**实验五 重量测量**

21200903

**一、实验目的&实验要求**

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。

**三、实验原理**

本实验主要可以分为两大部分，第一部分是利用 AD 转换器测量物品重量，第二部 分是将重量值送到液晶屏进行显示。具体来看，称重托盘下的重量传感器利用压敏电阻采集应变，产生变化的阻值，通过放大电路将其转化为电压值，再通过 AD 转换器将电压值转化为 CPU 可以处理的数字信号。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. **AD 转换器**

AD 转换器通过模拟多路开关，将通过 ADC0~7 的模拟量输入送给比较器。 用数/模转换器(DAC)转换的模拟量与本次输入的模拟量通过比较器进行比较，将比较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。A/D 转换结束后，最终的转换结果保存到 ADC 转换结果寄存器 ADC\_RES 和 ADC\_RESL，同时置位 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 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DC\_POWER 位。

**2. 液晶显示屏 LCM**

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，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。数据显示格式如下图所示：重要寄存器：

1) 状态字寄存器：

状态字寄存器是 LCM 与单片机通讯时唯一的“握手”信号。状态字寄存器向单片机表示其当前工作状态，尤其是状态字中的“忙”标志位是单片机在每次对 LCM 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，此时任何操作都将是无效的。

2）显示起始行寄存器：

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。

3）显示开/关触发器：

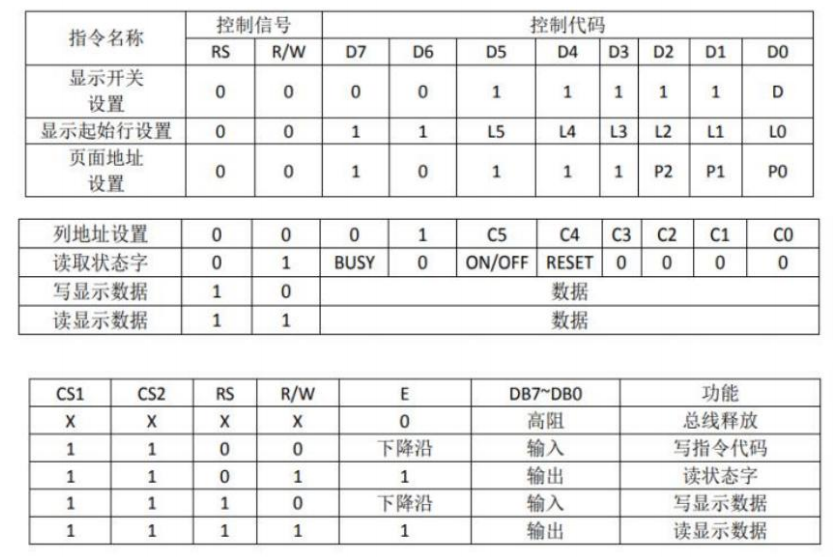
显示开/关触发器的作用就是控制显示驱动输出的电平以控制显示屏的开关。

4）复位端/RES：

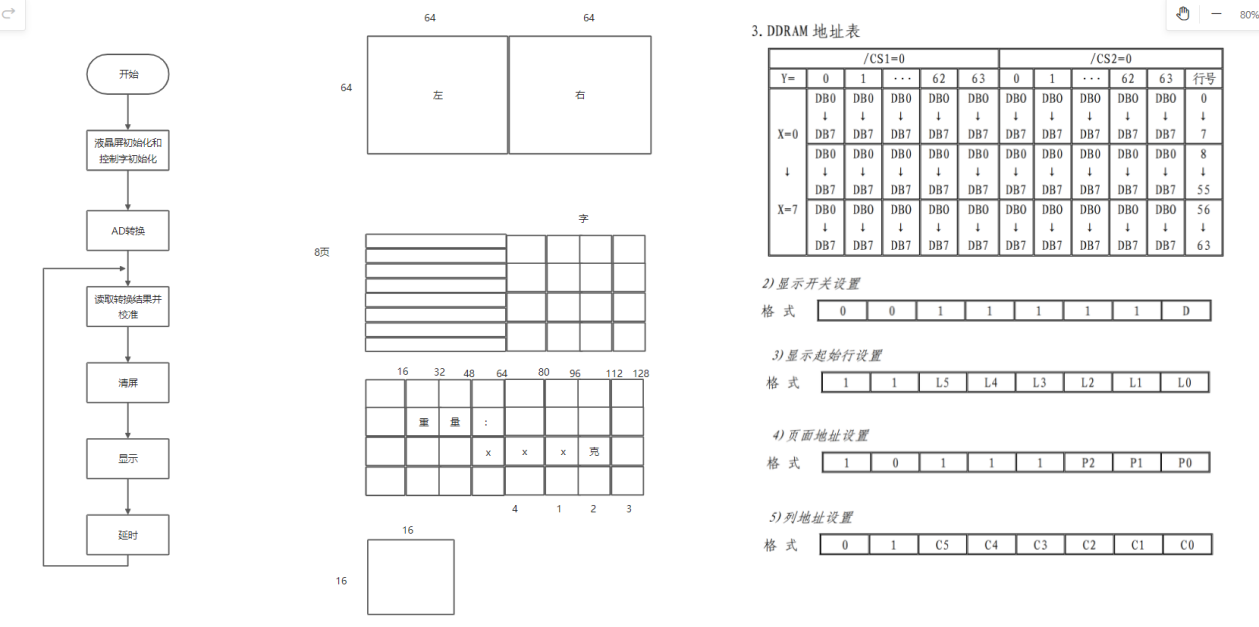
复位端/RES 用于在 LCM 上电时或需要时实现硬件电路对 LCM 的复位。

5）I/O 缓冲器：

I/O 缓冲器为双向三态数据缓冲器。是 LCM 内部总线与单片机总线的结合部。其作用是将两个不同时钟下工作的系统连接起来实现通讯。I/O 缓冲器在片选信号 CS 有效状态下,I/O缓冲器开放，实现 LCM 与单片机之间的数据传递。对液晶屏的操作控制主要依靠如下图所示的指令表和时序表：



**四、流程图**

****

1. **实验代码**

#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

int yy;

void main()

{

init\_yejing();//

init\_adc();

calibrate();//校准

while(1)

{

weight=(get\_ad\_result()-cweight)/2.3;

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

yy=weight%10;

if (yy>7)

{

send\_all(4,4,(weight/10)%10+1);//十

send\_all(4,5,0);

}

else

{

send\_all(4,4,(weight/10)%10);//十

send\_all(4,5,0);

}

//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0,一定要软件清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. **思考题**

1. 调零的原理，软件调零和硬件调零的区别。

软件调零是采用软件进行补偿的方法，又称数字调零；硬件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。

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。

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. I 2 C 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 I 2 C 总线接口器件已经比较普遍。另外，通用的 I 2 C 总线接口器件，

如带 I 2 C 总线的单片机、RAM、ROM、A/D、D/A、LCD 驱动器等器件，也越来越多地应

用于计算机及自动控制系统中。I 2 C 总线通过 SDA（串行数据线）及 SCL（串行时钟线）

两根线在连到总线上的器件之间传送信息，并根据地址识别每个器件。目前在仪器仪表、

移动通信、密码控制等领域采用 I 2 C 总线接口器件已经比较普遍。另外，通用的 I 2 C 总

线接口器件，如带 I 2 C 总线的单片机、RAM、ROM、A/D、D/A、LCD 驱动器等器件，也

越来越多地应用于计算机及自动控制系统中。

1. **实验过程中遇到的问题及实验感想**

在实验过程中，发现大砝码测量时，误差过大，通过尝试改变代码中“ weight = (get\_ad\_result ()-cweight)/1.8;”处除数，发现调大除数能使测量值更精确，通过软件调零的方法，把其值改到2.3后，结果精确度提高。