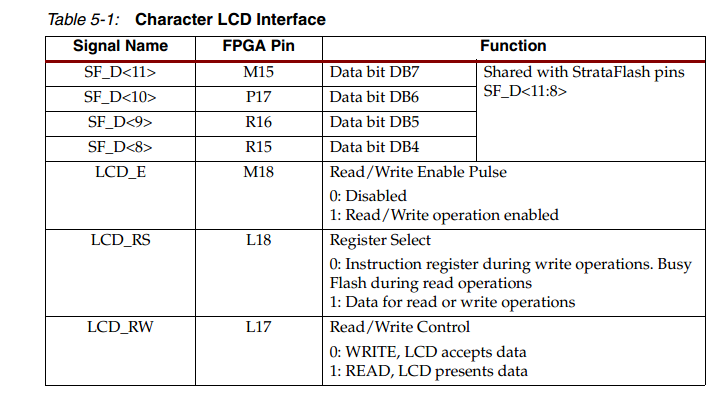
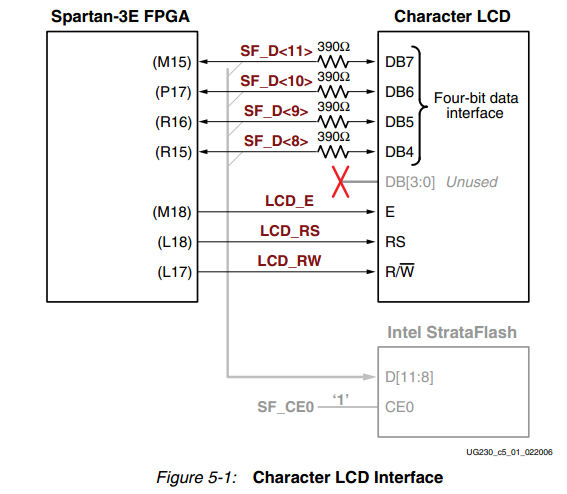
目前市场上的LCD1602液晶，其控制芯片主要有Samsung S6A0069X  or KS0066U，Hitachi HD44780，SMOS SED1278。Digilent的Spartan-3E开发板上配置的LCD1602的控制芯片是Sitronix的ST7066U，而且开发板配套的开发说明书上写“为了减少使用I/O口，采用4位传输模式”，真心蛮坑的，选择4位或者8位传输模式应该让用户选择嘛，呵呵……废话不说了，开始正文了。

        开发板的字符型LCD的接口情况如Figure5-1所示，其中，SF\_D<11-8>为数据传输信号，LCD\_E是LCD的读写使能信号。LCD\_RS是LCD的片选寄存器控制信号，在进行写指令操作的时候，应该将其置0，而在进行读写数据的时候，应该置1。LCD\_RW为读写控制位信号，为0时表示写数据（LCD接收来自FPGA的数据），为1时表示读数据（LCD显示寄存器中的数据）。关于LCD接口的详细说明如Table5-1所示。



         液晶控制芯片内部有三组存储器，分别为显示数据存储器(DD RAM)、字符发生器存储器(CG ROM)和字符产生器存储器(CG RAM)。关于这三组存储器的功能以及相应的概念，网上有很多，这里不再赘述。百度空间里有这么一篇文章可以帮助对这三组存储器的理解：<http://hi.baidu.com/%C4%E1%BF%A8njord/blog/item/f070efcbc55f6fe753664f2e.html>。

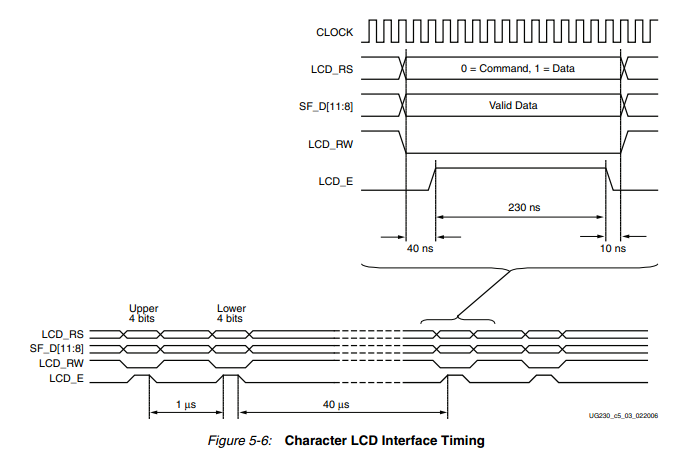


        Figure5-6的上半部分说明了LCD采用4位数据总线时输入命令和数据的时序情况，所有的命令和数据均以8位形式传送给字符显示屏控制芯片，采用4位数据总线传输时，每一次8位传送操作必须分两次4位传输操作才能完成，先传送高4位，再传送低4位，其间隔时间至少1μs。待完成一个8位的传输操作后，与下一次传输操作的时间间隔至少要超过40 μs。而在传输清屏指令之后，与下一次传输操作的时间间隔至少要超过1.64 ms。上电后，必须向液晶显示屏控制芯片传送初始化命令。由于接通电源后，液晶显示屏控制芯片默认为第一次写数据操作是8位数据传输，而实验开发板上的液晶显示屏控制芯片的DB3～DB0没有连接，仅连接了4位数据总线DB7～DB4，这就需要重复写入功能控制字0X3后再写入采用4位数据总线传输方式，传输功能设置控制字中的DL=0，以及写入功能控制字0X2。操作步骤如下：  
(1) 电源接通后，等待15 ms以上，当时钟频率为50 MHz时，15 ms就等于750 000个时钟周期。  
(2) 传送控制字SF\_D<11:8>=0x3，数据稳定后，LCD\_E变成高电平，并且保持高电平12个时钟周期。  
(3) 等待4.1 ms或更长时间，当时钟为50 MHz时，即为205 000个时钟周期。  
(4) 传送控制字SF\_D<11:8>=0x3，LCD\_E变成高电平，并且保持高电平12个时钟周期。  
(5) 等待100 μs或更长时间，当时钟为50 MHz时，即为5000个时钟周期。  
(6) 传送控制字SF\_D<11:8>=0x2，LCD\_E保持高电平12个时钟周期。  
(7) 等待40 μs或更长时间，当时钟为50 MHz时，即为2000个时钟周期。  
(8) 传送功能设置命令控制字0X28，DL=0时，采用4位数据总线DB7～DB4；N=1时，双行显示；F =0时，5×7点阵。  
(9) 传送模式设置命令，0X06，设置显示屏自动增加地址指针。  
(10) 传送显示打开命令，0X0c，打开显示屏，关闭光标显示。  
(11) 等待至少1.64 ms(82 000个时钟周期)。

        初始化完成后，再传输指定地址计数器地址和显示的数据。当地址计数器配置为自动增1和显示多个字符时，依次传输多个字符编码，每个字符自动存储并显示在下个位置。

       以上就是在LCD上显示字符的介绍了，我第一次按照上面的步骤写下来一共整了54个状态，满屏的代码看着真心不舒服。后来采用分频的方法，分频到100us，这样就可以大大减少状态机的状态个数啦。在这个基础上，如果要想实现LCD上的字符滚动显示，我最初的想法就是，通过改变数据的DDRAM地址来实现，即完整的一个字符串写完后，清屏一下，然后将原来的DDRAM地址+1，再写一次，这样就相当于将整个字符串向右移动一位了，如果整个字符串移动到LCD最右端，让DDRAM地址回到最初的位置，如此反复。为了实现动态的效果，在写完一次字符串后，延时一段时间（我这里用的是1s），否则移动太快，液晶上什么也看不到。

      下面给出源程序：

**[plain]** [view plaincopyprint?](http://blog.csdn.net/sixthsence/article/details/7825498)

1. <SPAN style="FONT-SIZE: 12px">library IEEE;
2. use IEEE.STD\_LOGIC\_1164.ALL;
3. use IEEE.STD\_LOGIC\_ARITH.ALL;
4. use IEEE.STD\_LOGIC\_UNSIGNED.ALL;
6. ---- Uncomment the following library declaration if instantiating
7. ---- any Xilinx primitives in this code.
8. --library UNISIM;
9. --use UNISIM.VComponents.all;
11. entity LCD1602\_Update is
12. Port ( Clk      :   in      STD\_LOGIC;
13. Reset     :   in     STD\_LOGIC;
14. LCD\_DATA     :   out     STD\_LOGIC\_VECTOR (7 downto 4);
15. LCD\_EN   :   out     STD\_LOGIC;
16. LCD\_RS   :   out     STD\_LOGIC;
17. LCD\_RW   :   out     STD\_LOGIC);
18. end LCD1602\_Update;
20. architecture Behavioral of LCD1602\_Update is
22. Signal Clk\_scan : STD\_LOGIC := '0';
23. Signal Clk\_100us    : STD\_LOGIC ;
24. --Signal Reset      : STD\_LOGIC := '0';  --Method 2
26. Type string1 is array(0 to 5) of STD\_LOGIC\_VECTOR(7 downto 0);
27. constant data1 : string1 := (x"48",x"45",x"4c",x"4c",x"4f",x"21");      --"HELLO!"
29. Type        State\_Display is (
30. st\_disp1,st\_disp2,st\_disp3,st\_disp4,st\_disp5,
31. st\_disp6,st\_disp7,st\_disp8,st\_disp9,st\_disp10,
32. st\_disp11,st\_disp12,st\_disp13,st\_disp14,st\_disp15,
33. st\_disp16,st\_disp17,st\_disp18,st\_disp19,st\_disp20,
34. st\_disp21
35. );
36. Signal  Current\_Disp : State\_Display := st\_disp1;
38. begin
39. Proc\_CLK\_100us:process(Clk)     --Frequency division to 10KHz,which is 100us
40. variable cnt\_clk : integer range 0 to 5000 := 0;
41. begin
42. if(rising\_edge(Clk))    then
43. if(cnt\_clk < 2500)   then
44. cnt\_clk := cnt\_clk + 1;
45. Clk\_scan    <= '0';
46. elsif(cnt\_clk < 4999) then
47. cnt\_clk := cnt\_clk + 1;
48. Clk\_scan    <= '1';
49. else
50. cnt\_clk := 0;
51. end if;
52. Clk\_100us <= Clk\_scan;
53. end if;
54. end process Proc\_CLK\_100us;
56. Proc\_Display:process(Clk\_100us,Reset)
57. variable cnt\_disp :     integer range 0 to 10000;
58. variable cnt\_rw     :   integer range 0 to 6  := 0;
59. variable cnt\_loop :     integer range 0 to 16 := 0;
60. begin
61. --      Method 1 : Use a button(always High) to set the signal port "Reset"
62. if(Reset = '0') then
63. LCD\_RS  <= '0';
64. LCD\_RW  <= '0';
65. LCD\_EN  <= '0';
66. LCD\_DATA <= "0000";
67. cnt\_disp := 0;
68. cnt\_rw  := 0;
69. cnt\_loop := 0;
70. Current\_Disp <= st\_disp1;
71. --          Method 2 : Use delay to set the signal "Reset" to 1
72. --          if(cnt\_disp < 50000) then
73. --              cnt\_disp := cnt\_disp + 1;
74. --          else
75. --              cnt\_disp := 0;
76. --          end if;
77. --          Reset <= '1';
78. elsif(rising\_edge(CLK\_100us)) then
79. case Current\_Disp is
80. when st\_disp1 =>
81. if(cnt\_disp < 150) then              --Wait for 15ms or longer
82. cnt\_disp := cnt\_disp + 1;
83. Current\_Disp <= st\_disp1;
84. else
85. cnt\_disp := 0;
86. Current\_Disp <= st\_disp2;
87. end if;
88. when st\_disp2 =>         --Write SF\_D<11:8>=0x3, pulse LCD\_E High for 240ns.
89. LCD\_DATA <= x"3";
90. LCD\_EN  <= '1';
91. if(cnt\_disp < 41) then               --Wait for 4.1ms or longer
92. cnt\_disp := cnt\_disp + 1;             --实验发现这样写没有问题！！！但最好还是再用一个状态
93. Current\_Disp <= st\_disp2;
94. else
95. cnt\_disp := 0;
96. LCD\_EN  <= '0';
97. Current\_Disp <= st\_disp3;
98. end if;
99. when st\_disp3 =>     --Write SF\_D<11:8>=0x3, pulse LCD\_E High for 240ns.
100. if(cnt\_disp < 1) then                --Wait 100 μs or longer
101. cnt\_disp := cnt\_disp + 1;
102. LCD\_DATA <= x"3";
103. LCD\_EN  <= '1';
104. Current\_Disp <= st\_disp3;
105. else
106. cnt\_disp := 0;
107. LCD\_EN  <= '0';
108. Current\_Disp <= st\_disp4;
109. end if;
110. when st\_disp4 =>     --Write SF\_D<11:8>=0x3, pulse LCD\_E High for 240ns.
111. if(cnt\_disp < 1) then                --Wait 40 μs or longer
112. cnt\_disp := cnt\_disp + 1;
113. LCD\_DATA <= x"3";
114. LCD\_EN  <= '1';
115. Current\_Disp <= st\_disp4;
116. else
117. cnt\_disp := 0;
118. LCD\_EN  <= '0';
119. Current\_Disp <= st\_disp5;
120. end if;
121. when st\_disp5 =>     --Write SF\_D<11:8>=0x2, pulse LCD\_E High for 240ns.
122. if(cnt\_disp < 1) then                --Wait 40 μs or longer
123. cnt\_disp := cnt\_disp + 1;
124. LCD\_DATA <= x"2";
125. LCD\_EN  <= '1';
126. Current\_Disp <= st\_disp5;
127. else
128. cnt\_disp := 0;
129. LCD\_EN  <= '0';
130. Current\_Disp <= st\_disp6;
131. end if;
132. when st\_disp6 =>             -- Function Set   0x28
133. if(cnt\_disp < 1) then                --Wait 40 μs or longer
134. cnt\_disp := cnt\_disp + 1;
135. LCD\_DATA <= x"2";
136. LCD\_EN  <= '1';
137. Current\_Disp <= st\_disp6;
138. else
139. cnt\_disp := 0;
140. LCD\_EN  <= '0';
141. Current\_Disp <= st\_disp7;
142. end if;
143. when st\_disp7 =>
144. if(cnt\_disp < 1) then                --Wait 40 μs or longer
145. cnt\_disp := cnt\_disp + 1;
146. LCD\_DATA <= x"8";
147. LCD\_EN  <= '1';
148. Current\_Disp <= st\_disp7;
149. else
150. cnt\_disp := 0;
151. LCD\_EN  <= '0';
152. Current\_Disp <= st\_disp8;
153. end if;
154. when st\_disp8 =>             --Entry Mode Set  0x06
155. if(cnt\_disp < 1) then                --Wait 40 μs or longer
156. cnt\_disp := cnt\_disp + 1;
157. LCD\_DATA <= x"0" ;
158. LCD\_EN  <= '1';
159. Current\_Disp <= st\_disp8;
160. else
161. cnt\_disp := 0;
162. LCD\_EN  <= '0';
163. Current\_Disp <= st\_disp9;
164. end if;
165. when st\_disp9 =>
166. if(cnt\_disp < 1) then                --Wait 40 μs or longer
167. cnt\_disp := cnt\_disp + 1;
168. LCD\_DATA <= x"6";
169. LCD\_EN  <= '1';
170. Current\_Disp <= st\_disp9;
171. else
172. cnt\_disp := 0;
173. LCD\_EN  <= '0';
174. Current\_Disp <= st\_disp10;
175. end if;
176. when st\_disp10 =>            --Display On/Off  0x0C
177. if(cnt\_disp < 1) then                --Wait 40 μs or longer
178. cnt\_disp := cnt\_disp + 1;
179. LCD\_DATA <= x"0";
180. LCD\_EN  <= '1';
181. Current\_Disp <= st\_disp10;
182. else
183. cnt\_disp := 0;
184. LCD\_EN  <= '0';
185. Current\_Disp <= st\_disp11;
186. end if;
187. when st\_disp11 =>
188. if(cnt\_disp < 1) then                --Wait 40 μs or longer
189. cnt\_disp := cnt\_disp + 1;
190. LCD\_DATA <= x"C";
191. LCD\_EN  <= '1';
192. Current\_Disp <= st\_disp11;
193. else
194. cnt\_disp := 0;
195. LCD\_EN  <= '0';
196. Current\_Disp <= st\_disp12;
197. end if;
198. when st\_disp12 =>            --Clear Display  0x01
199. if(cnt\_disp < 1) then                --Wait 40 μs or longer
200. cnt\_disp := cnt\_disp + 1;
201. LCD\_DATA <= x"0";
202. LCD\_EN  <= '1';
203. Current\_Disp <= st\_disp12;
204. else
205. cnt\_disp := 0;
206. LCD\_EN  <= '0';
207. Current\_Disp <= st\_disp13;
208. end if;
209. when st\_disp13 =>
210. if(cnt\_disp < 1) then                --Wait 40 μs or longer
211. cnt\_disp := cnt\_disp + 1;
212. LCD\_DATA <= x"1";
213. LCD\_EN  <= '1';
214. Current\_Disp <= st\_disp13;
215. else
216. cnt\_disp := 0;
217. LCD\_EN  <= '0';
218. Current\_Disp <= st\_disp14;
219. end if;
220. when st\_disp14 =>
221. if(cnt\_disp < 17) then               --Wait 1.64ms or longer
222. cnt\_disp := cnt\_disp + 1;
223. Current\_Disp <= st\_disp14;
224. else
225. cnt\_disp := 0;
226. Current\_Disp <= st\_disp15;
227. end if;
228. when st\_disp15 =>                --Set DDRAM Address 0x00+0x80
229. if(cnt\_disp < 1) then                --Wait 40 μs or longer
230. cnt\_disp := cnt\_disp + 1;
231. LCD\_DATA <= x"8";
232. LCD\_EN  <= '1';
233. Current\_Disp <= st\_disp15;
234. else
235. cnt\_disp := 0;
236. LCD\_EN  <= '0';
237. Current\_Disp <= st\_disp16;
238. end if;
239. when st\_disp16 =>                      --Shift the DDRAM Address
240. if(cnt\_disp < 1) then                --Wait 40 μs or longer
241. cnt\_disp := cnt\_disp + 1;
242. LCD\_DATA <= x"0" + conv\_std\_logic\_vector(cnt\_loop,4);
243. if(cnt\_loop < 15) then
244. cnt\_loop := cnt\_loop + 1;
245. else
246. cnt\_loop := 0;
247. end if;
248. LCD\_EN  <= '1';
249. Current\_Disp <= st\_disp16;
250. else
251. cnt\_disp := 0;
252. LCD\_EN  <= '0';
253. Current\_Disp <= st\_disp17;
254. end if;
255. when st\_disp17 =>            --Write Data to DDRAM(st\_disp17-st\_disp20)
256. if(cnt\_rw < 6) then
257. LCD\_DATA <= data1(cnt\_rw)(7 downto 4);   --Write the Upper Nibble
258. LCD\_EN  <= '1';
259. LCD\_RS   <= '1';
260. Current\_Disp <= st\_disp18;
261. else
262. cnt\_rw  := 0;
263. Current\_Disp <= st\_disp21;
264. end if;
265. when st\_disp18 =>
266. LCD\_EN  <= '0';
267. Current\_Disp <= st\_disp19;
268. when st\_disp19 =>
269. LCD\_DATA <= data1(cnt\_rw)(3 downto 0);       --Write the Lower Nibble
270. LCD\_EN  <= '1';
271. LCD\_RS   <= '1';
272. Current\_Disp <= st\_disp20;
273. when st\_disp20 =>
274. LCD\_EN  <= '0';
275. cnt\_rw  := cnt\_rw + 1;
276. Current\_Disp <= st\_disp17;
277. when st\_disp21 =>
278. if(cnt\_disp < 10000) then    --Delay 1 min to display the next string
279. cnt\_disp    := cnt\_disp + 1;
280. Current\_Disp <= st\_disp21;
281. else
282. cnt\_disp    := 0;
283. LCD\_RS   <= '0';
284. Current\_Disp <= st\_disp12;
285. end if;
286. when others => null;
287. end case;
288. end if;
289. end process Proc\_Display;
291. end Behavioral;