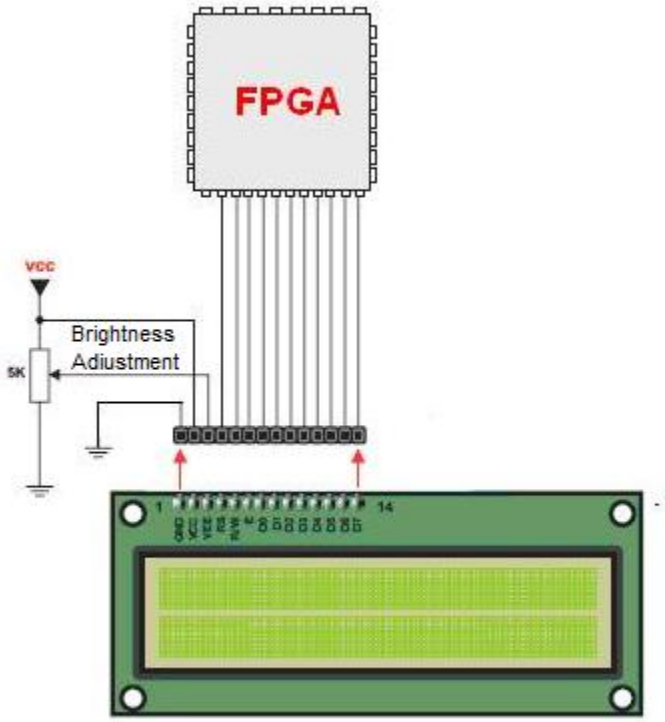
การควบคุมโมดูล LCD ด้วย FPGA

วัตถุประสงค์

ต้องการควบคุมการแสดงผลของ โมดูล LCD ด้วย FPGA



รูปที่ 1 การใช้งาน

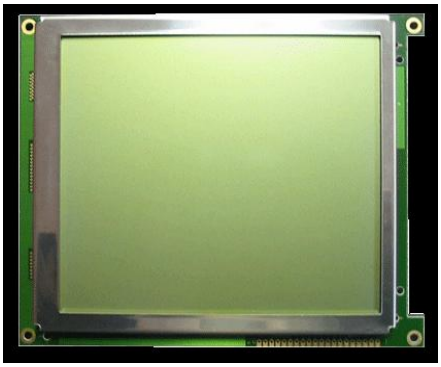
โมดูลแอลซีดี (LCD Module)

เป็นอุปกรณ์แผงแสดงผล ที่สามารถแสดงได้ทั้งข้อความ ตัวเลข และรูปภาพ เหมาะสำหรับงานแสดงผลที่ต้องการให้สิ้นเปลืองพลังงานต่ำ โดยทั่วๆไป โมดูลแอลซีดี สามารถแบ่งออกได้เป็น 3 ประเภท ได้แก่

- **Character LCD Module** แบบนี้สามารถแสดงผลได้ทั้งตัวอักษรและตัวเลข โดยจัดแบ่งเป็นแถวๆ เช่น แบบ 1 แถว แถวละ 16 ตัวอักษร หรือแบบ 4 แถว แถวละ 16 ตัวอักษรเป็นต้น ตามรูปที่ 15.1 เป็นแบบ 1 แถว 16 ตัวอักษร

รูปที่ 2 โมดูลแอลซีดีแบบอักขระ

- Graphic LCD Module แบบนี้สามารถแสดงผลเป็นรูปภาได้ และสามารถทำเป็นรูปภาพของอักษรแบบต่างๆได้

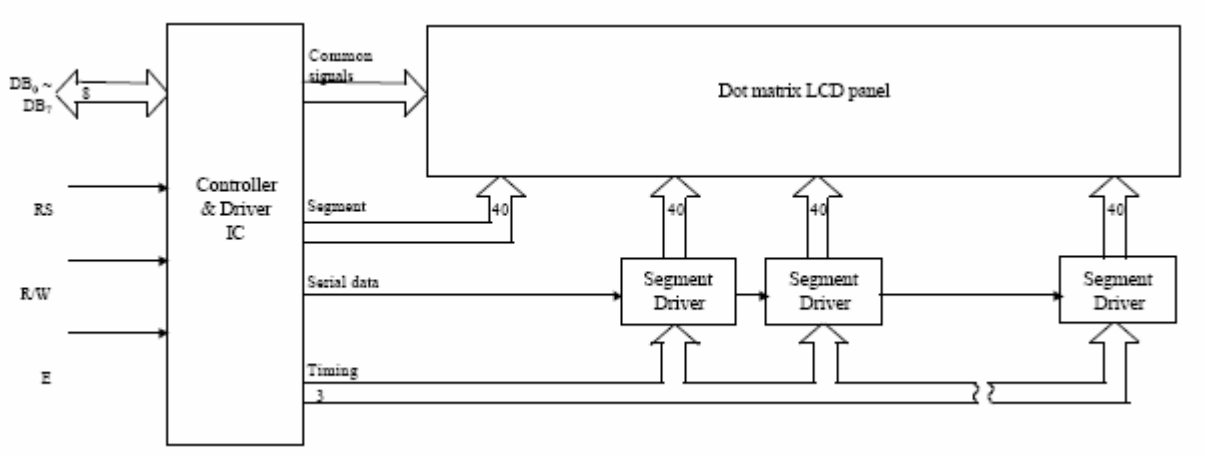


รูปที่ 3 โมดูลแอลซีดีแบบกราฟ

- Segment Display LCD Module เป็นแบบที่มีรูปแบบการแสดงจำกัด ไม่กี่แบบ

รูปที่ 4 โมดูลแอลซีดีแบบเซกเมนท์ แสดงผลเป็นตัวเลข

ส่วนประกอบของโมดูลแอลซีดี



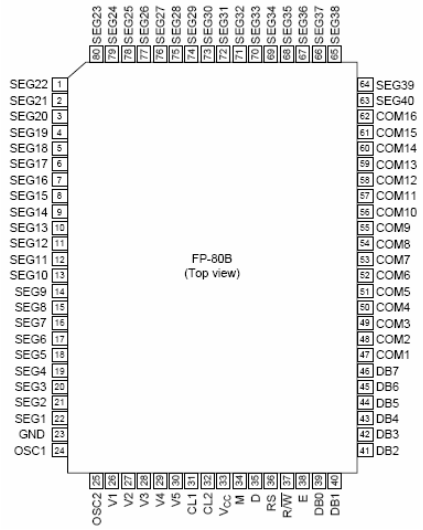
รูปที่ 5 บล็อกไดอะแกรมของโมดูลแอลซีดี

ส่วนประกอบของโมดูลแอลซีดีประกอบด้วยกัน 3 ส่วนคือ

- Dot Matrix LCD เป็นตัวแสดงผล ทำงานในลักษณะของการปิดหรือเปิด ตัวเองกับแสง
- Driver เป็นตัวขับ LCD รับสัญญาณมาจากส่วนควบคุม เบอร์ที่นิยมใช้ได้แก่ HD44100H และ MSM5259
- Controller เป็นตัวรับข้อมูลจากอุปกรณ์ภายนอก แล้วควบคุมการทำงานของ LCD เบอร์ที่นิยมใช้ สำหรับแบบ Character ได้แก่ HD4478 ส่วนแบบ Graphic ได้แก่ HD61830

HD44780

HD 44780 เป็นส่วนควบคุมของแอลซีดีโมดูลที่นิยมใช้กันมาก มีรูปแบบการจัดขาและบล็อกไดอะแกรมตามรูปที่ 15.5 และ 15.6 ตามลำดับ



รูปที่ 6 รูปแบบและการจัดตำแหน่งขาของ HD44780

คุณสมบัติพอสังเขปของ HD44780

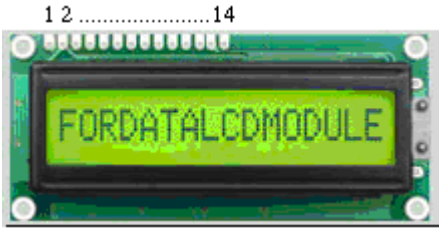
- จำนวนจุดต่อตัวอักษรทำได้ทั้งแบบ 5 x 8 และ 5 x 10
- ใช้แหล่งจ่ายกำลังงานต่ำ 2.7 – 5.5 โวลท์
- ให้กำลังขับแอลซีดีได้ย่านกว้าง 3.0 - 11 โวลท์
- ใช้กับระบบบัสของซีพียูได้ความถี่ถึง 2 MHz (ที่ VCC = 5V)
- สามารถอินเตอร์เฟสกับซีพียูแบบ 4- บิตหรือ 8 บิตก็ได้
- หน่วยความแรมจำสำหรับการแสดงผล (DDRAM = Display Data RAM) มีขนาดสูงสุด 80 X 8-bit (80 ตัวอักษร)
- คาแรคเตอร์ เจนเนอเรเตอร์รอม (CGROM = Character Generator ROM) มีขนาด 9,920 บิต ซึ่งใช้สำหรับเป็น ตัวอักษร 240 ตัว

- ตัวอักษรขนาด 5 X 8 จุด ได้ 208 ตัว
- ตัวอักษรขนาด 5 X 10 จุด ได้ 32 ตัว

- มีแรมภายในสำหรับคาแรคเตอร์ เจนเนอเรเตอร์(CGRAM = Character Generator RAM) ขนาด 64 x 8-bit
  - 8 character fonts (5 X 8 จุด)
  - 4 character fonts (5 X 10 จุด)
- ตัวขับภาคแสดงผลมีขนาด 16-common X 40-segment
- มีคำสั่งควบคุมการทำงานหลายอย่าง
  - Display clear, cursor home, display on/off, cursor on/off, display character blink, cursor shift, display shift
- มีการรีเซ็ตวงจรโดยอัตโนมัติเมื่อเริ่มจ่ายไฟเลี้ยง
- มีวงจรออสซิลเลเตอร์ภายในที่ใช้เพียงตัวต้านทานภายนอก
- กำลังงานสูญเสียต่ำ

ขาของโมดูลแอลซีดี

ขาที่	สัญญาณ	อินพุท/เอาทพุท	ต่อกับ	รายละเอียด
1	Vss		แหล่งจ่าย	0 V Gnd
2	Vcc		แหล่งจ่าย	+ 5V
3	Vee		แหล่งจ่าย	ใช้ปรับความสว่างของ LCD ถ้าต่อลงดินจะสว่างที่สุด
4	RS	อินพุท	ซีพียู	สัญญาณ Register Select ใช้เลือกรีจิสเตอร์ควบคุมหรือ หน่วยความจำแสดงผล
5	R/W	อินพุท	ซีพียู	สัญญาณควบคุมการอ่าน/เขียน "0" เขียนหรือส่งข้อมูลให้แก่โมดูล "1" อ่านข้อมูลจากโมดูล
6	E	อินพุท	ซีพียู	Enable - สัญญานสั่งให้เริ่มต้นการทำงาน สำหรับการอ่าน/เขียนข้อมูล การรับส่งข้อมูลจะเกิดเมื่อเป็น '1' และขอบขาลง
7 ~ 0	DB0 ~ DB3	อินพุท/เอาทพุท	ซีพียู	เป็นบัสแบบสองทิศทางใช้สำหรับส่งถ่ายข้อมูลระหว่างซีพียูกับโมดูล
11~ 14	DB4 ~ DB7	อินพุท/เอาทพุท	ซีพียู	เป็นบัสแบบสองทิศทางใช้สำหรับส่งถ่ายข้อมูลระหว่างซีพียูกับโมดูล



รูปที่ 7 แสดงตำแหน่งขาของโมดูลแอลซีดี

สัญญาณ RS หรือ Register Select

ใช้กำหนดว่าต้องการติดต่อกับรีจิสเตอร์ควบคุมหรือหน่วยความจำแสดงผล

ถ้าเป็น “0” แสดงว่าต้องการติดต่อกับรีจิสเตอร์ควบคุม

ถ้าเป็น “1” แสดงว่าต้องการติดต่อกับริจิสเตอร์แสดงผล

สัญญาณ R/W

เป็นสัญญาณควบคุมการอ่านหรือเขียน

ถ้าเป็น "0" ” แสดงว่าต้องการเขียนหรือส่งข้อมูลให้แก่โมดูล

ถ้าเป็น "1" แสดงว่าต้องการอ่านข้อมูลจากโมดูล

สัญญาณ RS และ R/W ทำงานร่วมกันโดยมีความหมายดังนี้

RS	R/W	การทำงาน
0	0	ส่งคำสั่งไปควบคุมโมดูล
0	1	อ่านแฟลก Busy flag หรือค่าตำแหน่ง
1	0	ส่งตัวอักขระออกแสดงผล
1	1	อ่านอักขระที่ตำแหน่ง

สัญญาณ DB0 – DB3

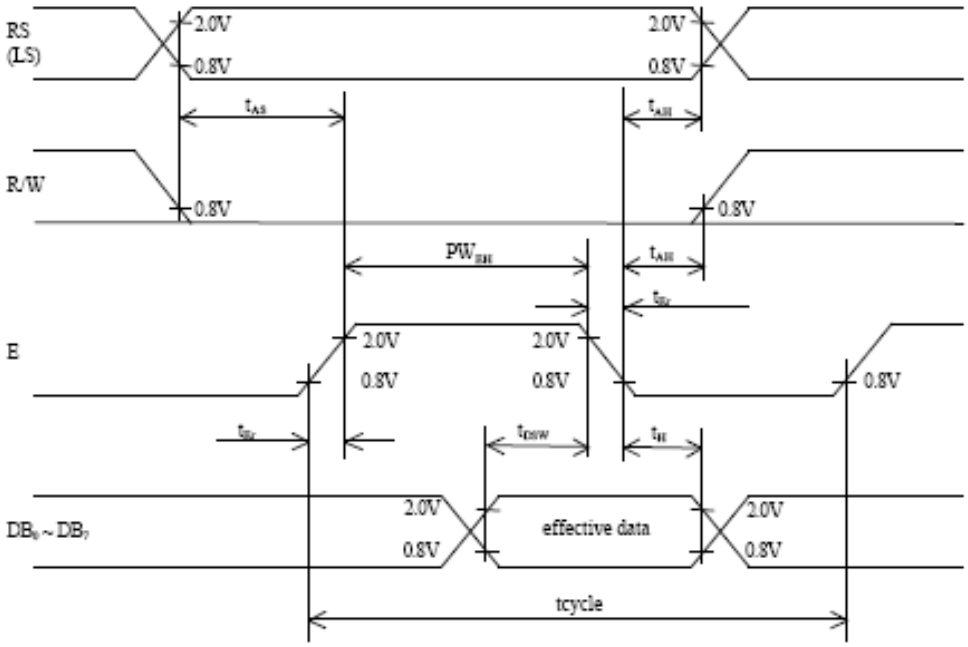
เป็นบัสแบบสองทิศทางใช้สำหรับส่งถ่ายข้อมูลระหว่างซีพียูกับโมดูล ในกรณีทีการทำงานเป็นแบบ 4 บิต บัสนี้ไม่ได้ใช้และควรต่อลงดินด้วย แต่ถ้าเป็นการทำงานแบบ 8 บิต บัสนี้จะเป็น 4 บิตต่ำ ใช้เพื่อการส่งถ่ายข้อมูล

สัญญาณ DB4 – DB7

เป็นบัสแบบสองทิศทางใช้สำหรับส่งถ่ายข้อมูลระหว่างซีพียูกับโมดูล ในกรณีทีการทำงานเป็นแบบ 4 บิต จะใช้บัสนี้ส่งถ่ายข้อมูล แต่ถ้าเป็นการทำงานแบบ 8 บิต บัสนี้จะเป็น 4 บิตสูง นอกจากนี้ DB7 ยังใช้เป็นบิตแสดงสถานะ Busy ด้วย

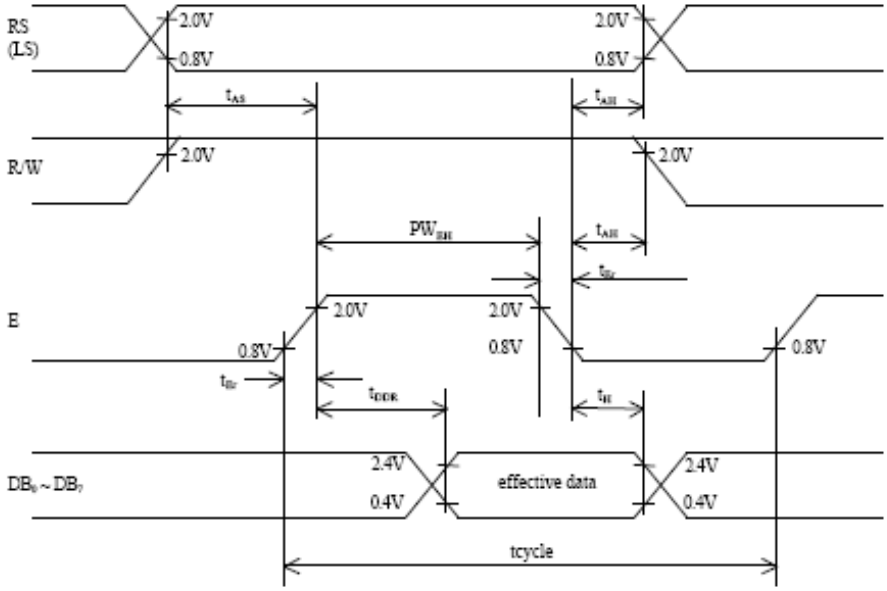
สัญญาณต่างๆเหล่านี้ ซีพียูจะใช้สำหรับติดต่อกับโมดูล โดยมีรูปแบบการเขียนและอ่านดังนี้

การเขียนข้อมูลให้โมดูล



รูปที่ 8 ไดอะแกรมเวลาขณะเขียนข้อมูลให้กับ โมดูลแอลซีดี

การอ่านข้อมูลจากโมดูล



รูปที่ 9 ไดอะแกรมเวลาขณะอ่านข้อมูลจากโมดูลแอลซีดี

คำสั่งควบคุม

คำสั่งควบคุมการทำงานของโมดูลแอลซีดีก็จะเป็นคำสั่งควบคุมของตัวควบคุมนั่นเองในที่นี้ก็หมายถึงคำสั่งของ HD44780 ก่อนที่จะกล่าวถึงคำสั่ง ขอให้เข้าใจความหมายของคำต่อไปนี้ก่อน

ความหมาย

1. DDRAM (Display Data Ram) คือหน่วยความจำภายในตัวโมดูลแอลซีดีที่เป็นบัฟเฟอร์ของข้อมูล ถ้าเขียนรหัส ASCII ใดๆ ลงในหน่วยความจำนี้ก็จะปรากฏเป็นตัวอักษรที่จอแสดงผลทันที

2.CGRAM (Character Generator Ram) เป็นหน่วยความจำภายในตัวโมดูล ใช้สำหรับเก็บภาพตัวอักษรที่ผู้ใช้สร้างขึ้นเอง (สร้างได้ 8 ตัว) โดยอ้างตำหน่งได้ 64 ไบต์ (8 ตัวอักษรคูณด้วย 8 แถว)

3. การเขียนข้อมูลให้โมดูลแต่ละครั้งต้องตรวจสอบความพร้อมของโมดูล โดยตรวจสอบได้จาก Busy Flag หรือระยะเวลาการทำงานของโมดูล ซึ่งดูได้จากตารางคำสั่ง ดังนั้นเมื่อเขียนข้อมูลหนึ่งๆไปเราต้องหน่วงเวลารอให้โมดูลพร้อม จึงจะเขียนชุดใหม่ต่อไป

4. การเขียนข้อมูลให้โมดูลสามารถทำได้ทั้งแบบ 4 บิตและแบบ 8 บิตขึ้นอยู่กับการเชื่อมต่อกับโมดูล ถ้าเป็นการเชื่อมแบบ 4 บิต การเขียนข้อมูลหรืออ่านข้อมูลต้องทำ 2 ครั้งโดยครั้งแรกต้องเป็นบิต 4 ถึงบิต 7 และครั้งที่ 2 จะเป็นบิตที่ 0 ถึง 3

5. ระยะเวลาที่ให้ไว้ในตารางใช้ Fosc = 250kHz.

ตารางที่ 2 ความหมายบิตต่างๆที่ใช้ในตารางที่ 3

Bit name	Setting / Status	
I/D	0 = Decrement cursor position	1 = Increment cursor position
S	0 = No display shift	1 = Display shift
D	0 = Display off	1 = Display on
C	0 = Cursor off	1 = Cursor on
B	0 = Cursor blink off	1 = Cursor blink on
S/C	0 = Move cursor	1 = Shift display
R/L	0 = Shift left	1 = Shift right
DL	0 = 4-bit interface	1 = 8-bit interface
N	0 = 1/8 or 1/11 Duty (1 line)	1 = 1/16 Duty (2 lines)
F	0 = 5x7 dots	1 = 5x10 dots
BF	0 = Can accept instruction	1 = Internal operation in progress

ตารางที่ 3 ชุดคำสั่งของ HD44780

Instruction	Code										Description	Execution time**
	RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0		
Clear display	0	0	0	0	0	0	0	0	0	1	Clears display and returns cursor to the home position (address 0).	1.64mS
Cursor home	0	0	0	0	0	0	0	0	1	*	Returns cursor to home position (address 0). Also returns display being shifted to the original position. DDRAM contents remains unchanged.	1.64mS
Entry mode set	0	0	0	0	0	0	0	1	I/D	S	Sets cursor move direction (I/D), specifies to shift the display (S). These operations are performed during data read/write.	40uS
Display On/Off control	0	0	0	0	0	0	1	D	C	B	Sets On/Off of all display (D), cursor On/Off (C) and blink of cursor position character (B).	40uS
Cursor/display shift	0	0	0	0	0	1	S/C	R/L	*	*	Sets cursor-move or display-shift (S/C), shift direction (R/L). DDRAM contents remains unchanged.	40uS
Function set	0	0	0	0	1	DL	N	F	*	*	Sets interface data length (DL), number of display line (N) and character font(F).	40uS
Set CGRAM address	0	0	0	1	CGRAM address						Sets the CGRAM address. CGRAM data is sent and received after this setting.	40uS
Set DDRAM address	0	0	1	DDRAM address							Sets the DDRAM address. DDRAM data is sent and received after this setting.	40uS
Read busy-flag and address counter	0	1	BF	CGRAM / DDRAM address							Reads Busy-flag (BF) indicating internal operation is being performed and reads CGRAM or DDRAM address counter contents (depending on previous instruction).	0uS
Write to CGRAM or DDRAM	1	0	write data								Writes data to CGRAM or DDRAM.	40uS
Read from CGRAM or DDRAM	1	1	read data								Reads data from CGRAM or DDRAM.	40uS

สรุปคำสั่ง

Initial Setting Commands Function Settings

Command	Description
0x3D	LCD 1 line, 5x8 dot matrix, 4bit interface
0x24	LCD 1 line, 5x10 dot matrix, 4 bit interface
0x28	LCD 2 line, 5x8 dot matrix, 4 bit interface
0x2C	LCD 2 line, 5x10 dot matrix, 4 bit interface
0x30	LCD 1 line, 5x8 dot matrix, 8 bit interface
0x34	LCD 1 line, 5x10 dot matrix, 8 bit interface
0x38	LCD 2 line, 5x8 dot matrix, 8 bit interface
0x20	LCD 2 line, 5x10 dot matrix, 8 bit interface

Display Clear/Reset

Command	Description
0x01	Display Clear
0x02	Cursor Home
0x03	Cursor Home

Input Mode Commands

Command	Description
0x04	After each character displayed on the LCD, shift the cursor to the left
0x05	After each character displayed on the LCD, shift the cursor and display left to the left
0x06	After each character displayed on the LCD, shift the cursor to the right
0x07	After each character displayed on the LCD, shift the cursor and display to the right

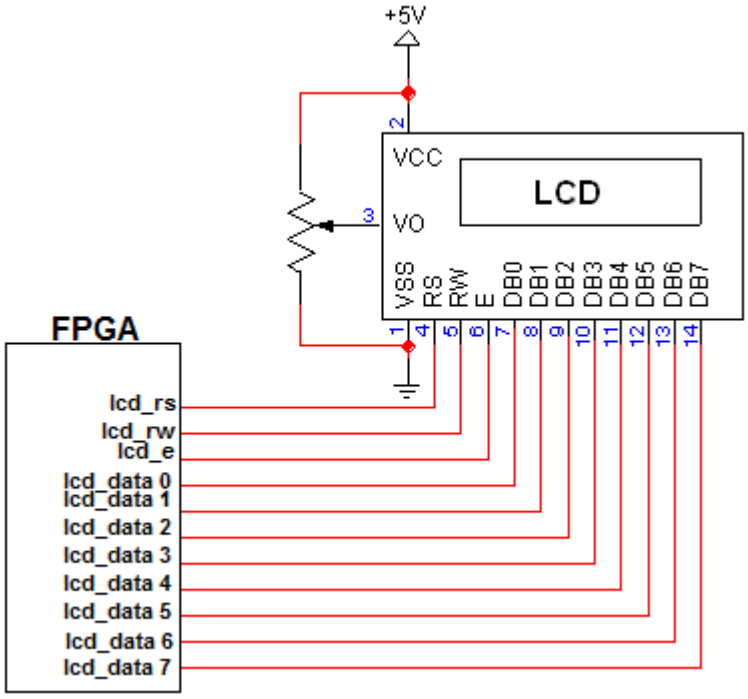
Display ON/OFF and Cursor Commands

Command	Description
0x08	Display OFF, Underscore cursor OFF, Cursor blink OFF
0x09	Display OFF, Underscore cursor OFF, Cursor blink ON
0x0A	Display OFF, Underscore cursor ON, Cursor blink OFF
0x0B	Display OFF, Underscore cursor ON, Cursor blink ON
0x0C	Display ON, Underscore cursor OFF, Cursor blink OFF
0x0D	Display ON, Underscore cursor OFF, Cursor blink ON
0x0E	Display ON, Underscore cursor ON, Cursor blink OFF
0x0F	Display ON, Underscore cursor ON, Cursor blink ON

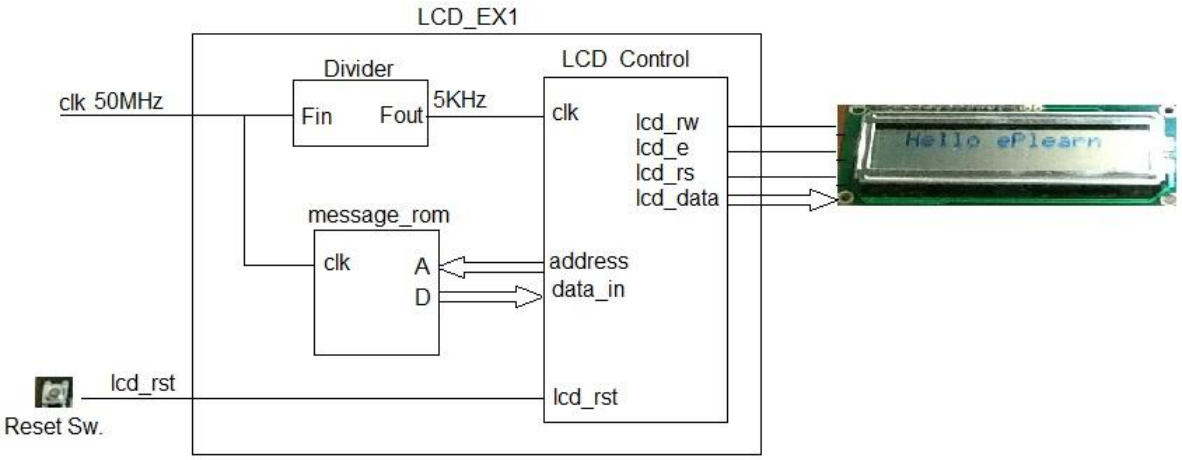
Display/Cursor Shift Commands

Command	Description
0x10	Shift cursor to the left
0x14	Shift cursor to the right
0x18	Shift display to the left
0x1C	Shift display to the right

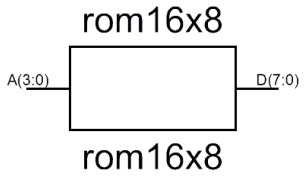
การเชื่อมต่อ โมดูล LCD กับ FPGA



โครงสร้าง



Message_rom




```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity rom16x8 is
port (A : in std_logic_vector (3 downto 0);
      D : out std_logic_vector (7 downto 0));
end rom16x8;
architecture rom16x8_beh of rom16x8 is
  signal Q:character;

type rom_array is array(0 to 15) of character;

constant rom : rom_array :=('H','e','l','l','o',' ',' ',' ',' ',
','W','o','r','l','d',character'val(10),character'val(13));
begin
   Q <= rom(to_integer(unsigned(A)));
   D <= STD_LOGIC_VECTOR(TO_UNSIGNED(CHARACTER'POS(Q), 8));
end rom16x8_beh;
```

LCD Control

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity lcd_control is
port ( clk : in std_logic;   ----clock i/p
           lcd_rst: in std_logic;       ---- Reset i/p
           data_in : in std_logic_vector(7 downto 0);  ---data input
           address : out integer range 0 to 31; -- address
           lcd_rw : out std_logic;   ---read&write control
           lcd_e  : out std_logic;   ----enable control
           lcd_rs : out std_logic;   ----data or command control
           lcd_data   : out std_logic_vector(7 downto 0)  ---data line
           );
end lcd_control;

architecture Behavioral of lcd_control is
         constant N: integer := 4; --:=22;
   type arr is array (1 to N) of std_logic_vector(7 downto 0);
         constant datas : arr := (X"38",  --2 Line, 5x8 and 8 bit interface
                                  X"0c",  --Display On, cursor off
                                  X"06",  --shift display right
                                  X"01"   --Clear display screen
                                  ); --command and data to display

         signal clk5k : std_logic := '0';
```

```
        type state_type is(initial_lcd,set_cursor,lcd_print);
        signal state : state_type := initial_lcd ;
        signal e : std_logic := '0';

begin

DIVIDER1 : entity work.DIVIDER
   generic map(fin => 50000000,
                fout => 5000 )
   port map  (CLK=>clk,
        Q => clk5k );

        lcd_rw <= '0'; ----lcd write
        process(clk5k)
            variable j : integer := 1;
        begin
            if clk5k'event and clk5k = '1' then
                if lcd_rst='1' then
                    state <= initial_lcd;
                    e <= '0';
                else
                    case state is
                        when initial_lcd =>
                            lcd_rs <= '0';   ---command signal
                            if e <= '0' then
                                lcd_data <= datas(j)(7 downto 0);
                                e <= '1';
                            else
                                e <= '0';
                                j := j+1;
                            end if;
                            if j > 4 then
                                state <= set_cursor;
                            end if;
                        when set_cursor =>
                            lcd_rs <= '0';   ---command signal
                            if e = '0' then
                                lcd_data <= "10000000";        --set cursor to beginning of the 1st line
                                e <= '1';
                            else
                                e <= '0';
                                j := 0;
                                state <= lcd_print;
                            end if;
                        when lcd_print =>
                            lcd_rs <= '1';   ---data signal
                            if e = '0' then
                                address <= j;
                                lcd_data <= data_in;
                                e <= '1';
                            else
                                e <= '0';
                                j := j+1;
                            end if;
                            if j > 15 then
                                state <= set_cursor;
```

```
                        end if;
                    end case;
                end if;
            end if;
        end process;
        lcd_e <= e;
end Behavioral;
```

ตัวอย่าง LCD_EX1

```
----------------------------------------------------------------------------------

-- Create Date:    17:04:11 11/29/2015
-- Design Name:    Narong Buabthong
-- Module Name:    lcd_ex1 - Behavioral
----------------------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use ieee.numeric_std.all;

entity lcd_ex1 is
port ( clk : in std_logic;   -- clock i/p
        lcd_rst: in std_logic;  -- Reset i/p
        lcd_rw : out std_logic;   -- read&write control
        lcd_e  : out std_logic;   -- enable control
        lcd_rs : out std_logic;   -- data or command control
        lcd_data   : out std_logic_vector(7 downto 0));  ---data line
end lcd_ex1;

architecture Behavioral of lcd_ex1 is

  signal addr_d :integer range 0 to 31;
  signal addr_q :integer range 0 to 31;
  signal data : std_logic_vector (7 downto 0);
  begin

  message_rom :  entity work.message_rom
         port map (
            clk =>clk,
            A  => addr_q,
            D  => data
            );

  lcd_control:entity work.lcd_control
        port map (
            clk  => clk,
            lcd_rst => lcd_rst,
            data_in => data,
            address => addr_d,
            lcd_rw  => lcd_rw,
            lcd_e   => lcd_e,
            lcd_rs  => lcd_rs,
            lcd_data => lcd_data
        );
process (clk)
  begin
    if clk = '1' and clk'event then
```

```
            addr_q <= addr_d;
    end if;
end process;

end Behavioral;
```