Chapter

The LCD Interfacing and Programming

Dahlan Sitompul and Poltak Sihombing

Abstract

This chapter will discuss 10 subchapters that will make it more detailed and easier for the reader to master and implement them in their project. Before discussing the subchapters in detail the author discusses the wide use of LCD in various equipment that needs display and the superiority of it compared to the conventional existing displays especially in the low energy consumption of it compare to the rest of the displays, then the author ended this general discussion by mentioning the type of LCD known in the market right now (passive matrix and active matrix). After discussing the LCD in general, the author starts discussing the detailed 10 subchapters. The 10 subchapters are 1. 2 × 16 LCD; 2. LCD controller; 3. LCD instructions; 4. LCD initialization; 5. More instructions; 6. LCD initialization subroutine; 7. Displaying a character on the LCD; 8. Displaying more than 1 character on the LCD; 9. A 4-bit mode 2 × 16 LCD module. To give the readers with a succinct overview of important details or interesting information, the author provides the summary of this chapter in subchapter 10. The author also provided the glossary to enable the readers to quickly study the general terms used in this chapter. Finally, the author provides some questions to enable the reader to test their own knowledge of this chapter. The references is also provided to enable the readers to refer to some articles as the sources of this subchapter and to enable them to enrich their knowledge of this chapter.

Keywords: LCD, microcontroller, interfacing, programming

1. Introduction

LCD stands for Liquid-Crystal Display. From the name, we know that this equipment is used as a display. Many types of electronic equipment (Laptop, Digital Voltmeter, Mobile Phone, ATM-Automatic Teller Machine, PC-Personal Computer, Compact Disk Player, Digital Thermometer, Clothe Washer, Flat TV-Television, etc.) has used this equipment to display images, Alpha Numeric information, and video. LCD is electronic equipment (display) made from liquid-crystal layers which operate if it gets varied voltage/currents which change the optical properties (light modulating properties) of this display; which causes this tool to continue or block the light through it. LCD uses its own light source known as Back Plane Light (BPL). Compared to other types of display such as CRT (Cathode Ray Tube), LED (Light Emitting Diode), plasma gas, LCD consumes smaller power, this is because it does not produce light, but this tool displays images with the principle of inhibiting or forwarding light; the changing of the optical properties of the LCD when it gets a varying voltage/current on the LCD layer and the existence of a light source (BPL)

causes this equipment to display the object/character that can be seen by our eyes. At present, there are two types of LCDs known in general, namely,

- Passive matrix
- Active matrix

Matrix active is often referred to as AMLCD (Active Matrix LCD). Some people call AMLCD with TFT (thin-film transistor) this is due to a transistor that is useful as a large controller of the current through the intersection of pixels. With the use of transistors as controlling currents, controlling the current through intersection becomes faster which ultimately increases the refresh time of this tool which then produces subtler images compared to the LCD-passive matrix [1, 2].

The manufacture or the fabrication of AMLCD is far more complicated than the fabrication of passive matrix LCD which causes its price to be more expensive than the passive matrix LCD [2].

The Microcontroller System 51-MCS51 (Microcontroller 8051 family) can also be connected with the LCD, where the LCD is used as a display. On this occasion, we will discuss LCDs commonly used in various microcontroller applications namely LCD 2 × 16; 2 rows with 16 characters/row.

The LCD module basically consists of two units namely the LCD itself and the controller. The LCD module is mounted and pinned to the controller and then the microcontroller is connected to the LCD controller.

1.1 2 × 16 LCD

There are various types of LCDs commonly used in various microcontroller applications, namely 1 × 16, 2 × 16, 4 × 20 [3]. The LCD type is known by the number of lines and the number of characters per line it has. LCD with type 1 × 16 means, this LCD can only display 1 line of characters, and can only display 16 alphanumeric characters. Likewise, LCD with types 2 × 16 and 4 × 20 each has 2 and 4 consecutive lines, and the first type can only display 16 alphanumeric characters/line, while the last type is capable of displaying 20 alphanumeric characters/line. On this occasion we will only discuss the connection/the usage of the second type of LCD, type 2 × 16 (2 lines × 16 characters/line); by understanding the 2 × 16 type LCD, then you can easily read and learn the manual of other types to connect them to MCS51 [4].

Figure 1 below shows the physical appearance of this display,

From **Figure 1** it can be seen that the number of lines of the LCD is 2. The first line contains the words “this is a 2 × 16” while the second line contains the words “ Line LCD Display” [5].

Figure 1.
2 × 16 LCD physical picture [5].

1.2 LCD controller

As previously mentioned above to build the LCD module, the LCD (M1632) is mounted on and pinned to the HD44780 controller. To use the LCD module in MCS-51 the controller is connected to the MCS-51 (**Figure 2**) [3, 6].

The LCD controller can be used for various types of LCDs with various numbers of characters per line; from 8 to 80 characters per line and the number of lines between 1 and 4 lines [4]. The controller has DISPLAY DATA RAM (DDRAM) in it with a memory map as shown in **Figure 3** below [7].

From **Figure 3** above we can see that the DDRAM controller should have a memory map of 104 bytes (00H to 67H = 68H = 104D), but the location that can be used is only 80 bytes 00H-27H (40 bytes) and 40H-67H (40 bytes); Note that the 28H-3FH address (24 location) is not visible.

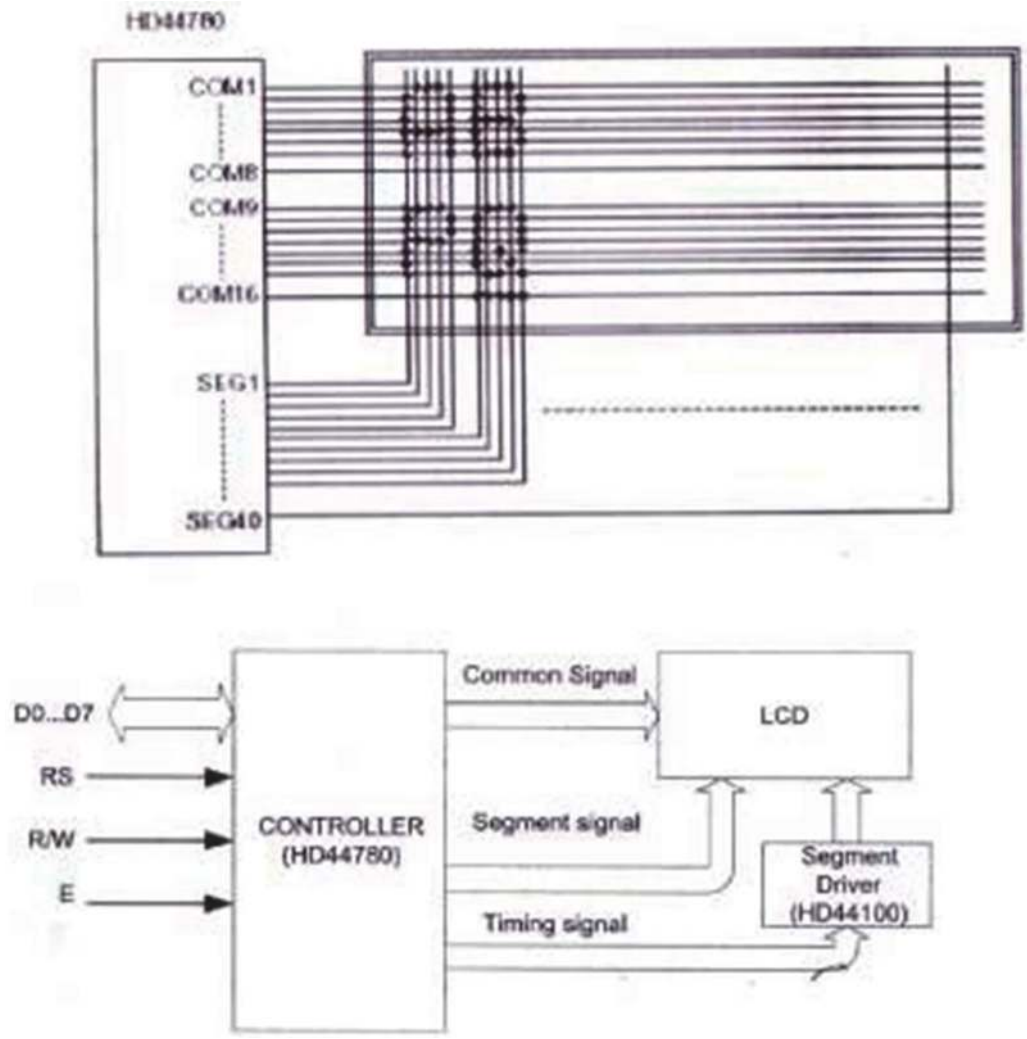

Figure 2.
Block diagram of 2 × 16 LCD module [3, 6].

00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E	1F	20	21	22	23	24	25	26	27
40	41	42	43	44	45	46	47	48	49	4A	4B	4C	4D	4E	4F	50	51	52	53	54	55	56	57	58	59	5A	5B	5C	5D	5E	5F	60	61	62	63	64	65	66	67

Figure 3.
2 × 16 LCD DDRAM- memory map [6, 7].

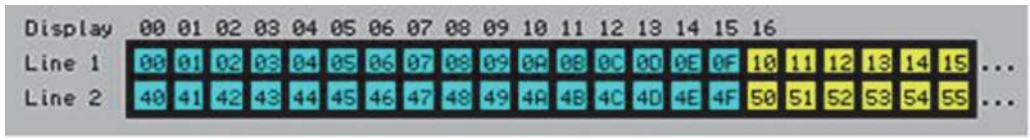

Figure 4.
2 × 16 LCD DDRAM address [8].

Each place on the LCD has an address that corresponds to the address on the DDRAM on the LCD controller) see **Figures 3** and **4** for more details. The DDRAM memory map on the 2 × 16 bit LCD controller that can be used for LCD with type M1632 is 00H-0FH for the first line (16 addresses) and 40H-4FH (16 addresses) for the second line (so the total address location 32 address locations), although as shown in **Figure 3** DDRAM memory maps of the LCD controller basically amounted to 80 locations.

However, the display looks just having 00H-0FH addresses for the first line and 40H-4FH addresses for the second line as shown in **Figure 4** above [8].

LCD connected to this controller will adjust itself to the memory map of this DDRAM controller; each location on the LCD will take 1 DDRAM address on the controller. Because we use 2 × 16 type LCD, the first line of the LCD will take the location of the 00H-0FH addresses and the second line will take the 40H-4FH addresses of the controller DDRAM; so neither the addresses of the 10H-27H on the first line or the addresses of the 50H-67H on the second line on DDRAM is used.

To be able to display a character on the first line of the LCD, we must provide written instructions (80h + DDRAM address where our character is to be displayed on the first line) in the Instruction Register-IR and then followed by writing the ASCII code of the character or address of the character stored on the CGROM or CGRAM on the LCD controller data register, as well as to display characters in the second row we must provide written instructions (C0H + DDRAM address where our character to be displayed on the second line) in the Instructions Register-IR and then followed by writing the ASCII code or address of the character on CGROM or CGRAM on the LCD controller data register.

The characters to be displayed can be in the form of American Standard Code For Information Interchange-ASCII code, characters originating from CGROM (various characters are prepared by the controller manufacturer or producer and are permanently stored in it), and from the character generator RAM-CGRAM (special place on CGROM with the addresses 00000000B-00000111B (consecutive eight locations) prepared for storing characters specifically to be displayed by the user/custom use); the total address of the CGROM with the CGRAM address in it and allocated specially for custom use is 256D (00000000B-111111111B) locations see **Figure 5** below; please do not confuse CGROM/CGRAM address with DDRAM address they are two completely different and independent memory locations. The character stored on the CGRAM address is temporary (volatile) and must be stored in it before retrieve and displayed it on the LCD; the data stored in CGRAM will disappear if the source of electrical voltage is cut off; in contrast, the characters stored in the CGROM is permanently reserved in it.

As mentioned above, to display a character (ASCII) you want to show on the LCD, you need to send the ASCII code to the LCD controller data register-DR. For characters from CGROM and CGRAM we only need to send the address of the character where the character is stored; unlike the character of the ASCII code, we must write the ASCII code of the character we want to display on the LCD controller data register to display it. For special characters stored on CGRAM, one must first save the special character at the CGRAM address (prepared 64 addresses, namely addresses 0–63); A special character with a size of 5 × 8 (5 columns × 8

DOI: http://dx.doi.org/10.5772/intechopen.102408

Figure 5.
CGRAM [9].

lines) requires eight consecutive addresses to store it, so the total special characters that can be saved or stored on the CGRAM addresses are only eight (8) characters. To be able to save a special character at the first CGRAM address we must send or write 40H instruction to the Instruction Register-IR followed by writing eight consecutive bytes of the data in the Data Register-DR to save the pattern/image of a special character that you want to display on the LCD [9, 10].

We can easily connect this LCD module (LCD + controller) with MCS51, and we do not need any additional electronic equipment as the interface between MCS51 and it; This is because this LCD works with the TTL logic level voltage—Transistor-Transistor Logic.

Based on 2 × 16 LCD data sheets of the Hitachi 44780 trademark we can connect it with MCS-51 as shown in **Figure 6** [11].

From **Figure 6** above it can be seen clearly that this display has 16 Pins that can be grouped into 4 as shown in **Table 1**.

The voltage source of this display is +5 V connected to Pin 2 (VCC) and GND power supply connected to Pin 1 (VSS) and Pin 16 (GND); Pin 1 (VSS) and

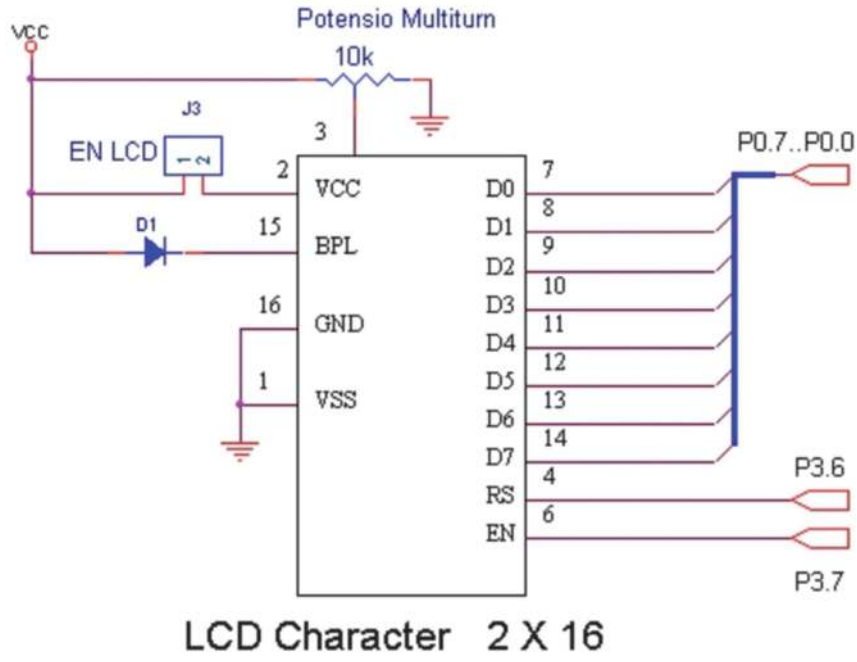

Figure 6.
Connection of 2 × 16 LCD module (8 bit mode) with MCS51 [11].

Pin	Name Pin	Function
1, 2, dan 16	VSS, VCC, GND	Power supply
7–14	D0-D7	Data bus
4, 5, and 6	RS, R/$\overline{W}$, and $\overline{EN}$	Control
3 and 15	VEE and BPL	BPL

Table 1.
Pin function of 2 × 16 LCD.

Pin 16 (GND) are combined together and connected to the GND of the power supply.

Pins 7–14 (8 Pins) of the display function as a channel to transmit either data or instruction with a channel width of 1 byte (D0-D7) between the display and MCS51. In **Figure 6**, it can be seen that each Pin connected to the data bus (D0-D7) of MCS51 in this case P0 (80h); P0.0-P0.7 MCS-51 connected to D0-D7 of the LCD.

Pins 4–6 are used to control the performance of the display. Pin 4 (Register Select-RS) is in charge of selecting one of the 2 display registers. If RS is given logic 0 then the selected register is the Instruction Register-IR, otherwise, if RS is given logic 1 then the selected register is the Data Register-DR. The implication of this selection is the meaning of the signal sent down through the data bus (D0-D7), if RS = 0, then the signal sent from the MCS-51 to the LCD is an instruction; usually used to configure the LCD, otherwise if RS = 1 then the data sent from the MCS-51 to the LCD (D0-D7) is the data (object or character) you want to display on the LCD. From **Figure 6** Pin 4 (RS) is connected to Pin 16 (P3.6/$\overline{W}$) of MCS-51 with the address (B6H).

Pin 5 (R/$\overline{W}$)) of the LCD does not appear in **Figure 6** is used for read/write operations. If Pin 5 is given logic 1, the operation is a read operation; reading the data from the LCD. Data will be copied from the LCD data register to MCS-51 via

the data bus (D0-D7), namely Pins 7–14 of the LCD. Conversely, if Pin 5 is given a voltage with logical 0 then the operation is a write operation; the signal will be sent from the MCS51 to LCD through the LCD Pins (Pins 7–14); The signal sent can be in the form of data or instructions depending on the logic level input to the Register Select-RS Pin, as described above before if RS = 0 then the signal sent is an instruction, vice versa if the RS = 1 then the signal sent/written is the data you want to display. Usually, Pin 5 of the LCD is connected with the power supply GND, because we will never read data from the LCD data register, but only send instructions for the LCD work configuration or the data you want to display on the LCD.

Pin 6 of the LCD ($\overline{EN}$) is a Pin used to enable the LCD. The LCD will be enabled with the entry of changes in the signal level from high (1) to low (0) on Pin 6. If Pin 6 gets the voltage of logic level either 1 or 0 then the LCD will be disabled; it will only be enabled when there is a change of the voltage level in Pin 6 from high logic level to low logic level for more than 1000 microseconds (1 millisecond), and we can send either instruction or data to processed during that enable time of Pin 6.

Pin 3 and Pin 15 are used to regulate the brightness of the BPL (Back Plane Light). As mentioned above before the LCD operates on the principle of continuing or inhibiting the light passing through it; instead of producing light by itself. The light source comes from LED behind this LCD called BPL. Light brightness from BPL can be set by using a potentiometer or a trimpot. From **Figure 6** Pin 3 (VEE) is used to regulate the brightness of BPL (by changing the current that enters BPL by using a potentiometers/a trimpot). While Pin 15 (BPL) is a Pin used for the sink of BPL LED.

Table 2 below is a summary of the function of each Pin of the 16 Pins of 2 × 16 LCD.

Pin	Name	Function
1	VSS	GND of the source voltage.
2	VCC	+5 V of the source voltage.
3	VEE	Adjust the brightness of the BPL by using an adjustable/variable resistor (potentiometer or trimpot).
4	RS	Register selector on the LCD, if RS = 0 then the selected register is an instruction register (the operation to be performed is a write operation/LCD configuration if Pin 5 (R/$\overline{W}$) is given a logic 0), if RS = 1 then the selected register is a data register; if (R/$\overline{W}$) = 0 then the operation performed is a data write operation to the LCD, otherwise if (R/$\overline{W}$) = 1 then the operation performed is a read operation (data will be sent from the LCD to μC (microcontroller); it is usually used to read the busy bit/Busy Flag- BF of the LCD (bit 7/D7).
5	(R/$\overline{W}$)	Sets the operating mode, logic 1 for reading operations and logic 0 for write operations, the information read from the LCD to μC is data, while information written to the LCD from μC can be data to be displayed or instructions used to configure the LCD. Usually, this Pin is connected to the GND of the power supply because we will never read data from the LCD but only write instructions to configure it or write data to the LCD register to be displayed.
6	$\overline{Enable}$	The LCD is not active when Enable Pin is either 1 or 0 logic. The LCD will be active if there is a change from logic 1 to logic 0; information can be read or written at the time the change occurs.
7–14	D0-D7	Data/instruction transmission channel between the LCD and the μC.
15	BPL	Sink for LED BPL (Back Plane Light). The VCC is connected to the LED anode and the cathode is connected to the BPL.
16	GND	Power supply ground. This Pin is connected to the power supply ground together with Pin 1 (VSS) of the LCD.

Table 2.
Summary of functions of each Pin on a 2 × 16 LCD.

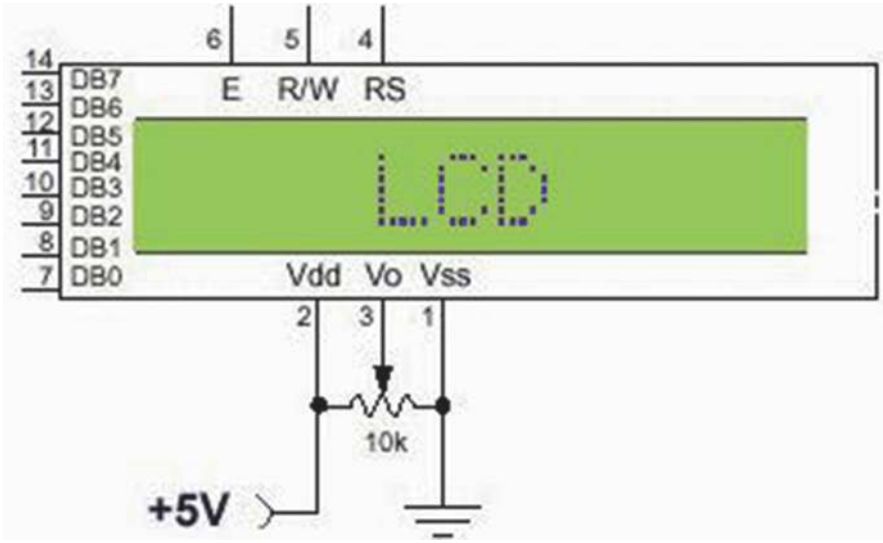

Figure 7.
2 × 16 LCD module [12].

1.3 LCD Instructions

Just like the microcontroller, the LCD controller must also be programmed first to operate. Programming the LCD controller is an easy thing, all we have to do is only to give the appropriate logic level to the three Pins RS, R/$\overline{W}$, E (Pins 4, 5, and 6), and DB0-DB7 Pin (Pin 7–14); see **Figure 7**. Each time we want to execute an instruction (read or write operation) there must be a change of logic level of Pin 6 (E-Enable); from logic 1 to logic 0 so that the instruction given to the controller can be executed properly, see **Figure 7** below [12].

The following are instructions for LCD controllers

A. Clear display

Clear display is an instruction that will clean the display and restore the cursor to the initial position (00h address). The format of this instruction is as follows,

RS	R/$\overline{W}$	D7	D6	D5	D4	D3	D2	D1	D0
0	0	0	0	0	0	0	0	0	1

Note that RS = 0 which means we choose Instruction Register-IR and R/$\overline{W}$ = 0 which indicates a write operation. Thus, the instruction we have to write on IR is 01h.

B. Home cursor

This instruction serves to return the cursor to the initial position (line 0, column 0) without changing the data on DDRAM. The format to do so is shown in the table below.

RS	R/$\overline{W}$	D7	D6	D5	D4	D3	D2	D1	D0
0	0	0	0	0	0	0	0	1	X

D0(X) can be filled with either 0 or 1 (do not care) and in this case, we select X = 0. Thus, the instruction that must be written to return the cursor to the initial position on the IR is 02H [10, 13].

C. Entry mode set

This instruction will determine the direction of the cursor movement whether it is left or right and determines whether to shift the display or not. The format of this instruction is as follows.

RS	R/$\overline{W}$	D7	D6	D5	D4	D3	D2	D1	D0
0	0	0	0	0	0	0	1	I/D	S

Please pay attention to D1 (I/D-Increment/Decrement) it is a "bit" that determines whether the cursor position goes up or down. If D1 (I/D) = 0 (decrement), the cursor position will decrease, which will shift towards the left of one column. Conversely, if D1 (I/D) = 1 (increment), the cursor position will shift to the right of one column.

Likewise, the Bit D0(S) if D0(S) = 0 then the operation is normal, the cursor shifts but the display will stay in the same place, but if D0(S) = 1 then the display will shift one column to the right if D1 = 0 (the cursor will remain in the same place/no movement. So, if we want a normal operating display and the cursor shifts to the right then the instructions, we have to give to IR is 06h and is 04h if you want to cursor shift to the left. **Table 3** below summarizes the instructions given and the result of the cursor and the display movement.

D. Display ON/OFF

This instruction is used to ON or OFF the display, display or not display the cursor, and set the cursor to blink or not. The format of this instruction is as follows.

RS	R/$\overline{W}$	D7	D6	D5	D4	D3	D2	D1	D0
0	0	0	0	0	0	1	D	C	B

When D2 (D-Display) = 1 then the display will be on, but if D2 = 0 then the display will be off, just like in the case of D2, D1 (C-cursor) also has the same property if D1 = 1 then the cursor will be displayed otherwise (if D1 = 0) then the cursor will not be shown. D0 (B-blink) as mentioned above is used to set the cursor to blink or not if D0 = 0 then the cursor will not blink, but if D0 = 1 then the cursor will blink.

From **Table 4** it can be seen clearly that the display will be ON when the instruction is 0Ch or bigger.

Instruction	Cursor movement direction	Display movement direction
04h	To the left one column	Stay in the same place/no movement
05h	Stay in the same place/no movement	Move to the right one column
06h	To the right one column	Stay in the same place/ no movement
07h	Stay in the same place/no movement	Move to the left one column

Table 3.
Entry mode set, cursor movement, and display movement.

Instruction/command	Display	Cursor (display)	Cursor (blinking)
08h	OFF	NO	NOT
09h	OFF	NO	NOT
0Ah	OFF	NO	NOT
0Bh	OFF	NO	NOT
0Ch	ON	NO	NOT
0Dh	ON	NO	NOT
0Eh	ON	YES	NOT
0Fh	ON	YES	YES

Table 4.
Display, cursor, and blinking.

E. Function set

Function set (set data width, number of lines, and font size) with the format of instruction as follows,

RS	R/$\overline{W}$	D7	D6	D5	D4	D3	D2	D1	D0
0	0	0	0	1	DL	N	F	X	X

D4 (DL) ⇒ Data Length. If DL = 0, data width = 4 bits (D4-D7). If DL = 1, data width = 8 bits (D0-D7). D3 (N) ⇒ number of Line. If N = 0 number of lines = 1, if N = 1 number of lines is 2. D2 (F) = > font size. If F = 0 then the font size is 5 × 7 dot, if F = 1 then the font size is 5 × 10 dot. When data width of 4 bits is chosen, data must be sent twice.

F. Set CGRAM address

As mentioned earlier in Chapter 2 LCD controller, CGRAM (Character Generator RAM) is a part of CGROM (Character Generator ROM); See **Figure 5** below [9]. From **Figure 5** we can see that CGRAM with an address of 00000000b-00000111b (eight addresses) which is a part of CGROM is deliberately prepared by the manufacturer to be able to store custom use characters needed by the user and not be prepared by the manufacturer such characters as, battery picture, alarm picture, or other characters at the CGRAM addresses.

The format of instruction to program the CGRAM can be seen below.

RS	R/$\overline{W}$	D7	D6	D5	D4	D3	D2	D1	D0
0	0	0	1	CGRAM address					

It is obviously seen that we need an instruction of 40h + the location of the CGRAM address where the pattern wants to be written to the Instruction Register-IR and followed by a write of the data to the Data Register (DR); this data pattern will be stored in the CGRAM address; the CGRAM is only 6 bits in length (D0-D5), thus the number of CGRAM addresses is only 64 addresses (address 0–63) [14]. To display a custom use character we need to save the

patterns (8 patterns in maximum) into the 64-byte size memory location of CGRAM (0D-63D). The first pattern to be displayed will be referred to just like a normal CGROM address but the address is 00h. To save the pattern in the CGRAM address we need to run a command (40H + the first address of the pattern saved in the CGRAM locations) to the IR and followed by eight consecutives write data to the first eight locations of the 64D locations of the CGRAM. To display the CGRAM pattern we need to run the same instruction as to display the pattern in the CGROM, thus run the write command (80H + the DDRAM address where the pattern wants to be displayed on the LCD followed by the run of the write data instruction; the data is 00h. The same steps must be taken for the rest of the pattern saved in the CGRAM addresses (8D-63D); to study how to display a custom use pattern on the LCD in-depth one can visit the following site and download the paper, http://www.arpnjournals.org/jeas/research_papers/rp_2016/jeas_1016_5086.pdf [15].

G. Set DDRAM address

As mentioned before (above) to display one character on the specific location of the LCD we first have to write an instruction to IR; the instruction is "80h + the location of the character to be displayed on the LCD that has the associated addresses of DDRAM" which is on the controller before it can be displayed on the LCD; see the instruction format below.

After sending the mentioned above instruction to the Instruction Register-IR we need to write the data of the character (ASCII) to display on the LCD to the Data Register; to display the character stored on the CGROM or CGRAM all we need to send is the location of the character stored on either CGROM or CGRAM instead of the code of the character. The format of the instructions to do that is as follows,

RS	R/$\overline{W}$	D7	D6	D5	D4	D3	D2	D1	D0
0	0	1	DDRAM address						

From the instruction format above we can see that the DDRAM address has a memory space of 128 locations (0000000b-1111111b) or 00H-7FH but the addresses that can be accessed only 80 locations from 00H-27H and 40H-67H each of them is 40 location memory spaces. For 2 × 16 LCD the addresses are limited to 32 locations with the first line DDRAM addresses of 00H-0FH (0D-15D) and the second line with the associated addresses range 40H-4FH (64D-79D).

H. Read busy flag and address

Before we can display data on the LCD, or read, change the data on the DDRAM first we must ensure the LCD module is in a not-busy state. The format to read busy Flag-BF of the LCD status is shown below,

RS	R/$\overline{W}$	D7	D6	D5	D4	D3	D2	D1	D0
0	1	0	0	0	0	0	0	0	1

The results of the reading from busy flag will appear as shown in the table below.

D7	D6	D5	D4	D3	D2	D1	D0
BF (*busy flag*)			AC (*address counter*) contents				

From the table above we can see the busy flag-BF can be read on the 7th bit (D7), while the D0-D6 bit contains the address counter (cursor position). If D7 (BF) = 0 indicates the controller is not busy, on the contrary, if D7 (BF) = 1 indicates the controller is in a busy state. For example, the program to check busy bits or busy flag-BF can be seen in a published paper in the International Journal with the following website address, https://www.sciencedirect.com/science/article/pii/s1110016817300546 [16].

I. To write data to DDRAM or CGRAM

The format of the instruction to write data to DDRAM or CGRAM has the same instruction format as shown below. The operation is a writing data to the Data Register-DR; RS = 1 and R/$\overline{W}$ = 0.

RS	R/$\overline{W}$	D7	D6	D5	D4	D3	D2	D1	D0
1	0	DDRAM or CGRAM address							

Instruction format to read data from DDRAM or CGRAM has the same instruction format as shown in the table below,

RS	R/$\overline{W}$	D7	D6	D5	D4	D3	D2	D1	D0
1	1	DDRAM or CGRAM address							

The operation is a read operation (R/$\overline{W}$ = 1) of the content of DDRAM or CGRAM (RS = 1).

1.4 LCD initialization

Before it can be used as a display for the MCS51 or other microprocessor or microcontroller, the LCD controller must first be configured or initialized by sending instructions to the Instruction Register-IR of the LCD controller. Before you can send instructions from the MCS51 to the LCD controller through P0 (P0.0-P0.7) we must choose the Instruction Register-IR on the LCD controller, by making RS = 0, resetting R/$\overline{W}$ (R/$\overline{W}$ = 0), and make a change of signal level to the $\overline{Enable}$ Pin; from logic 1 to logic 0, see **Table 2** for more details.

The LCD configuration process is to set the number of lines to be used (1 line or 2 lines), character font size (5 × 8 dot or 5 × 10 dot), the data width used (4 bits or 8 bits), and others. In this chapter, we only discuss the initialization of the LCD controller with a data width of 8 bits and 5 × 8 dot character font size.

We will develop some subroutines for this initialization process; to make it easier to use them in the future (just by calling the needed subroutines). The subroutines to be developed includes the write instruction subroutine, the write data subroutine, and the delay subroutine.

Because the initialization process is a write operation, we name this subroutine as a write_Insli subroutine. Before we build this subroutine, let us recall

the Pins connection between the LCD module and MCS-51 for this initialization process.

For the initialization process the RS Pin must be reset (RS = 0) this is because we want to access the Instruction Register- IR to provide initialization instructions/commands. RS Pin of the LCD controller is connected to P3.6 of MCS-51, for this reason, to reset this Pin (RS = 0) we can do it by running the CLR P3.6 statement.

As mentioned above initialization is a write operation, Pin R/$\overline{W}$ must be reset (R/$\overline{W}$ = 0) but due to the corresponding Pin being connected directly to the power supply GND we do not need to develop a subroutine to do that.

$\overline{Enable}$ Pin of the LCD is connected to P3.7 of MCS-51; it must change from logic 1 to logic 0 to enable the process to be executed properly during this slot of time then we can write or develop some instructions for this process as follows,

```
SETB P3.7                    ; Enable (P3.7 = 1)
CALL delay                   ; call delay subroutine
CLR P3.7                     ; Enable (P3.7 = 0)
```

Thus, we can build the write_Insli subroutine as shown in subroutine write_Insli below.

```
write_instruction   :
MOV P0,r1           ; fill P0 (D0-D7) with initialization instruction
                    ; which is in r1
CLR P3.6            ;RS = 0 to select instruction register
SETB P3.7           ; Enable get logic 1
CALL delay          ; give appropriate delay in accord
                    ; with 2 × 16 LCD datasheet
CLR P3.7            ; Enable get logic 0
RET                 ; return to the main program from the write_instruction subroutine
```

Based on datasheets from LCD 2 × 16 and the delay time needed to be around 1000 μs [17]. For that, we can build the subroutine delay as follows,

```
delay:
                    MOV r0,#0h          ; fill r0 with 0
delay1:             MOV r7,#0fh         ; fill r7 with 0Fh (15 s)
delay2:             DJNZ r7,$           ; r7 = r7-1, if r7 is not equal to 0 jump to delay 2
                    DJNZ r0,delay1      ; r0 = r0-1, if r0 is not equal to 0 jump to delay 1
                    RET                 ; return to the calling program or the main program
```

To analyze the delay time **Table 5** below can be used [18, 19],

From **Table 5** above it can be seen clearly that the delay time is 546 MC (546 × 10^{-6} s) or can be written as follows 546 μs; you can adjust the content of register r0 and r7 to obtain the recommended 1000 μs (enable cycle time)

	Instructions	Number of MC-machine cycles	Time in seconds (1MC = 10^{-6} s)
	MOV r0,#00h	1	10^{-6} s
delay1:	MOV r7,#0fh	1	10^{-6} s
delay2:	DJNZ r7, delay2	30 (15 × 2)	30×10^{-6} s
	DJNZ r0, delay1	512 (256 × 2)	512×10^{-6} s
	RET	2	2×10^{-6} s

Table 5.
Analyzing delay [18, 19].

to ensure the LCD is really ready to receive the next command (not in a busy state).

To write data to the LCD is a very similar process to write instruction to the LCD (Instruction Register-IR) as shown in the write_inst subroutine; the only difference between them is the logic state of RS; instead of RS = 0 we give RS a logic 1 (RS = 1) for this process; write data to the Data Register-DR. The subroutine to write data to the Data Register-DR is called write_data as shown below,

```
write_data:
MOV P0,r1        ; fill in P0 (D0-D7) with the data you want to display
                 ; which is in r1
SETB P3.6        ;RS = 1 to select data register
SETB P3.7        ; Enable get logic 1
CALL delay       ; call delay subroutine to give a delay in accord
                 ; with 2 × 16. LCD datasheet
CLR P3.7         ; Enable get logic 0
RET              ; return from subroutine write_instruction
```

From the 2 × 16 LCD control data [20], there are some things we have to do for the initialization process, namely

- *Function set* (set the width of the data, number of lines, and character font size) with the following instructions format

D7	D6	D5	D4	D3	D2	D1	D0
0	0	1	DL	N	F	X	X

NB:

DL ⇒ Data Length. If DL = 0, data width = 4 bits (D4-D7).

If DL = 1, data width = 8 bits (D0-D7).

N ⇒ number of line. If N = 0 the number of lines is 1, if N = 1 the number of lines is 2.

F ⇒ font size. If F = 0 then the font size is 5 × 8 dot, if F = 1 then the font size is 5 × 10 dot.

If we choose the data width of 4 bits, the process of sending or receiving data or instruction must be done twice. On this occasion we only discuss the 8-bit data width (Data Length-DL) with (DL = 1), the font size 5 × 8 dot (F = 0), the number of lines 2 (N = 1), bits D1, and D0 are filled with logic 0; therefore the instruction

that must be given to set the function set is 00111000B = 38H and we can use the instructions or statements below to do the function set.

MOV r1,#38h;	fill r1 with instruction (function set) 38h
CALL write_instruction;	CALL write_instruction

- *Entry mode set,* this initialization instruction has the following instructions format

D7	D6	D5	D4	D3	D2	D1	D0
0	0	0	0	0	1	I/D	S

Note Below (NB):

D1 (I/D) ⇒ Increment/decrement, if D1 (I/D) = 0 this bit will tell the controller to move the cursor to the left one column (D1 = 0) or move to right one column (D1 = 1) on condition D0(S) =0. But if D0(S) =1 the whole display (the cursor and the character or characters) will move to the left one column when D1 = 1 and to the right one column when D1 = 0; this instruction also increases or decrease the DDRAM address based on the D1 and D0 status. If D0 = 1 and D1 = 0 the DDRAM address will increase one column, but in the contrary if D0 = 1 and D1 = 1 the DDRAM address will decrease one column.

In this book, we will set S and I/D thus the instruction we send to configure the Entry Mode Set is 0000 0111b (07h). The instructions for initialization Entry Mode Set of the LCD we can use the statements below,

MOV r1,#07h;	fill r1 with instructions (entry mode set) 07h
CALL write_instruction;	call the write_instruction subroutine

- *Display on/off cursor,* these instructions have a format as shown below

D7	D6	D5	D4	D3	D2	D1	D0
0	0	0	0	1	D	C	B

This instruction is used to on or off the display, display or does not display the cursor, and set the cursor to blink or not.

When D2 (D-Display) = 1 then the display will be on the opposite if D2 = 0 then the display will be off, so is with the case of D1 (C-cursor) if D1 = 1 then the cursor will be displayed otherwise if D1 = 0 then the cursor will not be shown.

D0 (B-blink) as mentioned above is used to set the cursor to blink or not to blink if D0 = 0 then the cursor will not blink, but if D0 = 1 then the cursor will blink. To initialize the on /off cursor display we use the initialization of display on, cursor off, and blink off thus the instructions that must be sent to IR for it is (0Ch). The instructions used for the initialization of the display on/off cursor are shown below.

MOV r1,#0Ch;	fill r1 with instruction (display on/off cursor) 0Ch
CALL write_instruction;	call the write_instruction subroutine

- *Clear display* initialization instructions to clean the display; it has an instruction format as shown in the table below.

D7	D6	D5	D4	D3	D2	D1	D0
0	0	0	0	0	0	0	1

Thus for the initialization process to clean the display, the instruction that must be given to IR is (01h) and can be done by running the instructions below.

```
MOV r1,#01h                    ; fill r1 with instruction (clear display) 01h
CALL write_instruction         ; call the write_instruction subroutine
```

- *Shift cursor and display,* this initialization instruction is used to shift the cursor position or the entire display to the right and to the left. The format of instructions for this operation is as follows

D7	D6	D5	D4	D3	D2	D1	D0
0	0	0	1	S/C	R/L	X	X

The conducted operation is based on the logic status (logic level) of both bits D3 (S/C) and D2 (R/L) as shown in the table below (**Table 6**) [6, 14],

1.5 More instructions

The following are other commands besides the initialization command, write data, and write instructions mentioned above. These instructions or commands determine the Display Data RAM (DDRAM) address, read the status of the LCD (Busy Flag-BF), and specify the Character Generator RAM (CGRAM address).

1.5.1 Specifying DDRAM addresses

As mentioned earlier that each location on the LCD has an address that corresponds to the address on DDRAM; See Chapter 2 LCD Controller. To display a character on the LCD you must write an instruction (80h + the address location on the LCD where the character wants to be displayed) to the Instruction Register followed by writing the data (the character code you want to display) to the Data Register-DR.

Each time we want to read or change data on certain addresses on DDRAM we must write an instruction to the Instruction Register-IR to be able to access the address before carrying out the activities mentioned above (displaying characters on the LCD, read data on certain addresses from the DDRAM, and change data at the address). The format of the instruction for it is shown as follows,

D3 (S/C)	D2 (R/L)	Explanation
0	0	Shift the cursor position to the left
0	1	Shift the cursor position to the right
1	0	Shift the whole display to the right
1	1	Shift the whole display to the left

Table 6.
Shift cursor and dispaly [6, 14].

D7	D6	D5	D4	D3	D2	D1	D0
1	Address of DDRAM						

From the instruction format above we can see that the instructions we have to write to the Instruction Register-IR is 80H + the DDRAM addresses where the character is to be displayed; we also have to send signals (RS = 0, R/$\overline{W}$ = 0). For example, to put the cursor on the first line of the second column (on the LCD then we must provide the following instructions,

```
MOV r1,#82h                    ; fill r1 with 82h (80h + 02h)
CALL write_instruction         ; call the write_instruction subroutine
```

Likewise, if we want to put the cursor in the position of the first line of column 15 (0fh) then we must provide the following instructions, [10, 13].

```
MOV r1,#8Fh                    ; fill r1 with 8Fh (80h + 0Fh)
CALL write_instruction         ; call the subroutine write_instruction
```

1.5.2 Reading LCD busy status

Before we display data on the LCD, or read, change data on the DDRAM first we must ensure the LCD module is in a non-busy state. Format to read busy LCD status is shown below,

D7	D6	D5	D4	D3	D2	D1	D0
BF (*busy flag*)	AC (*address counter*) contents						

To be able to read the busy status of this LCD module we must reset RS (RS = 0), and set R/$\overline{W}$ (R/$\overline{W}$ = 1). As shown in the format of the busy/not busy of the status of LCD Bit 7 is the BF (busy flag) and bits 0- bit 6 are the contents of the Address Counter. In this book we will never read BF status; It is assumed that the LCD module remains on standby [6, 12, 15]. Most of the time people use a delay loop to make sure that LCD is ready not in a busy state; this is due to the grounded of Pin 5 (R/$\overline{W}$) of the LCD to the ground (GND) of the power supply. If you want to study how to read the BF status of the LCD module you can download and study a research paper published on the following site, https://www.sciencedirect.com/science/article/pii/S1110016817300546 [16].

1.5.3 Specifying the CGRAM address

To be able to read or write data on the CGRAM (Character Generator RAM), we must first determine the CGRAM address to be accessed. The format to access CGRAM can be seen below (RS = 0, R/$\overline{W}$ = 0); a write to the Instruction Register of the LCD with the command/instruction 40h.

D7	D6	D5	D4	D3	D2	D1	D0
0	1	CGRAM address					

The instructions below can be used to determine the CGRAM address (10H = 00010000B) [6, 12].

MOV r1,#01010000b	; fill r1 with (40h + 10h = 50h) to determine the CGRAM; address
	; at address 10h
CALL write_instruction	; call the write_instruction subroutine

From the format to access the addresses of the CGRAM we can see that the address of the CGRAM which is possible to access is from 000000b to 111111b or if written in the format of the hexadecimal is 00h to 3fh and in the format of decimal is 0d to 63d, thus the total address of the CGRAM is 64 bytes.

To save one custom pattern character with a size of 5 × 8 dot (5 columns × 8 lines) it takes eight address locations on CGRAM, thus 64 locations on CGRAM can only accommodate or store eight custom patterns; 1 memory location is occupied by 1 custom pattern line. If we make a storage memory map of CGRAM we can see it as shown on the CGRAM memory map below.

Pattern no.	**CGRAM address**	
	Hexa decimal	**Decimal**
1	00-07	0–7
2	08-0F	8–15
3	10-17	16–23
4	18-1F	24–31
5	20-27	32–39
6	28-2F	40–47
7	30-37	48–55
8	38-3F	56–63

For example, to create the character of the letter "T" then this data can be stored in eight consecutive CGRAM addresses (**Table 7**) [6].

And to make characters with alarm images can be seen in **Table 8** below,

If the custom pattern character size used is 5 × 10 dot then CGRAM will only be able to accommodate 6 custom pattern characters. Examples of programs to display special characters on the LCD can be seen in a published paper on the International Journal with the following website address, http://www.arpnjournals.org/jeas/research_papers/rp_ [15].

	Custom pattern						
	1	**8**	**4**	**2**	**1**	**Hexa**	**Decimal**
Row 1						1F	31
Row 2						04	4
Row 3						04	4
Row 4						04	4
Row 5						04	4
Row 6						04	4
Row 7						04	4
Row 8						04	4

Table 7.
Graph of the formation of a custom pattern for the letter T with a size of 5 × 8 dots [21].

	Custom pattern						
	1	**8**	**4**	**2**	**1**	**Hexa**	**Decimal**
Line 1						04	4
Line 2						0E	14
Line 3						0E	14
Line 4						0E	14
Line 5						0E	14
Line 6						1F	31
Line 7						04	4
Line 8						00	0

Table 8.
Graph of the formation of a custom pattern alarm image with a size of 5 × 8 dots.

1.5.4 To return the cursor to the initial position

To restore the cursor to the initial position (line 0, column 0) without changing the data on DDRAM. The format to do this is as follows (RS = 0, R/$\overline{W}$ = 0), D0 is marked with X we can fill it with either 0 or 1 (do not care) and we choose X = 0. Thus the instruction that must be written to the IR (Instruction Register) is 02H [10, 13].

D7	**D6**	**D5**	**D4**	**D3**	**D2**	**D1**	**D0**
0	0	0	0	0	0	1	X

The instructions below will return the cursor to the starting position (home).

```
MOV r1,#02h                    ; fill r1 with immediate data 02h
CALL write_instruction         ; call the write_instruction subroutine
```

1.6 LCD initialization subroutine

As mentioned earlier, before the LCD can be used it must be initialized/configuration initially. After studying the instructions for the initialization process above, we will build a subroutine for the initialization process and call it Init_LCD. In developing this subroutine all that needs to be done is to combine the entire initialization instructions that have been discussed previously. The init_LCD subroutine can be seen as follows,

```
init_LCD:
MOV r1,#01h                    ; clear display
CALL write_instruction
MOV r1,#38h                    ; function set, data size of 8 bit, 2 lines,
                               ; font size 5 × 8 dot
CALL write_instruction
MOV r1,#0Ch                    ; display ON, cursor OFF, blink OFF
CALL write_instruction
```

```
    MOV r1,#06h                    ; entry mode set, the display will not
                                   ; move, but the cursor will
                                   ; move to the right one column
    CALL write_instruction
    RET
```

1.7 Displaying a character on the LCD

The instruction or command to display a character on the LCD is 80h. To display a character at a certain location we must add 80H with the location address on the LCD DDRAM where the character is to be displayed. For example, if we want to display a character in the first row and the second column on the LCD we must add 80h + 01h = 81h, while to display the character on the second line of column 4 then we must write 80H + 43H = C3H to the Instruction Register-IR.

The program below is an example of a program to display character A on the second line of column 4 (C3H) [21].

```
                   Org 0h
                   CALL init_LCD          ; initialization of the LCD
    Start:
                   MOV r1,#C3h            ; fill r1 with C3h, put the character
                                          ; on row 2 column 4 of the LCD
                   CALL write_instruction ; call write_instruction
                   MOV r1,#'A'            ; fill r1 with 'A' the character
                                          ; to be displayed on the LCD
                   CALL write_data        ; call write_data
                   SJMP $                 ; halt or stop
```

The paper that can be used as a learning media for reading a 4 × 4 keypad and displaying the results of typing characters on the LCD can be seen at the following site address, http://www.arpnjournals.org/jeas/research_papers/rp_2018/jeas_0418_7024.pdf [22].

1.8 Displaying more than 1 character on the LCD

To display more than one character is almost the same as to display a single character. The program below is an example of a program to display seven characters "MANDATE" in the first line of the first column (80h) [21, 23].

```
                   Org 0h
                   CALL init_LCD          ; LCD initialization
    start:
                   MOV r1,#80h            ; fill r1 with 80h, put the character
                                          ; on LCD row 0, column 0
                   CALL write_instruction ; call write_instruction
```

	MOV r1,#'M'	; fill r1 with 'M' the character
		; to be displayed on the LCD
	CALL write_data	; call write_data
	MOV r1,#'A'	; fill r1 with 'A' the character
		; to be displayed on the LCD
	CALL write_data	; call write_data
	MOV r1,#'N'	; fill r1 with 'N' the character
		; to be displayed on the LCD
	CALL write_data	; call write_data
	MOV r1,#'D'	; fill r1 with 'D' the character
		; to be displayed on the LCD
	CALL write_data	; call write_data
	MOV r1,#'A'	; fill r1 with 'A' the character
		; to be displayed on the LCD
	CALL write_data	; call write_data
	MOV r1,#'T'	; fill r1 with 'T' the character
		; to be displayed on the LCD
	CALL write_data	; call write_data
	MOV r1,#'E'	; fill r1 with 'E' the character
		; to be displayed on the LCD
	CALL write_data	; call write_data
	SJMP $	; halt or stop

1.9 A 4-bit mode 2 × 16 LCD module

Figure 8 below shows a 4 bit mode 2 × 16 LCD module. Unlike the LCD in 8 bit mode in 4-bit mode only used 6 Pins as the LCD interface to other equipment (a microcontroller, a microprocessor, or a computer), 4 bidirectional data bus (D4-D7) (Pin 11-Pin 14), and 2 control signal RS (Pin 4) and E (Pin 6) [6, 24].

Data channels (Pin 7–Pin 10) D0-D3 are not used and connected to the ground of the power supply. Note the Pin (R/$\overline{W}$) is connected to the ground, thus in this mode, one can not read the status or data from the module all you can do only to write either instruction or data to the appropriate register of the LCD controller. If you want the status of the LCD and the data in it to be read then Pin (R/$\overline{W}$) must be connected to another of the controller that can provide a high signal for a read operation or a low signal for writing operations.

The advantage of using LCD in 4-bit mode compared to 8-bit mode is the number of Pins used as interfaces with microprocessors, microcontrollers, or computers is lesser, but also it has some shortcomings in programming; more complicated (data transmission must be done twice, the first is the high nibble data (D4-D7) then followed by sending low nibble (D0-D3) through D4-D7 data bus. Because the channels used are only 4 bits (D4-D7), a shift operation must be carried out to shift data from low nibble to high nibble (D4-D7) or swap instruction that exchanges between high nibble and low nibble.

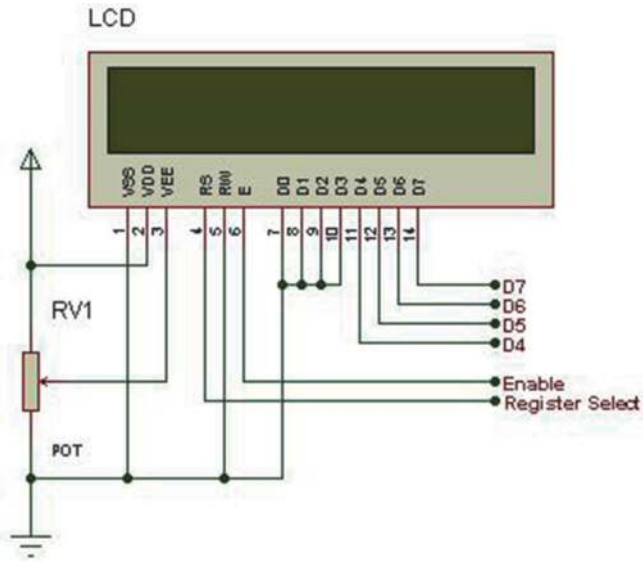

Figure 8.
4 bit mode 2 × 16 LCD module [6, 24].

1.10 Summary

LCD is an alternative display to the Light Emitting Diode-LED and seven segment display which are commonly used in various microcomputer and microcontroller applications or projects. LCDs are available in various sizes; size is usually based on the number of rows and columns. LCD is basically a module consisting of a display and controller. The controller is a microcontroller that has two types of memory (the DDRAM and the CGROM with CGRAM resides in it; CGROM is a non-volatile memory but not the CGRAM, it is a volatile memory) and two registers (the instruction register and the data register) in it. The LCD modules that are commonly used are LCD with a size of 2 × 16; has two lines and each line can display 16 characters.

The memory in the controller can be divided into two, the Display Data RAM-DDRAM, and Character Generator ROM-CGROM. DDRAM is a RAM memory (volatile memory) each address on the DDRAM has a corresponding address to each location on the LCD; where the character will be displayed. LCD with a size of 2 × 16, the first line has the DDRAM addresses corresponding to the LCD addresses of 00h-0Fh, while the second line has a DDRAM address corresponding to the LCD addresses of 40h-4Fh.

CGROM is a non-volatile memory where various characters are permanently stored in it by the manufacturer, see **Figure 5** to be displayed on the LCD by sending the address where the character is stored in the CGROM of the LCD controller; The command (80h + column in the first line where the character is to be displayed) is written to the LCD controller's instruction register.

To display the character on the second line, command C0h + column on the second line where the character is to be displayed in the LCD controller instruction register. The LCD controller manufacturer has also prepared a special location 00h-07h on the CGROM is a place to store special characters that are not provided by the LCD controller manufacturer. This special character is built by the user by sending a 40h command to the instruction register and sending the next eight bytes (the graph of the constructed character) to the data register eight times in sequence.

LCD can be operated in two modes, namely eight-bit mode and four-bit mode. In the eight-bit mode, the eight data busses of the LCD controller are connected to

the data bus of the microcontroller used to communicate between the two. In 4-bit mode, only four channels of the 8-bit width data bus of both LCD controllers and microcontrollers are used, namely D7-D4; taken from the data bus of the LCD controller, D3-D0 is not used and is connected to the power supply ground (GND). The eight-bit mode does use more wires than the four-bit mode, but is simpler in programming; the four-bit mode requires fewer wires but is more complicated to program and requires sending instructions and data twice.

In order to use the LCD module, it must be programmed first. By programming it, we can display the characters we want to display on the LCD. To enable it to display the characters, the LCD must first be configured/initialized first. Initialization is a process where we tell the LCD controller to display the characters on the LCD in the mode we want; font type, shift the cursor or the display to the left or to the right direction, blinking cursor or not, characters are displayed on the desired LCD row and column and so on. Basically, LCD programming consists of two operations, namely read and write operations; write or read the instructions to or from the instruction registers-IR or write the data to the data registers-DR. The write operations are used more than the read operation. The read operation is usually used to find out whether the LCD is busy or not (by reading the LCD controller's busy flag-BF); if the LCD is in a busy state, sending or reading the instructions/data to or from it will be useless. If you have connected Pin (R/$\overline{W}$) to the power supply ground and you could not read the busy flag-BF status anymore, you can use the delay loop time suggested by the LCD datasheet instead.

2. Questions

1. Explain the reasons for the widespread use of LCDs in the market today and mention equipment that uses LCDs as their display.

2. What is the real difference between active matrix LCD and passive matrix LCD?

3. Explain how the LCD works; associated with the term light modulating properties.

4. What is the light source of the LCD to operate and explain how it works?

5. Name the main parts of the LCD module.

6. Name and explain the types of memory owned by the LCD.

7. What does the term "initialization" mean?

8. What steps should be taken in carrying out write operations on the LCD module?

9. What steps should be taken in carrying out the read operation on the LCD module?

10. State the steps that must be taken in displaying the characters stored in the CGROM onto the LCD.

11. Plan to develop a program to display the character in the first row and 9th column [21, 25, 26].

12. State the steps to store a special character at address 4 (05h) of the CGRAM.

13. Draw a block diagram and explain the advantages and disadvantages of using LCD in four-bit and eight-bit modes.

14. By using the ASM-51build a program to display the character "A" in the second row and fourth column on the LCD [6, 25, 26].

15. From question no. 12 above, state the steps to be taken to display the special characters that have been stored at address 5 (05h) CGRAM on the LCD [6, 25, 26].

16. By using the ASM-51 build a program to display your name on the first line; the first letter of your name appears in the first column [6, 25, 26].

17. From problem no. 16 above plan and develop a program to display your name and shift it to the right one character each shift to the end of the first line and reappear in the first row and first column continuously [6, 25, 26].

Glossary

CGROM CGROM stands for Character Generator ROM; it is a ROM of the LCD module. This ROM is filled with 208 Dot Character Pattern 5×8 and 32 dot character pattern 5×10 that have been filled during its fabrication; Cannot be modified anymore. The address of the characters stored on this ROM will be sent to a certain memory location on DDRAM to be displayed on the LCD.

CGRAM CGRAM stands for Character Generator RAM; is one type of RAM of the LCD module. This RAM is basically a part of CGROM which is intentionally emptied by its manufacturer in order to be used by users to build special characters that are not prepared by the manufacturer to be displayed on the LCD (DDRAM); contains a point character made by the user; custom use character. There are eight characters with a 5×8 format and six characters with a 5×10 format. This memory is a volatile memory where the data will be lost if the energy source is removed from it; LCD does not get a voltage source.

DDRAM DDRAM stands for Display Data RAM. The DDRAM is a RAM used to save for a while the 8-bit character that will be displayed on the LCD; Both characters from CGROM and characters built by the user themselves and stored in CGRAM; The stored characters in CGRAM are volatile (will disappear if the power source is removed from it). The LCD with a size of 2 lines $\times$ 16 Characters, the address of the DDRAM for the first line is 00H-0FH and for the second line is 40H-4FH.

BPL BPL is an abbreviation of the Back Plane Light, it is the Sink or Ground of the light source for the LCD, in this case, is a LED; Cathode (−) Lead of the LED is connected to the BPL Pin and LED Anode (+) Lead is connected to the VCC voltage source. Light emitted by the LED is used to illuminate the displayed character on the LCD. Character brightness displayed on the LCD can be set up by using a variable resistor such as a potentiometer or a trimpot.

Write operation it is an operation to place data/instructions from outside the LCD controller to each register (the data register and the instruction register) inside the LCD controller to be further processed by the LCD controller; data to be displayed on the LCD and instructions to be executed.

Read operation it is an operation to read the busy bit of the LCD; to read the content of the instruction register. The read operation is generally used to read the

status (busy/not busy) of the LCD. If the LCD is busy we send neither data nor instructions to it; data/instructions will be lost if we send them to LCD.

Initialization it is a program that tells the LCD controller to set up the characteristics of the characters that we want to display on the LCD; set the number of lines of the LCD that we will use (1 line or 2 lines), the character font size (5×8 or 5×10); 5×8 (5 columns $\times$ 8 rows), 5×10 (5 columns $\times$ 10 rows), the LCD controller data width used (4 bits or 8 bits) and others.

The busy bit the busy bit is the 7th bit of the LCD controller data bus. We need to read the 7th bit (Busy Flag-BF) before giving further data or instructions to the LCD controller. This 7th bit will be a logic 1 (High) if it is busy and a logic 0 (Low) if it is not busy. The operation to read the 7th bit (Busy Bit) is an operation to read the contents of the Instruction Register IR; bit 7 in this case [22].

Author details

Dahlan Sitompul[1*] and Poltak Sihombing[2]

1 Fakultas Ilmu Komputer dan Teknologi Informasi, Department of Teknik Informatika, ATI-Immanuel Medan, and Program Studi S1 Ilmu Komputer, Universitas Sumatera Utara, Indonesia

2 Program Studi S1 Ilmu Komputer, Fakultas Ilmu Komputer dan Teknologi Informasi, Universitas Sumatera Utara, Indonesia

*Address all correspondence to: drps62@yahoo.com

References

[1] Difference between Active Matrix LCD and Passive Matrix LCD. Available from: https://www.geeksforgeeks.org/difference-between-active-matrix-lcd-and-passive-matrix-lcd

[2] Passive LCD. Available from: https://www.usmicroproducts.com/displays/custom-passive-lcd

[3] LCD block diagram. Available from: https://lh3.googleusercontent.com/0mMcFMaYqEzSjY2YLvcSSMR-NRy-xYPsUcKEMJoj_8iZCxG20T9cBA2ZTayD7d5ISJcrgA=s85

[4] The types and the size of the LCD. Available from: https://www.engineersgarage.com/character-lcd-pinout-working

[5] Interfacing LCD to 8051. Available from: https://www.dnatechindia.com/images/stories/8051_Tutorial/Interfacing_LCD_to_8051_/2x16_Line_Alphanumeric_LCD_Display.jpg

[6] Poltak Sihombing DS. A-Z Microcontroller 8051 Perangkat keras, Antar Muka, Pemograman dan Aplikasi. Medan: USU Press; 2019

[7] LCD Addressing. Available from: http://web.alfredstate.edu/faculty/weimandn/lcd/lcd_addressing/lcd_addressing_index.html

[8] Two line LCD. Available from: https://lh3.googleusercontent.com/IgFjlJwrahfBNE4l4-6_C44w-YB_rLEMhr8ox4qvdhQaTGLolft7ZHNEZY9hSV4FeaUm=s170

[9] LCD16 × 2 CGROM and CGRAM. Available from: https://www.handsonembedded.com/wp-content/uploads/2018/04/char_codes.png

[10] LCD Commands HD44780 instruction set. Available from: https://mil.ufl.edu/3744/docs/lcdmanual/commands.html

[11] LCD character 2 × 16. Available from: https://www.mytutorialcafe.com/image/gambar41.gif

[12] The Schematics of the LCD. Available from: https://www.microcontrollerboard.com/lcd.html

[13] Commands and Instruction set. Available from: https://www.8051projects.net/lcd-interfacing/commands.php

[14] LCD Commands. HD44780 instruction set. Available from: https://mil.ufl.edu/3744/docs/lcdmanual/commands.html#Sca

[15] Mhd Furqan DRPS. Developing a teaching media of microcontroller 8051 in displaying CGRAM character on LCD by using the MCU 8051 IDE and ASM51 in supporting ALFHE. ARPN Journal of Engineering and Applied Sciences. 2016; **11**(19):11363-11367. Available from: http://www.arpnjournals.org/jeas/research_papers/rp_2016/jeas_1016_5086.pdf

[16] Dahlan Sitompul PS. A teaching media of using busy bit and SDCC in displaying character string on LCD in MCU 8051 IDE. Alexandria Engineering Journal. 2018;**57**(2):813-818. Available from: https://www.sciencedirect.com/science/article/pii/S1110016817300546

[17] Bus timing characteristics. p. 49. Available from: http://lcd-linux.sourceforge.net/pdfdocs/hd44780.pdf

[18] 8051/8052 Microcontroller Tutorial. 8051/8052 Instruction Set. Available from: https://www.hobbyprojects.com/8051_tutorial/8052_instruction_set.html#is

[19] Philips 80C51 Family Programmer's Guide. Available from: https://www.keil.com/dd/docs/datashts/philips/p51_pg.pdf

[20] All about circuits. How to Interface a 16×2 LCD Module with an MCU. Available from: https://www.allaboutcircuits.com/technical-articles/how-to-a-162-lcd-module-with-an-mcu/

[21] 8051 Microcontroller Assembly Language Programming. Available from: https://www.electronicshub.org/8051-microcontroller-assembly-language-programming/

[22] Dahlan Sitompul SDF. An active learning media of interfacing microcontroller 8051 to 4 × 4 keypad using mcu 8051 IDE and ASM-51. ARPN Journal of Engineering and Applied Sciences. 2018;**13**(8):2987-2992. Available from: http://www.arpnjournals.org/jeas/research_papers/rp_2018/jeas_0418_7024.pdf

[23] Microprocessors & Microcontrollers 8051 Assembly language Programming. Available from: https://amiestudycircle.com/free-samples/amie/chapters/8051-programming-sec-b-elect-m&m.pdf

[24] LCD connections in 4-bit mode. Available from: https://www.8051projects.net/lcd-interfacing/lcd-4-bit.php

[25] The 8051 Simulator for Teachers and Students. Available from: https://www.edsim51.com/

[26] MCU 8051 IDE. Available from: https://sourceforge.net/projects/mcu8051ide/