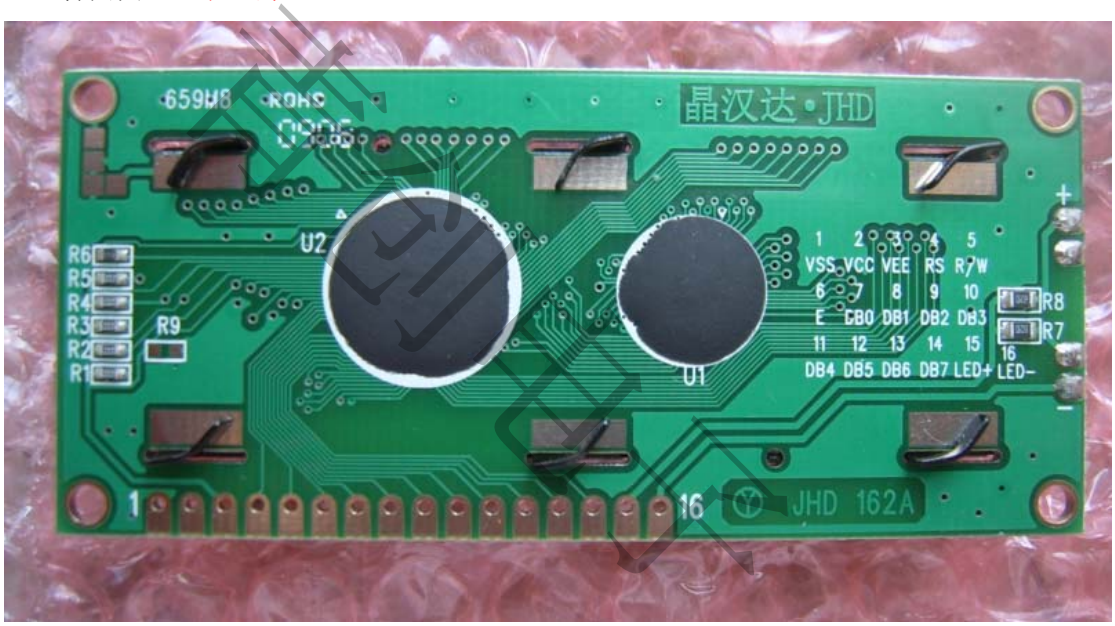
液晶的操作虽然比较繁琐，但不复杂，只要理解了相关的内容，掌握了关键知识点，操作起来还是比较简单的，而且比较有趣，比较有成就感。

一块液晶，最关键的就是其驱动芯片，我们选用的字符液晶的驱动芯片是 HD44780 型号的，市面上字符液晶绝大多数都是基于它制造的，所以本节所编写的控制程序可以很方便地应用于市面上大部分的字符型液晶。学会了这个驱动，就又为自己增添了开发利器了！

液晶实物正面图:

背面图：（双芯片）



管脚定义如下：（我们把这个表先列在这儿，下面遇到这些信号时可以在这个表里查下代表什么意思，不用去记住它们的）

管脚定义表格

引脚号	引脚名	电平	输入/输出	作用
1	Vss			电源地
2	Vcc			电源(+5V)
3	Vee			对比调整电压
4	RS	0/1	输入	0=输入指令 1=输入数据
5	R/W	0/1	输入	0=向LCD写入指令或数据 1=从LCD读取信息
6	E	1,1→0	输入	使能信号，1时读取信息， 1→0(下降沿)执行指令
7	DB0	0/1	输入/输出	数据总线line0(最低位)
8	DB1	0/1	输入/输出	数据总线line1
9	DB2	0/1	输入/输出	数据总线line2
10	DB3	0/1	输入/输出	数据总线line3
11	DB4	0/1	输入/输出	数据总线line4
12	DB5	0/1	输入/输出	数据总线line5
13	DB6	0/1	输入/输出	数据总线line6
14	DB7	0/1	输入/输出	数据总线line7(最高位)
15	A	+Vcc		LCD背光电源正极
16	K	接地		LCD背光电源负极

最后的效果显示图：

你是不是也想在液晶上写点自己的东西，那就赶快开始吧！

我们下面的讲解主要关注编程时需要掌握的环节，其它次要环节将略过（详细可查看手册），防止内容过多，显得又乱又杂，条理不清。

HD44780 内置了 DDRAM、CGROM 和 CGRAM。DDRAM—Display data RAM,即显示数据 RAM，用来寄存待显示的字符代码；CGROM—Character generator ROM,即字符产生 ROM，保存了 192 个常用字符的字模；CGRAM—Character generator RAM,即字符产生 RAM，里面可保存 8 个用户自定义的字符。要在液晶上的某个位置上显示某个字符，就是要向 DDRAM 的某个地址写入要显示的数据代码。这里只要掌握两个问题即可：1.如何确定位置？2.如何写入数据代码？继续往下看，你马上可以让液晶显示你所需要的字符了。

确定位置：

首先，所谓的 1602 指的是整块液晶屏幕物理上分为 16 列，最大可显示 2 行。其屏幕物理位置与 DDRAM 地址的对应关系如下：

屏幕物理位置与 DDRAM 地址的对应关系

列	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
行 1	00H	01H	02H	03H	04H	05H	06H	07H	08H	09H	0AH	0BH	0CH	0DH	0EH	0FH
2	40H	41H	42H	43H	44H	45H	46H	47H	48H	49H	4AH	4BH	4CH	4DH	4EH	4FH

注意：两行对应的 DDRAM 地址是不连续的。

比如，我们想向液晶上的第 1 行第 8 列位置写入一个 A 字，那么我们就需要向 DDRAM 的 07H 地址写入 A 的显示代码。那么如何写入 A 的显示代码呢？继续往下看！

输入显示代码：

液晶要显示某个字符时，实质上就是显示该字符的字模。当然我们我们程序没有必要产生这个模板，因为 HD44780 芯片已经做好了 192 个常用字符的字模，我们只要把它们调出来即可。

至于如何把这些字模调出来，也很简单，就是向 DDRAM 里写数就行。要想知道“调用哪个字模需要写入什么数据”，请看下表：

写入DDRAM数据与调出的字模对照表

所以，我们若想调出 A 的字模，就向 DDRAM 写入“01000001”，即 41H。若想显示其它的字母就在这里面找吧！（若你没能找到你要的字模，那就需要自己构造自模了，会用到上表中写有 CGRAM 的内容，这个我们在下一节 CGRAM 操作里讲）

到这里。我们就已经从理论上掌握了如何显示字符了。下面的实际操作就相对简单了，只是写写指令罢了。

先趁热看下如何实现上述两个功能的指令，即如何确定 DDRAM 地址和如何向 DDRAM 写数据。

1..设定 DDRAM 地址指令

指令功能	指令编码										执行时间 /us
	RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0	
设定DDRAM 地址	0	0	1	DDRAM 的地址(7位)							40

功能：设定下一个要存入数据的 DDRAM 的地址。

注意：写地址时，只要写入低 7 位地址就可以了，最高位已经固定为 1 了。

2.数据写入 DDRAM 或 CGRAM 指令:

指令功能	指令编码										执行时间 /us
	RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0	
数据写入到 DDRAM或 CGRAM	1	0	要写入的数据 D7~D0								40

功能：<1> 将字符码写入 DDRAM，以使液晶显示屏显示出相对应的字符;
<2> 将使用者自己设计的图形存入 CGRAM。

注意：如何区别执行该指令时是执行那个功能呢？手册中规定，执行这条指令之前，都需要先设定地址：所以，若先前执行的是“设置 CGRAM 地址”指令，那么接下来执行这个语句时，就是执行功能 2；若先前执行的是“设定 DDRAM 地址”的指令，那么接下来执行这个指令时会执行功能 1。呵呵。两个功能不会混淆的。

有了这两条指令，显示字符的功能基本就可以实现了。但液晶也是一个小系统，也有一些系统功能，如清屏，开关显示，光标设置，功能设置等，我们把它们的指令一一列举如下：

3.清屏指令

指令功能	指令编码										执行时间 /ms
	RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0	
清屏	0	0	0	0	0	0	0	0	0	1	1.64

功能：<1> 清除液晶显示器，即将 DDRAM 的内容全部填入"空白"的 ASCII 码 20H;
<2> 光标归位，即将光标撤回液晶显示屏的左上方;
<3> 将地址计数器(AC)的值设为 0。

4.光标归位指令

指令功能	指令编码										执行时间 /ms
	RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0	
光标归位	0	0	0	0	0	0	0	0	1	X	1.64

功能：<1> 把光标撤回到显示器的左上方;
<2> 把地址计数器(AC)的值设置为 0;
<3> 保持 DDRAM 的内容不变。

5 进入模式设置指令

指令功能	指令编码										执行时间 /us
	RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0	
进入模式设置	0	0	0	0	0	0	0	1	I/D	S	40

功能：设定每次定入 1 位数据后光标的移位方向，并且设定每次写入的一个字符是否移动。参数设定的
情况如下所示：
位名 设置
I/D 0=写入新数据后光标左移，AC 自减 1 1=写入新数据后光标右移，AC 自增 1
S 0=写入新数据后显示屏不移动 1=写入新数据后显示屏整体右移 1 个字符

6.显示开关控制指令

指令功能	指令编码										执行时间 /us
	RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0	
显示开关控制	0	0	0	0	0	0	1	D	C	B	40

功能：控制显示器开/关、光标显示/关闭以及光标是否闪烁。参数设定的情况如下：
位名 设置
D 0=显示功能关 1=显示功能开
C 0=无光标 1=有光标
B 1=光标闪烁 0=光标不闪烁

7 设定显示屏或光标移动方向指令

指令功能	指令编码										执行时间/us
	RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0	
设定显示屏或光标移动方向	0	0	0	0	0	1	S/C	R/L	X	X	40

功能：使光标移位或使整个显示屏幕移位。参数设定的情况如下：

S/C	R/L	设定情况
0	0	光标左移 1 格，且 AC 值减 1
0	1	光标右移 1 格，且 AC 值加 1
1	0	显示器上字符全部左移一格，但光标不动
1	1	显示器上字符全部右移一格，但光标不动

8.功能设置指令

指令功能	指令编码										执行时间/us
	RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0	
功能设定	0	0	0	0	1	DL	N	F	X	X	40

功能：设定数据总线位数、显示的行数及字型。参数设定的情况如下：

位名	设置	
DL	0=数据总线为 4 位	1=数据总线为 8 位
N	0=显示 1 行	1=显示 2 行
F	0=5×7 点阵/每字符	1=5×10 点阵/每字符

9.设定 CGRAM 地址指令

指令功能	指令编码										执行时间/us
	RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0	
设定CGRAM地址	0	0	0	1	CGRAM的地址(6位)						40

功能：设定下一个要存入数据的 CGRAM 的地址。

10.读取忙信号或 AC 地址指令

指令功能	指令编码										执行时间/us
	RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0	
读取忙碌信号或AC地址	0	1	FB	AC内容(7位)							40

功能：<1> 读取忙碌信号 BF 的内容，BF=1 表示液晶显示器忙，暂时无法接收单片机送来的数据或指令;当 BF=0 时，液晶显示器可以接收单片机送来的数据或指令;

<2> 读取地址计数器(AC)的内容。

注意：在编程时，为了简化流程，只要等待足够长的时间，就可以略掉该指令。

11.从 CGRAM 或 DDRAM 读出数据指令

指令功能	指令编码										执行时间/us
	RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0	
从CGRAM或DDRAM读出数据	1	1	要读出的数据 D7~D0								40

功能：读取 DDRAM 或 CGRAM 中的内容。

我们只要把这些指令按照一定的顺序输入到液晶驱动芯片里，液晶就可以显示了。我们的指令输入顺序是（括号内是指令编号）：

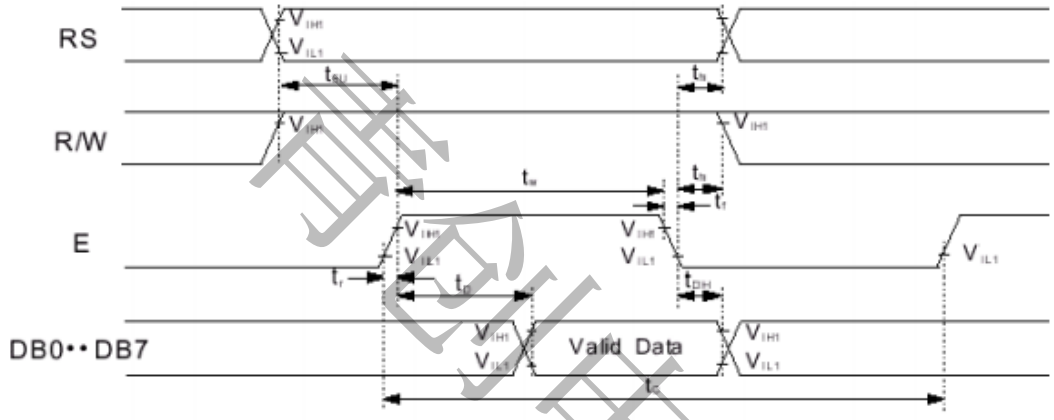
清屏指令(3)->功能设置指令(8)->进入模式设置指令(5)->显示开关控制指令(6)->设定 DDRAM 地址指令(1)->数据写入 DDRAM 指令(2)

为了简单期间，我们可以写完指令或数据后等待足够长的时间后再开始下一次的指令或数据写入，这样我们可以省去了读取忙信号指令，以简化我们的操作。

好了，恭喜你，你已经掌握了液晶的基本操作，现在跟着我们解析一下程序，用程序语言把上述的指令过程描述一下。

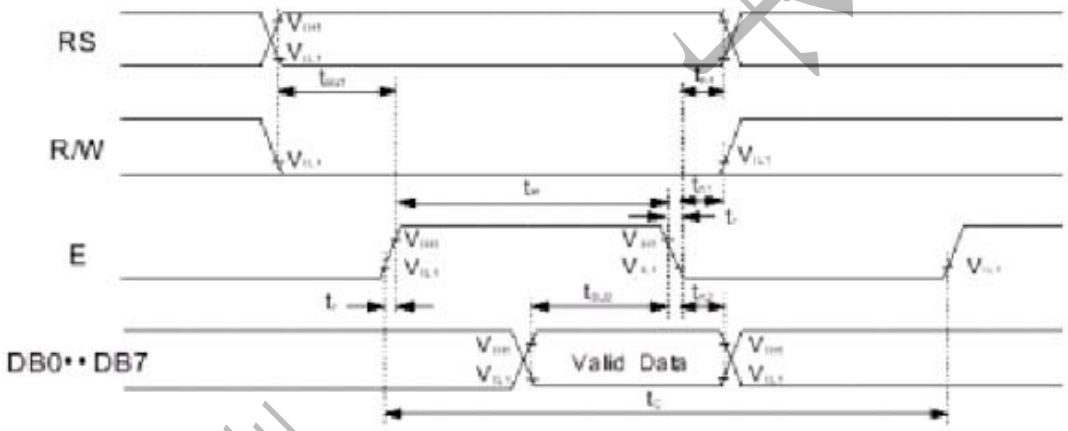
当然在具体硬件操作时，我们需要有个液晶读写时序，如下图：

从液晶读数据时序：



Read Mode Timing Diagram

把数据写入液晶时序：



Write Mode Timing Diagram

对于以上两个时序，只要关注一点就可以：**在 E 的下降沿时，数据要有效。我们编程的时候只要注意这个就可以。**

现在贴出程序源代码，该段程序功能是：液晶上显示两行，第一行为"Welcome to QC!"，第二行为"LCD DISPLAY"。

```
--lcd1602控制模块,在液晶上显示两行，第一行为"Welcome to QC!",第二行为"LCD DISPLAY"
--从CGROM中取出数据显示
--lcd_e: 1-使能有效，0-使能无效
--lcd_rw:1-读操作，0-写操作  lcd_rs:1-输入数据，0-输入指令
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;

entity CGROM is
    port
    (
        clkin,resetin:in std_logic;--时钟，复位信号输入
        lcd_data:inout std_logic_vector(7 downto 0);--lcd数据总线
        lcd_e,lcd_rw,lcd_rs:out std_logic--lcd控制信号
        --
        --clk_1M_tst:out std_logic;
        --delay_cnt_tst:out integer range 0 to 127
    );
end CGROM;
```

```
architecture behave of CGROM is
    component gen_div is--分频元件调用声明
    generic(div_param:integer:=2);--默认是4分频
    port
    (
        clk:in std_logic;
        bclk:out std_logic;
        resetb:in std_logic
    );
    end component;
----
type state is (set_Func,set_DispSwitch,set_EntryMd,clr_Disp,set_DDAd1,set_DDAd2,Display1,Display2,over);
type data_array is array(0 to 13) of std_logic_vector(7 downto 0);
constant data1:data_array:=(X"57",X"65",X"6C",X"63",
                            X"6F",X"6D",X"65",X"20",
                            X"74",X"6F",X"20",X"51",
                            X"43",X"21");--"Welcome to QC!"
constant data2:data_array:=(X"4C",X"43",X"44",X"20",
                            X"44",X"49",X"53",X"50",
                            X"4C",X"41",X"59",X"20",
                            X"20",X"20");--"LCD DISPLAY"
signal lcd_state:state:=clr_Disp;--初始为"清屏"状态
signal clk_yj:std_logic;--100k时钟, T=10us

begin
--clk_1M_tst<=clk_yj;
---
gen_clk_yj: --分频产生1M脉冲
        gen_div generic map(200)--400分频的,产生100k脉冲
        port map--分频元件例化
        (
            clk=>clkin,
            resetb=>not resetin,
            bclk=>clk_yj
        );
----
Display:
    process(resetin,clk_yj)
    variable delay_cnt:integer range 0 to 127:=0;--延时计数器
    variable char_cnt:integer range 0 to 15:=0;--字符计数器
    begin
    if resetin='0' then--复位
        lcd_state<=clr_Disp;
        delay_cnt:=0;
        char_cnt:=0;
        lcd_e<='0';
    else
        if rising_edge(clk_yj) then
            case lcd_state is
                when clr_Disp=>--刚开始时就需要清屏幕
                    delay_cnt:=delay_cnt+1;
                    if delay_cnt<=2 then
                        lcd_rs<='0';
                        lcd_rw<='0';
                        lcd_e<='1';
                    elsif delay_cnt<=4 then
                        lcd_data<=X"01";
                    elsif delay_cnt<=6 then
                        lcd_e<='0';--下降沿产生
                    elsif delay_cnt>=100 then--等待>40us时间,使指令写入完毕
                        delay_cnt:=0;
                        lcd_state<=set_Func;
                    end if;
```

```
85              when set_Func=>--功能设置
86                  delay_cnt:=delay_cnt+1;
87                  if delay_cnt<=2 then
88                      lcd_rs<='0';--指令
89                      lcd_rw<='0';--写
90                      lcd_e<='1';
91                  elsif delay_cnt<=4 then
92                      lcd_data<=X"38";--DL=1,8位, N=1,2行,F=0,5*7字体
93                  elsif delay_cnt<=6 then
94                      lcd_e<='0';--下降沿产生
95                  elsif delay_cnt>=100 then--等待>40us时间,使指令写入完毕
96                      delay_cnt:=0;
97                      lcd_state<=set_EntryMd;
98                  end if;
99              when set_EntryMd=>--设置输入模式
100                 delay_cnt:=delay_cnt+1;
101                 if delay_cnt<=2 then
102                     lcd_rs<='0';
103                     lcd_rw<='0';
104                     lcd_e<='1';
105                 elsif delay_cnt<=4 then
106                     lcd_data<=X"06";--I/D=1,自增, S=0,显示不移动
107                 elsif delay_cnt<=6 then
108                     lcd_e<='0';--下降沿产生
109                 elsif delay_cnt>=100 then--等待>40us时间,使指令写入完毕
110                     delay_cnt:=0;
111                     lcd_state<=set_DispSwitch;
112                 end if;

113             when set_DispSwitch=>--设置显示开关
114                 delay_cnt:=delay_cnt+1;
115                 if delay_cnt<=2 then
116                     lcd_rs<='0';
117                     lcd_rw<='0';
118                     lcd_e<='1';
119                 elsif delay_cnt<=4 then
120                     lcd_data<=X"0C";--D=1,显示开, C=0光标关闭,B=0,光标所在字符不闪烁
121                 elsif delay_cnt<=6 then
122                     lcd_e<='0';--下降沿产生
123                 elsif delay_cnt>=100 then--等待>40us时间,使指令写入完毕
124                     delay_cnt:=0;
125                     lcd_state<=set_DDAd1;
126                 end if;
127             when set_DDAd1=>--设置成DDRAM第一行的地址首地址为0x81
128                 delay_cnt:=delay_cnt+1;
129                 if delay_cnt<=2 then
130                     lcd_rs<='0';
131                     lcd_rw<='0';
132                     lcd_e<='1';
133                 elsif delay_cnt<=4 then
134                     lcd_data<=X"81";
135                 elsif delay_cnt<=6 then
136                     lcd_e<='0';--下降沿产生
137                 elsif delay_cnt>=100 then--等待>40us时间,使指令写入完毕
138                     delay_cnt:=0;
139                     lcd_state<=Display1;
140                     char_cnt:=0;--清字符计数器
141                 end if;
```

```
142                 when Display1=>--向第一行DDRAM写数据
143                     if char_cnt<14 then --显示"Welcome to QC!"
144                         delay_cnt:=delay_cnt+1;
145                         if delay_cnt<=2 then
146                             lcd_rs<='1';--数据
147                             lcd_rw<='0';--写
148                             lcd_e<='1';
149                         elsif delay_cnt<=4 then
150                             lcd_data<=data1(char_cnt);
151                         elsif delay_cnt<=6 then
152                             lcd_e<='0';--下降沿产生
153                         elsif delay_cnt>=100 then--等待>40us时间,使数据写入完毕
154                             delay_cnt:=0;
155                             lcd_state<=Display1;
156                             char_cnt:=char_cnt+1;--字符计数器自增
157                         end if;
158                     else
159                         delay_cnt:=0;
160                         lcd_state<=set_DDAd2;
161                         char_cnt:=0;
162                     end if;

163                 when set_DDAd2=>--设置DDRAM第二行的地址首地址为0xc2
164                         delay_cnt:=delay_cnt+1;
165                         if delay_cnt<=2 then
166                             lcd_rs<='0';
167                             lcd_rw<='0';
168                             lcd_e<='1';
169                         elsif delay_cnt<=4 then
170                             lcd_data<=X"C2";
171                         elsif delay_cnt<=6 then
172                             lcd_e<='0';--下降沿产生
173                         elsif delay_cnt>=100 then--等待>40us时间,使指令写入完毕
174                             delay_cnt:=0;
175                             lcd_state<=Display2;
176                             char_cnt:=0;
177                         end if;

178                 when Display2=>--向第二行DDRAM写数据
179                     if char_cnt<11 then --显示"LCD DISPLAY"
180                         delay_cnt:=delay_cnt+1;
181                         if delay_cnt<=2 then
182                             lcd_rs<='1';--数据
183                             lcd_rw<='0';--写
184                             lcd_e<='1';
185                         elsif delay_cnt<=4 then
186                             lcd_data<=data2(char_cnt);
187                         elsif delay_cnt<=6 then
188                             lcd_e<='0';--下降沿产生
189                         elsif delay_cnt>=100 then--等待>40us时间,使数据写入完毕
190                             delay_cnt:=0;
191                             lcd_state<=Display2;
192                             char_cnt:=char_cnt+1;
193                         end if;
194                     else
195                         delay_cnt:=0;
196                         lcd_state<=over;
197                         char_cnt:=0;
198                     end if;
199                 when over=>
200                     null;
201             end case;
202         end if;
203     end if;
204     --delay_cnt_tst<=delay_cnt;
205     end process;
206 end behave;
```

逐行解释：

13：输入时钟信号 40M，复位信号低电平有效，即低电平复位

14：液晶 8 位数据总线

15：液晶的控制信号。具体功能见上面的管脚定义表格。
23-31：分频元件声明。我们在程序中并没有直接用到输入的 40M 时钟，而是用了其分频后的时钟 100kHz。
33：自定义一种枚举数据类型 state，里面的每个类型表示液晶操作的一个状态：set_Func—设置功能，set_DispSwitch—设置显示开关,set_EntryMd—设置输入模式,clr_Disp—清屏,set_DDAd1—设置第一行的 DDRAM 地址,set_DDAd2—设置第二行的 DDRAM 地址,Display1—向第一行 DDRAM 写入数据,Display2—向第二行 DDRAM 写入数据,over—操作结束。整个程序通过这些状态构成一个状态机。
34：自定义一种数组类型 data_array，每个 data_array 类型有 14 个成员，每个成员是 std_logic_vector(7 downto 0)类型。我们要把写入 DDRAM 的数据保存在这个数组里。
36-42：保存了写入 DDRAM 的数据。你可以通过“写入 DDRAM 数据与调出的字模对照表”找出你要显示的字模所对应的 DDRAM 数据。
43:定义了一个 state 类型的信号，其初始状态是 clr_Disp，即清屏
44：这个 clk_yj 是 100kHz，从输入 40M 中分频得来，作为状态机的基准时钟。
49-56：分频元件实例化。从 40M 输入时钟 clkin 分出 100kHZ 的时钟 clk_yj
60：延时变量。因为每条指令执行都需要一定的时间，在下一条指令执行前要有足够的延时以让上一条指令执行完毕。
61：显示字符指针变量。我们用这个变量依次从先前定义的 data1 和 data2 里取出数据写到 DDRAM.
63-67：复位操作。状态机置到清屏状态 clr_Disp，清空计数器，并将 E 信号置为无效。
70-201：一个 case 语句，构成液晶操作的状态机。

71-84：**清屏操作**。

73-76：lcd_rs<='0'—输入指令；lcd_rw<='0'—写操作；lcd_e<='1’— E 拉高

77-78：把清屏指令对应的指令数据送到数据总线上。

79-80 ：lcd_e<='0’—E 拉低，形成下降沿，这时 HD44780 就会把刚才放在总线上的清屏指令数据读进去

81-83：延时，至少 40us 时间，等待 HD44780 处理完清屏指令。然后将状态机状态指向下一个状态 set_Func—功能设置指令

85-98：**功能设置指令**。主要设置：8 位数据线，2 行字符，5*7 点阵/字符

87-90 ：lcd_rs<='0'—输入指令；lcd_rw<='0'—写操作；lcd_e<='1’— E 拉高

91-92：根据上述的设置，对照功能设置指令，得到 X"38"这个指令数据

93-94：lcd_e<='0’—E 拉低，形成下降沿，这时 HD44780 就会把刚才放在总线上的功能设置指令数据读进去

95-97：延时等待，然后转到下一个状态 set_EntryMd—设置输入模式

99-112：**设置输入模式**。主要设置：写入新数据后光标右移，写入新数据后显示屏不移动

整个动作和上面清屏或功能设置指令类似，在此不再赘述。

113-126：**设置显示开关**。主要设置：D=1--0 显示开，C=0--光标关闭,B=0--光标所在字符不闪烁

整个操作和上面的操作类似，在此不再赘述。

127-141：**设置第一行起始字符所对应的 DDRAM 地址**。

129-132：lcd_rs<='0'—输入指令；lcd_rw<='0'—写操作；lcd_e<='1’— E 拉高

133-134：给出第一行起始字符的 DDRAM 地址。我们对照屏幕物理位置和 DDRAM 地址的对应关系和设置 DDRAM 地址的指令，设置第一个字符的起始位置在第一

行第二列。

135-136：lcd_e<='0’—E 拉低，形成下降沿，这时 HD44780 就会把刚才放在总线上的数据读进去

147-140：延时等待。并将下一个状态转入 Display1—写入第一行显示字符对应的 DDRAM 数据

142-157：**设置第一行 DDRAM 的数据**。我们将从数组 data1 里依次取出数据写入到 DDRAM 里去。在这个过程中每向 DDRAM 写一个数，DDRAM 地址为自增 1，所以 DDRAM 的地址就不需要再设置了。地址会在第一次设置的地址基础上不断自增的。

143-158：循环在状态 Display1 里，直到第一行的数据全部进入进去。

145-148：lcd_rs<='1'—输入数据；lcd_rw<='0'—写操作；lcd_e<='1’— E 拉高

149-150：从数组 data1 取出数据放到总线上

151-152：lcd_e<='0’—E 拉低，形成下降沿，这时 HD44780 就会把刚才放在总线上的数据读进去

153-157：延时等待。并清计数器，准备下一个数据的写操作。状态机还是处在同一个状态。字符计数器 char_cnt 自增 1，指向下一个需要写入的数据。然后又回到 142 开始循环写入下一个数据。

159-162：第一行数据写入完毕，状态机转到 set_DDAd2—设置第二行起始字符对应的 DDRAM 地址

163-177：**设置第二行起始字符对应的 DDRAM 的地址**。操作流程类似于 127-141。

178-198：**设置第二行的 DDRAM 的数据**。操作流程类似于 142-157。

199-200：**显示操作结束**。状态机在结束状态里不执行任何动作。

状态机跳转图，由于贴在这里不清楚，可以看 Quartus2 生产的图形，在这里就不贴了。最后的现实效果图就是

：

这一讲的液晶操作就讲到这里了。若你想了解如何显示自定义的图形，那就请看 CGRAM 夹下的操作说明。