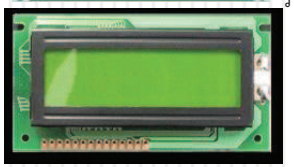
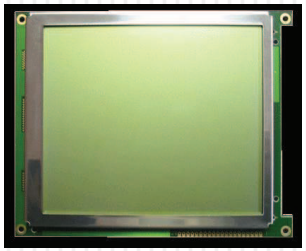
AVR และ ARDUINO
กับการเชื่อมต่อกับ LCD แบบ Dot Matrix

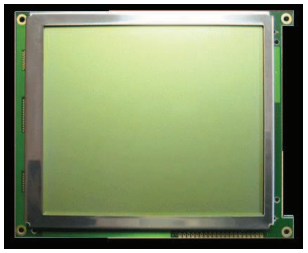


โดย
รศ.ณรงค์ บวบทอง
ภาควิชาวิศวกรรมไฟฟ้าและคอมพิวเตอร์
คณะวิศวกรรมศาสตร์
มหาวิทยาลัยธรรมศาสตร์

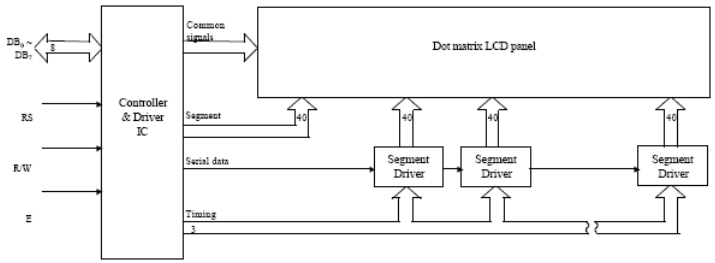


Dot Matrix LCD Module

- Character LCD Module
- Graphic LCD Module
- Segment Display LCD Module

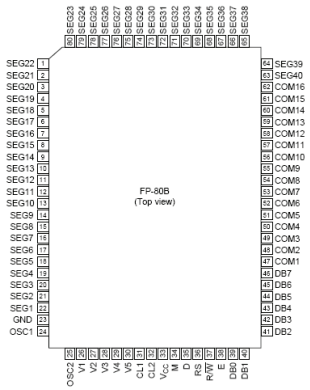


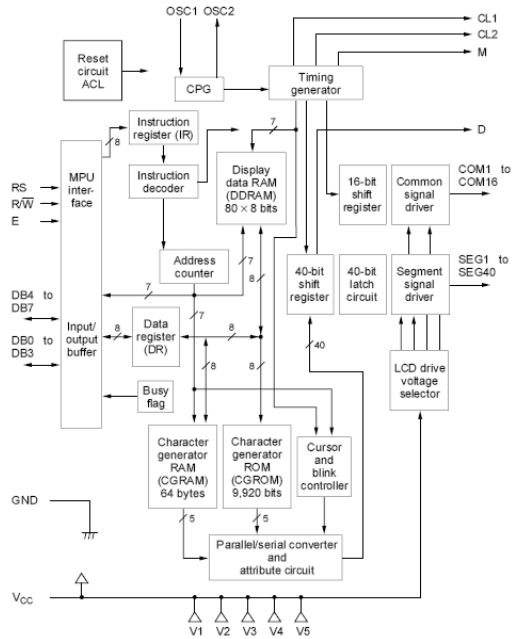
ส่วนประกอบของ LCD Module



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

Block diagram of HD44780

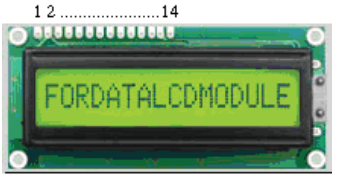




คุณสมบัติพอสังเขป

5 x 8 and 5 x 10 dot matrix possible
Low power operation support:
2.7 to 5.5V
Wide range of liquid crystal display driver power
3.0 to 11V
Liquid crystal drive waveform
A (One line frequency AC waveform)
Correspond to high speed MPU bus interface
2 MHz (when VCC = 5V)
4-bit or 8-bit MPU interface enabled
80 ′ 8-bit display RAM (80 characters max.)
9,920-bit character generator ROM for a total of 240 character fonts
208 character fonts (5 ′ 8 dot)
32 character fonts (5 ′ 10 dot)
64 x 8-bit character generator RAM
8 character fonts (5 ′ 8 dot)
4 character fonts (5 ′ 10 dot)
16-common ′ 40-segment liquid crystal display driver
Programmable duty cycles
1/8 for one line of 5 ′ 8 dots with cursor
1/11 for one line of 5 ′ 10 dots with cursor
1/16 for two lines of 5 ′ 8 dots with cursor
Wide range of instruction functions:
Display clear, cursor home, display on/off, cursor on/off, display character blink, cursor shift, display shift
Pin function compatibility with HD44780S
Automatic reset circuit that initializes the controller/driver after power on
Internal oscillator with external resistors
Low power consumption

ขาของ LCD Module

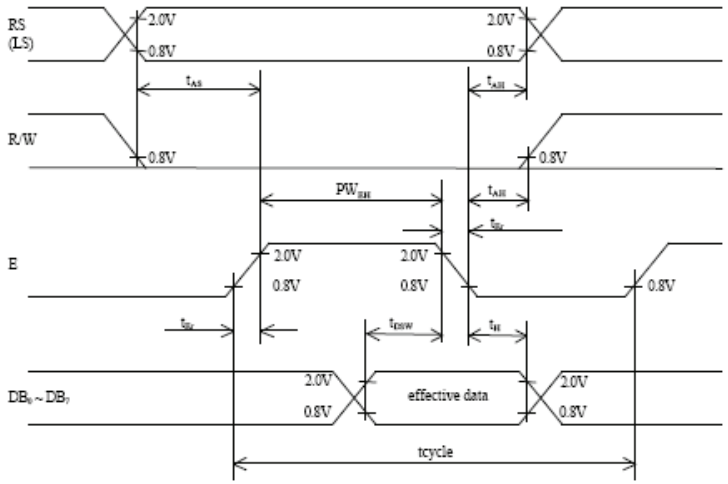


ขาที่	สัญญาณ	รายละเอียด
1	Vss	0 V Gnd
2	Vcc	+ 5V
3	Vee	ใช้ปรับความสว่างของ LCD ถ้าต่อลงดินจะสว่างที่สุด
4	RS	สัญญาณ Register Select ใช้เลือกรีจิสเตอร์ควบคุม หรือหน่วยความจำแสดงผล - ถ้าเป็น "0" แสดงว่าต้องการติดต่อกับรีจิสเตอร์ควบคุม - ถ้าเป็น "1" แสดงว่าต้องการติดต่อกับริจิสเตอร์แสดงผล
5	R/W	สัญญาณควบคุมการอ่าน/เขียน ถ้าเป็น "0" " แสดงว่าต้องการเขียนหรือส่งข้อมูลให้แก่โมดูล ถ้าเป็น "1" แสดงว่าต้องการอ่านข้อมูลจากโมดูล

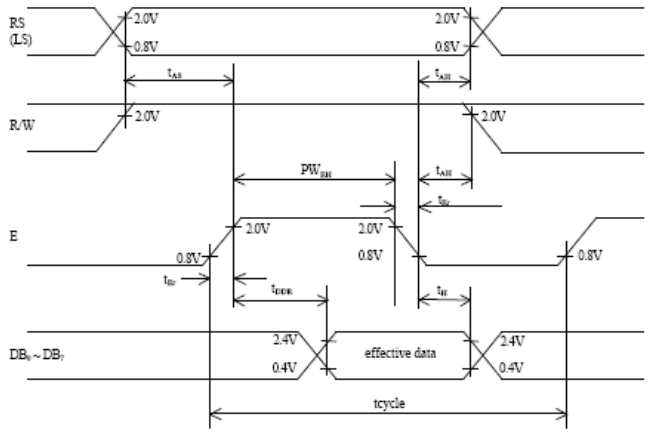
ขาของ LCD Module

ขาที่	สัญญาณ	รายละเอียด
6	E	Enable - สัญญานสั่งให้เริ่มต้นการทำงาน สำหรับการอ่าน/เขียนข้อมูล การรับส่งข้อมูลจะเกิดเมื่อเป็น '1' และขอบขาลง
7 ~ 0	DB0 ~ DB3	เป็นบัสแบบสองทิศทางใช้สำหรับส่งถ่ายข้อมูลระหว่างซีพียูกับโมดูลในกรณีทีการ ทำงานเป็นแบบ 4 บิต บัสนี้ไม่ได้ใช้และควรต่อลงดินด้วย แต่ถ้าเป็น การทำงานแบบ 8 บิต บัสนี้จะเป็น 4 บิตต่ำ ใช้เพื่อการส่งถ่ายข้อมูล
11~ 14	DB4 ~ DB7	เป็นบัสแบบสองทิศทางใช้สำหรับส่งถ่ายข้อมูลระหว่างซีพียูกับโมดูลในกรณีทีการทำงานเป็นแบบ 4 บิต จะใช้บัสนี้ส่งถ่ายข้อมูล แต่ถ้าเป็นการทำงานแบบ 8 บิต บัสนี้จะเป็น 4 บิตสูง นอกจากนี้ DB7 ยังใช้เป็นบิตแสดงสถานะBusy ด้วย

การเขียนข้อมูลให้โมดูล



การอ่านข้อมูลจากโมดูล



RS:
0=Command
1=Data

คำสั่งควบคุม

- DDRAM (Display Data Ram) คือหน่วยความจำภายในตัวโมดูลแอลซีดีที่เป็นบัฟเฟอร์ของข้อมูล ถ้าเขียนรหัส ASCII ใดๆ ลงในหน่วยความจำนี้ก็จะปรากฏเป็นตัวอักษรที่จอแสดงผลทันที
- CGRAM (Character Generator Ram) เป็นหน่วยความจำภายในตัวโมดูล ใช้สำหรับเก็บภาพตัวอักษรที่ผู้ใช้สร้างขึ้นเอง (สร้างได้ 8 ตัว) โดยอ้างตำหน่งได้ 64 ไบต์ (8 ตัวอักษรคูณด้วย 8 แถว)
- การเขียนข้อมูลให้โมดูลแต่ละครั้งต้องตรวจสอบความพร้อมของโมดูล โดยตรวจสอบได้จาก Busy Flag หรือระยะเวลาการทำงานของโมดูล ซึ่งดูได้จากตารางคำสั่ง ดังนั้นเมื่อเขียนข้อมูลหนึ่งๆไปเราต้องหน่วงเวลารอให้โมดูลพร้อม จึงจะเขียนชุดใหม่ต่อไป
- การเขียนข้อมูลให้โมดูลสามารถทำได้ทั้งแบบ 4 บิตและแบบ 8 บิตขึ้นอยู่กับการเชื่อมต่อกับโมดูล ถ้าเป็นการเชื่อมแบบ 4 บิต การเขียนข้อมูลหรืออ่านข้อมูลต้องทำ 2 ครั้งโดยครั้งแรกต้องเป็นบิต 4 ถึงบิต 7 และครั้งที่ 2 จะเป็นบิตที่ 0 ถึง 3

ตารางคำสั่ง

INSTRUCTION	RS	R/W	DATA BIT								EXE. TIME (MicroS)
			7	6	5	4	3	2	1	0	
CLEAR DISPLAY	0	0	0	0	0	0	0	0	0	1	1640
CURSOR AT HOME	0	0	0	0	0	0	0	0	1	*	1640
ENTRY MODE SET	0	0	0	0	0	0	0	1	I/D	S	40
DISPLAY ON/OFF	0	0	0	0	0	0	1	D	C	B	40
DISPLAY SHIFT	0	0	0	0	0	1	S/C	R/L	*	*	40
FUNCTION SET	0	0	0	0	1	DL	N	F	*	*	40
SET CGRAM ADD.	0	0	0	1	CGRAM ADDRESS						40
SET DDRAM ADD.	0	0	1	DDRAM ADDRESS							40
BUSY,ADD. READ	0	1	BF	ADDRESS							0
CGRAM,DDRAM WR	1	0	WRITE DATA								40
CGRAM,DDRAM RD	1	1	READ DATA								40

คำสั่ง Entry Mode Set

RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	0	0	0	0	0	0	1	I/D	S

I/D=0 กำหนดทิศทางของ Cursor และ DDRAM ให้เป็นแบบ Decrement
I/D=1 กำหนดทิศทางของ Cursor และ DDRAM ให้เป็นแบบ Increment
S=0 เมื่อเขียนข้อมูลแล้ว ตัวCursorจะถูกเลื่อนไปตามทิศทางของค่า I/D
S=1 เมื่อเขียนข้อมูลแล้วตัว Cursor จะอยู่กับที่และตัวอักษรจะถูกดันไปตามทิศทางของค่า I/D

4.DISPLAY ON/OFF

D=0 กำหนดให้ Off Display
D=1 กำหนดให้ On Display
C=0 กำหนดให้ Off Cursor
C=1 กำหนดให้ On Cursor โดย Cursor จะเป็นเส้นขีดใต้ตัวอักษร
B=0 กำหนดให้ไม่มีการกระพริบที่ตำแหน่ง Cursor
B=1 กำหนดให้มีการกระพริบที่ตำแหน่ง Cursor (กระพริบเป็นรูป □)

RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	0	0	0	0	0	1	D	C	B

5.DISPLAY SHIFT

RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	0	0	0	0	1	S/C	R/L	*	*

S/C=0 กำหนดให้เลื่อน Cursor ตามทิศทาง R/L ไป 1 ตำแหน่ง
S/C=1 กำหนดให้เลื่อนข้อความบนแผงแสดงตามทิศทาง R/L ไป 1 Column (เลื่อนทุกบรรทัด)
R/L=0 กำหนดให้มีทิศทางไปทางซ้าย R/L=1 กำหนดให้มีทิศทางไปทางขวา

6.FUNCTION SET

DL=0 กำหนดให้การติดต่อกับ LCD Module เป็นแบบ 4 bit
DL=1 กำหนดให้การติดต่อกับ LCD Module เป็นแบบ 8 bit จะสังเกตว่าการกำหนดค่า D/L นี้สามารถกระทำได้ที่ DB4-DB7 ซึ่งถ้ามีการกำหนดให้เป็นแบบ 4 bit ตั้งแต่ครั้งแรกหลังจากจ่ายไฟเลี้ยงก็จะทำให้ LCD Module มีการรับข้อมูลแบบ 4 bit ทันที
N=0 กำหนดจำนวนบรรทัดแบบ 1/8 Duty และ 1/11 Duty
N=1 กำหนดจำนวนบรรทัดแบบ 1/16 Duty
F=0 กำหนดให้ตัวอักษรเป็นแบบ 5*7 Dots
F=1 กำหนดให้ตัวอักษรเป็นแบบ 5*10 Dots
(กรณีที่ LCD Module เป็นแบบ 5*7 อยู่แล้วก็จะไม่มีผลอะไร)

RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	0	0	0	1	DL	N	F	*	*

7. SET CGRAM ADDRESS

RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	0	0	1	CGRAM ADDRESS					

สำหรับการกำหนด Address ของ CGRAM เมื่อได้ทำการกำหนดไว้แล้วการอ่านและเขียน Data ที่ต่อจากนี้จะเป็นไปตามAddressที่กำหนดทันที

8. SET DDRAM ADDRESS

RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	0	1	DDRAM ADDRESS						

สำหรับการกำหนด Address ของ DDRAM เมื่อได้ทำการกำหนดไว้แล้วการอ่านและเขียน Data ที่ต่อจากนี้จะเป็นไปตามAddressที่กำหนดทันที ตำแหน่งของ Address ในแต่ละรุ่นจะมีความแตกต่างกันบ้าง เพราะจำนวนตัวอักษรต่อบรรทัดไม่เท่ากัน ซึ่งแสดงดังตารางต่อไปนี้(ตารางนี้จะกำหนดให้บิทที่ 7 เท่ากับ 1 เสมอเพื่อความสะดวกในการเรียกใช้)

รุ่น DMC 082

80	81	82	83	84	85	86	87
C0	C1	C2	C3	C4	C5	C6	C7

รุ่น DMC 202

80	81	82	83	84	85	86	87	88	89	8A	8B	8C	8D	8E	8F	90	91	92	93
C0	C1	C2	C3	C4	C5	C6	C7	C8	C9	CA	CB	CC	CD	CE	CF	D0	D1	D2	D3

รุ่น DMC 204

80	81	82	83	84	85	86	87	88	89	8A	8B	8C	8D	8E	8F	90	91	92	93
C0	C1	C2	C3	C4	C5	C6	C7	C8	C9	CA	CB	CC	CD	CE	CF	D0	D1	D2	D3
94	95	96	97	98	99	9A	9B	9C	9D	9E	9F	A0	A1	A2	A3	A4	A5	A6	A7
D4	D5	D6	D7	D8	D9	DA	DB	DC	DD	DE	DF	E0	E1	E2	E3	E4	E5	E6	E7

รุ่น DMC 161

80	81	82	83	84	85	86	87	C0	C1	C2	C3	C4	C5	C6	C7

รุ่น DMC 162

80	81	82	83	84	85	86	87	88	89	8A	8B	8C	8D	8E	8F
C0	C1	C2	C3	C4	C5	C6	C7	C8	C9	CA	CB	CC	CD	CE	CF

รุ่น DMC 164

80	81	82	83	84	85	86	87	88	89	8A	8B	8C	8D	8E	8F
C0	C1	C2	C3	C4	C5	C6	C7	C8	C9	CA	CB	CC	CD	CE	CF
90	91	92	93	94	95	96	97	98	99	9A	9B	9C	9D	9E	9F
D0	D1	D2	D3	D4	D5	D6	D7	D8	D9	DA	DB	DC	DD	DE	DF

ตารางคำสั่งที่ใช้บ่อยๆ

No.	Instruction	Hex	Decimal
1	Function Set: 8-bit, 1 Line, 5x7 Dots	0x30	48
2	Function Set: 8-bit, 2 Line, 5x7 Dots	0x38	56
3	Function Set: 4-bit, 1 Line, 5x7 Dots	0x20	32
4	Function Set: 4-bit, 2 Line, 5x7 Dots	0x28	40
5	Entry Mode	0x06	6
6	Display off Cursor off (clearing display without clearing DDRAM content)	0x08	8
7	Display on Cursor on	0x0E	14
8	Display on Cursor off	0x0C	12
9	Display on Cursor blinking	0x0F	15
10	Shift entire display left	0x18	24
11	Shift entire display right	0x1C	30
12	Move cursor left by one character	0x10	16
13	Move cursor right by one character	0x14	20
14	Clear Display (also clear DDRAM content)	0x01	1
15	Set DDRAM address or coursor position on display	0x80 + address*	128 + address*
16	Set CGRAM address or set pointer to CGRAM location	0x40 + address**	64 + address**

ตัวอย่างการเขียนโปรแกรมจาก

http://www.mikrocontroller.net/articles/AVR-GCC-Tutorial/LCD-Ansteuerung

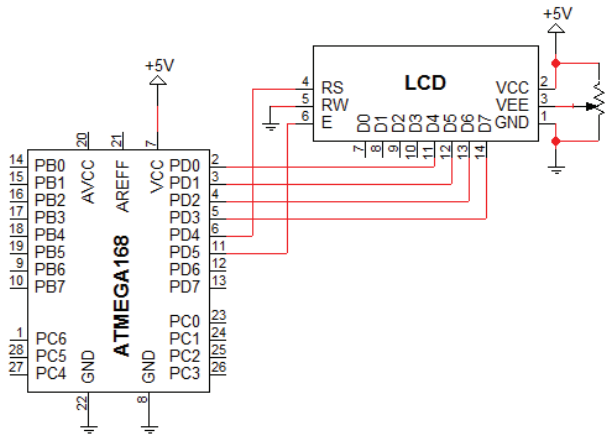
ตัวอย่างฟังก์ชั่น

```
// Sends a command to the LCD
void lcd_command (  uint8_t data )
{
   LCD_PORT & = ~ ( 1 << LCD_RS ) ;  // Set the RS to 0

   lcd_out ( data ) ;                // first the upper
   lcd_out ( data << 4  ) ;          // then send the lower 4 bits

   _delay_us ( LCD_COMMAND_US ) ;
}
```



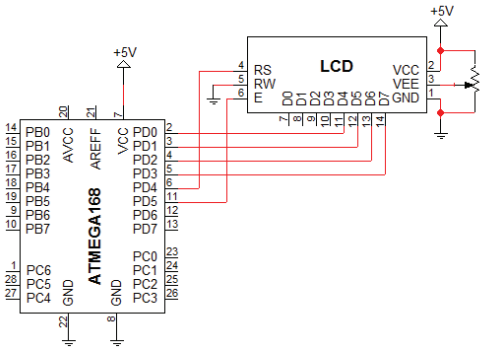
```
// Sends a 4-bit output operation to the LCD
static  void lcd_out (  uint8_t data )
{
   data & =  0xF0 ;                        // mask upper 4 bits

   LCD_PORT & = ~ ( 0xF0 >> ( 4 - LCD_DB ) ) ;     // mask
   delete
   LCD_PORT | =  ( data >> ( 4 - LCD_DB ) ) ;      // bits set
   lcd_enable ( ) ;
}

// Creates an enable pulse
static  void lcd_enable (  void  )
{
   LCD_PORT | =  ( 1 < < LCD_EN ) ;      // Enable to 1.
   _delay_us ( LCD_ENABLE_US ) ;   // short break
   LCD_PORT & = ~ ( 1 << LCD_EN ) ;     // Enable to 0
}
```


ตัวอย่างการต่อ LCD กับ ATMEGA168



คำสั่งจำลองการทำงานของ LCD บน VMLAB

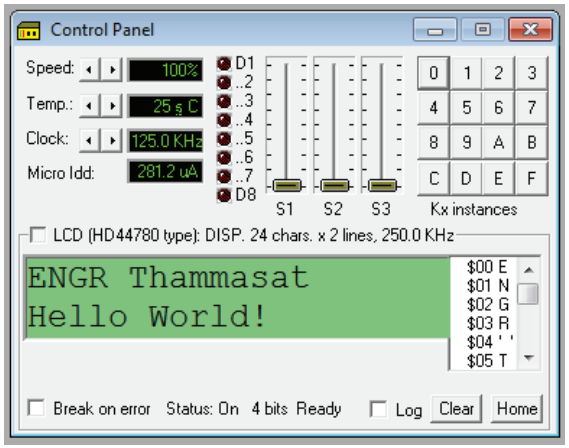
```
.SOURCE    "lcd_r.c" "lcd_4.c"

;X[inst_name]  LCD(chars lines oscil_freq) RS RW E D7 D6 D5 D4 D3 D2 D1 D0
Xdisp LCD(24 2 250K) PD4  VSS  PD5   PD3 PD2 PD1 PD0   nc3 nc2 nc1 nc0
```



ตัวอย่างโปรแกรม

```
#include <avr\io.h>
#include <avr\signal.h>
#include <avr/io.h>
#include "lcd_r.h"    //Add LCD routine

int main(void)
{
  lcd_init();  //LCD Initialization

  // Print character
 lcd_data( 'E' );
 lcd_data( 'N' );
 lcd_data( 'G' );
 lcd_data( 'R' );
 lcd_data( ' ' );
 lcd_data( 'T' );
 lcd_data( 'h' );
 lcd_data( 'a' );
 lcd_data( 'm' );
 lcd_data( 'm' );
 lcd_data( 'a' );
 lcd_data( 's' );
 lcd_data( 'a' );
 lcd_data( 't' );

 lcd_setcursor( 0, 2 ); //  Set cursor

 // Print string
 lcd_string("Hello World!");

 while(1)
 {
 }
}
```


หน้าจอ Control Panel แสดงการทำงานของ LCD



LCD Library สำหรับ Arduino

- LiquidCrystal()
- begin()
- clear()
- home()
- setCursor()
- write()
- print()
- cursor()
- noCursor()
- blink()
- noBlink()
- display()
- noDisplay()
- scrollDisplayLeft()
- scrollDisplayRight()
- autoscroll()
- noAutoscroll()
- leftToRight()
- rightToLeft()
- createChar()

รายละเอียดดูได้จาก **http://arduino.cc/en/Reference/LiquidCrystal**

LiquidCrystal()

Description

Creates a variable of type LiquidCrystal. The display can be controlled using 4 or 8 data lines. If the former, omit the pin numbers for d0 to d3 and leave those lines unconnected. The RW pin can be tied to ground instead of connected to a pin on the Arduino; if so, omit it from this function's parameters.

Syntax

LiquidCrystal(rs, enable, d4, d5, d6, d7)

LiquidCrystal(rs, rw, enable, d4, d5, d6, d7)

LiquidCrystal(rs, enable, d0, d1, d2, d3, d4, d5, d6, d7)

LiquidCrystal(rs, rw, enable, d0, d1, d2, d3, d4, d5, d6, d7)

LiquidCrystal()

Parameters

rs: the number of the Arduino pin that is connected to the RS pin on the LCD

rw: the number of the Arduino pin that is connected to the RW pin on the LCD (optional)

enable: the number of the Arduino pin that is connected to the enable pin on the LCD

d0, d1, d2, d3, d4, d5, d6, d7: the numbers of the Arduino pins that are connected to the corresponding data pins on the LCD. d0, d1, d2, and d3 are optional; if omitted, the LCD will be controlled using only the four data lines (d4, d5, d6, d7).

begin()

Description

Specifies the dimensions (width and height) of the display.

Syntax

lcd.begin(cols, rows)

Parameters

lcd: a variable of type LiquidCrystal

cols: the number of columns that the display has

rows: the number of rows that the display has

clear()

Description

Clears the LCD screen and positions the cursor in the upper-left corner.

Syntax

lcd.clear()

Parameters

lcd: a variable of type LiquidCrystal

home

Description

Positions the cursor in the upper-left of the LCD. That is, use that location in outputting subsequent text to the display. To also clear the display, use the clear() function instead.

Syntax

lcd.home()

Parameters

lcd: a variable of type LiquidCrystal

setCursor()

Description

Position the LCD cursor; that is, set the location at which subsequent text written to the LCD will be displayed.

Syntax

lcd.setCursor(col, row)

Parameters

lcd: a variable of type LiquidCrystal

col: the column at which to position the cursor (with 0 being the first column)

row: the row at which to position the cursor (with 0 being the first row)

write()

Description

Write a character to the LCD.

Syntax

lcd.write(data)

Parameters

lcd: a variable of type LiquidCrystal

data: the character to write to the display

Returns

byte

write() will return the number of bytes written, though reading that number is optional

printf

Description

Prints text to the LCD.

Syntax

lcd.print(data)

lcd.print(data, BASE)

Parameters

lcd: a variable of type LiquidCrystal

data: the data to print (char, byte, int, long, or string)

BASE (optional): the base in which to print numbers: BIN for binary (base 2), DEC for decimal (base 10), OCT for octal (base 8), HEX for hexadecimal (base 16).

Returns

byte

print() will return the number of bytes written, though reading that number is optional

cursor()

Description

Display the LCD cursor: an underscore (line) at the position to which the next character will be written.

Syntax

lcd.cursor()

Parameters

lcd: a variable of type LiquidCrystal

noCursor()

Description

Hides the LCD cursor.

Syntax

lcd.noCursor()

Parameters

lcd: a variable of type LiquidCrystal

blink()

Description

Display the blinking LCD cursor. If used in combination with the cursor() function, the result will depend on the particular display.

Syntax

lcd.blink()

Parameters

lcd: a variable of type LiquidCrystal

noBlink()

Description

Turns off the blinking LCD cursor.

Syntax

lcd.noBlink()

Parameters

lcd: a variable of type LiquidCrystal

display()

Description

Turns on the LCD display, after it's been turned off with noDisplay(). This will restore the text (and cursor) that was on the display.

Syntax

lcd.display()

Parameters

lcd: a variable of type LiquidCrystal

noDisplay()

Description

Turns off the LCD display, without losing the text currently shown on it.

Syntax

lcd.noDisplay()

Parameters

lcd: a variable of type LiquidCrystal

scrollDisplayLeft()

Description

Scrolls the contents of the display (text and cursor) one space to the left.

Syntax

lcd.scrollDisplayLeft()

Parameters

lcd: a variable of type LiquidCrystal

scrollDisplayRight()

Description

Scrolls the contents of the display (text and cursor) one space to the right.

Syntax

lcd.scrollDisplayRight()

Parameters

lcd: a variable of type LiquidCrystal

autoscroll()

Description

Turns on automatic scrolling of the LCD. This causes each character output to the display to push previous characters over by one space. If the current text direction is left-to-right (the default), the display scrolls to the left; if the current direction is right-to-left, the display scrolls to the right. This has the effect of outputting each new character to the same location on the LCD.

Syntax

lcd.autoscroll()

Parameters

lcd: a variable of type LiquidCrystal

noAutoscroll()

Description

Turns off automatic scrolling of the LCD.

Syntax

lcd.noAutoscroll()

Parameters

lcd: a variable of type LiquidCrystal

leftToRight()

Description

Set the direction for text written to the LCD to left-to-right, the default. This means that subsequent characters written to the display will go from left to right, but does not affect previously-output text.

Syntax

lcd.leftToRight()

Parameters

lcd: a variable of type LiquidCrystal

rightToLeft()

Description

Set the direction for text written to the LCD to right-to-left (the default is left-to-right). This means that subsequent characters written to the display will go from right to left, but does not affect previously-output text.

Syntax

lcd.rightToLeft()

Parameters

lcd: a variable of type LiquidCrystal

createChar()

Description

Create a custom character (gylph) for use on the LCD. Up to eight characters of 5x8 pixels are supported (numbered 0 to 7). The appearance of each custom character is specified by an array of eight bytes, one for each row. The five least significant bits of each byte determine the pixels in that row. To display a custom character on the screen, write() its number.

Syntax

lcd.createChar(num, data)

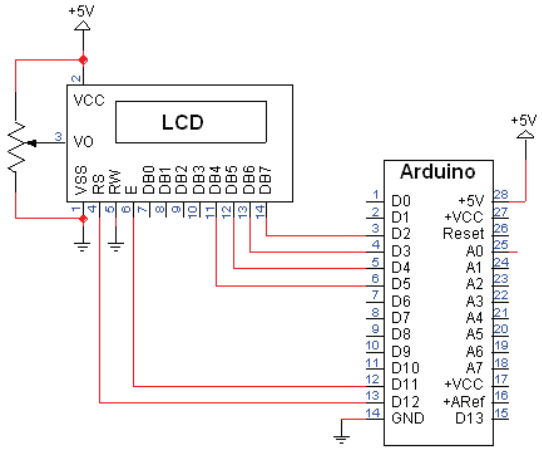
Parameters

lcd: a variable of type LiquidCrystal

num: which character to create (0 to 7)

data: the character's pixel data

ตัวอย่างการต่อวงจร LCD แบบ 4 บิต กับบอร์ด ARDUINO

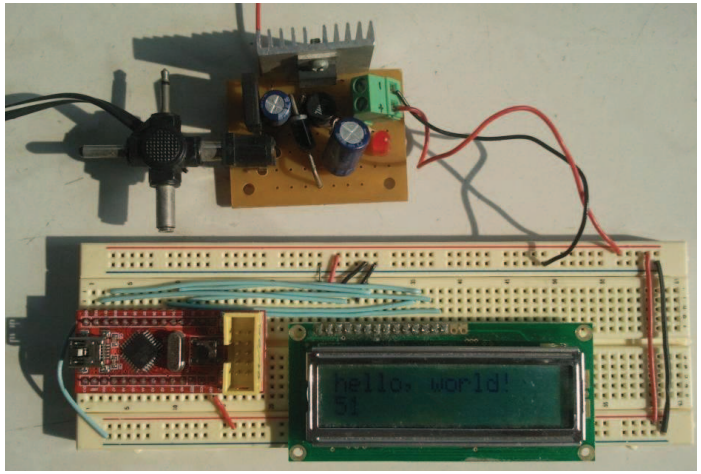


ข้อควรระวัง1. แรงดัน +5V ที่จ่ายเข้าขา 28 ของบอร์ด Arduino อย่าให้เกิน

2. ขา 3 ของ LCD สามารถต่อลง GND ได้เลยจะสว่างที่สุด ไม่ต้องใช้ R ปรับค่า

ตัวอย่างโปรแกรม Hello world

```
#include <LiquidCrystal.h>

LiquidCrystal lcd(12, 11, 5, 4, 3, 2); // initialize the library with the numbers of the interface pins

void setup() {
 lcd.begin(16, 2);                    // set up the LCD's number of columns and rows:
 lcd.print("hello, world!");           // Print a message to the LCD.
}

void loop() {
  lcd.setCursor(0, 1);                // set the cursor to column 0, line 1
                                      // (note: line 1 is the second row, since counting begins with 0):
 lcd.print(millis()/1000);            // print the number of seconds since reset:
}
```

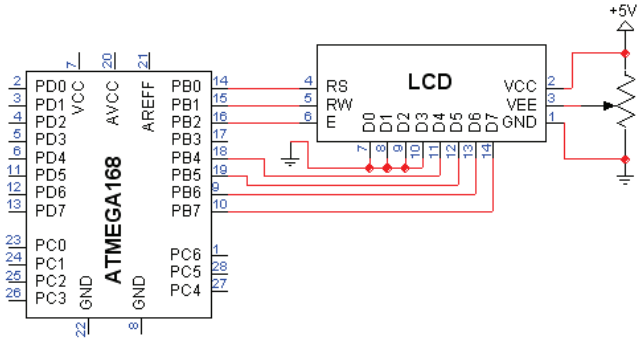



ตัวอย่างโปรแกรมเพิ่มเติม

- https://sites.google.com/site/eplearn/arduino-project/02_arduino_lcd

ตัวอย่างการต่อวงจร LCD แบบ 4 บิต กับ ATmega 8

ประมาณ 0.75 โวลท์

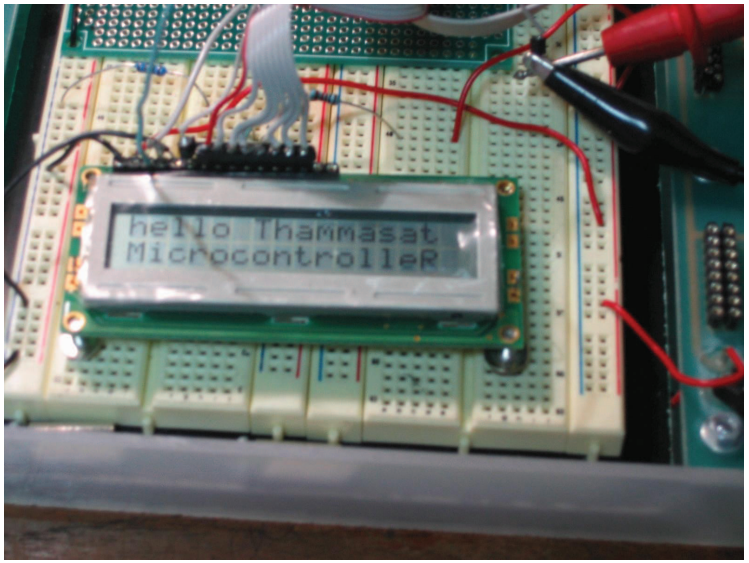


ตัวอย่างโปรแกรม

```
// ------- Includes ------//
#include <avr/io.h>             // include I/O definitions (port names, pin
names, etc)
#include <avr/signal.h>         // include "signal" names (interrupt names)
#include <avr/interrupt.h>      // include interrupt support

//-------- Procyon AVRlib -------//
#include "timer.c"              // include Timer function library
#include "lcd.c"                // include LCD function library
```



```
//----- Main Functions -----//
int main(void)
{
        unsigned int maxprogress=500;
        unsigned int progress=0;
        unsigned char length=16;
        unsigned char set_updown=1;
        // initialize all timers
        timerInit();
        // initializes the LCD display
        lcdInit();
        lcdGotoXY(0,0);
        lcdPrintData("+-- LCD Demo --+",16);

  while (1) {           // Loop forever
                lcdGotoXY(0,1);
                lcdProgressBar(progress, maxprogress, length);
                timerPause(100);            // Delay 100ms
                if (progress >= maxprogress) set_updown = 0;
                if (progress <= 0) set_updown = 1;
                if (set_updown)
                        progress +=10;
                else
                        progress -=10;
        }
        return 0;
}
```

```
#include <stdio.h>

int sprintf (char *buffer, /* storage buffer */ const char *fmtstr, /* format
string */ [, argument]...); /* additional arguments */


n = sprintf (buf,"%2.2f",volt);
```


การแปลงจำนวนให้เป็นตัวอักษร

```
#include <stdio.h>

int sprintf (char *buffer, /* storage buffer */ const char *fmtstr, /*
                format string */ [, argument]...); /* additional arguments */
```

ตัวอย่าง

```
n = sprintf (buf,"%2.2f",volt);
```

- ตัวแปรตัวเลขที่ต้องการแปลง
- รูปแบบตัวเลขที่ต้องการให้แสดง
- ตัวแปรอะเรย์ที่ใช้เก็บตัวอักษร
- จำนวนตัวอักษรที่แปลงได้

อ้างอิง

1. http://www.mikrocontroller.net/articles/AVR-GCC-Tutorial/LCD-Ansteuerung
2. http://arduino.cc/en/Tutorial/LiquidCrystal