实验五 重量测量

1. **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。

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 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）。

A/D转换器的具体使用方法参见STC单片机指南的第九章。

6. 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。

1. **实验器材**

1.单片机测控实验系统

2.重量测量实验板/砝码

3.Keil开发环境

4.STC-ISP程序下载工具

**四、实验内容**

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。

1. **实验步骤**

同PPT所述。

1. **思考题**
2. 调零的原理，软件调零和调零调零的区别。

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。

1. 模/数和数/模的信号转换原理。

模/数和数/模的信号转换原理：ADC 的转换原理根据的电路形式有所不同。ADC 电路通常由两部分组成，它们是：采样、保持电路和量化、编码电路。其中量化、编码电路是最核心的部件，任何 AD 转换电路都必须包含这种电路。ADC 电路的形式很多，通常可以并为两类：间接法：它是将采样保持的模拟信号先转换成与模拟量成正比的时间或频率，然后再把它转换为数字量。这种通常是采用时钟脉冲计数器，它又被称为计数器式。它的工作特点是：工作速度低，转换精度高，抗干扰能力强。直接法：通过基准电压与采样保持信号进行比较，从而转换为数字量。它的工作特点是：工作速度高，转换精度容易保证。DA 转换器是由数码寄存器、模拟电子开关电路、解码电路、求和电路及基准电压及部分组成。数字量是以串行或并行方式输入并存储在数码寄存器中，寄存器输出的每位数码驱动对应数位上的电子开关将电阻解码网络中获得的相应数位权值送入数位求和电路中 2，求和电路将各位权值相加就得到与数字量相应的模拟量。

1. I2C总线在信号通讯过程中的应用。

I2C 总线在信号通讯过程中的应用：I2C 总线是 Phihps 公司开发的一种双向两线多主机总线，利用两根信号线来实现设备之间的信息传递，一根为数据线 SDA，一根为时钟线 SCL。它能方便地实现芯片间的数据传输与控制。通过两线缓冲接口和内部控制与状态寄存器，可方便地完成多机间的非主从通信或主从通信。基于 I2C 总线的多机通信电路结构简单、程序编写方便，易于实现系统软硬件的模块化和标准化，被广泛用于系统内部微控制器和外部设备之间的串行通讯。

实验记录

1. 代码

#include <reg52.h>

#include <intrins.h>

#define uchar unsigned char

#define uint unsigned int

sbit CS1=P1^7;

sbit CS2=P1^6;//选择左右

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;//当前为运行状态

sfr ADC\_CONTR = 0xBC; ///ADC control register

sfr ADC\_RES = 0xBD; ///ADC hight 8-bit result register

sfr ADC\_LOW2 = 0xBE; ///ADC low 2-bit result register

sfr P1ASF = 0x9D;//P1 secondary function control register: P1口模拟功能控制寄存器

sfr AURX1 = 0xA2; //AURX1中的ADRJ位用于转换结果寄存器的数据格式调整控制

#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; ///模拟输入通道选择

uchar code zima[20][32]=

{

0x00,0x00,0xC0,0xE0,0x30,0x10,0x08,0x08,0x08,0x08,0x08,0x18,0x30,0xE0,0xC0,0x00,

0x00,0x00,0x07,0x0F,0x18,0x10,0x20,0x20,0x20,0x20,0x20,0x10,0x18,0x0F,0x07,0x00,///\*"0"\*0/

0x00,0x00,0x00,0x10,0x10,0x10,0x10,0xF0,0xF8,0x00,0x00,0x00,0x00,0x00,0x00,0x00,

0x00,0x00,0x00,0x20,0x20,0x20,0x20,0x3F,0x3F,0x20,0x20,0x20,0x20,0x00,0x00,0x00,///\*"1"\*1/

0x00,0x00,0x60,0x50,0x10,0x08,0x08,0x08,0x08,0x08,0x08,0x98,0xF0,0x70,0x00,0x00,

0x00,0x00,0x20,0x30,0x28,0x28,0x24,0x24,0x22,0x22,0x21,0x20,0x30,0x18,0x00,0x00,///\*"2"\*2/

0x00,0x00,0x30,0x30,0x08,0x08,0x88,0x88,0x88,0x88,0x58,0x70,0x30,0x00,0x00,0x00,

0x00,0x00,0x18,0x18,0x20,0x20,0x20,0x20,0x20,0x20,0x31,0x11,0x1F,0x0E,0x00,0x00,///\*"3"\*3/

0x00,0x00,0x00,0x00,0x00,0x80,0x40,0x20,0x10,0xF0,0xF8,0xF8,0x00,0x00,0x00,0x00,

0x00,0x04,0x06,0x05,0x05,0x04,0x24,0x24,0x24,0x3F,0x3F,0x3F,0x24,0x24,0x24,0x00,///\*"4"\*4/

0x00,0x00,0x00,0xC0,0x38,0x88,0x88,0x88,0x88,0x88,0x88,0x88,0x08,0x08,0x00,0x00,

0x00,0x00,0x18,0x29,0x21,0x20,0x20,0x20,0x20,0x20,0x30,0x11,0x1F,0x0E,0x00,0x00,///\*"5"\*5/

0x00,0x00,0x80,0xE0,0x30,0x10,0x98,0x88,0x88,0x88,0x88,0x88,0x98,0x10,0x00,0x00,

0x00,0x00,0x07,0x0F,0x19,0x31,0x20,0x20,0x20,0x20,0x20,0x20,0x11,0x1F,0x0E,0x00,///\*"6"\*6/

0x00,0x00,0x30,0x18,0x08,0x08,0x08,0x08,0x08,0x88,0x48,0x28,0x18,0x08,0x00,0x00,

0x00,0x00,0x00,0x00,0x00,0x00,0x38,0x3E,0x01,0x00,0x00,0x00,0x00,0x00,0x00,0x00,///\*"7"\*7/

0x00,0x00,0x70,0x70,0xD8,0x88,0x88,0x08,0x08,0x08,0x08,0x98,0x70,0x70,0x00,0x00,

0x00,0x0C,0x1E,0x12,0x21,0x21,0x20,0x21,0x21,0x21,0x23,0x12,0x1E,0x0C,0x00,0x00,///\*"8"\*8/

0x00,0xE0,0xF0,0x10,0x08,0x08,0x08,0x08,0x08,0x08,0x18,0x10,0xF0,0xC0,0x00,0x00,

0x00,0x00,0x11,0x33,0x22,0x22,0x22,0x22,0x22,0x32,0x11,0x1D,0x0F,0x03,0x00,0x00,///\*"9"\*9/

0x08,0x08,0x0A,0xEA,0xAA,0xAA,0xAA,0xFF,0xA9,0xA9,0xA9,0xE9,0x08,0x08,0x08,0x00,

0x40,0x40,0x48,0x4B,0x4A,0x4A,0x4A,0x7F,0x4A,0x4A,0x4A,0x4B,0x48,0x40,0x40,0x00,///\*"重"\*10/

0x40,0x40,0x40,0xDF,0x55,0x55,0x55,0xD5,0x55,0x55,0x55,0xDF,0x40,0x40,0x40,0x00,

0x40,0x40,0x40,0x57,0x55,0x55,0x55,0x7F,0x55,0x55,0x55,0x57,0x50,0x40,0x40,0x00,///\*"量"\*11/

0x00,0x00,0x00,0x00,0x00,0x00,0xC0,0xC0,0xC0,0xC0,0x00,0x00,0x00,0x00,0x00,0x00,

0x00,0x00,0x00,0x00,0x00,0x00,0x30,0x30,0x30,0x30,0x00,0x00,0x00,0x00,0x00,0x00,///\*":"\*12/

0x00,0x04,0x04,0xE4,0x24,0x24,0x24,0x3F,0x24,0x24,0x24,0xE4,0x04,0x04,0x00,0x00,

0x00,0x00,0x80,0x43,0x31,0x0F,0x01,0x01,0x01,0x3F,0x41,0x43,0x40,0x40,0x70,0x00,///\*"克"\*13/

};

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;//初始重量，用于校准

int weight;//真实重量

void main()

{

init\_yejing();//

init\_adc();

calibrate();//校准

while(1)

{

weight=(get\_ad\_result()-cweight)/1.81;

clearscreen();//清屏

send\_all(1,1,10);//重

send\_all(1,2,11);//量

send\_all(1,3,12);//:

send\_all(4,3,weight/100);//百

send\_all(4,4,(weight/10)%10);//十

send\_all(4,5,weight%10);//个

send\_all(4,6,13);//克

delay(50000);

}

}

void init\_yejing()

{

send\_byte(192,1,1);//设置起始行，规定了显示屏上最顶一行所对应的显示存储器的行地址，默认格式的最高两位是1，所以是在192的基础上加

send\_byte(63,1,1);//打开显示开关显示开关设置，默认格式为0011111D，D为1时候显示，为0不显示

}

void send\_byte(uchar dat,uchar cs1,uchar cs2)

{

P2=0xff;

CS1=cs1; CS2=cs2;

RS=0; RW=1; E=1;//读状态字

while(BUSY) ;

//送数据或者控制字

E=0;

RS=!(cs1&&cs2),RW=0;//写指令代码

P2=dat;

E=1; delay(3);

E=0;//总线释放

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;//选择页面 184-页面地址设置，也就是X的设置，默认格式的高五位是10111，所以是在184的基础上加

send\_byte(64+lie\*16-(lie>3)\*64,1,1);//选择列号，也就是Y的设置，默认格式中最高两位是01，所以是在64的基础上加，Y是自动加一的

for(j=0;j<16;++j)

send\_byte(zima[offset][k++],lie<4,lie>=4);//送数

}

}

void init\_adc()

{

P1ASF = 1;//Set P1.0 as analog input port

AURX1 |= 0X04;//AURX1中的ADRJ位用于转换结果寄存器的数据格式调整控制

ADC\_RES = ADC\_LOW2 = 0; //Clear previous result

ADC\_CONTR = ADC\_POWER | ADC\_SPEEDLL | ADC\_START | ch;//ch=0 ADC channel NO.0

delay(4);//ADC power-on delay and Start A/D conversion

}

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\_();//Must wait before inquiry

while (!(ADC\_CONTR & ADC\_FLAG)); //Wait complete flag

ADC\_result = (ADC\_RES & 0x03) \*256 + ADC\_LOW2;//ADC\_RES中存高2位

ADC\_CONTR &= ~ADC\_FLAG;//Close ADC flag位置0

return ADC\_result;

}

void calibrate()//校正

{

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|line

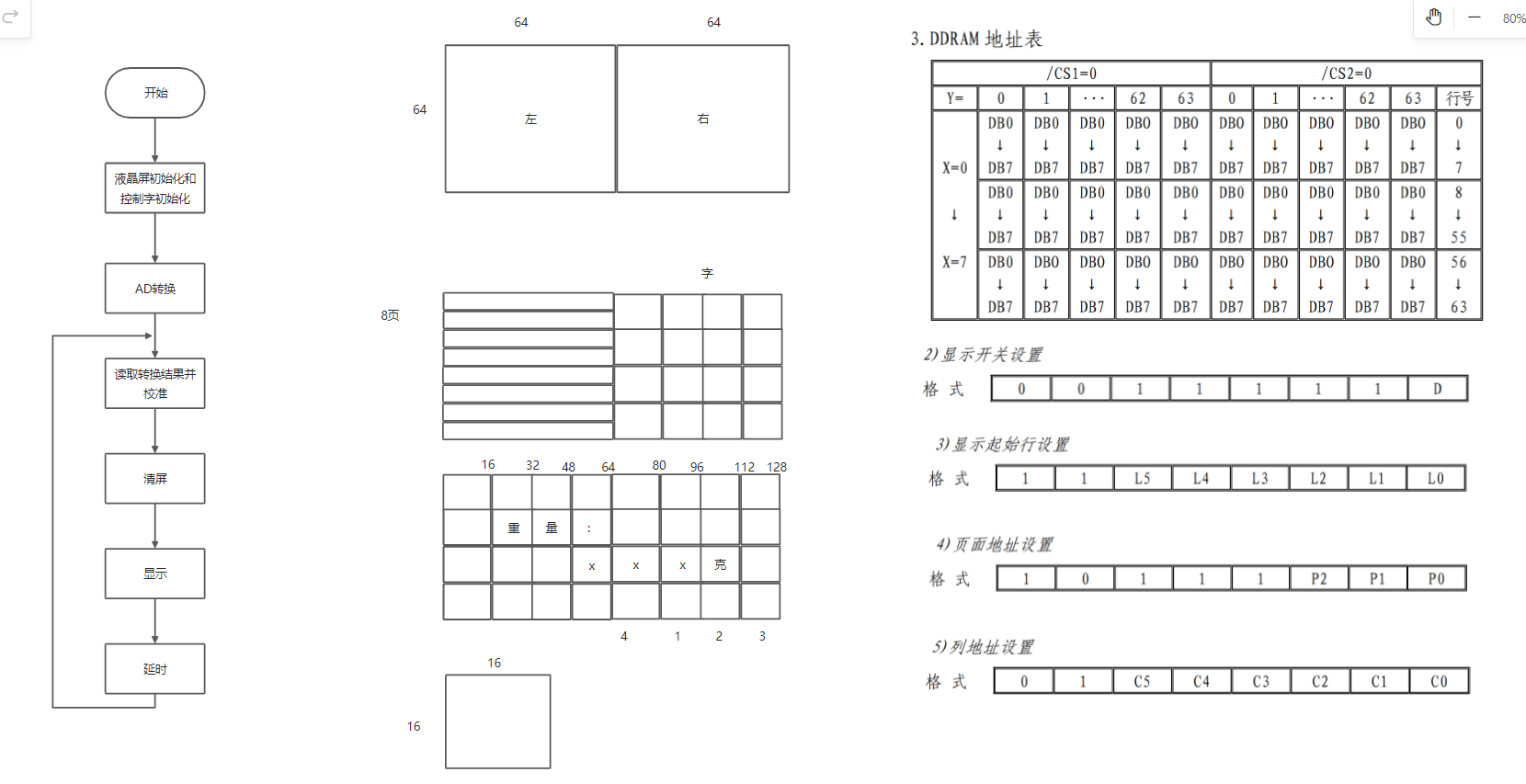
for(j=0;j<64;++j)

{

send\_byte(0x00,0,1);

send\_byte(0x00,1,0);

}}}

1. 流程图
2. 修改的代码：滚动显示

#include <reg52.h>

#include <intrins.h>

#define uchar unsigned char

#define uint unsigned int

sbit CS1=P1^7;

sbit CS2=P1^6;//选择左右

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;//当前为运行状态

sfr ADC\_CONTR = 0xBC; ///ADC control register

sfr ADC\_RES = 0xBD; ///ADC hight 8-bit result register

sfr ADC\_LOW2 = 0xBE; ///ADC low 2-bit result register

sfr P1ASF = 0x9D;//P1 secondary function control register: P1口模拟功能控制寄存器

sfr AURX1 = 0xA2; //AURX1中的ADRJ位用于转换结果寄存器的数据格式调整控制

int yy;

int xx;

uchar ch = 0; ///模拟输入通道选择

uchar code zima[20][32]=

{

0x00,0x00,0xC0,0xE0,0x30,0x10,0x08,0x08,0x08,0x08,0x08,0x18,0x30,0xE0,0xC0,0x00,

0x00,0x00,0x07,0x0F,0x18,0x10,0x20,0x20,0x20,0x20,0x20,0x10,0x18,0x0F,0x07,0x00,///\*"0"\*0/

0x00,0x00,0x00,0x10,0x10,0x10,0x10,0xF0,0xF8,0x00,0x00,0x00,0x00,0x00,0x00,0x00,

0x00,0x00,0x00,0x20,0x20,0x20,0x20,0x3F,0x3F,0x20,0x20,0x20,0x20,0x00,0x00,0x00,///\*"1"\*1/

0x00,0x00,0x60,0x50,0x10,0x08,0x08,0x08,0x08,0x08,0x08,0x98,0xF0,0x70,0x00,0x00,

0x00,0x00,0x20,0x30,0x28,0x28,0x24,0x24,0x22,0x22,0x21,0x20,0x30,0x18,0x00,0x00,///\*"2"\*2/

0x00,0x00,0x30,0x30,0x08,0x08,0x88,0x88,0x88,0x88,0x58,0x70,0x30,0x00,0x00,0x00,

0x00,0x00,0x18,0x18,0x20,0x20,0x20,0x20,0x20,0x20,0x31,0x11,0x1F,0x0E,0x00,0x00,///\*"3"\*3/

0x00,0x00,0x00,0x00,0x00,0x80,0x40,0x20,0x10,0xF0,0xF8,0xF8,0x00,0x00,0x00,0x00,

0x00,0x04,0x06,0x05,0x05,0x04,0x24,0x24,0x24,0x3F,0x3F,0x3F,0x24,0x24,0x24,0x00,///\*"4"\*4/

0x00,0x00,0x00,0xC0,0x38,0x88,0x88,0x88,0x88,0x88,0x88,0x88,0x08,0x08,0x00,0x00,

0x00,0x00,0x18,0x29,0x21,0x20,0x20,0x20,0x20,0x20,0x30,0x11,0x1F,0x0E,0x00,0x00,///\*"5"\*5/

0x00,0x00,0x80,0xE0,0x30,0x10,0x98,0x88,0x88,0x88,0x88,0x88,0x98,0x10,0x00,0x00,

0x00,0x00,0x07,0x0F,0x19,0x31,0x20,0x20,0x20,0x20,0x20,0x20,0x11,0x1F,0x0E,0x00,///\*"6"\*6/

0x00,0x00,0x30,0x18,0x08,0x08,0x08,0x08,0x08,0x88,0x48,0x28,0x18,0x08,0x00,0x00,

0x00,0x00,0x00,0x00,0x00,0x00,0x38,0x3E,0x01,0x00,0x00,0x00,0x00,0x00,0x00,0x00,///\*"7"\*7/

0x00,0x00,0x70,0x70,0xD8,0x88,0x88,0x08,0x08,0x08,0x08,0x98,0x70,0x70,0x00,0x00,

0x00,0x0C,0x1E,0x12,0x21,0x21,0x20,0x21,0x21,0x21,0x23,0x12,0x1E,0x0C,0x00,0x00,///\*"8"\*8/

0x40,0xC0,0xC0,0xC0,0xC0,0xE0,0xFC,0xFF,0xFC,0xE0,0xC0,0xC0,0xC0,0xC0,0x40,0x00,

0x00,0x00,0x41,0x39,0x3F,0x1F,0x1F,0x0F,0x1F,0x1F,0x3F,0x39,0x41,0x00,0x00,0x00,

//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,

0x20,0x22,0x2A,0x2A,0xAA,0x6A,0x3A,0x2E,0x29,0x29,0x29,0x29,0x29,0x20,0x20,0x00,

0x08,0x04,0x02,0x01,0xFF,0x55,0x55,0x55,0x55,0x55,0x55,0xFF,0x00,0x00,0x00,0x00,

//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,

0x20,0x24,0x24,0x24,0xFE,0x23,0x22,0x20,0x20,0xFF,0x20,0x22,0x2C,0xA0,0x20,0x00,

0x00,0x08,0x48,0x84,0x7F,0x02,0x41,0x40,0x20,0x13,0x0C,0x14,0x22,0x41,0xF8,0x00,

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/

};

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(int yy);

void calibrate();

int get\_ad\_result();

void clearscreen();

void main()

{

send\_byte(63,1,1);//打开显示开关显示开关设置，默认格式为0011111D，D为1时候显示，为0不显示

while(1)

{

yy=0;

for (;yy<64;yy++)

{

init\_yejing(yy);

clearscreen();//清屏

send\_all(4,2,9);//

send\_all(4,3,10);//看

send\_all(4,4,11);//我

delay(500000);

delay(500000);

}

}

}

void init\_yejing(int xx)

{

send\_byte(192+xx,1,1);//设置起始行，规定了显示屏上最顶一行所对应的显示存储器的行地址，默认格式的最高两位是1，所以是在192的基础上加

//send\_byte(63,1,1);//打开显示开关显示开关设置，默认格式为0011111D，D为1时候显示，为0不显示

}

void send\_byte(uchar dat,uchar cs1,uchar cs2)

{

P2=0xff;

CS1=cs1; CS2=cs2;

RS=0; RW=1; E=1;//读状态字

while(BUSY) ;

//送数据或者控制字

E=0;

RS=!(cs1&&cs2),RW=0;//写指令代码

P2=dat;

E=1; delay(3);

E=0;//总线释放

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;//选择页面 184-页面地址设置，也就是X的设置，默认格式的高五位是10111，所以是在184的基础上加

send\_byte(64+lie\*16-(lie>3)\*64,1,1);//选择列号，也就是Y的设置，默认格式中最高两位是01，所以是在64的基础上加，Y是自动加一的

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);///10111000|page

send\_byte(64,1,1);///01000000|line

for(j=0;j<64;++j)

{

send\_byte(0x00,0,1);

send\_byte(0x00,1,0);

}}}

实验感想

（1）问题分析：在调试校准中出现问题（其中软件调零通过测量点对数据进行曲线拟合））

（2）实验体会：通过本次实验，学会了 A/D 转换的原理，以及液晶显示屏的使用方法。