbit KEY1=P3^6;
sbit KEY2=P3^7;
int speed = 0,t_speed = 0;
int obj_speed = 0;
int X = 0;//累加进位法控制占空比中的累加变量
int N = 30,M = 256;//累加进位法中的 M 和 N
void ex_int0() interrupt 0
{
    t_speed = t_speed + 1;
}
void t0_int0() interrupt 1//定时器 0 溢出中断
{
    TR0 =0;
    If(t0_cnt > 18)
    {
        t0_cnt = t0_cnt + 1;
        TH0 = ??;
        TH1 = ??;
        TR1 = 1;
        return;
    }
    t0_cnt = 0;
    speed = t_speed;
     t_speed=0;
    if(speed > obj_speed)
        N = N + 1;
    else N = N - 1;//修改控制变量
    TH0 = ??;
    TH1 = ??;//50ms
    TR1 = 1;
}
void t1_int0() interrupt 3//定时器 1 溢出中断
```

```
{
    TR0 =0;
    X = X + N;
    if((!KEY1)&&(!KEY2)) //当 KEY1 和 KEY2 都被按下，则电机加速
        out = 0;
    else
    {
        if(!KEY1)
            obj_speed = 50;//最小速度
        else if(!KEY2)
            obj_speed = 200;//最大速度
        else if(KEY1&&KEY2)
            obj_speed = 125;//中等速度
        //累加进位法
        if(X > M)
        {
            X = X - M;
            out = 0;
        }
        else
            out = 1;
    }
    TH0 = ??;
    TH1 = ??;//0.1ms
    TR1 = 1;
}
init()
{
    TMOD = 0x11;//16 位定时器 0、1，为模式 1
    TH0 = ??;
    TL0 = ??;
    TH1 = ??;
    TH1 = ??;
    EA = 1;
    TR0 = 0;//定时器 0 停止运行
    TR1 = 1;//定时器 1 停止工作
    ET0 = 1;//定时器 0 开中断
    ET1 = 1;//定时器 1 开中断
    EX0 = 1;//使能外部中断
    IT0 = 1;//使外部中断下降沿触发
    TR0 = 1;
    TR1 = 1;
}
void main()
```

```
{
    init();
}
```

七、思考题

1. 讨论脉宽调速和电压调速的区别、优缺点和应用范围。

脉宽调速：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)，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。并且需要的外围器件较少，特别适合于单片机控制领域。

电压调速：直接改变电压模拟量从而改变电机转速的方法。电压便于平滑性调节，可以实现无级调速，损耗小，调速经济性好。

2. 说明程序原理中累加进位法的正确性。

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。每次循环中都有 M/N 次输出，而其中只有一次输出为 1，即总共输出了 N 个 1，而且总输出次数是 M 次，1 的比例就是 N/M。

3. 计算转速测量的最大可能误差，讨论减少误差的方法。

外部因素：减少摩擦力。

内部因素：让电机的转速保持在 200~40 转/s 之间的速度，并保持一个比较低的速度(速度不要过快)。

八、代码

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

#define TIMER 10
#define MAXSPEED 180
#define MIDSPEED 90
#define MINSPEED 50

sbit CS1=P1^7;
sbit CS2=P1^6;
sbit RST=P1^5;
sbit E=P3^3;
sbit RW=P3^4;
sbit RS=P3^5;
sbit BUSY=P2^7;
sbit RESET=P2^4;
sbit KEY1=P3^6;
sbit KEY2=P3^7;
```

```
sbit OUT=P1^1;

sfr P4=0xc0;
sfr P4SW=0xbb;
sbit DCLK=P4^4;
sbit LED=P4^5;

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;
char code ledCode[10]={0xC0,0xF9,0xA4,0xB0,0x99,0x92,0x82,0xF8,0x80,0x90};
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*/
                        };
char                                                                    code
number[10][16]={{0x00,0xE0,0x10,0x08,0x08,0x10,0xE0,0x00,0x00,0x0F,0x10,0x20,0x20,0x10,0x
0F,0x00},/*"0",0*/

    {0x00,0x10,0x10,0xF8,0x00,0x00,0x00,0x00,0x00,0x00,0x20,0x20,0x3F,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){
    while(count--){
        _nop_();
    }
}

void send2LED(char temp){
    for(d=0;d<8;d++){
        DCLK=0;
        LED=temp & 0x80;
        DCLK=1;
        temp<<=1;
    }
}

void outLed(int num){
    a=num/100;
    b=(num-a*100)/10;
    c=num-a*100-b*10;
    send2LED(ledCode[c]);
    send2LED(ledCode[b]);
```

```
        send2LED(ledCode[a]);
}


void checkReady(bit cs){
        CS1= cs==0 ? 1:0;
        CS2= cs==1 ? 1:0;
        P2=0xff;
        E=1;
        RS=0;
        RW=1;
        while(BUSY==1);
        E=0;
        CS2=0;
        CS1=0;
}
void write_command(bit cs,char com){
        checkReady(cs);
        CS1= cs==0 ? 1:0;
        CS2= cs==1 ? 1:0;
        RS=0;
        RW=0;
        E=1;
        P2=com;
        wait(20);
        E=0;
        CS1=0;
        CS2=0;
}
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;
                        E=1;
                        RS=1;
                        RW=0;
                        P2=0x00;
                        wait(20);
                        E=0;
                }
        }
```

```
}
void write_bytes(bit cs,char * buffer,int len){
	for(i=0;i<len;i++){
		checkReady(cs);
		CS1= cs==0 ? 1:0;
		CS2= cs==1 ? 1:0;
		E=1;
		RS=1;
		RW=0;
		P2=*(buffer+i);
		wait(20);
		E=0;
	}
}
void outChina(bit cs,char x,char y,char china[][16]){
	write_command(cs,x+0xb8);
	write_command(cs,y+0x40);
	write_bytes(cs,*china,16);

	write_command(cs,x+1+0xb8);
	write_command(cs,y+0x40);
	write_bytes(cs,*(china+1),16);
}
void outNumber(bit cs,char x,char y,int num){
	write_command(cs,x+0xb8);
	write_command(cs,y+0x40);
	write_bytes(cs,number[num],8);
	write_command(cs,x+1+0xb8);
	write_command(cs,y+0x40);
	write_bytes(cs,number[num]+8,8);
}
void showNow(int num){
	a=num/1000;
	b=(num-a*1000)/100;
	c=(num-a*1000-b*100)/10;
	d=num-a*1000-b*100-c*10;
	outNumber(1,4,43,d);
	outNumber(1,4,33,c);
	outNumber(1,4,23,b);
	outNumber(1,4,13,a);
}
void showTarget(int num){
	a=num/1000;
	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);
    outNumber(0,4,33,c);
    outNumber(0,4,23,b);
    outNumber(0,4,13,a);
}
void initLCD(){
    cls(1);
    write_command(0,0x3f);
    cls(0);
    write_command(1,0x3f);
    outChina(0,1,5,mubiao);
    outChina(0,1,35,mubiao+2);
    outChina(1,1,5,dangqian);
    outChina(1,1,35,dangqian+2);
}
void ex_int0() interrupt 0
{
    count=count+1;
}
int getStepLen(int num){
    if(num<6){
        return 0;
    }
    if(num<10){
        return 1;
    }
    temp=num/100;
    temp=(num-temp*100)/10;
    temp=temp/3*2+1;
    if((N-temp)<1){
        return 1;
    }
    return temp;
}
void t0_int0() interrupt 1
{
    TR0=0;

    timer=timer-1;
    if(timer==0){
        if(countS==0){
            countS=count;
```

```
            currentSpeed=count*2;
        }else{
            currentSpeed=count+countS;
            countS=0;
        }
        timer=TIMER;
        count=0;
        if(currentSpeed<objSpeed){
            N=N+getStepLen(objSpeed-currentSpeed);
        }
        if(currentSpeed>objSpeed){
            N=N-getStepLen(currentSpeed-objSpeed);
        }
    }

    TH0=0x3c;
    TL0=0xb0;
    TR0=1;
}
void t1_int0() interrupt 3
{
    TR1=0;

    SUM=SUM+N;
    if((!KEY1)&&(!KEY2)){
        OUT=0;
    }else{
        if(!KEY1){
            objSpeed=MAXSPEED;
        }
        if(!KEY2){
            objSpeed=MINSPEED;
        }
        if(KEY1 && KEY2){
            objSpeed=MIDSPEED;
        }
        if(SUM>M){
            OUT=0;
            SUM=SUM-M;
        }else{
            OUT=1;
        }
    }
```

```
    TH1=0xff;
    TL1=0x9c;
    TR1=1;
}
void initTimer(){
    P4SW=0x30;

    TMOD=0x11;
    TH0=0x3c;
    TL0=0xb0;
    TH1=0xff;
    TL1=0x9c;

    IT0=1;
    EA=1;
    ET0=1;
    ET1=1;
    EX0=1;

    TR0=1;
    TR1=1;
}
void waitL(int count){
    k=0xffff;
    while(count--){
        while(k--);
    }
}
void main(){
    initLCD();
    initTimer();
    while(1){
        if((!KEY1)&&(!KEY2)){
            outLed(currentSpeed);
        }else{
            showTarget(objSpeed);
            showNow(currentSpeed);
        }
        waitL(0x05);
    }
}
```

第八次实验（温度测量与控制）报告

一、实验原理

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、实验器材

1. 单片机测控实验系统
2. 温控实验模块
3. Keil 开发环境
4. STC-ISP 程序下载工具

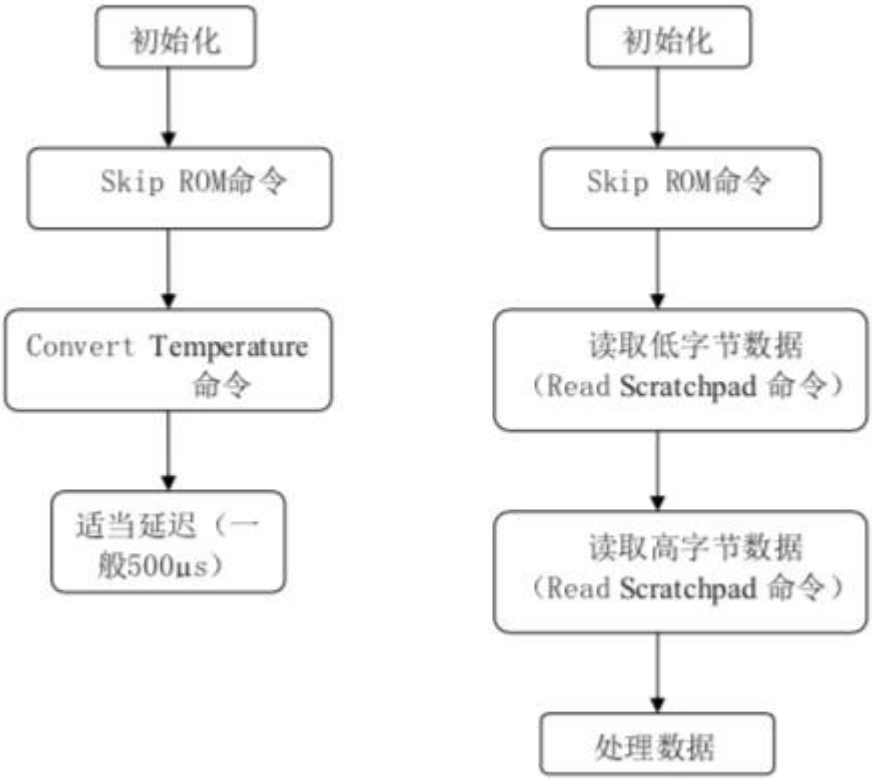
三、实验内容

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、实验过程

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、实验流程图



六、思考题

1. 进行精确的延时程序有几种方法?各有什么优缺点?

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。

定时器方法：通过定时器来进行延时，通常可以规定 2-3 个计时器同时进行延时，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。

七、代码

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

sbit CS1=P1^7;
```

```
sbit CS2=P1^6;
sbit RST=P1^5;
sbit E=P3^3;
sbit RW=P3^4;
sbit RS=P3^5;
sbit BUSY=P2^7;
sbit KEY1=P3^6;
sbit KEY2=P3^7;

sbit DQ=P1^4;
sbit RQ=P1^1;

bit bt;

int i,j,k,a,b,c,d,e,f,objTemp=0,nowTemp=0;
int firstDiff=0,secondDiff=0,N=0,M=256,SUM=0,x;
unsigned char t,temp,status=0x01,tempNum=0x00;
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*/
                    };
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){
    while(count--){
        _nop_();
        _nop_();
    }
}
void checkReady(bit cs){
    CS1= cs==0 ? 1:0;
    CS2= cs==1 ? 1:0;
    P2=0xff;
    E=1;
    RS=0;
    RW=1;
    while(BUSY==1);
    E=0;
```

```
	CS2=0;
	CS1=0;
}
void write_command(bit cs,char com){
	checkReady(cs);
	CS1= cs==0 ? 1:0;
	CS2= cs==1 ? 1:0;
	RS=0;
	RW=0;
	E=1;
	P2=com;
	wait(20);
	E=0;
	CS1=0;
	CS2=0;
}
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;
			E=1;
			RS=1;
			RW=0;
			P2=0x00;
			wait(20);
			E=0;
		}
	}
}
void write_bytes(bit cs,char * buffer,int len){
	for(i=0;i<len;i++){
		checkReady(cs);
		CS1= cs==0 ? 1:0;
		CS2= cs==1 ? 1:0;
		E=1;
		RS=1;
		RW=0;
		P2=*(buffer+i);
		wait(20);
		E=0;
	}
```

```
}
void outChina(bit cs,char x,char y,char china[][16]){
	write_command(cs,x+0xb8);
	write_command(cs,y+0x40);
	write_bytes(cs,*china,16);

	write_command(cs,x+1+0xb8);
	write_command(cs,y+0x40);
	write_bytes(cs,*(china+1),16);
}
void outNumber(bit cs,char x,char y,int num){
	write_command(cs,x+0xb8);
	write_command(cs,y+0x40);
	write_bytes(cs,number[num],8);
	write_command(cs,x+1+0xb8);
	write_command(cs,y+0x40);
	write_bytes(cs,number[num]+8,8);
}
void showNow(int num){
	a=num/1000;
	b=(num-a*1000)/100;
	c=(num-a*1000-b*100)/10;
	d=num-a*1000-b*100-c*10;
	outNumber(1,4,43,d);
	outNumber(1,4,33,c);
	outNumber(1,4,23,b);
	outNumber(1,4,13,a);
}
void showTarget(int num){
	a=num/1000;
	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);
	outNumber(0,4,33,c);
	outNumber(0,4,23,b);
	outNumber(0,4,13,a);
}
void initLCD(){
	cls(1);
	write_command(0,0x3f);
	cls(0);
	write_command(1,0x3f);
	outChina(0,1,5,mubiao);
```

```
        outChina(0,1,35,mubiao+2);
        outChina(1,1,5,dangqian);
        outChina(1,1,35,dangqian+2);
}
bit initDS(void)
{
   DQ = 0;
   wait(480);
   DQ = 1;
   wait(60);
      bt=DQ;
      wait(420);
   return bt;
}

unsigned char readByte(void)
{
      temp=0;
   for (i=8;i>0;i--)
   {
            temp>>=1;
      DQ = 0;
            wait(1);
      DQ = 1;
            wait(1);
      if(DQ){
            temp|=0x80;
            }
            wait(60);
   }
   return temp;
}

void writeByte(unsigned char dat)
{
   for (i=8; i>0; i--)
   {
      DQ = 0;
            wait(8);
            dat>>=1;
            DQ=CY;
            wait(60);
      DQ = 1;
      wait(10);
```

```
    }
}

int readTemperature(void)
{
    initDS();
        while(!DQ);
    writeByte(0xcc);
    writeByte(0x44);
        while(!DQ);
    initDS();
        while(!DQ);
    writeByte(0xcc);
    writeByte(0xbe);
    a=readByte();
    b=readByte();
        P0=a;
        wait(255);
        x=b;
        x<<=8;
        x|=a;
        return x*0.0625;
}
void waitL(int count){
        k=0xffff;
        while(count--){
                while(k--);
        }
}
int getStep(int diff1,int diff2){
        return (diff2+(diff1+diff2)/2+(diff2-diff1))/2;
}
void main(){
        initLCD();
        while(1){
                SUM+=N;
                if(SUM>M){
                        RQ=1;
                        SUM-=M;
                }else{
                        RQ=0;
                }
                if(!KEY1){
                        objTemp=70;
```

```
            }else{
                objTemp=00;
            }
            nowTemp=readTemperature();
            if(nowTemp!=objTemp){
                if(firstDiff==0){
                    firstDiff=objTemp-nowTemp;
                }else{
                    secondDiff=objTemp-nowTemp;
                    N+=getStep(firstDiff,secondDiff);
                    if(N<0){
                        N=0;
                    }
                    if(N>M+1){
                        N=M;
                    }
                    firstDiff=0;
                }
            }
            showTarget(objTemp);
            showNow(nowTemp);
            waitL(0x03);
        }
    }
```