光存储与显示技术

实验指导书

付丽　罗钧　编

重庆大学光电工程学院

2012 年 9 月

目 录

实验一　LED 点阵显示实验

一、实验目的

1、掌握 LED 点阵显示的基本原理和实现方法。
2、掌握点阵扫描输出接口技术，完成显示汉字功能。

二、实验原理

1、8X8 点阵 WTD3088 模块工作原理
8X8 点阵 WTD3088 LED 结构如下图所示：

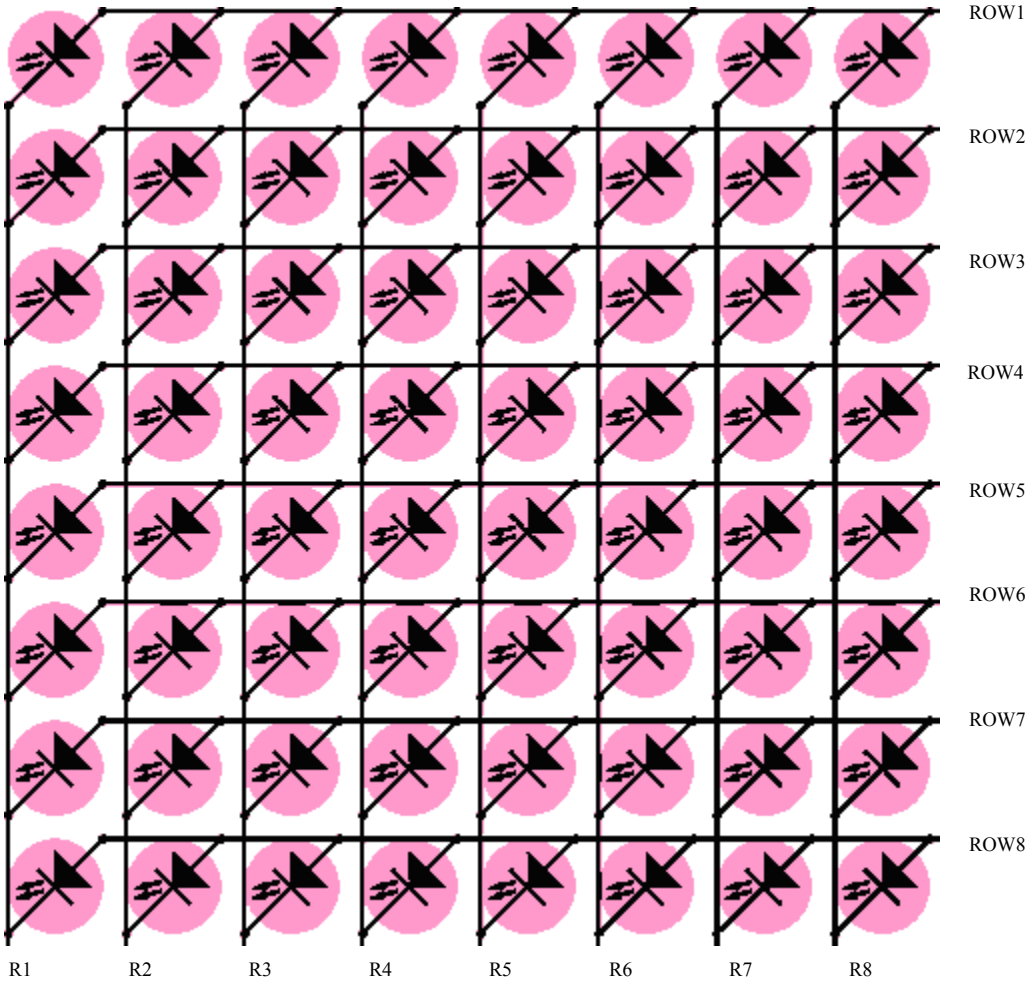
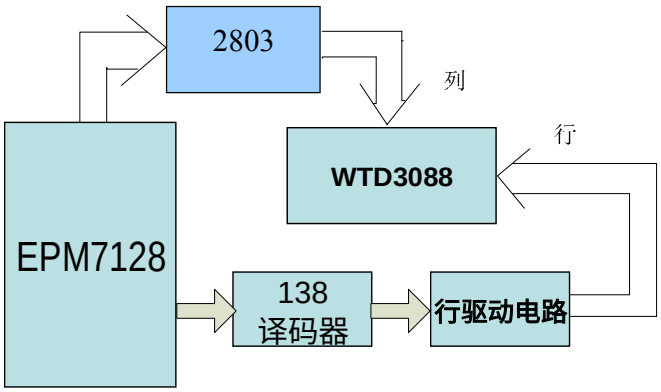


图 1.1　WTD3088 8X8 点阵 LED 原理图

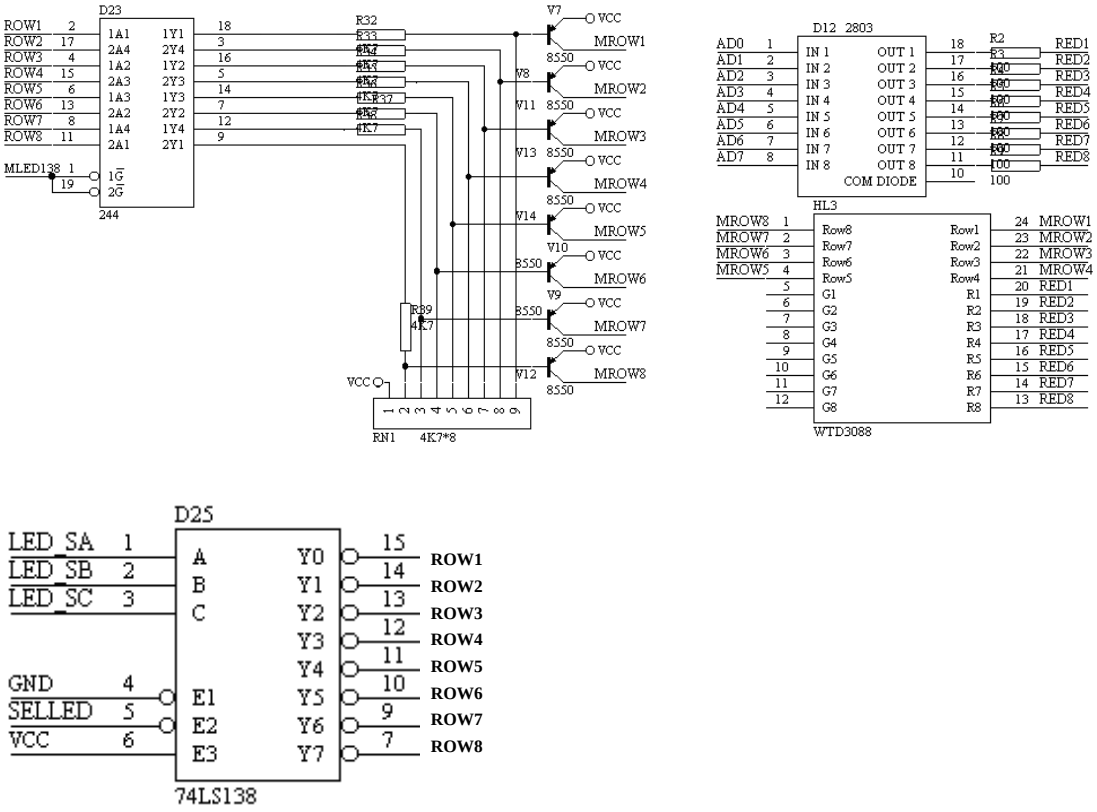
从图 1.1 中可以看出，WTD3088 点阵模块共有 64 个发光二极管组成，且每个发光二极管是放置在行线和列线的交叉点上，若当对应的某一列置低电平，某一行置高电平，则相应的二极管就亮。

实验电路原理框图和电路图如图 1.2 所示，WTD3088 列输入线（RED1~RED8）接至内部 LED 的阴极端；行输入线（MROW1～MROW8）接至内部 LED 的阳极端。本次实验用

实验箱中 138 译码器输出驱动 LED 点阵中的行信号。2803 芯片输出反向驱动 LED 点阵列信号。



(a)　LED 点阵系统框图



(b) LED 点阵电路图

图 1.2　LED 点阵原理图

备注：图中标号相同的引脚是相连接的。

2、显示代码原理

例子“ 0 ”的显示如下图所示：

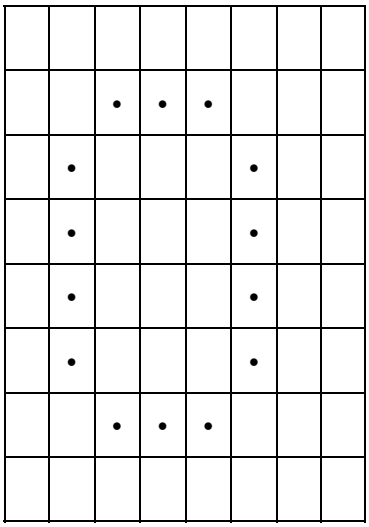


图 1.3　显示数字“ 0 ”LED 点阵示意图

每行形成的列代码为　00H，38H，44H，44H，44H，44H，38H，00H；通过 138 译码器输出依次扫描每一行，同时依次把相应的列数据送到 WTD3088 的列输入线上，如此反复。

三、实验内容

1、验证性实验：验证用 VHDL 语言实现 LED 点阵显示“ 0 ”。
2、设计性实验：修改程序，自己编码，用 LED 点阵显示汉字“ 日 ”字。

四、实验步骤

1、参考附 1.2 “ QUARTUS II4.0 的使用步骤参考 ”步骤 1～14，在 quartus II 下建立工程，编译等。
2、按照表 1.1 连线，并对模块 23 中的开关按照以下设置：

S2-1，OFF　　蜂鸣器关
S2-2，OFF　　交通灯不显示
S2-3，OFF　　骰子灯不显示
S2-4，OFF　　LED 点阵行选择不使用 CPLD 输出
S2-5，ON　　LED 点阵行选择使用三八译码器输出
S2-6，ON　　用 LED 数码管的 138 译码
S2-7，OFF　　L8-L1 不显示

3、参考附 1.2 “ QUARTUS II4.0 的使用步骤参考 ”步骤 15～16，下载目标文件。

注：P83 直接连接到脉冲源模块的 1KHZ

表 1.1　　适配卡连线表

输入信号名	对应芯片端子名	引入端子名	功能	输出信号名	对应芯片端子名]	引入端子名	功能
CLK1	P83	1KHZ	驱动时钟	LED_S0	P55	LSA	38 译码器输入
RESET	P81	RST	RESET	LED_S1	P58	LSB	
				LED_S2	P67	LSC	
				AD0	P11	AD0	数据
				AD1	P15	AD1	数据

				AD2	P20	AD2	数据
				AD3	P25	AD3	数据
				AD4	P30	AD4	数据
				AD5	P35	AD5	数据
				AD6	P40	AD6	数据
				AD7	P45	AD7	数据

附：LED 点阵显示数字“ 0 ”vhdl 程序

```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity LEDDOT is
      port (
            CLK1:          in STD_LOGIC;--点阵时钟信号
            RESET:         in STD_LOGIC;
            LED_S:   out STD_LOGIC_VECTOR (2 downto 0);--点阵行扫描译码信号
            AD:        out STD_LOGIC_VECTOR (7 downto 0));--点阵列数据信号
end LEDDOT;

architecture LEDDOT_ARCH of LEDDOT is
type romtable is array (0 to 7) of std_logic_vector(7 downto 0);
   constant roma:romtable:=romtable'(
                        "00000000",
                        "00111000",
                        "01000100",
                        "01000100",
                        "01000100",
                        "01000100",
                        "00111000",
                        "00000000"
);                  --"0"字符编码列数据存储器
begin
process(clk1,reset)
variable cnt: STD_LOGIC_VECTOR (0 TO 2);
variable COL_COUNT: STD_LOGIC_VECTOR (0 TO 2);
      begin
      if reset='1' then
            COL_COUNT:="000";
            cnt:="000";
            AD<=roma(conv_integer(cnt));
            LED_S<="000";
      else if (clk1'event and clk1='1')then
            cnt:=cnt+1;
            AD<=roma(conv_integer(cnt));--送列数据
            COL_COUNT:=COL_COUNT+1;
            LED_S<=COL_COUNT;--行扫描
            end if;
      end if;
end process;
end LEDDOT_ARCH;
```

附 1.2　　QUARTUS II4.2 的使用步骤参考

以下介绍如何使用 QUARTUS II 建立工程、程序编译、下载以及运行。

1．新建一个工程，File->New Project Wizard。

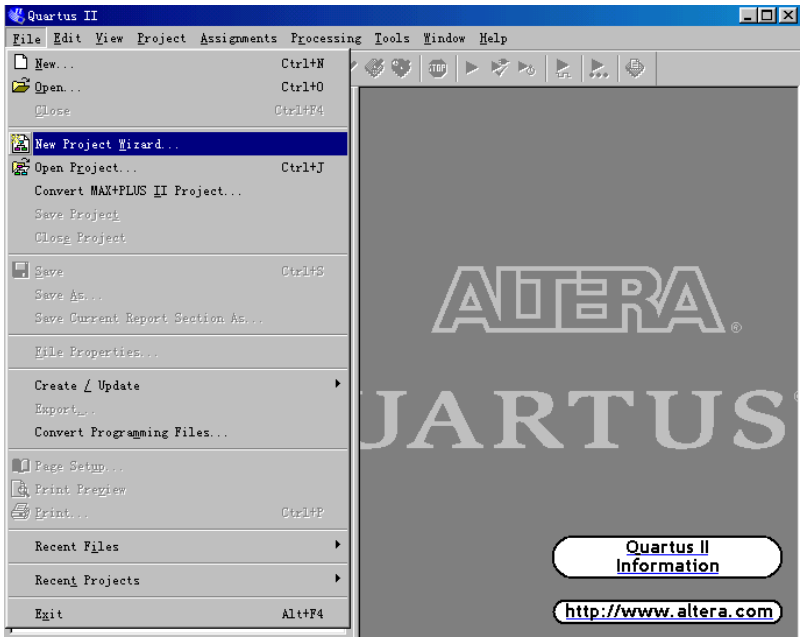


图 1.4 新建工程

2．在出现的 New Project Wizard: Introduction 中单击“ NEXT ”。则出现下图所示的对话框，分别输入工程路径、工程名、实体名，工程名和实体名可以相同。

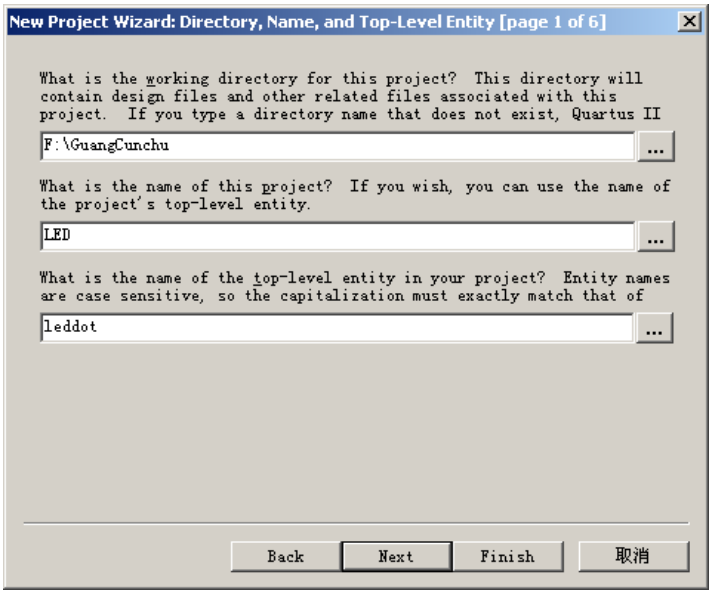


图 1.5 新建工程向导

3．点击上图中的“ Finish ”，完成工程的建立。

4．File->New，在 New 窗口中的 Device Design Files 中选择 VHDL File 建立 VHDL 文

件。

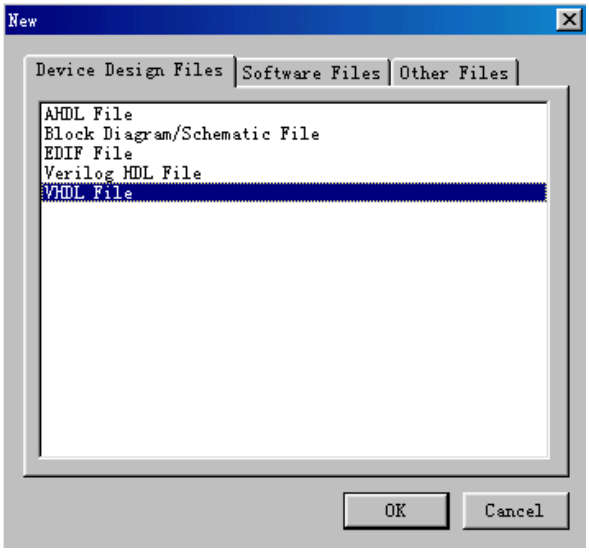


图 1.6 新建 vhd 文件

5．编写 VHDL 程序并保存。（从给定的 vhd 程序直接拷贝）

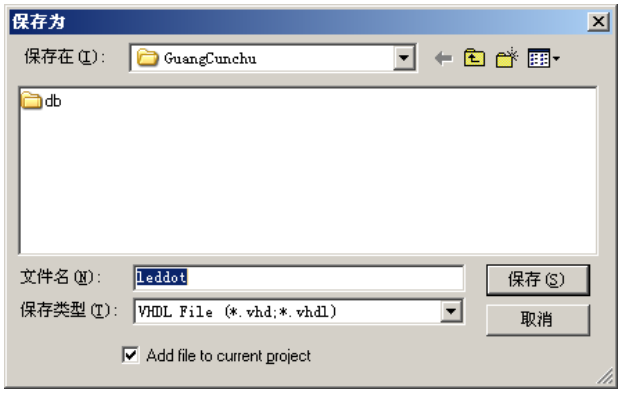


图 1.7 保存 vhd 文件

6．在下图所示的工程浏览窗口中，选择 vhd 文件，利用快捷键，选择“ Creat Symbol Files for Current File ”从 VHDL 文件生成 Symbol 文件。

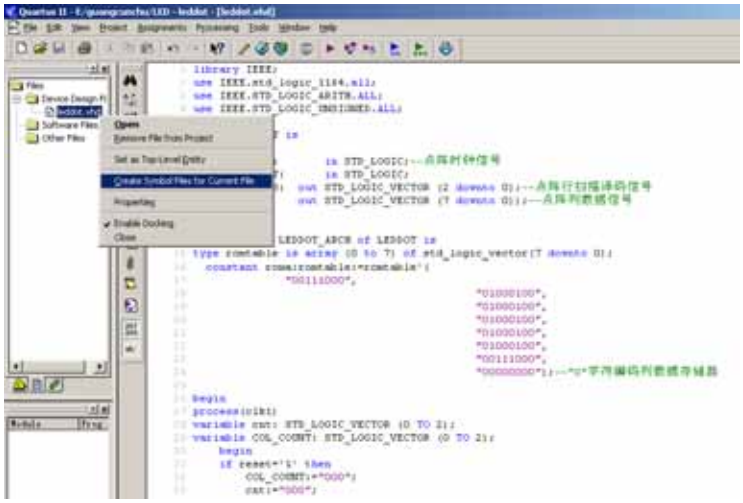


图 1.8　生成 Symbol 文件

7．建立 bdf 文件，选择 File->New，在 New 窗口中的 Device Design Files 选择 Block Diagram/Schematic File。

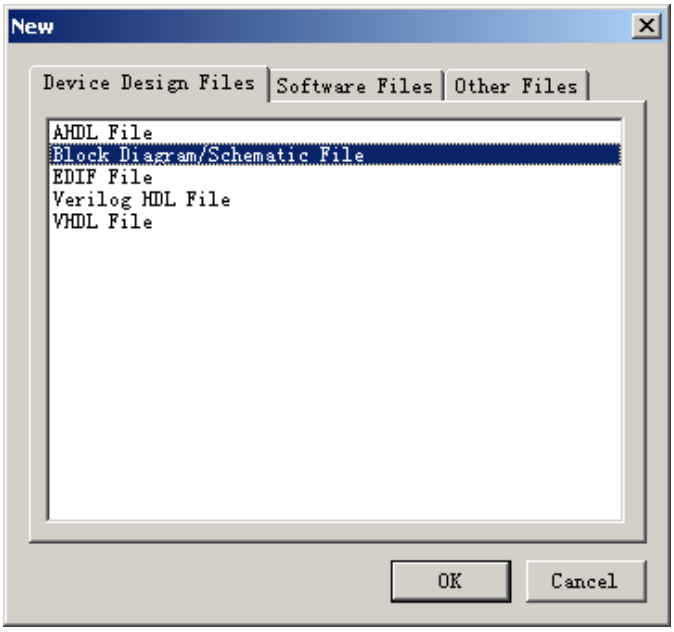


图 1.9　建立 bdf 文件

8．选择 Edit->Insert Symbol，出现 Symbol 窗口。

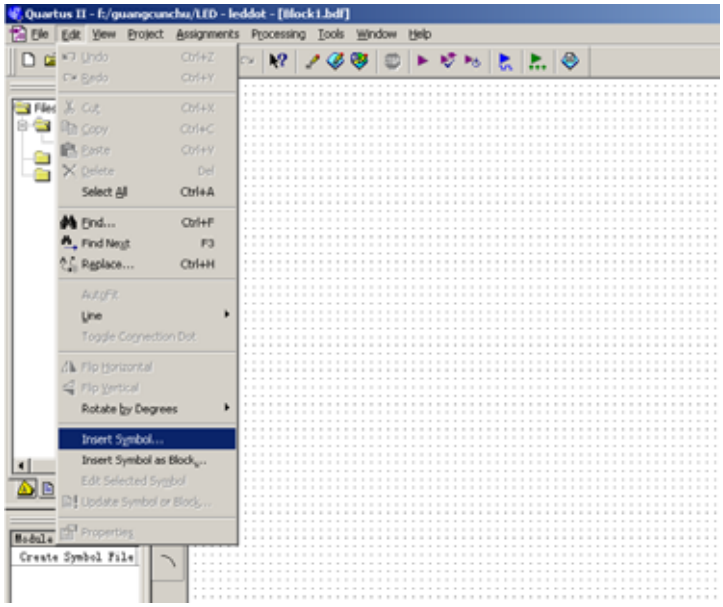


图 1.10 插入符号

9．如下图，选择 Project 目录下 vhd 文件生成的符号，并点击 OK 进行添加符号。

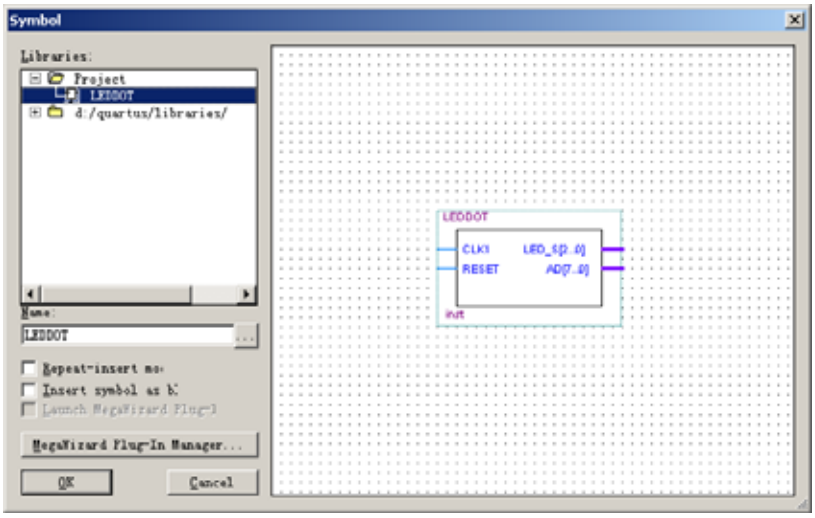


图 1.11 添加符号

10．再在 Symbol 窗口中选择输入、输出符号并连接，修改符号名称，使其与符号中的输入/输出名一致（也可不相同）。如下图所示。

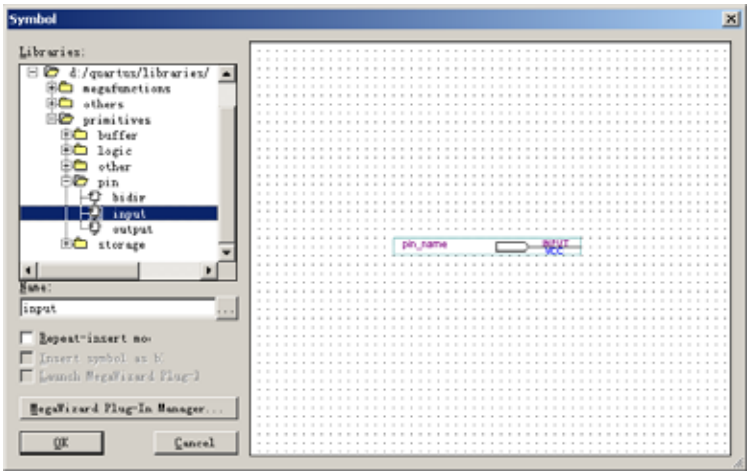


图 1.12 添加输入与输出符号

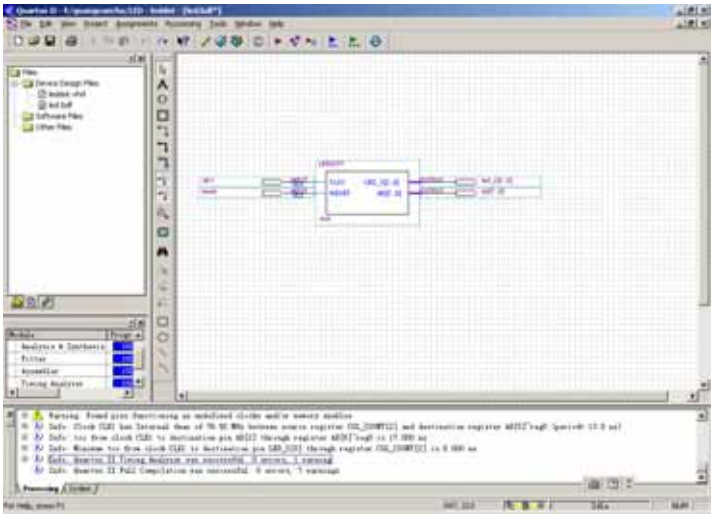


图 1.13 修改符号名称

11．保存 bdf 文件。Processing->Start Compilation

12．芯片选择 Assigment->Device，按照下图进行设置。

注意：“Target device ”中不能选择：Auto device selected by Fitter from the ‘Available device’ list.

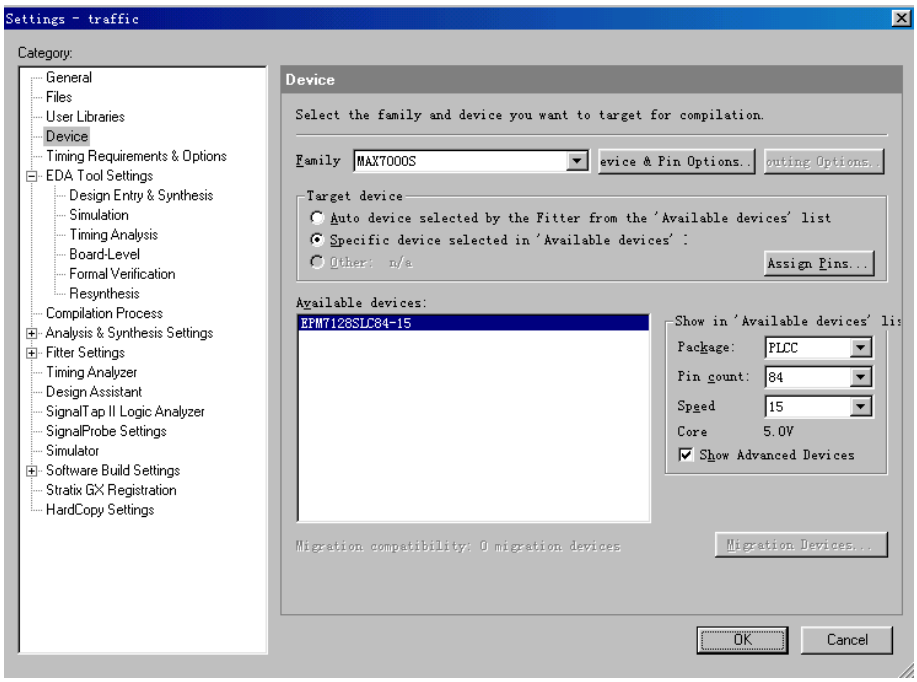


图 1.14　分配芯片

13．定义引脚：在 bdf 文件中单击信号，快捷键中 Assignment Editor。

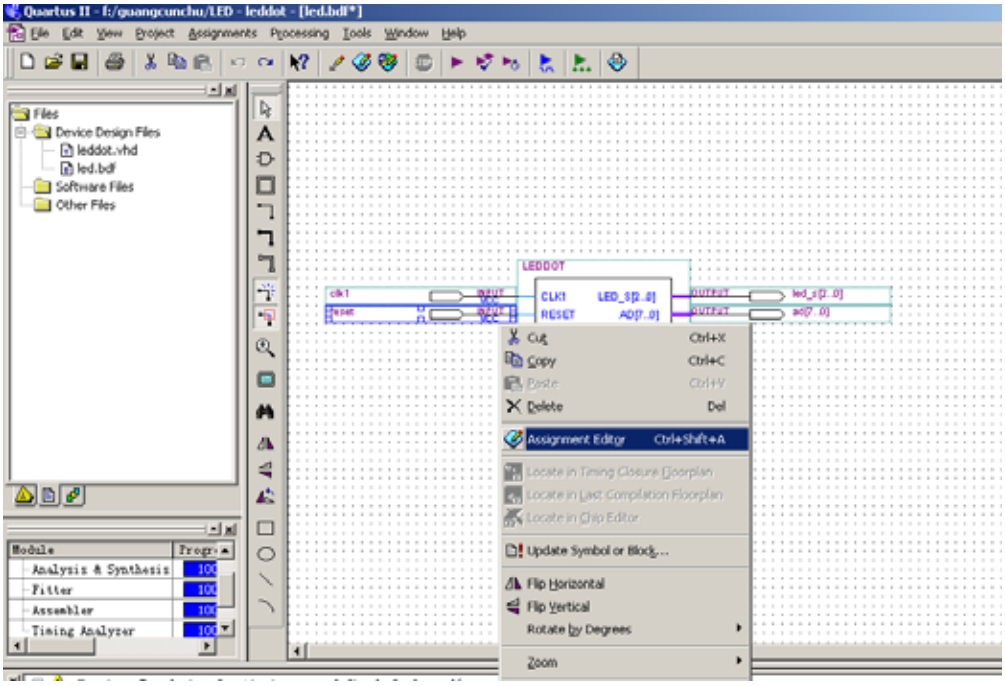


图 1.15 绑定引脚

14．在 Assignment Editor 的窗体中点击 Pin 按钮，出现如下所示窗体：

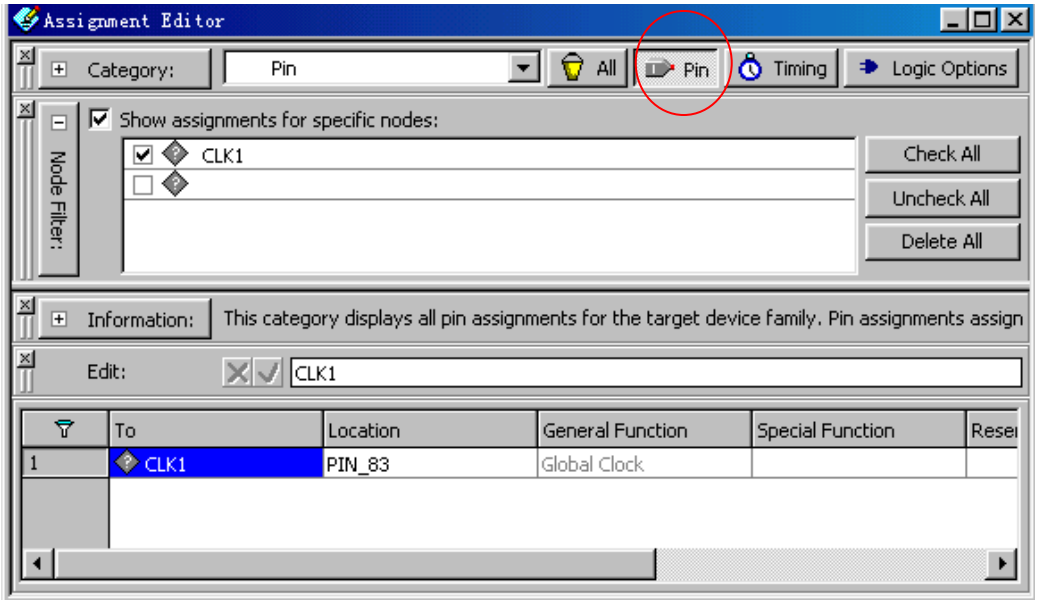


图 1.16 绑定引脚窗口

在 Location 处选择相应的 CPLD 的引脚，关闭该窗口，在出现如下所示窗口中，点击“是”保存。

按照上述方法设置所有信号的引脚。然后进行 Processing->Start Compilation 编译。

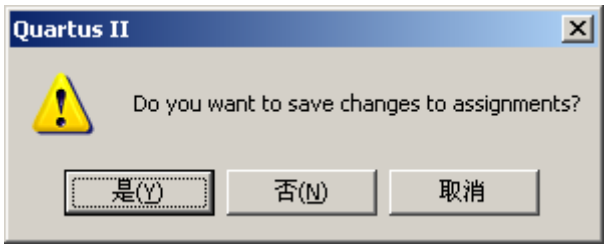


图 1.17 保存引脚的分配

15．下载程序：选择 Tools->Programmer，弹出如下所示窗口。如果该处显示 ByteBlasterMV [LPT1]，则可直接进入步骤 16。否则按照以下步骤进行设置。

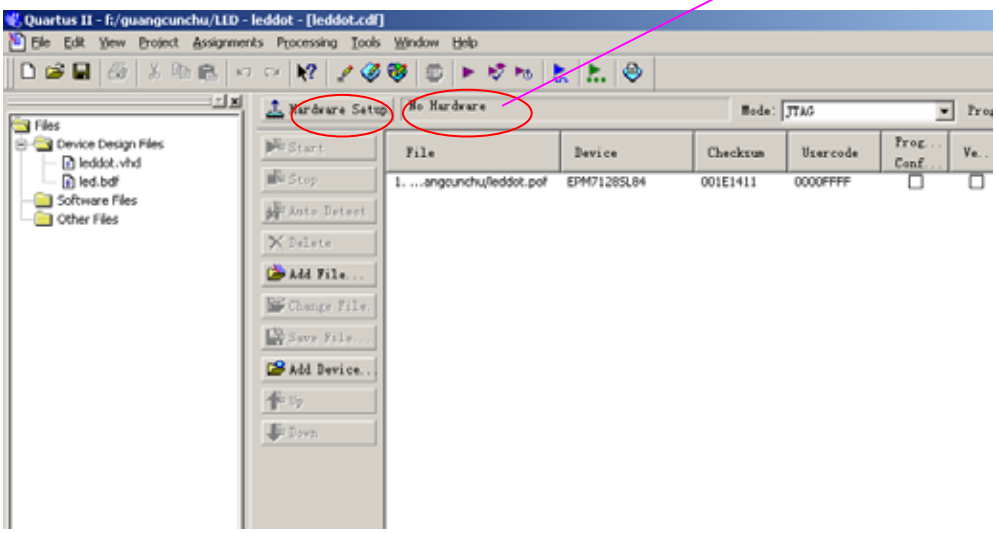


图 1.18 保存引脚的分配

点击上图所示窗口中 Hardware Setup，在弹出的下图所示窗口中选择 “ Add Hardware…… ”

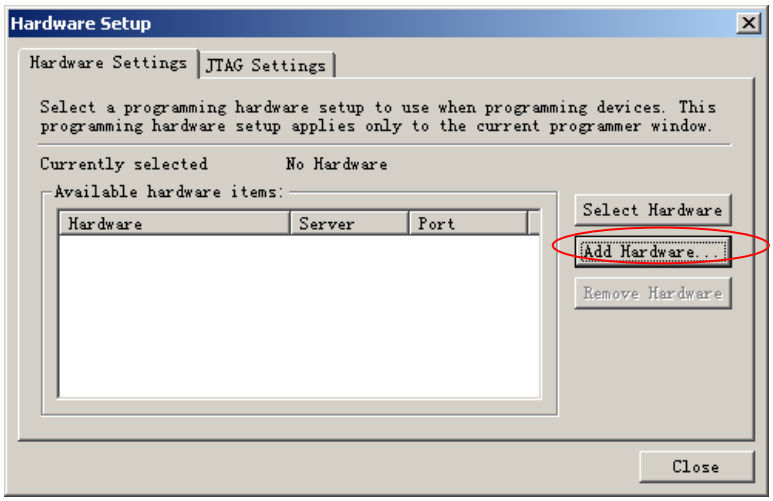


图 1.19 Hardfware Setup 窗口

按照如下所示窗口进行设置：

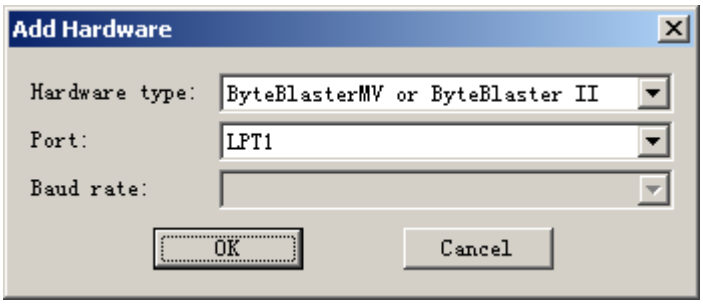


图 1.20　Add Hardware 窗口

单击 ByteBlasterMV，然后点击“ Select Hardware ”，在此位置出现“ ByteBlasterMV [LPT1] ”，然后关闭该窗口。

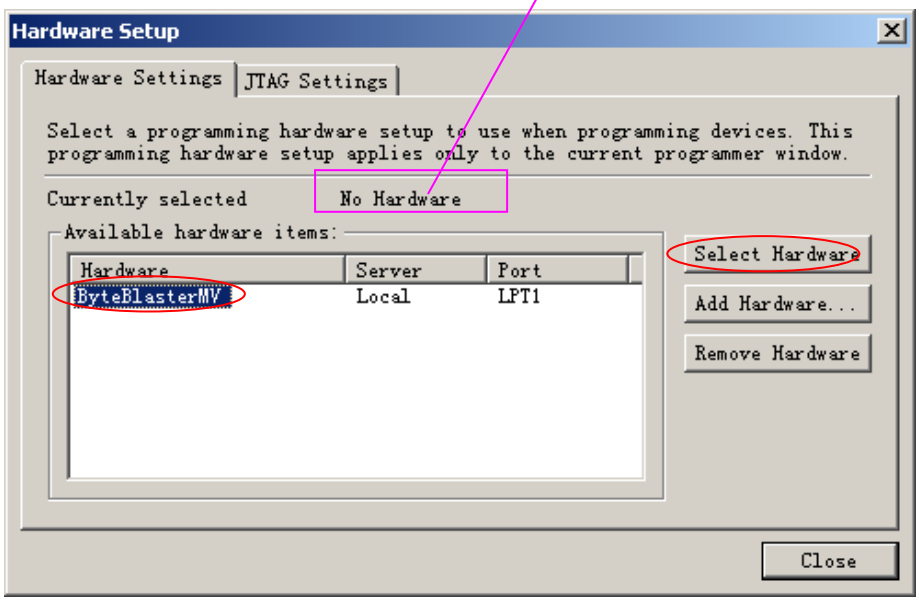


图 1.21　Hardware Setup 窗口

16．在如下所示窗口中选择 Program Configure，然后点击 Start 按钮开始下载。

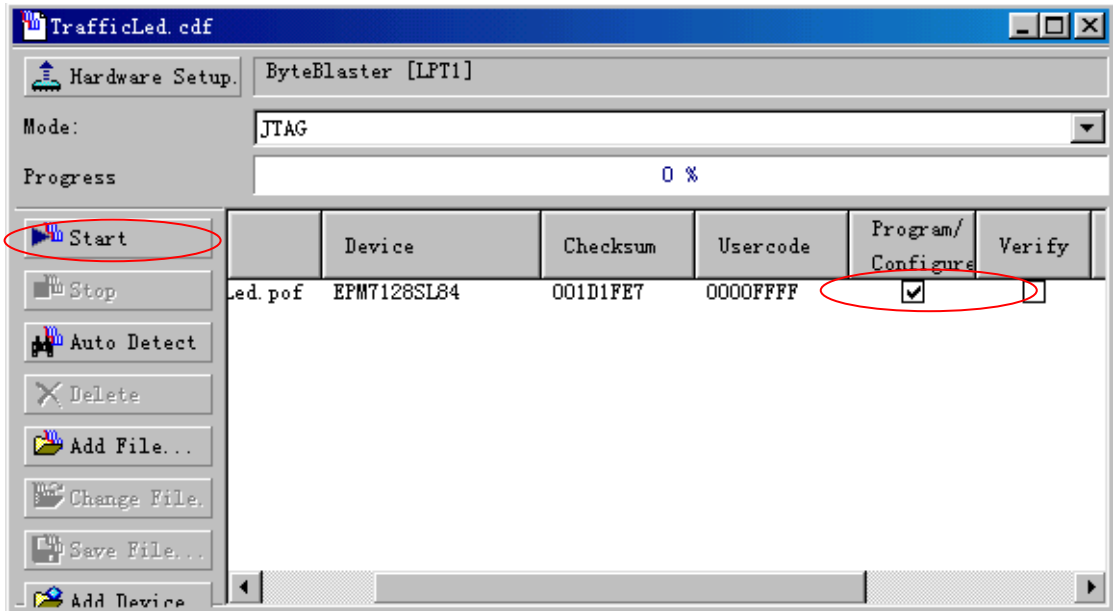


图 1.22　下载目标文件

17．观察运行结果。

实验二　液晶显示实验

一、实验目的

验证用画逻辑图设计 LCD 译码驱动电路，用 VHDL 程序驱动 LCD 显示数字 0-9。

二、实验原理

下图是FM1602J液晶模块：R44、R45决定液晶的对比度。

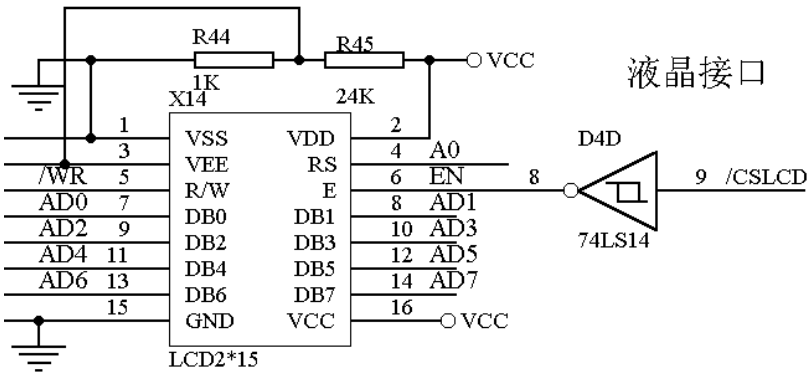


图2.1　FM1602J液晶模块

三、实验内容

1、验证实验：验证字符型 LCD 译码驱动程序，在液晶上循环显示 0-7。

2、设计实验：修改验证实验程序，在液晶的第一行显示 0～F，显示完后清除显示，然后又从第一行第一个位置开始显示 0～F，重复以上过程。

要求：只显示第一行，不显示第二行。

四、实验步骤

1、在 ***quartus II*** 下建立工程，按照附 2.1 的实验例程建立 vhd 文件。

2、选择器件 EPM7128SLC84－15，对工程进行编译，编译无误后按照表 2.1 绑定引脚并进行编译。注意： **P83 直接连接到脉冲源模块的 100HZ。**

表2.1　　适配卡连线表

输入信号名	对应芯片端子名	引入端子名	功　能	输出信号名	对应芯片端子名	引入端子名	功　能
Clk	P83	100HZ	读信号	EN	P45	/LCD	片选
Reset	P4	RST	写信号	RS	P52	A0	寄存器选择
				RW	P60	A8	读写
				AD0	P8	AD0	数据线 AD0
				AD1	P12	AD1	数据线 AD1
				AD2	P16	AD2	数据线 AD2
				AD3	P21	AD3	数据线 AD3
				AD4	P24	AD4	数据线 AD4
				AD5	P30	AD5	数据线 AD5
				AD6	P34	AD6	数据线 AD6
				AD7	P37	AD7	数据线 AD7

3、按照表 2.1 在 EDA 实验箱中的适配卡上连线，并对模块 23 中的开关按照如下进行设置：

S2-1，OFF　　蜂鸣器关

S2-2，OFF　　交通灯不显示

S2-3，OFF　　骰子灯不显示

S2-4，OFF　　LED 点阵行选择不使用 CPLD 输出

S2-5，OFF　　LED 点阵行选择不使用三八译码器输出

S2-6，OFF　　LED 数码管不显示

S2-7，0FF　　L8-L1 不允许显示

4、下载目标文件。

5、观察实验结果。

6、按照设计实验要求修改实验例程，编译无误后下载目标文件进行验证。

附 2.1　　液晶显示 VHDL 程序

```
--循环显示 0～7，两行显示;
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity LCD1602 is
   Port ( Clk : in std_logic;
            Reset:in std_logic;
            RS : out std_logic; --命令/数据选择
            RW : out std_logic; --读写
            EN : out std_logic; --片选
            AD : out std_logic_vector(7 downto 0));
end LCD1602;
architecture Behavioral of LCD1602 is
   signal state:integer range 0 to 5;
  begin
EN <=Clk;
RW <='0';
control:process (reset,state,clk)
   variable cnt1 :std_logic_vector(4 downto 0);
   variable cnt2 :std_logic_vector(3 downto 0);
   type ram1 is array(0 to 31) of std_logic_vector(7 downto 0);
constant dgram1:ram1:=(("10000000"),
("10000001"),
("10000010"),
("10000011"),
("10000100"),
("10000101"),
("10000110"),
("10000111"),
("10001000"),
("10001001"),
("10001010"),
("10001011"),
("10001100"),
("10001101"),
("10001110"),
("10001111"),
("11000000"),
("11000001"),
```

```
("11000010"),
("11000011"),
("11000100"),
("11000101"),
("11000110"),
("11000111"),
("11001000"),
("11001001"),
("11001010"),
("11001011"),
("11001100"),
("11001101"),
("11001110"),
("11001111"));
   type ram2 is array(0 to 15) of std_logic_vector(7 downto 0);
constant dgram2:ram2:=(("00110000"),--"0"  的 ASCII 码
("00110001"),--"1"  的 ASCII 码
("00110010"),
("00110011"),
("00110100"),
("00110101"),
("00110110"),
("00110111"),--"7"  的 ASCII 码
("00110000"),
("00110001"),
("00110010"),
("00110011"),
("00110100"),
("00110101"),
("00110110"),
("00110111"));
begin
if reset='1' then
      state<=0;
      RS<='0'; --命令选择
      cnt1:="00000";
      cnt2:="0000";
elsif clk'event and clk='0'then
      if    (state=5) then
      state<=4;
      cnt2:=cnt2+1;
      else if (state=4) then
      state<=5;
      cnt1:=cnt1+1;
```

```
        else state<=state+1;
        end if;
end if;
end if;
case state is
        when 0=>
                AD<="00000001";--01h
        when 1=>
                AD<="00111000";--38h
        when 2=>
                AD<="00001111";--0fH
        when 3=>
                AD<="00000110";--06H
        when 4=>
                RS<='0'; --命令选择
                AD<=dgram1(conv_integer(cnt1)); --写显示的地址

        when 5=>
                RS<='1'; --数据选择
                AD<=dgram2(conv_integer(cnt2)); --写显示的数据
        when others=>null;
end case;
end process control;
end Behavioral;
```

附 2.2　　1602 液晶显示模块的 11 条指令表

序号	指令	RS	R/W	D7	D6	D5	D4	D3	D2	D1	D0
1	清显示	0	0	0	0	0	0	0	0	0	1
2	光标返回	0	0	0	0	0	0	0	0	1	*
3	置输入模式	0	0	0	0	0	0	0	1	I/D	S
4	显示开/关控制	0	0	0	0	0	0	1	D	C	B
5	光标或字符移位	0	0	0	0	0	1	S/C	R/L	*	*
6	设置功能	0	0	0	0	1	DL	N	F	*	*
7	设置字符发生存储器地址	0	0	0	1	字符发生存储器地址（ACG）					
8	设置数据存储器地址	0	0	1	显示数据存储器地址（ADD）						
9	读忙标志或地址	0	1	BF	计数器地址（AC）						
10	写数到 CGRAM（字符发生器）或 DDRAM（显示存储器）	1	0	要写的数据							
11	从 CGRAM 或 DDRAM	1	1	读出数据							

说明：“ * ”表示任意电平。

指令 1：清显示，指令码 01H，光标复位到地址 00H 位置。
指令 2：光标复位，光标返回到地址 00H。
指令 3：光标和显示模式设置
I/D：光标移动方向，高电平右移，低电平左移
S：屏幕上所有文字是否左移或右移，高电平表示有效，低电平则无效。
指令 4：显示开关控制
D：控制整体显示的开与关，高电平表示开显示，低电平表示关显示
C：控制光标的开与关，高电平表示有光标，低电平表示无光标
B：控制光标是否闪烁，高电平闪烁，低电平不闪烁。
指令 5：光标或显示移位
S/C：高电平时移动显示的文字，低电平时移动光标。
指令 6：功能设置命令
DL：高电平时为 4 位总线，低电平时为 8 位总线；
N：低电平时为单行显示，高电平时双行显示；
F：低电平时显示 5×7 的点阵字符，高电平时显示 5×10 的点阵字符。
指令 7：字符发生器 RAM 地址设置。
指令 8：DDRAM 地址设置。
指令 9：读忙信号和光标地址
BF：为忙标志位，高电平表示忙，此时模块不能接收命令或者数据，如果为低电平表示不忙。
指令 10：写数据。
指令 11：读数据。

实验三　LCD 投影机实验

一、实验目的

1、 了解 LCD 投影机的结构和特点。
2、 熟练掌握投影机的使用方法。

二、实验原理

LCD 投影机是利用液晶的光电效应，即液晶分子的排列在电场作用下发生变化，影响其液晶单元的透光率或反射率，从而影响它的光学性质，产生具有不同灰度层次及颜色的图像。

按照液晶板的片数，LCD 投影机分为三片机和单片机，本次实验使用三片 LCD 板投影机，其原理是光学系统把强光通过分光镜形成 RGB 三束光，分别透射过 RGB 三色液晶板；信号源经过 AD 转换，调制加到液晶板上，通过控制液晶单元的开启、闭合，从而控制光路的通断，RGB 光最后在棱镜中汇聚，由投影镜头投射在屏幕上形成彩色图像。

三、实验内容

1、观察投影机各部件的结构。
2、投影机的使用操作。

四、实验步骤

1、拆开报废的给定 LCD 投影机。
2、认真观察和了解机内分光镜、液晶板、光源、冷却风扇、变压器等部件的相对位置及其各个部件的作用，并画出投影机各部件的结构示意图以及投影机的成像原理图，并记录在实验报告纸上。
3、安装好投影机。
4、使用一台可以正常使用的投影机，把投影机通过视频线连接到 PC，挂好银幕，调节投影机的各个操作部件，总结出影像的大小、上下左右位置、正反、清晰度、亮度变化的调节规律。
5、实验完毕，切断电源，整理好仪器。