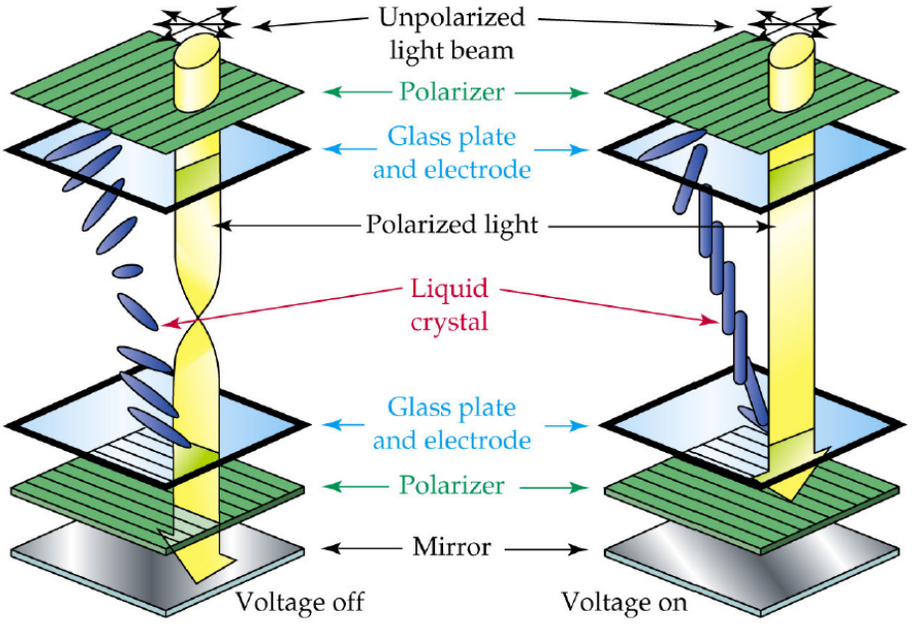
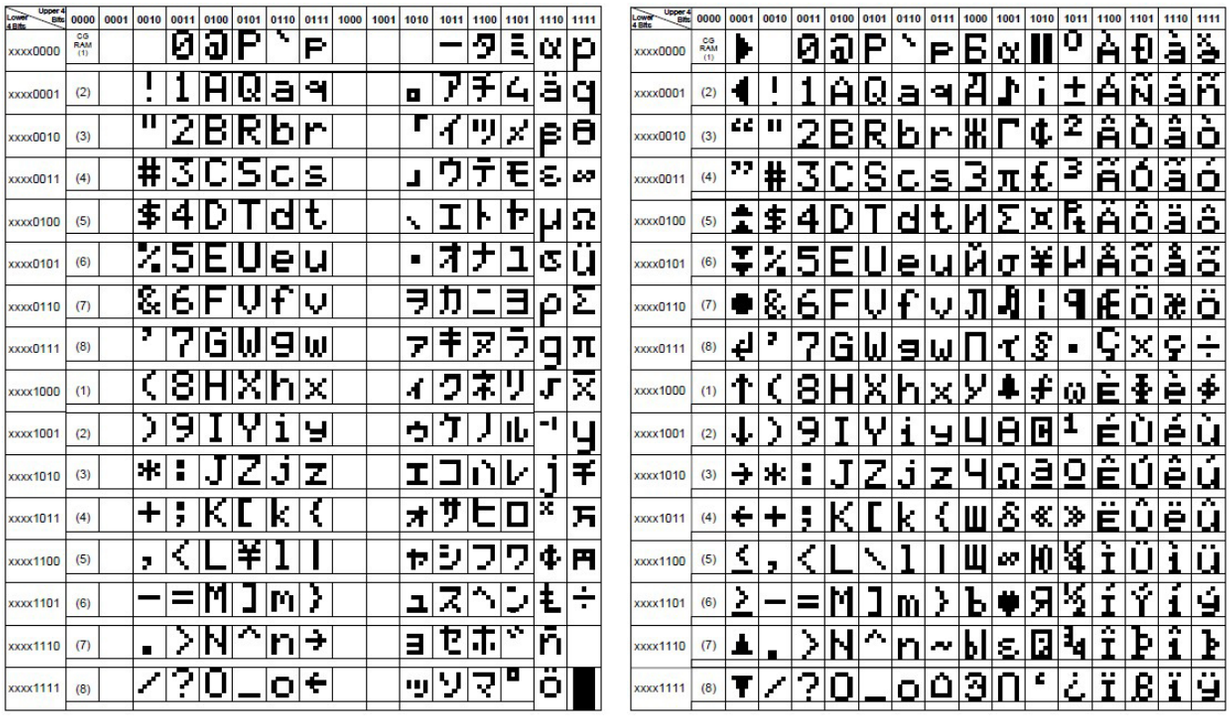
การแสดงผลตัวอักษรโดยใช้ Liquid Crystal Display หรือจอภาพ LCD หลักการทำงานจะมีการส่งแสงไฟที่ฉากด้านหลังจอด้วย LED Backlight โดยที่แสงจะเป็นคลื่นแม่เหล็กไฟฟ้า ประกอบด้วยสนามแม่เหล็กและไฟฟ้าที่สั่นตั้งฉากกันอยู่บนระนาบที่ตั้งฉากกับทิศทางการแผ่ของคลื่น และเป็นคลื่นตามขวางโดยใช้ปรากฏการณ์โพลาไรเซชัน แสงโพลาไรซ์สามารถผลิตได้โดยการส่งผ่านโพลาไรเซอร์ แสงจะถูกส่งออกมาทางด้านหน้าผ่านทางของเหลวที่เป็นฉาก แสงโพลาไรซ์ที่ประกอบด้วยสนามไฟฟ้า ซึ่งสั่นในแนวใดแนวหนึ่ง จะผ่านฉากของจอ LCD ที่ควบควบคุมด้วยแรงดันไฟฟ้าเข้าไประหว่างผลึกของสารเหลวซึ่งมีคุณสมบัติการบิดแกนโพลาไรซ์ของแสง ทำให้ดัชนีการหักเหของแสงไม่เท่ากันเกิดการเคลื่อนที่ของแสงด้วยความเร็วไม่เท่ากันในแต่ละทิศทางของผลึกเกิดการหักเหหมุนบิดตัวให้อยู่ในแนวตรงกันข้าม เป็นการเปลี่ยนโพลาไรซ์ของแสง เมื่อผ่านตัวกรองโพลาไรซ์ที่จะทำให้คลื่นแสงของโพลาไรซ์เพียงแนวเดียวผ่านได้ ที่เป็นโพลาไรเซอร์แนวตรงข้ามกัน เช่น แนวดิ่งกับแนวราบ จึงทำให้แสงสามารถลอดผ่านออกมาได้เกิดเห็นเป็นสีขาว และถ้าควบคุมแรงดันไฟฟ้าไม่ให้มีการบิดแกนโพลาไรซ์แสงจะทะลุผ่านออกมาไม่ได้จึงเห็นเป็นสีดำ



จอแสดงผล LCD ที่ใช้ในการทดลองจะมีอิเล็กโทรดที่บังคับการทำงานของผนึกเหลวนี้ มาประกอบกันเป็นเซลล์เล็กๆที่มีความละเอียดในรูปแบบ Dot Matrix ทำให้สามารถควบคุมสร้างให้เป็นตัวอักษรได้ โดยมีขนาดของตัวอักษรเป็นแบบ Matrix ประกอบด้วยความกว้าง 5 pixel กับสูง 8 pixel มีจำนวนตัวอักษรแถวละ 16 ตัว และมีจำนวนบรรทัดทั้งหมด 2 แถว ภายในวงจรที่ใช้ควบคุมจะใช้ไอซี HD44780 เป็นไอซีที่ทำหน้าที่ Dot Matrix Liquid Crystal Display Controller/Driver ถูกพัฒนาโดยบริษัท Hitachi เพื่อใช้ในการควบคุมการแสดงผลตัวอักษรแบบ Dot Matrix บนจอ LCD โดยที่ LCD จะมีการแสดงข้อความเป็นแบบขาวดำ โดยมากมักจะถูกนำมาใช้ในเครื่องถ่ายเอกสาร เครื่องรับโทรศัพท์ เครื่องแฟกซ์ เครื่องพิมพ์เลเซอร์ อุปกรณ์ทดสอบทางอุตสาหกรรม อุปกรณ์เครือข่าย และอุปกรณ์จัดเก็บข้อมูล

ชุดของตัวอักษรในไอซี HD44780 จะมีใช้โดยทั่งไปได้ 2 แบบ คือ แบบเอเชียและแบบยุโรป โดยที่เป็นชุดตัวอักษรหนึ่งในสองชุดที่แตกต่างกัน ดังนี้



HD44780 สามารถนำไปใช้ควบคุมจอ LCD โดยมีโหมดการเชื่อมต่อได้ 2 แบบคือ โหมด 4 Bit (DB4 - DB7) ใช้สายทั้งหมด 8 เส้น และโหมด 8 Bit (DB0 - DB7) ใช้สายทั้งหมด 12 เส้น โมดูลอะแดปเตอร์ของหน้าจอLCD ที่ใช้ทั่วไปจะนิยมแบบการเชื่อมต่อโหมด 4 บิต การเชื่อมต่อโหมดแบบ 4 บิตนี้จะต้องใช้ส่งข้อมูลไปทั้งหมด 2 ครั้ง เพื่อให้ได้ข้อมูลครบ 8 บิต โดยการส่ง 4 บิตแรกที่มีนัยสำคัญสูงสุด(MSB)ไปก่อน แล้วตามด้วย 4 บิตหลังที่มีนัยสำคัญต่ำสุด(LSB) ซึ่งการที่ต้องส่งข้อมูลถึง 2 ครั้งนั้นเวลาที่ใช้ไม่ได้เป็นอุปสรรคเนื่องจากความเร็วในการส่งข้อมูลก็ยังมากจนสายตาของเราไม่สามารถแยกออกได้ ส่วนการสั่งงานให้โมดูล LCD ทำงานนั้นจะต้องเขียน Software ควบคุมจากไมโครคอนโทรลเลอร์ไปสร้างสัญญาณ Timing เพื่อควบคุมไอซี HD44780 ให้ทำงาน ดังนั้นนอกจากบัสข้อมูล 4 เส้นแล้วจำเป็นต้องมีสายควบคุมเพิ่มเข้ามาได้แก่

RS (Register Select) ขานี้ทำหน้าที่เพื่อเลือกว่าจะถ่ายโอนข้อมูล หรือ ส่งคำสั่งระหว่างไมโครคอนโทรลเลอร์กับโมดูล LCD ถ้าขานี้เป็น High คือ Data register จะเป็นการอ่านหรือเขียนข้อมูลของไบต์ที่ตำแหน่งปัจจุบันของเคอร์เซอร์ใน LCD ที่เวลานั้น และถ้าขานี้เป็น Low จะมี 2 โหมดคือ 1. Instruction Register สำหรับการเขียน จะเป็นการส่งคำสั่งไปยัง LCD , 2. Read busy flag (DB7) and address counter (DB0 to DB6) สำหรับการอ่าน จะเป็นการอ่านสถานะการดำเนินการของคำสั่งสุดท้ายกลับมา (ไม่ว่าจะเสร็จสมบูรณ์หรือไม่ก็ตาม)

R/W (Read/Write) ขานี้ทำหน้าที่กำหนดทิศทางของข้อมูลว่าจะเป็นการอ่านหรือเขียนข้อมูล แต่จะไม่สามารถอ่านสถานะย้อนกลับจากจอแสดงผลได้

E (Enable) คือ Starts data read/write. ขานี้ใช้เพื่อเริ่มการอ่านหรือเขียนข้อมูลภายใน LCD โดยการกำหนดให้ขานี้เป็น High ก่อนที่จะส่งข้อมูลหรือคำสั่งใดๆ หลังจากให้ดีเลย์และส่งอาร์กิวเมนต์ไปเรียบร้อยแล้วต้องให้ขานี้เป็น Low

HD44780 มีรีจิสเตอร์อยู่ 2 ตัว คือ

Instruction Register (IR) มีหน้าที่เก็บคำสั่งเช่น Clear display, Cursor shift , ข้อมูลของตำแหน่งที่ Display ใน Display Data RAM (DDRAM) และ Character Generator RAM (CGRAM) โดยที่ IR จะถูกเขียนจากไมโครโพรเซสเซอร์

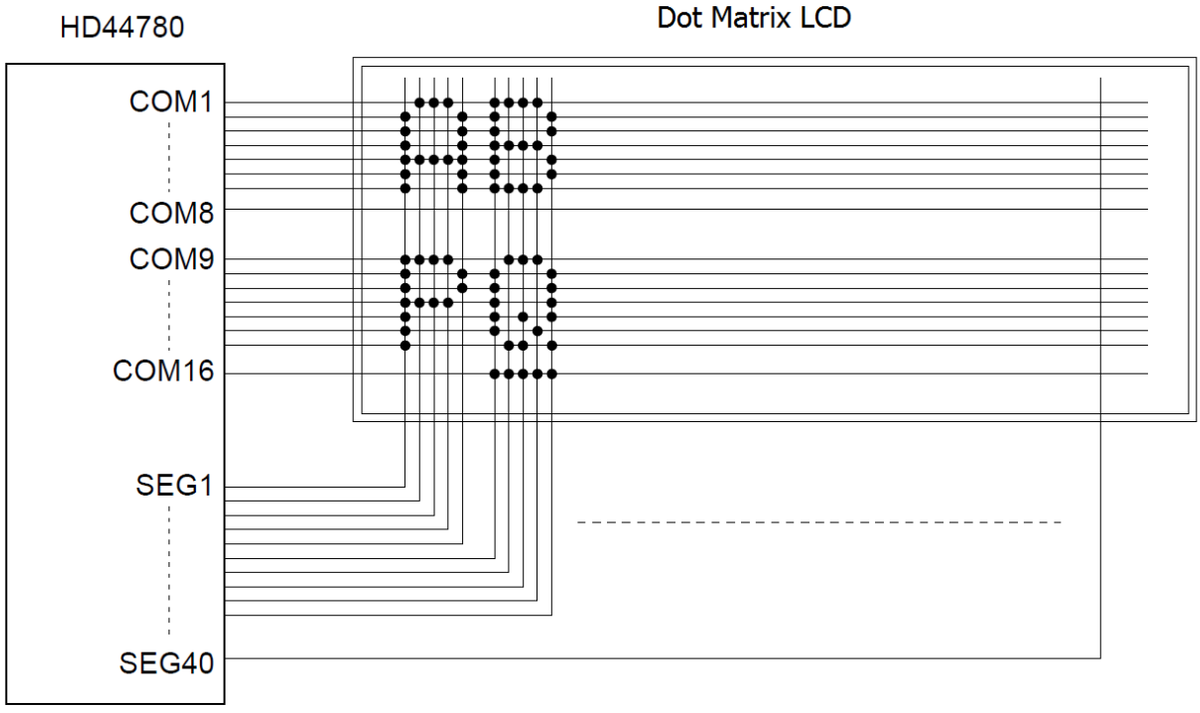
Data Register (DR) ใช้เก็บข้อมูลชั่วคราวที่จะเขียนลงใน Data Display RAM (DDRAM) หรือ Character Generator RAM (CGRAM) และเก็บข้อมูลชั่วคราวที่อ่านจาก DDRAM หรือ CGRAM

RS	R/W	การทำงาน
0	0	เขียนคำสั่งลงใน IR
0	1	อ่าน Busy Flag (DB7) และ Address Counter (DB0 – DB6)
1	0	เขียนข้อมูลลงใน DD RAM หรือ CG RAM
1	1	อ่านข้อมูลจาก DD RAM หรือ CG RAM

ชุดคำสั่ง (Instruction set) เพื่อควบคุมการทำงานของ Dot Matrix LCD HD44780 มีดังนี้

	Code										
Instruction	**RS**	**R/$\overline{W}$**	**DB7**	**DB6**	**DB5**	**DB4**	**DB3**	**DB2**	**DB1**	**DB0**	**Description**
Clear display	0	0	0	0	0	0	0	0	0	1	Clears entire display and sets DDRAM address 0 in address counter.
Return home	0	0	0	0	0	0	0	0	1	—	Sets DDRAM address 0 in address counter. Also returns display from being shifted to original position. DDRAM contents remain unchanged.
Entry mode set	0	0	0	0	0	0	0	1	I/D	S	Sets cursor move direction and specifies display shift. These operations are performed during data write and read.
Display on/off control	0	0	0	0	0	0	1	D	C	B	Sets entire display (D) on/off, cursor on/off (C), and blinking of cursor position character (B).
Cursor or display shift	0	0	0	0	0	1	S/C	R/L	—	—	Moves cursor and shifts display without changing DDRAM contents.
Function set	0	0	0	0	1	DL	N	F	—	—	Sets interface data length (DL), number of display lines (N), and character font (F).
Set CGRAM address	0	0	0	1	ACG	ACG	ACG	ACG	ACG	ACG	Sets CGRAM address. CGRAM data is sent and received after this setting.
Set DDRAM address	0	0	1	ADD	ADD	ADD	ADD	ADD	ADD	ADD	Sets DDRAM address. DDRAM data is sent and received after this setting.
Read busy flag & address	0	1	BF	AC	AC	AC	AC	AC	AC	AC	Reads busy flag (BF) indicating internal operation is being performed and reads address counter contents.
Write data to CG or DDRAM	1	0	Write data								Writes data into DDRAM or CGRAM.
Read data from CG or DDRAM	1	1	Read data								Reads data from DDRAM or CGRAM.
	I/D = 1: Increment I/D = 0: Decrement S = 1: Accompanies display shift S/C = 1: Display shift S/C = 0: Cursor move R/L = 1: Shift to the right R/L = 0: Shift to the left DL = 1: 8 bits, DL = 0: 4 bits N = 1: 2 lines, N = 0: 1 line F = 1: 5 × 10 dots, F = 0: 5 × 8 dots BF = 1: Internally operating BF = 0: Instructions acceptable						–				DDRAM: Display data RAM CGRAM: Character generator RAM ACG: CGRAM address ADD: DDRAM address (corresponds to cursor address) AC: Address counter used for both DD and CGRAM addresses

เมื่อจ่ายไฟเข้ากับจอแสดงผล LCD แล้วจะอยู่ในสถานะพร้อมทำงาน ซึ่งแสดงโดยบรรทัดทั้งหมดเป็นสีดำ หลังจากเริ่มต้นระบบสามารถส่งคำสั่งไปยังจอแสดงผลเพื่อย้ายเคอร์เซอร์ไปยังตำแหน่งที่ต้องการ สำหรับจอแสดงผล LED ขนาดตัวอักษร 2×16 บรรทัดแรกอักษรจะมีแอดเดรสอยู่ที่ 0x80-0x8F ในขณะที่บรรทัดที่สองแอดเดรสจะอยู่ที่ 0xC0-0xCF และถ้า LCD ที่ใช้นั้นมีจำนวนบรรทัดมากกว่าสอง จะทำให้มีบรรทัดที่สามแอดเดรสอยู่ที่ 0x90-0x9F และบรรทัดที่สี่แอดเดรสจะอยู่ที่ 0xD0-0xDF ตัวอย่างวงจรภายในสำหรับการต่อ Dot Matrix ขนาด 5x8 แปดตัวอักษร สองบรรทัด แสดงได้ดังนี้



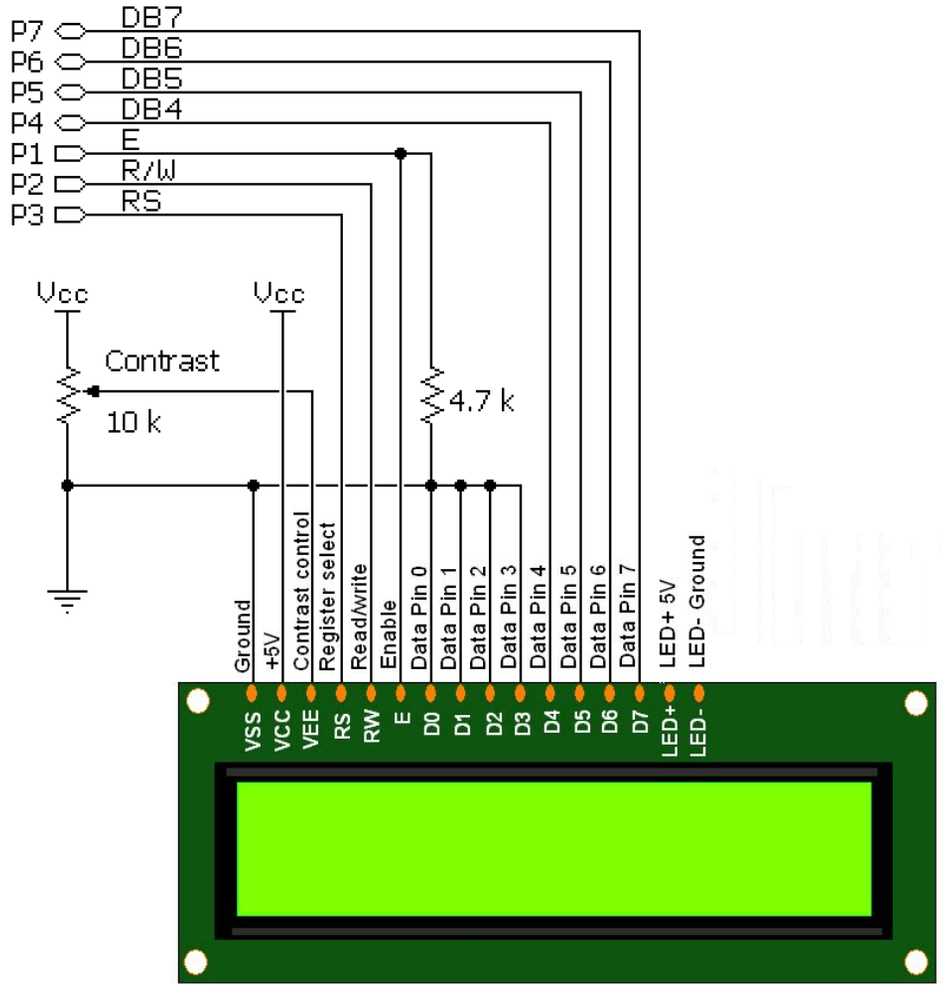
Example of a 5 × 8 dot, 8-character × 2-line display (1/5 bias, 1/16 duty cycle)

หน้าจอ LCD ที่สามารถนำมาใช้งานร่วมกันกับไอซี HD44780 นี้ได้ผลิตขึ้นมาในรูปแบบมาตรฐานต่างๆ ขนาดทั่วไปคือมีตั้งแต่หนึ่งแถวแปดตัวอักษร (8x1) สองแถวสิบหกตัวอักษร (16×2) นอกจากนี้ยังมีรูปแบบขนาดที่กำหนดใหญ่ขึ้นอีก ประกอบด้วยอักษร 20, 32, 40 หรือ 80 ตัว และมีจำนวนแถวตั้งแต่ 1, 2, 4 หรือ 8 บรรทัด การผลิตที่มีหน้าจอขนาดใหญ่ขึ้นนั้นโดยทั่วไปมีจำนวนตัวอักษร 40x4 ซึ่งต้องใช้คอนโทรลเลอร์ HD44780 ที่กำหนดแอดเดรสแยกกันสองตัว เนื่องจากไอซี HD44780 ตัวเดียวสามารถระบุตัวอักษรได้สูงสุดไม่เกิน 80 ตัว ส่วน Backlight ที่ใช้ในการให้ความสว่างเพื่อให้พื้นหลังเรืองแสง ทำให้เห็นตัวอักษรของหน้าจอ LCD ได้นั้น อาจจะเป็นหลอด Fluorescent หรือ LED ก็ได้

หน้าจอแสดงผล LCD ที่ใช้ HD44780 ส่วนใหญ่จะใช้ขั้วต่ออินเทอร์เฟซแบบ 16 ขาติดต่อกัน ซึ่งใช้หมุดสำหรับการเชื่อมต่อระยะห่างระหว่างกัน 0.1 นิ้ว (2.54 มม.) โดยมีแต่ละขาที่ต่อเป็นดังนี้

1. Ground (0V)
2. Vcc Power supply ต่อกับไฟเลี้ยง 2.7V ถึง 5.5V
3. Contrast adjustment (VO) เป็นอินพุตแบบอะนาล็อก ผู้ใช้จะต้องควบคุมแรงดันไฟฟ้าที่ขานี้เพื่อเพิ่มประสิทธิภาพในการมองเห็นของจอแสดงผลที่แตกต่างกัน ถ้าปรับไม่ถูกต้องจะไม่เห็นตัวอักษรที่หน้าจอแสดงผล ซึ่งขานี้จะเชื่อมต่อกับแรงดันไฟฟ้าที่ได้จากการแบ่งแรงดันไฟฟ้า (Voltage divider) ของตัวต้านทานปรับค่าได้ 10k

4. Register Select (RS) ถ้า RS=0 จะเป็นประเภทคำสั่ง (Command) และ RS=1 จะเป็นข้อมูล (Data)
5. Read/Write (R/W) ถ้า R/W=0 จะเป็น Write และ R/W=1 จะเป็น Read
6. Enable (E) เป็นเสมือนสัญญาณ Clock ใช้ขอบขาลงในการทริก
7. DB0 (ไม่ได้ใช้ในการทำงานแบบ 4 บิต)
8. DB1 (ไม่ได้ใช้ในการทำงานแบบ 4 บิต)
9. DB2 (ไม่ได้ใช้ในการทำงานแบบ 4 บิต)
10. DB3 (ไม่ได้ใช้ในการทำงานแบบ 4 บิต)
11. DB4
12. DB5
13. DB6
14. DB7
15. Backlight Anode (+) ต่อกับแรงดันไฟบวก
16. Backlight Cathode (-) ต่อกับกราวด์



การเขียนโปรแกรม Character Pattern การกำหนดรูปแบบตัวอักษรจะต้องมีความสอดคล้องระหว่าง Address และ Data ที่ใช้ในการเขียนโปรแกรมรูปแบบอักษรจะถูกสร้างแล้วเก็บอยู่ใน EPROM ROM HD44780U สามารถสร้างรูปแบบตัวอักษรขนาด 5x8 จุด และ 5x10 จุด โดยมีรูปแบบตัวอักษรทั้งหมด 240 แบบ

ค่า Address กับค่า Data ของ EPROM และรูปแบบ Pattern ของตัวอักษรจะมีความสอดคล้องกัน ในการสร้างตัวอักษรขนาด 5x8 ตัวอย่างอักษร b สามารถแสดงได้ดังในตาราง

EPROM Address												Data				
A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0	O4	O3	O2	O1	LSB O0
								0	0	0	0	1	0	0	0	0
								0	0	0	1	1	0	0	0	0
								0	0	1	0	1	0	1	1	0
								0	0	1	1	1	1	0	0	1
								0	1	0	0	1	0	0	0	1
								0	1	0	1	1	0	0	0	1
								0	1	1	0	1	1	1	1	0
0	1	1	0	0	0	1	0	0	1	1	1	0	0	0	0	0
								1	0	0	0	0	0	0	0	0
								1	0	0	1	0	0	0	0	0
								1	0	1	0	0	0	0	0	0
								1	0	1	1	0	0	0	0	0
								1	1	0	0	0	0	0	0	0
								1	1	0	1	0	0	0	0	0
								1	1	1	0	0	0	0	0	0
								1	1	1	1	0	0	0	0	0

← Cursor position

Character code — Line position

โดยที่ค่าตามในตารางจะเป็นดังนี้

1. EPROM แอดเดรส A11 ถึง A4 เป็นค่าตำแหน่งของรหัสตัวอักษร
2. EPROM แอดเดรส A3 ถึง A0 เป็นค่าตำแหน่งแต่ละแถวของ Pattern ในตัวอักษรนั้น
3. ข้อมูล EPROM O4 ถึง O0 เป็นค่าข้อมูลแต่ละแถวของ Pattern ตัวอักษรนั้น
4. ข้อมูล EPROM O5 ถึง O7 เป็นค่าข้อมูลของตัวอักษรกำหนดให้เป็น 0
5. ตำแหน่ง Pixel ของตัวอักษรที่ต้องการแสดงกำหนดให้เป็น 1
6. ค่าข้อมูลของ Pattern ตั้งแต่แถวที่ 9 เป็นต้นไปจะว่างกำหนดให้เป็น 0 สำหรับฟอนต์อักษรขนาด 5×8 Pixel

การเขียนโปรแกรมเพื่อแสดงผลบนโมดูล Dot Matrix LCD จะใช้คำสั่งที่ได้จาก Library ที่ชื่อว่า Wire และ hd44780 โดยนำการสื่อสารแบบ I2C มาใช้ แล้วเลือก Address ของอุปกรณ์ให้ตรงกับฮาร์ดแวร์ รูปแบบฟังก์ชัน API ที่มีอยู่ในไลบรารี hd44780 จะเป็นดังนี้

begin(cols, rows) initialize communication interface and LCD
clear() clear the display and home the cursor
home() home the cursor
setCursor(col, row) set cursor position
write(data) send data byte to the display returns 1 on success
write(*str) send C string to the displayreturns characters written
write(*buffer, size) send size bytes to the displayreturns characters writtenreturns size on success
print(...) print formatted data on the display(from Print class)returns characters output
cursor() turn on underline cursor

noCursor() turn off/hide cursor
blink() enable blinking at cursor position
noBlink() disable blinking at cursor position
display() enable pixels on display
noDisplay() disable/hide pixels on display
scrollDisplayLeft() shift display contents left
scrollDisplayRight() shift display contents right
autoscroll() enable left/right autoshifting for new characters
noAutoscroll() disable left/right autoshifting
leftToRight() write left to right, set autoshift to left
rightToLeft() write right to left, set autoshift to right
createChar(charval, charmap[ ]) create a custom character
setRowOffsets(row0) set address for start of line
setRowOffsets(row0, row1) set address for start of each line
command(cmd) send raw 8bit hd44780 command to LCD
backlight() turn on backlight (max brightness)
noBacklight() turn off backlight
lineWrap() turn on automatic line wrapping(wraps lines but does not scroll display)
noLineWrap() turn off automatic line wrapping
moveCursorLeft() move cursor one space to right
moveCursorRight() move cursor one space to left
read() read data byte from LCD(requires r/w signal control)returns negative value on failure
setExecTimes(chUs, insUs) configure clear/home and instruction/data times
setBacklight(dimvalue) set backlight brightness (0-255)
setContrast(contvalue) set contrast (0-255)
on() turn on LCD pixels and backlight
off() turn off LCD pixels and backlight
status() read status byte (busy flag & address)(requires r/w signal control) : returns negative value on failure
cmdDelay(CmdDelay, CharDelay) use setExecTimes() instead
cursor_on() use cursor() instead
cursor_off() use noCursor() instead
blink_on() use blink() instead
blink_off() use noBlink() instead
load_custom_character(char_num, Rows[ ]) use createChar() instead
setCursor(row, col) row,col is backwards from Liquidcrystal

ปกติแล้วการเชื่อมต่อโมดูล LCD แบบ 4 Bit จะใช้สายจำนวนมาก ทำให้เปลือง Pin ที่อาจจะต้องนำไปใช้งานกับอุปกรณ์อื่นๆ แต่เนื่องจากโหมดการเชื่อมต่อแบบ 4 Bit สามารถแปลงสายสัญญาณให้เข้ากับโมดูลที่มีอะแดปเตอร์ I2C ได้ ทำให้ต่อสายสัญญาณได้ง่ายขึ้น ดังนั้นในการทดลองนี้จะได้นำเอาโมดูล I2C มาประยุกต์ใช้งานกับโมดูล LCD เพื่อที่จะลดจำนวน Pin ที่ต้องใช้งานจำนวนมากลงให้เหลือเพียง 2 สาย คือ SCL และ SDA เท่านั้น และยังสามารถติดต่อกับอุปกรณ์อื่นที่ใช้ I2C ร่วมกันได้บนสายสัญญาณเดียวกันได้อีกด้วย

1. ให้เชื่อมต่อโมดูล Dot Matrix LCD ที่มี 16 Pin ต่อเข้ากับโมดูล I2C Board
2. จากโมดูล I2C Board ให้เชื่อมต่อกับพอร์ต I2C ของ Arduino โดยต่อสาย SDA เข้ากับขา A4 ของ Arduino และสาย SCL เข้ากับขา A5 ของ Arduino ในกรณีที่ต่อใช้อุปกรณ์ I2C หลายตัวพร้อมกันจะต้องเปลี่ยนค่า Address ของอุปกรณ์แต่ละตัวไม่ให้ซ้ำกันด้วย โดยใช้โปรแกรมเพื่อตรวจหา Address ของอุปกรณ์ต่างๆที่ต่ออยู่กับพอร์ต I2C
3. ให้ใส่ libraries hd44780 ลงในโปรแกรม Arduino โดยที่ hd44780.h จะเป็นฟังก์ชัน API ของ Dot Matrix LCD HD44780 ส่วนที่ติดต่อสื่อสารผ่านทาง I2C จะใช้ hd44780_I2Cexp.h ร่วมด้วย
4. ทำการเขียนโปรแกรมที่ทำหน้าที่สั่งงานให้ Dot Matrix LCD แสดงผลลัพธ์ที่หน้าจอเป็นคำว่า LCD I2C โดยในโปรแกรมมีการกำหนดให้ LCD นี้มีจำนวน Column ทั้งหมดเท่ากับ 16 และมี Row ทั้งหมดเท่ากับ 2 จากนั้นทำการคอมไพล์แล้วทำการ Upload โปรแกรมที่ได้ลงบนบอร์ด Arduino

```
#include <Wire.h>
#include <hd44780.h>                  // main hd44780 header
#include <hd44780_I2Cexp.h>           // i2c expander i/o class header
```

```
hd44780_I2Cexp lcd;              // declare lcd object: auto locate & config exapander chip

const int LCD_COLS = 16;
const int LCD_ROWS = 2;

void setup()
{
int status;
    status = lcd.begin(LCD_COLS, LCD_ROWS);
    if(status)                               // non zero status means it was unsuccesful
    {
        // begin() failed so blink error code using the onboard LED if possible
        hd44780::fatalError(status);         // does not return
    }

    lcd.print("LCD I2C");
}

void loop()
{
}
```

5. ให้แก้ไขโปรแกรมในข้อ 4 โดยเพิ่มโปรแกรมสั่งงานให้ LCD นี้แสดงผลลัพธ์เพิ่มเติมที่หน้าจอในบรรทัดที่ 2 โดยให้ใส่ไว้ใน void setup() ดังนี้

```
lcd.setCursor(0, 1);
lcd.print("ASCII Code Test");
```

6. ให้แก้ไขโปรแกรมในข้อ 5 โดยย้ายโปรแกรมในข้อ 5 ไปใส่ไว้ใน void loop() แทน แล้วให้อธิบายผลที่ได้

ตัวหนังสือกระพริบไป

7. ให้แก้ไขโปรแกรมในข้อ 6 โดยเพิ่มโปรแกรมสั่ง Clear หน้าจอ ต่อท้ายโปรแกรมเดิม แล้วให้อธิบายผลที่ได

```
lcd.clear();
```

จะแสดง ASCII Code Test

แล้วก็จะเคลียร์หน้าจอใหม่เรื่อยๆ

8. จากโปรแกรมในข้อ 7 เพื่อให้โปรแกรมทำงานได้ปกติจะต้องเพิ่มคำสั่งอะไรหลังคำสั่ง Print หน้าจอ ให้ทำการทดลองและอธิบายผลที่ได้

ใส่ delay(100) เข้าไป

9. ให้แก้ไขโปรแกรมในข้อ 8 โดยเพิ่มโปรแกรมแสดงชุดของตัวอักษรทั้งหมดภายใน Dot Matrix LCD HD44780 ตั้งแต่ ASC II ตัวที่ 0 ถึง 255 โดยใช้คำสั่งเพื่อคำนวณหาตำแหน่งของ Cursor ด้วยดังนี้

```
lcd.cursor();                  // turn on cursor so you can see where it is row = 0;
col = 0;
for(int i=0; i<256; i++)       // start at the character for the number zero
{
    lcd.setCursor(col, row);
    lcd.print(char(i));        // print ASCII chars
    col = col + 1;
    if (col == 16)
    {
        col = 0;
```

```
            row = row + 1;
            if (row == 2)
                row =0;
        }
        delay(50);                    // slow things down to watch the printing & wrapping
    }
    delay(2000);
```

10. จากโปรแกรมในข้อ 9 ให้แก้ไขโปรแกรมเพื่อให้แสดงชุดของตัวอักษรทั้งหมดภายใน Dot Matrix LCD HD44780 ตั้งแต่ ASC II ตัวที่ 0 ถึง 255 โดยเปลี่ยนเป็นดังนี้ แล้วให้อธิบายผลที่ได้ว่าเกิดข้อผิดพลาดอะไรบ้าง

```
lcd.cursor();                        // turn on cursor so you can see where it is
for(int i=0; i<256; i++)             // start at the character for the number zero
{
    lcd.write(i);                    // lcd.print(char(i));  print ASCII chars
    delay(50);                       // slow things down to watch the printing & wrapping
}
delay(2000);
```

Run ตัวอักษรล้นไปเป็นบรรทัด

11. จากโปรแกรมในข้อ 10 เพื่อให้โปรแกรมทำงานได้ปกติ ให้ทดลองเพิ่มคำสั่ง lcd.lineWrap(); โดยให้ใส่ไว้ใน void setup() เสร็จแล้วให้ทำการทดลองและอธิบายผลที่ได้ว่าคำสั่งนี้ทำหน้าที่อะไร

ใส่ลงไปแล้ว จะทำให้โปรแกรมสามารถปรับบรรทัดให้อัตโนมัติ รับมาใช้แค่ 0-17 จะได้

row 1 ; 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

2 ; 16 17

12. จากข้อ 11 ให้ทดลองกำหนดตัวแปรสร้างตัวอักษรในรหัส ASC II ขึ้นมาใหม่ในตำแหน่งตั้งแต่ตัวที่ 0 ถึง 5 โดยที่อักษรแต่ละตัวจะเป็นขนาด 5x8

```
uint8_t char0[8] = {0x00,0x00,0x03,0x04,0x08,0x19,0x11,0x10};    // Char 0 - Upper-left
uint8_t char1[8] = {0x00,0x1F,0x00,0x00,0x00,0x11,0x11,0x00};    // Char 1 - Upper-middle
uint8_t char2[8] = {0x00,0x00,0x18,0x04,0x02,0x13,0x11,0x01};    // Char 2 - Upper-right
uint8_t char3[8] = {0x12,0x13,0x1b,0x09,0x04,0x03,0x00,0x00};    // Char 3 - Lower-left
uint8_t char4[8] = {0x00,0x11,0x1f,0x1f,0x0e,0x00,0x1F,0x00};    // Char 4 - Lower-middle
uint8_t char5[8] = {0x09,0x19,0x1b,0x12,0x04,0x18,0x00,0x00};    // Char 5 - Lower-right
```

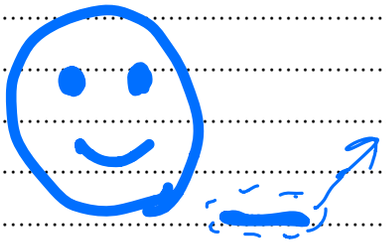
13. ให้แก้ไขโปรแกรมในข้อ 12 โดยเพิ่มโปรแกรม Create ตัวอักษร แล้วสั่งให้แสดงตัวอักษรนั้นบน Dot Matrix LCD จากนั้นให้อธิบายผลที่ได้

```
// create 6 custom characters
lcd.createChar(0, char0);
lcd.createChar(1, char1);
lcd.createChar(2, char2);
lcd.createChar(3, char3);
lcd.createChar(4, char4);
lcd.createChar(5, char5);

lcd.clear();
lcd.setCursor(0, 0);
lcd.write(0);              // write character 0
lcd.write(1);              // write character 1
```

```
lcd.write(2);              // write character 2
lcd.setCursor(0, 1);
lcd.write(3);              // write character 3
lcd.write(4);              // write character 4
lcd.write(5);              // write character 5
delay(2000);
```

ไสรูป



14. จากข้อ 13 ถ้านำเอาตัวอักษรหลายตัวต่อกันเป็นรูปภาพ โดยที่ไม่ต้องการให้เห็นตำแหน่ง Cursor จะต้องแก้ไขโปรแกรมอย่างไร ให้ทำการทดลองและอธิบายวิธีการแก้ไข

เมื่อใส่ lcd.noCursor(); จะได้

15. ให้เขียนโปรแกรมเพื่อแสดงชื่อนักศึกษาเป็นภาษาไทย บน Dot Matrix LCD โดยทำเป็นตัวอักษรวิ่งจากขวาไปซ้ายทีละตัว
16. จากข้อ 15 ให้นำสวิทช์มาต่อเพิ่ม 1 ตัว กำหนดเงื่อนไขไว้ว่า ถ้ากดสวิทช์ให้ตัวอักษรวิ่งจากซ้ายไปขวา และถ้าปล่อยสวิทช์ตัวอักษรจะวิ่งจากขวาไปซ้ายทีละตัว โดยให้แสดงทันทีที่สวิทช์ถูกกด
17. จากข้อ 16 ให้นำไปประยุกต์ใช้งานร่วมกับ Stepping motor ที่มีการทำงานร่วมกับ I2C โดยต้องกำหนด Address ของ I2C ทั้งสองตัวให้ต่างกันด้วย