单片机实验五实验报告

一、实验目的&实验要求

- 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。
- 掌握模拟/数字（A/D）转换方式
- 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

- 参考辅助材料，学习 C51 语言使用。
- 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。

- **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 ADC_POWER 位。

相关寄存器：

1) P1ASF：

设置将 8 路中的任何一路设置为 AD 转换，不需作为 AD 转换使用的口可继续作为 IO 口使用。

2) ADC_CONTR：

a) ADC_POWER：电源控制位。初次打开内部 AD 转换模拟电源，需适当延时，等内部模拟电源稳定后，再启动 AD 转换。

b) SPEED1，SPEED0：模数转换器转换速度控制位。

c) ADC_FLAG：模数转换器转换结束标志位，不管是 AD 转换完成后由该位申请产生中断，还是由软件查询该标志位 AD 转换是否结束，当 AD 转换完成后，ADC_FLAG = 1，要由软件清 0。

d) ADC_START：AD 转换启动控制位，设置为“1”时，开始转换，转换结束后为 0。

e) CHS2/CHS1/CHS0：模拟输入通道选择。
由于是 2 套时钟,所以,设置 ADC_CONTR 控制寄存器后,要加 4 个空操作延时才可以正确读到 ADC_CONTR 寄存器的值。

3) ADC_RES、ADC_RESL：
用于保存 A/D 转换结果，当 AUXR1 寄存器的 ADRJ 位（数据格式调整控制位）为 0 时，10 位 AD 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 为 1 时，10 位 AD 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。

4) IE：
中断允许寄存器。位 EA 是 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；位 EADC 是 AD 转换中断允许位，EADC=1，允许 AD 转换中断，EADC=0，禁止 AD 转换中断。

5) IPH/IP：
中断优先级控制寄存器高/低，共分为 4 个优先级。

- **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。

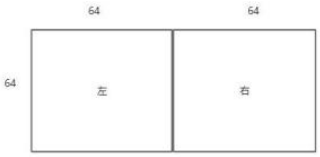
5）I/O 缓冲器：
I/O 缓冲器为双向三态数据缓冲器。是 LCM 内部总线与单片机总线的结合部。其作用是将两个不同时钟下工作的系统连接起来实现通讯。I/O 缓冲器在片选信号 CS 有效状态下,I/O 缓冲器开放，实现 LCM 与单片机之间的数据传递。对液晶屏的操作控制主要依靠如下图所示的指令表和时序表：

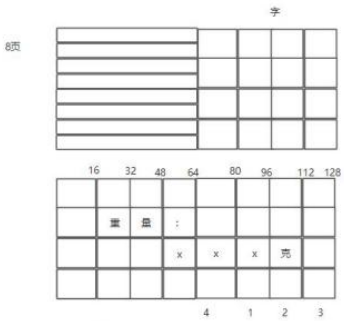
指令名称	控制信号		控制代码							
	RS	R/W	D7	D6	D5	D4	D3	D2	D1	D0
显示开关设置	0	0	0	0	1	1	1	1	1	D
显示起始行设置	0	0	1	1	L5	L4	L3	L2	L1	L0
页面地址设置	0	0	1	0	1	1	1	P2	P1	P0

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							

CS1	CS2	RS	R/W	E	DB7~DB0	功能
X	X	X	X	0	高阻	总线释放
1	1	0	0	下降沿	输入	写指令代码
1	1	0	1	1	输出	读状态字
1	1	1	0	下降沿	输入	写显示数据
1	1	1	1	1	输出	读显示数据

四、实验流程图





3. 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

2)显示开关设置

格式								
	0	0	1	1	1	1	1	D

3)显示起始行设置

格式								
	1	1	L5	L4	L3	L2	L1	L0

4)页面地址设置

格式								
	1	0	1	1	1	P2	P1	P0

5)列地址设置

格式								
	0	1	C5	C4	C3	C2	C1	C0

五、代码内容

```
#include <reg52.h>
#include <intrins.h>
#define uchar unsigned char
#define uint unsigned int
//初始化端口引脚、寄存器地址、用于控制的数值常量
sbit CS1=P1^7;//左屏
sbit CS2=P1^6;//右屏 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（忙状态位）
sfr ADC_CONTR = 0xBC; ///ADC 控制寄存器
sfr ADC_RES = 0xBD; ///ADC 高八位结果寄存器
sfr ADC_LOW2 = 0xBE; ///ADC 低二位结果寄存器
sfr P1ASF = 0x9D;//P1 口模拟功能控制寄存器，通道选择寄存器
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 开关控制
#define ADC_FLAG 0x10 //ADC 转换完成标志
#define ADC_START 0x08 //ADC 开始标记
#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; //ADC 输入通道号
//设置码值表，用于存放在液晶屏上显示的字符和数字
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,
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x38,0x3E,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,0xA9,0xA9,0xA9,0xE9,0x08,0x08,0x08,0x
00,
0x40,0x40,0x48,0x4B,0x4A,0x4A,0x4A,0x7F,0x4A,0x4A,0x4A,0x4B,0x48,0x40,0x40,0x0
0,///*"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/
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0xC0,0xC0,0xC0,0xC0,0x00,0x00,0x00,0x00,0x00,0x00,0x00
,
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x30,0x30,0x30,0x30,0x00,0x00,0x00,0x00,0x00,0x00,0x00,/
/
/*":"*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);//向液晶屏送入 8 位数据
void send_all(uint page,uint lie,uint offset);//写液晶屏，显示相应字
void delay(uint x);//延时
void init_adc();//初始化 AD 转换器
```

```
void init_yejing();//初始化液晶屏幕
void calibrate();//校准
int get_ad_result();//获取重量
void clearscreen();//清屏
int cweight;//初始重量，用于校准
int weight;//真实重量
int yy;//重量个位，消除机器误差
void main()
{init_yejing();//初始化液晶屏
init_adc();//初始化 AD 转换器
calibrate();//校准
while(1)
{
weight=(get_ad_result()-cweight)/2.05;
yy=weight%10; //结果四舍五入，消除机器误差
if (yy>=5)
{
weight = weight - yy + 10;
}
else
{
weight = weight - yy;
}
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);
}
}
//初始化液晶屏
void init_yejing()
{
//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(192,1,1);
//打开显示开关，默认格式为 0011111D，D 为 1 时候显示，为 0 不显示
send_byte(63,1,1);
}
//向液晶屏送入 8 位数据
```

```
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) ;//判断是否在忙，若忙则等待
//送数据和控制字
E=0;
RS=!(cs1&&cs2),RW=0;//写指令代码
P2=dat;//数据写入左半屏
E=1; //在 E 使能端口的上升沿写显示数据
delay(3);//必须有延时
E=0;//总线释放
CS1=CS2=0;
}
//写液晶屏，显示相应字，page 是页号，lie 是列号
void send_all(uint page,uint lie,uint offset)
{
uint i,j,k=0;
for(i=0;i<2;++i)
{
//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(184+i+page,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 地址计数器是自动加一的
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);//送数
}
}
//初始化 AD 转换器
void init_adc()
{
P1ASF = 1;//选择 P1.0 端口
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位
ADC_RES = ADC_LOW2 = 0; //清除之前的结果
//设置 ADC_CONTR 控制寄存器的值
ADC_CONTR = ADC_POWER | ADC_SPEEDLL | ADC_START | ch;//ch=0 ADC 通道选择 0
delay(4);//由于是 2 套时钟，需要 4 个延时空操作才能读到 ADC_CONTR 的值，开始转换
}
//获取 AD 转换器值 int get_ad_result()
{
int ADC_result;
```

```
ADC_RES = ADC_LOW2 = 0; //清除之前的结果
//设置 ADC_CONTR 控制寄存器的值
ADC_CONTR = ADC_POWER | ADC_SPEEDLL | ch | ADC_START;
_nop_(); _nop_(); _nop_(); _nop_(); _nop_(); _nop_();//在查询前必须有延时空操作
while (!(ADC_CONTR & ADC_FLAG)); //等待完成标志
ADC_result = (ADC_RES & 0x03) *256 + ADC_LOW2;//ADC_RES 中存高 2 位
ADC_CONTR &= ~ADC_FLAG;//关闭 ADC （
flag 位（中断标志位）置 0，必须软件清零
）
return ADC_result;//返回 ADC 结果
}
//校正
void calibrate()
{
cweight=get_ad_result();//获取 ADC 结果
}
//延时
void delay(uint x)
{
while(x--) ;
}
//清屏
void clearscreen()
{
int i,j;
for(i=0;i<8;++i)
{
send_byte(184+i,1,1);//选择页面地址，0-7 页，10111000 + i
send_byte(64,1,1);//选择列地址，自动加一，01000000
for(j=0;j<64;++j)
{
send_byte(0x00,0,1);
send_byte(0x00,1,0);
}
}
}
```

六、思考题

1. 调零的原理，软件调零和调零调零的区别。

软件调零是采用软件进行补偿的方法，又称数字调零；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。

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。