Chapter 6. LCD Display and IR Remote Control Applications

This chapter extends the (software enabled) serial communication of Chapter 5 into the applications of data display and IR remote controller which have many additional applications for projects and other designs.

1. LCD Displaying

Alphanumeric LCD display is very popular for many applications because we can quickly and easily display a result of calculation or measurement, or data for debugging purpose. Of course, as we discussed before, a computer monitor is an excellent tool for the same purpose, but when we build an embedded computing system, much smaller LCD is always useful. There also are graphic LCDs are available.

A LCD is different from a LCD module. A LCD is just a medium to display characters or graphics, it itself also cannot display. A LCD module contains, in addition to the display medium, an interface controller/driver for the LCD. A LCD controller/driver displays alphanumerics and symbols. The most popular LCD controller/driver is the Hitachi 44780 based LCD controller chip. A single HD44780 can display up to one 8-character line or two 8-character lines. It can be configured to drive a dot-matrix liquid crystal display under the control of a 4- or 8-bit microprocessor.

LCD Controller/Driver HD44780
Internally HD44780 has a 80x8-bit display data (DD) RAM for maximum 80 characters, and 9,920-bit character generator(CG) ROM for a total of 240 character fonts (208 character fonts with 5x8 dot size and 32 character fonts with 5x10 dot size), and a 64x8-bit character generator RAM for 8 character fonts (5x8 dot) and 4 character fonts (5x10 dot). It also covers Wide range of instruction functions, "HD44780 Standard Control and Command Code," such as display clear, cursor home, display on/off, cursor on/off, display character blink, cursor shift, and display shift. It contains a reset circuit that initializes the controller/driver after power on.

Display data RAM (DDRAM) stores display data represented in 8-bit character codes. Its extended capacity is 80x8 bits, or 80 characters. The area in display data RAM (DDRAM) that is not used for display can be used as general data RAM. The following table shows the relationships between DDRAM addresses and positions on the LCD.

Position	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
First line	00h	01h	02h	03h	04h	05h	06h	07h	08h	09h	0Ah	0Bh	0Ch	0Dh	0Eh	0Fh	10h	11h	12h	13h
Second line	40h	41h	42h	43h	44h	45h	46h	47h	48h	49h	4Ah	4Bh	4Ch	4Dh	4Eh	4Fh	50h	51h	52h	53h
Third line	14h	15h	16h	17h	18h	19h	1Ah	1Bh	1Ch	1Dh	1Eh	1Fh	20h	21h	22h	23h	24h	25h	26h	27h
Fourth Line	54h	55h	56h	57h	58h	59h	5Ah	5Bh	5Ch	5Dh	5Eh	5Fh	60h	61h	62h	63h	64h	65h	66h	67h

In addition to the CGRAM and DDRAM, HD44780 has two 8-bit registers: an instruction register (IR) and a data register (DR). The IR stores instruction codes, such as display clear and cursor shift, and address information for display data RAM (DDRAM) and character generator RAM (CGRAM). The IR can only be written from microprocessor. The DR temporarily stores data to be written into DDRAM or CGRAM and temporarily stores data to be read from DDRAM or CGRAM.

Data written into the DR from the microprocessor is automatically written into DDRAM or CGRAM by an internal operation. The DR is also used for data storage when reading data from DDRAM or CGRAM. When address information is written into the IR, data is read and then stored into the DR from DDRAM or CGRAM by an internal operation. Data transfer to the microprocessor is then completed when the microprocessor reads the DR. After the read, data in DDRAM or CGRAM at the next address is sent to the DR for the next read from the processor. By the register selector (RS) signal, these two registers can be selected. In 16F877 perspective, by controlling the RS line for IR or DR, and sending a DDRAM location for display position and a data for a character to display that position, we can display a character on a desired position.

In addition to the IR and DR, there is Address Counter (AC). The AC assigns addresses to both DDRAM and CGRAM. When an address of an instruction is written into the IR, the address information is sent from the IR to the AC. Selection of either DDRAM or CGRAM is also determined concurrently by the instruction. After writing into (reading from) DDRAM or CGRAM, the AC is automatically incremented by 1 (decremented by 1). The AC contents are then output to DB0 to DB6 when RS = 0 and RW=0.

There are two interfacing method to a microprocessor. The HD44780U can send data in either two 4-bit operations or one 8-bit operation. For 4-bit interface, only four bus lines (DB4 to DB7) are used for transfer: Bus lines DB0 to DB3 are disabled. The data transfer between the HD44780U and the microprocessor is completed only after the 4-bit data has been transferred twice. As for the order of data transfer, the high nibble (DB4 to DB7) are transferred before the low nibble (DB0 toDB3). The busy flag must be checked (one instruction) after the 4-bit data has been transferred twice. Two more 4-bit operations then transfer the busy flag and address counter data. For 8-bit interface, all eight bus lines (DB0 to DB7) are used.

This section will explore the control of a regular LCD module and a serial LCD module. One caution we all have to use is that not all LCD modules are the same: some with different characteristics and pin arrangement, etc. Therefore, before you try to connect a LCD to 16F877, you have to read the data sheet of the module you received or bought. However, once you make yourself familiar with the one presented in this section, on any module of LCD, you can easily change the physical connection and code to adapt to the changing characteristics.

LCD example

A regular LCD module we discuss here is one manufactured by Truly which can display 4 rows, 20 characters per row, with character dot matrix size of 5x8. The exact model number is MTC-C204. So we use 20x4 LCD display with HD44780 controller or equivalent.

The pin arrangement for the LCD module is listed below.

Pin NO.	Symbol	Level	Description
1	V_{SS}	0V	Ground
2	V_{DD}	5.0V	Supply voltage for logic
3	V_O	---	Input voltage for LCD
4	RS	H/L	H : Data, L : Instruction code
5	R/W	H/L	H : Read mode, L : Write mode

6	E	H, H → L	Chip enable signal
7	DB0	H/L	Data bit 0
8	DB1	H/L	Data bit 1
9	DB2	H/L	Data bit 2
10	DB3	H/L	Data bit 3
11	DB4	H/L	Data bit 4
12	DB5	H/L	Data bit 5
13	DB6	H/L	Data bit 6
14	DB7	H/L	Data bit 7
15	BLA	---	For LCD Backlight (Anode)
16	BLK	---	For LCD Backlight (Cathode)

A host microprocessor "talks" to the LCD cotnroller/Driver via the data bus and 3 control lines: Register Select (RS), Read/Write (RW) and Enable (E). This places minimal demands upon the microprocessor. Only when the host microprocessor writes to or reads from the LCD, is intercommunication required.

The Control and Display Command codes for communicating to HD44780 LCD controller/driver are shown below. These codes are good for any LCD module with HD44780 or equivalent processor as the controller/driver of the module.

Control/ Command	**Code** R S R W B 7 B 6 B 5 B 4 B 3 B 2 B 1 B 0	**Description**	**Execution Time with f=250Khz**
Clear Display	0 0 0 0 0 0 0 0 0 1	Clears all display and returns the cursor to the home position (Address 0)	1.64ms
Return Home	0 0 0 0 0 0 0 0 1 X	Returns the cursor to the home position (Address 0). Also returns the display being shifted to the original position.	1.64ms
Entry Mode Set	0 0 0 0 0 0 0 1 M S	Set cursor move direction (M=1 for increase, M=0 for decrease) and shift of display (S=1 for shifted and S=0 for not-shifted)	40µs
Display On/Off	0 0 0 0 0 0 1 D C B	Sets On/Off of a Display (D=1 for On and D=0 for Off), Cursor (C=1 for On and C=0 for Off), and Blinking (B=1 for Blink On and B=0 for Blink Off)	40µs
Shift	0 0 0 0 0 1 S R X X	Moves the cursor (S=1 for Shift and S=0 for Cursor Move) and shifts display (R=1 for Right and R=0 for Left Shift).	40µs
Set Function	0 0 0 0 1 L N F X X	Sets interface data length (L=1 for 8-bit and L=0 for 4-bit), number at display lines (N=1 for 2 line display and N=0 for 1 line display), and once character font (F=1 for 5x10 and F=0 for 5x7 dots)	40µs
Set CG RAM Address	0 0 0 1 <--Acg --->	Set the CG (Character Generator) RAM address (i.e., cursor address). CG RAM data is sent and received after this set.	40µs
Set DD RAM Address	0 0 1 <----Add---->	Set the DD (Display Data) RAM address. DD RAM data is sent and received after this set.	40µs
Read Busy Flag & Address	0 1 B <--Acount--->	Reads Bust Flag indicating internal operation is being performed (B=1 for Busy and B=0 for Ready) and read address counter (Acount) contents used for both DD and CG RAMs.	1ms
Write Data	1 0 <----DATA----->	Write DATA to DD or CG RAM	40µs
Read Data	1 1 <----DATA----->	Read DATA from DD or CG RAM	40µs

The LCD Waveform diagram below shows how a data is written to the LCD module. As seen, even though the data is written to the internal data register, it still cannot be displayed on the LCD unless a High-to-Low transition input of E(Enable) signal is provided to the module.

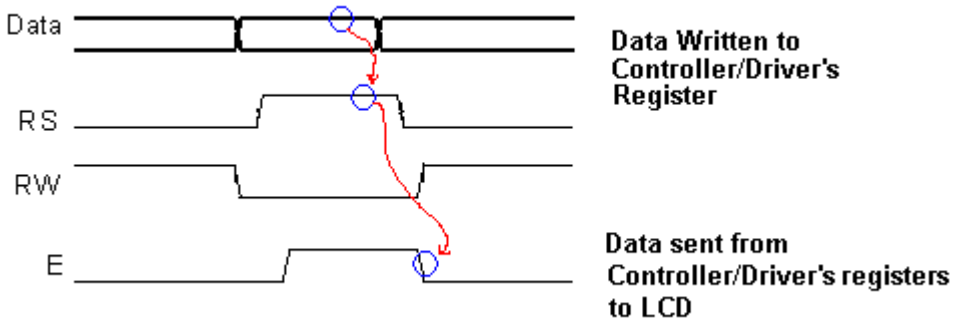


Fig. 24 LCD Waveform diagram

This High-to-Low transition input of E(Enable) signal is also needed when an instruction is written to the instruction register of the LCD controller/Driver. When your interface bit is 4, then we have to send the data twice, higher nibble then lower nibble. For each nibble write, we have to have the transitional E signal.

Initialization of LCD module

As mentioned above, HD44780 has an automatic reset circuit when power is on. The following instructions are executed during the initialization. The busy flag (B) is kept in the busy state until the initialization ends (B = 1). The busy state lasts for 10 ms after VCC rises to 4.5 V.

1. Display clear
2. Function set: 8-bit interface, 1-line display, 5x8 dot character font
3. Display on/off control: Display off, Cursor off, Blinking off
4. Entry mode set: Increment by 1, No shift (DDRAM is selected)

If the power supply condition does not reset properly, we have to initialize by instruction. Following is a usual LCD module initialization sequence by instruction.

1. Give power to the LCD module.
2. Wait for 15ms or more so that LCD is warm and ready to respond.
3. Set function for interface data length (i.e., 8 or 4 bits), number of display lines, and character dot matrix size.
4. Wait for 4.5 ms.
5. Check for Busy Flag.
6. Display Off.
7. Display Clear.
8. Set Entry mode.

Operation Example (8-bit interface with 8-digit 2 line display with internal reset)

We have many I/O ports in 16F877, so for this example, we try 4-bit interface and this requires total 11 pins. Assume that PORTB is assigned to the 8 data lines <DB7:DB0>. Since we usually do not read from we connect the RW line to the ground for always-reading status. The

busy flag checking, thus cannot be done by this configuration. However, giving enough time delay after writing an instruction or data is does the job. Then, we need two more lines for RS and E signal. Assume that they occupy two pins of PORTD.

LCDEven though the Truly LCD has four lines for display, internally, it is considered as 2 line display. It's done all by the DDRAM address selection as shown in the DDRAM address map. In other words, in 2 line mode, the first line can go from 00h to 40h, but since the LCD module can display only 20 characters, the first line starts from 00 but ends at 13h, then from 14h to 27h will be displayed at the third line. Similarly, the DDRAM addresses of 40h – 53h are displayed at the second line and those of 54-67h are displayed at the fourth line.

Display Position for 2 Lines

	1	2	3	4	5		20	21	22		39	40
DDRAM address (hexadecimal)	00	01	02	03	04	· · · · ·	14	15	· · · ·		26	27
	40	41	42	43	44	· · · · ·	54	55	· · · ·		66	67
	1	2					20	21	22		39	40

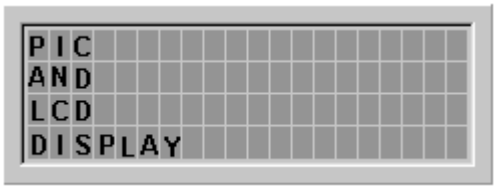
Display Position with 20x2 Format

		1	2	3	4	5		20	21	22		39	40
DDRAM address (hexadecimal)	Line 1	00	01	02	03	04	· · · · ·	14	15	· · · ·		26	27
	Line 2	40	41	42	43	44	· · · · ·	54	55	· · · ·		66	67
		1	2					20	21	22		39	40

Display Position with 20x4 Format

		1	2				20	1	2		19	20	
DDRAM address (hexadecimal)	Line 1	00	01	02	03	04	· · · · ·	14	15	· · · ·	26	27	Line 3
	Line 2	40	41	42	43	44	· · · ·	54	55	· · · ·	66	67	Line 4
		1	2				20	1	2		19	20	

The following example shows 8-bit interface (<DB7:DB0>) for 20x4 format with 5x8 dot matrix size. Note that RW is tied to ground for ever-writing mode. The steps and accompanying code will eventually display the four lines as shown below.



Step 1: Turn on Power to initialize the LCD. Give enough delay. There is no display.

A 16F877 instruction goes like this:

```
        call  delay10ms
        call  delay10ms   ;delay for 20ms
```

However, if you are not sure the power on reset actually work, you may have to follow the recommended initialization process. See the instructional initialization process.

Step 2: Function set for 8-bit, 2-line display, and 5x8 dot matrix.

RS=0

<DB7:DB0>= 0 0 1 1 1 0 X X

16F877 instruction for this is:

```
        movlw       0x38
        movwf       PORTB
        bcf         PORTD, RS
        bsf         PORTD, E
        bcf         PORTD, E          ;Transitional E signal
        call        delay10ms
```

The above instruction writing can be made into a subroutine.

```
;subroutine instw (instruction write)
;instruction to be written is stored in W before the call
instw movwf       PORTB
        bcf         PORTD, RS
        bsf         PORTD, E
        bcf         PORTD,E
        call        delay10ms
        return
```

Then, the above instruction can be rewritten to:

```
        movlw       0x38
        call        instw
```

Step 3. Display control: Display On, Cursor On, with no blinking are selected.

RS=0

<DB7:DB0>=0 0 0 0 1 1 1 0

Corresponding 16F877 code goes like:

```
        movlw       0x0E
        call        instw
```

Step 4: Entry mode set: Increment the DDRAM address by one and to shift the cursor to the right at the time of write to DDRAM. Display is not shofted

RS=0

<DB7:DB0>= 0 0 0 0 0 1 1 0

Corresponding 16F877 code goes like:

```
        movlw       0x0E
        call        instw
```

Step 5: Write data (i.e., 'P' of 50h in ASCII code) to DDRAM (The initial DDRAM address is set to 00h by the power on initialization.) So the line#1 position 1 is already selected by the reset. After this write, the cursor is incremented by 1 and shifted to the right.

RS=1
<DB7:DB0>= 0 1 0 1 0 0 0 0

Corresponding 16F877 code goes like:

```
      movlw       0x50
      movwf       PORTB
      bsf         PORTD, RS
      bsf         PORTD, E
      bcf         PORTD, E          ;Transitional E signal
      call        delay10ms
```

By changing the above code into a subroutine, we have the following code:

```
      movlw 0x50
      call  dataw

;subroutine dataw (data write)
dataw movwf       PORTB
      bsf         PORTD, RS
      bsf         PORTD, E
      bcf         PORTD, E          ;Transitional E signal
      call        delay10ms
      return
```

So we call `instrw` when RS=0 and `dataw` when RS=1.

Step 6: Write data (i.e., 'I' and 'C' next to 'P' in line #1) to DDRAM. Note that the DDRAM address in automatically incremented by one after each write, therefore, we do not write the DDRAM address (or position).

RS=1
<DB7:DB0>= 0 1 0 0 1 0 0 1 for 'I'
<DB7:DB0>= 0 1 0 0 0 0 1 1 for 'C'

Corresponding 16F877 code goes like:

```
      movlw 0x50              ;'I'
      call  dataw
      movlw 0x43
      call  dataw             ;'C'
```

Step 7. Set DDRAM address for the next 3 characters (A, N, and D) in line #2. The DDRAM address starts from 40h for the line #2.

RS=0
<DB7:DB0>= 1 1 0 0 0 0 0 0 for 1000000b

Corresponding 16F877 code goes like:

```
      movlw 0xC0              ;B'11000000'
      call  instw             ;RS=0
```

Step 8. Write the three characters, 'A', 'N', and 'D' to DDRAM. They are displayed at the line #2 from position 1.

RS=1
<DB7:DB0>= 0 1 0 0 0 0 0 1 for 'A'

<DB7:DB0>= 0 1 0 0 1 1 1 0 for 'N'
<DB7:DB0>= 0 1 0 0 0 1 0 0 for 'D'

Corresponding 16F877 code goes like:

```
movlw 0x41              ;'A'
call  dataw
movlw 0x4E
call  dataw             ;'N'
movlw 0x44
call  dataw             ;'D'
```

Step 9. Set DDRAM address for the next 3 characters (L, C, and D) in line #3. The DDRAM address starts from 14h for the line #3.

RS=0
<DB7:DB0>= 1 0 0 1 0 0 0 0 for 0010000b

Corresponding 16F877 code goes like:

```
movlw 0x94              ;B'10010100'
call  instw             ;RS=0
```

Step 10. Write the three characters, 'L', 'C', and 'D' to DDRAM. They are displayed at the line #3 from position 1.

RS=1
<DB7:DB0>= 0 1 0 0 1 1 0 0 for 'L'
<DB7:DB0>= 0 1 0 0 0 0 1 1 for 'C'
<DB7:DB0>= 0 1 0 0 0 1 0 0 for 'D'

Corresponding 16F877 code goes like:

```
movlw 0x4C              ;'L'
call  dataw
movlw 0x43
call  dataw             ;'C'
movlw 0x44
call  dataw             ;'D'
```

Step 11. Set DDRAM address for the next 7 characters (D, I, S, P, L, A, and Y) in line #4. The DDRAM address starts from 54h for the line #3.

RS=0
<DB7:DB0>= 1 1 0 1 0 1 0 0 for 11010100b

Corresponding 16F877 code goes like:

```
movlw 0xD4
call  instw             ;RS=0
```

Step 12. Write the seven characters, 'D', 'I', 'S', 'P', 'L', 'A', and 'Y' to DDRAM. They are displayed at the line #4 from position 1.

RS=1
<DB7:DB0>= 0 1 0 0 0 1 0 0 for 'D'
<DB7:DB0>= 0 1 0 0 1 0 0 1 for 'I'
<DB7:DB0>= 0 1 0 1 0 0 1 1 for 'S'
<DB7:DB0>= 0 1 0 1 0 0 0 0 for 'P'

<DB7:DB0>= 0 1 0 0 1 1 0 0 for 'L'
<DB7:DB0>= 0 1 0 0 0 0 0 1 for 'A'
<DB7:DB0>= 0 1 0 1 1 0 0 1 for 'Y'

Corresponding 16F877 code goes like:

```
movlw 0x44              ;'D'
call  dataw
movlw 0x49              ;'I'
call  dataw
movlw 0x53              ;'S'
call  dataw             ;
movlw 0x50              ;'P'
call  dataw
movlw 0x4C              ;'L'
call  dataw
movlw 0x41              ;'A'
call  dataw
movlw 0x59              ;'Y'
call  dataw
```

Step 13. Now let's move the cursor to the home position (position 1 of line #1) and set the DDRAM address to 0. This is done by the "return home" instruction.

RS=0
<DB7:DB0>= 0 0 0 0 0 0 1 0

Corresponding 16F877 code goes like:

```
movlw 0x02
call  instw             ;RS=0
```

Instructional initialization Process:

Step 1: When power on reset actually work, you have to follow the recommended initialization process and have the following codes at the very first line:

```
        call  delay10ms
        call  delay10ms
        movlw       0x30
        call        instw       ;see step 2 below for instw
```

Step 2: Function set for 8-bit, 2-line display, and 5x8 dot matrix. (Still part of initialization. And this step for setting is final and cannot be changed after this step.)

RS=0

<DB7:DB0>= 0 0 1 1 1 0 X X

16F877 instruction for this is:

```
        movlw       0x38
        call        instw
```

Step 3. Display off. (Still initialization process)

RS=0

<DB7:DB9>= 0 0 0 0 1 0 0 0

16F877 instruction for this step is:

```
        movlw       0x08
        call        instw
```

Step 4. Display Clear. (Still in the initialization process)

RS=0

<DB7:DB0>= 0 0 0 0 0 0 0 1

16F877 instruction for this step is:

```
        movlw       0x01
        call        instw
```

Step 5. Entry Mode Set (The last step of initialization) for increment and no shift

RS=0

<DB7:DB0>= 0 0 0 0 0 1 1 0

16F877 instruction for this step is:

```
        movlw       0x06
        call        instw
```

Hardware connection

Let's connect the 20x4 LCD module as shown below. Eight data bus lines are connected to PORTB, and E and RS are connected to PORTD<5> and PORTD<4>, respectively. RW is connected to PORTD<6>, but, as indicated above, since our main function is to write either command or data to LCD module, RW can be tied to the ground to make "write only" mode.

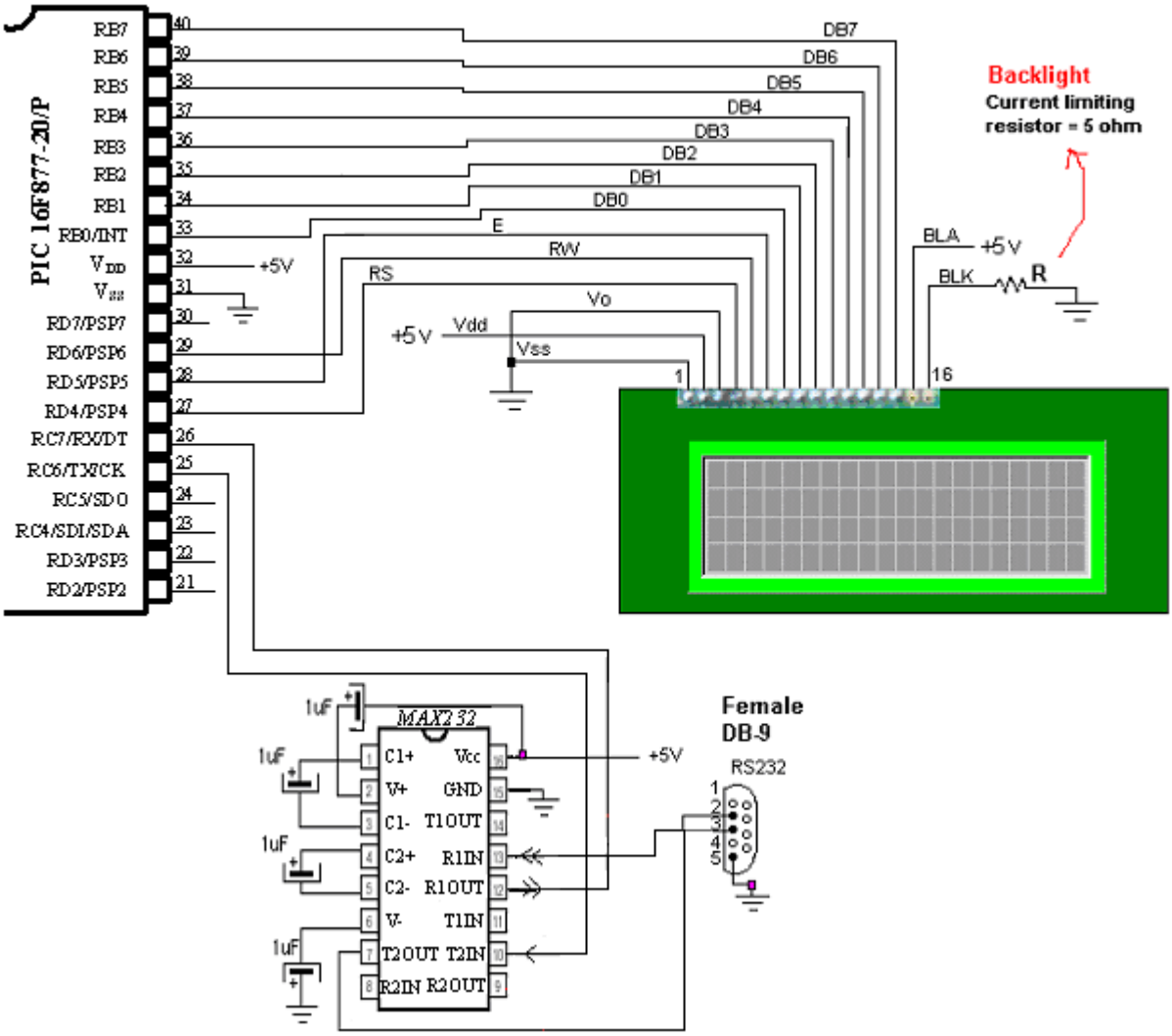


Fig 25. Hardware connection

Code example

Let's have an example code for the 8-bit interface mode control of a 20x4 LCD module. Follow the code carefully for instructions and comments.

```
;LCD-P.asm
;
;This program is to display an 20x4 LCD module
;by Truly (HD44780 compatible)
;
;8-bit interfacing
;
;Pin Connection from LCD to 16F877
;LCD (pin#) 16F877 (pin#)
```

```
;DB7 (14) -----RB7(40)
;DB6 (13) ----RB6(39)
;DB5 (12) ----RB5(38)
;DB4 (11) ----RB4(37)
;DB3 (10) ----RB3(36)
;DB2 (9)---- RB2(35)
;DB1 (8) ----RB1(34)
;DB0 (7) ----RB0(33)
;E (6) ------RD5(28)
;RW (5) -----RD6(29)
;RS (4) -----RD4(27)
;Vo (3) -----+5V
;Vdd (2) ----+5V
;Vss (1) -----GND
;
;Example code to display:
;     PIC
;     AND
;     LCD
;     DISPLAY
;

      list P = 16F877

STATUS      EQU   0x03
PORTB       EQU   0x06
TRISB       EQU   0x86
PORTD       EQU   0x08
TRISD       EQU   0x88
RS          EQU   0x04        ;RD4
E           EQU   0x05        ;RD5
RW          EQU   0x06        ;RW

;RAM arEA

      CBLOCK      0x20
            Kount120us   ;Delay count (number of instr cycles for delay)
            Kount100us
            Kount1ms
            Kount10ms
            Kount1s
            Kount10s
            Kount1m
      ENDC

;
;The Next 5 lines must be here
;because of bootloader arrangement
;Bootloader first execute the first 4 addresses
;then jump to the address what the execution directs
;=======================================================
        org       0x0000            ;line 1
        goto      START       ;line 2 ($0000)
        org       0x05

START
      BANKSEL     TRISD
; 1 for input, 0 for output
      movlw       0x00
      movwf       TRISD
      movwf       TRISB       ;RB<7:0> are all outputs

      banksel     PORTB
      clrf        PORTB
```

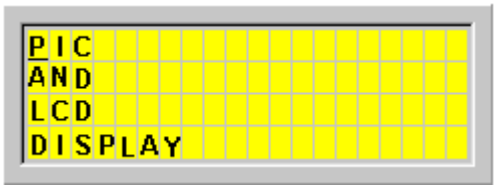
```
      clrf        PORTD       ;Here RW is pulled down to ground
;LCD routine starts
      call        delay10ms
      call        delay10ms
                  ;give LCD module to reset automatically
;Fundtion for 8-bit, 2-line display, and 5x8 dot matrix
      movlw       0x38
      call        instw
;Display On, CUrsor On, No blinking
      movlw       0x0E        ;0F would blink
      call        instw
;DDRAM address increment by one & cursor shift to right
      movlw       0x06
      call        instw
;DISPLAY CLEAR
      movlw       0x01
      call        instw

;Set DDRAM ADDRES
      movlw       0x80        ;00
      call  instw
;WRITE DATA in the 1st position of line 1
      movlw       0x50        ;P
      call        dataw

      movlw       0x49        ;I
      call        dataw

      movlw       0x43        ;C
      call        dataw

;Set DDRAM address for the 1st position of line 2 (40h)

      movlw       0xC0              ;B'11000000'
      call        instw             ;RS=0

;Write A, N, D

      movlw       0x41              ;A
      call        dataw
      movlw       0x4E
      call        dataw             ;N
      movlw       0x44
      call        dataw             ;D

;Set DDRAM address for the next 3 characters (L, C, and D) in line #3. (14h)
;The DDRAM address starts from 14h for the line #3.

      movlw       0x94              ;B'10010000'
      call        instw             ;RS=0

;Write the three characters, 'L', 'C', and 'D' to DDRAM.
;They are displayed at the line #3 from position 1.

      movlw       0x4C              ;L
      call        dataw
      movlw       0x43
      call        dataw             ;C
      movlw       0x44
      call        dataw             ;D

;Set DDRAM address for the next 7 characters (D, I, S, P, L, A, and Y) in line
#4.
;The DDRAM address starts from the line #4. (54h)
```

```
      movlw       0xD4
      call        instw             ;RS=0

;Write the seven characters, 'D', 'I', 'S', 'P', 'L', 'A', and 'Y' to DDRAM.
;They are displayed at the line #4 from position 1.

      movlw       0x44              ;D
      call        dataw
      movlw       0x49              ;I
      call        dataw
      movlw       0x53              ;S
      call        dataw             ;
      movlw       0x50              ;P
      call        dataw
      movlw       0x4C              ;L
      call        dataw
      movlw       0x41              ;A
      call        dataw
      movlw       0x59              ;Y
      call        dataw

;Now let's move the cursor to the home position (position 1 of line #1)
;and set the DDRAM address to 0.  This is done by the "return home"
instruction.

      movlw       0x02
      call        instw

IDLE  nop
      goto        IDLE

;====SUBROUTINES =====
;subroutine instw (instruction write)
;instruction to be written is stored in W before the call
instw movwf       PORTB
      call        delay1ms    ;delay may not be needed
      bcf         PORTD, RS
      call        delay1ms
      bsf         PORTD, E
      call        delay1ms
      bcf         PORTD,E
      call        delay10ms
      return

;subroutine dataw (data write)
dataw movwf       PORTB
      call        delay1ms    ;delay may not be needed
      bsf         PORTD, RS
      call        delay1ms
      bsf         PORTD, E
      call        delay1ms
      bcf         PORTD, E          ;Transitional E signal
      call        delay10ms
      return

;
;=========================================================
;DELAY SUBROUTINES

Delay120us
      banksel     Kount120us
      movlw       H'C5'       ;D'197'
      movwf       Kount120us
```

```
R120us
      decfsz      Kount120us
      goto        R120us
      return
;
Delay100us
      banksel     Kount100us
      movlw       H'A4'
      movwf       Kount100us
R100us
      decfsz      Kount100us
      goto        R100us
      return
;
;1ms delay
Delay1ms
      banksel     Kount1ms
      movlw       0x0A  ;10
      movwf       Kount1ms
R1ms  call        delay100us
      decfsz      Kount1ms
      goto        R1ms
      return
;
;10ms delay
; call 100 times of 100 us delay  (with some time discrepancy)
Delay10ms
      banksel     Kount10ms
      movlw       H'64' ;100
      movwf       Kount10ms
R10ms call        delay100us
      decfsz      Kount10ms
      goto        R10ms
      return
;
;

;1 sec delay
;call 100 times of 10ms delay
Delay1s
      banksel     Kount1s
      movlw       H'64'
      movwf       Kount1s
R1s   call        Delay10ms
      decfsz      Kount1s
      goto        R1s
      return
;
;
;10 s delay
;call 10 tiems of 1 s delay
Delay10s
      banksel     Kount10s
      movlw       H'0A'       ;10
      movwf       Kount10s
R10s  call        Delay1s
      decfsz      Kount10s
      goto        R10s
      return
;
;1 min delay
;call 60 times of 1 sec delay
Delay1m
      banksel     Kount1m
```

```
      movlw       H'3C' ;60
      movwf       Kount1m
R1m   call        Delay1s
      decfsz      Kount1m
      goto        R1m
      return
;=====================================================
      END
```

Run your program and see if you have the following display with an underscore cursor under 'P' of the fist line with lit backlight as shown below.



2. LCD Displaying: 4-bit Interface Example

Even though 16F877 has an ample amount of I/O pins, it's always wise to save a few pins for future use. Also, if we can achieve with fewer number of I/O pins the same function, there is no reason not to try the economical method. The 4-bit interface method is different from 8-bit interface only how we send the 8-bit data over 8 data lines or 4 data lines.
In 4-bit interface, we separate the 8-bit data by nibbles and send each nibble at a time. Therefore, for coding perspective, the only difference is the change in the subroutines of `instw` and `dataw`. Of course, we have to instruct the LCD module for 4-bit interface instead of 8-bit.

However, there is a slight odd step you have to have before setting the 4-bit interface. The HD44780 requires, for 4-bit interface only, to send the only the high nibble at the first step, and to send the high and low nibbles at the second step. In other words, the setting up for 4-bit interface has, unlike in 8-bit interface, an additional weird step. This is very important. If you miss this first step, you would some weird behavior from the LCD module such as one reset would show proper display and another would not.

The first step for function set for 4-bit interface:
RS=0
<DB7:DB4>=0 0 1 0
Then, the above instruction can be rewritten as:

```
      movlw 0x28
      call  hnibble4
```

with subroutine hnibble4;

```
hnibble4
      movwf       Temp        ;Temp storage
      movf        Temp,0            ;Now W also holds the data
      andlw       0xF0           ; get upper nibble
      movwf       PORTB         ; send data to lcd
```

```
      bcf         PORTB, RS
      bsf         PORTB, E
      call        delay1ms
      bcf         PORTB, E
      call        delay10ms         ;end of high nibble for 4-bit setup
      return
```

The second step for 4-bit interface now can set for for 4-bit, 2-line display, and 5x8 dot matrix:

RS=0

<DB7:DB0>= 0 0 1 0 1 0 X X

Then, the above instruction can be rewritten as (with X=0):

```
            movlw       0x28
            call        instw4
```

However, since we have to separate the byte into two nibbles and send each nibble separately, we have to change the `instw` subroutine to `instw4` subroutine.

```
;subroutine instw4 (4-bit interface instruction write)
;instruction to be written is stored in W before the call
instw4
      movwf       Temp        ;Temp storage
      movf        Temp,0      ;Now W also holds the data
      andlw       0xf0            ; get upper nibble
      movwf       PORTB          ; send data to lcd
      bcf         PORTB, RS
      bsf         PORTB, E
      call        delay1ms
      bcf         PORTB, E
      call        delay10ms         ;end of higher nibble
      swapf       Temp,0            ;get lower nibble to W
      andlw       0xf0
      movwf       PORTB             ;Write to LCD
      bcf         PORTB, RS
      bsf         PORTB, E
      call        delay1ms
      bcf         PORTB, E          ;end of lower nibble
      call        delay10ms
      return
```

Similarly, the data write subroutine `dataw` must also be changed to `dataw4` to reflect the change in data transmission.

```
dataw4
      movwf       Temp        ;Temp storage
      movf        Temp,0      ;Now W also holds the data
      andlw       0xf0            ; get upper nibble
      movwf       PORTB          ; send data to lcd
      bsf         PORTB, RS
      bsf         PORTB, E
      call        delay1ms
      bcf         PORTB, E
      call        delay10ms         ;end of higher nibble
      swapf       Temp,0            ;get lower nibble to W
      andlw       0xf0
      movwf       PORTB             ;Write to LCD
      bsf         PORTB, RS
      bsf         PORTB, E
      call        delay1ms
```

```
        bcf         PORTB, E          ;end of lower nibble
        call        delay10ms
        return
```

Additional change you have to bring to the code is to correctly assign the pins of RW, RS, and E to PORTB. As you see the following 4-bit interface illustration, we use only PORTB for a LCD module.

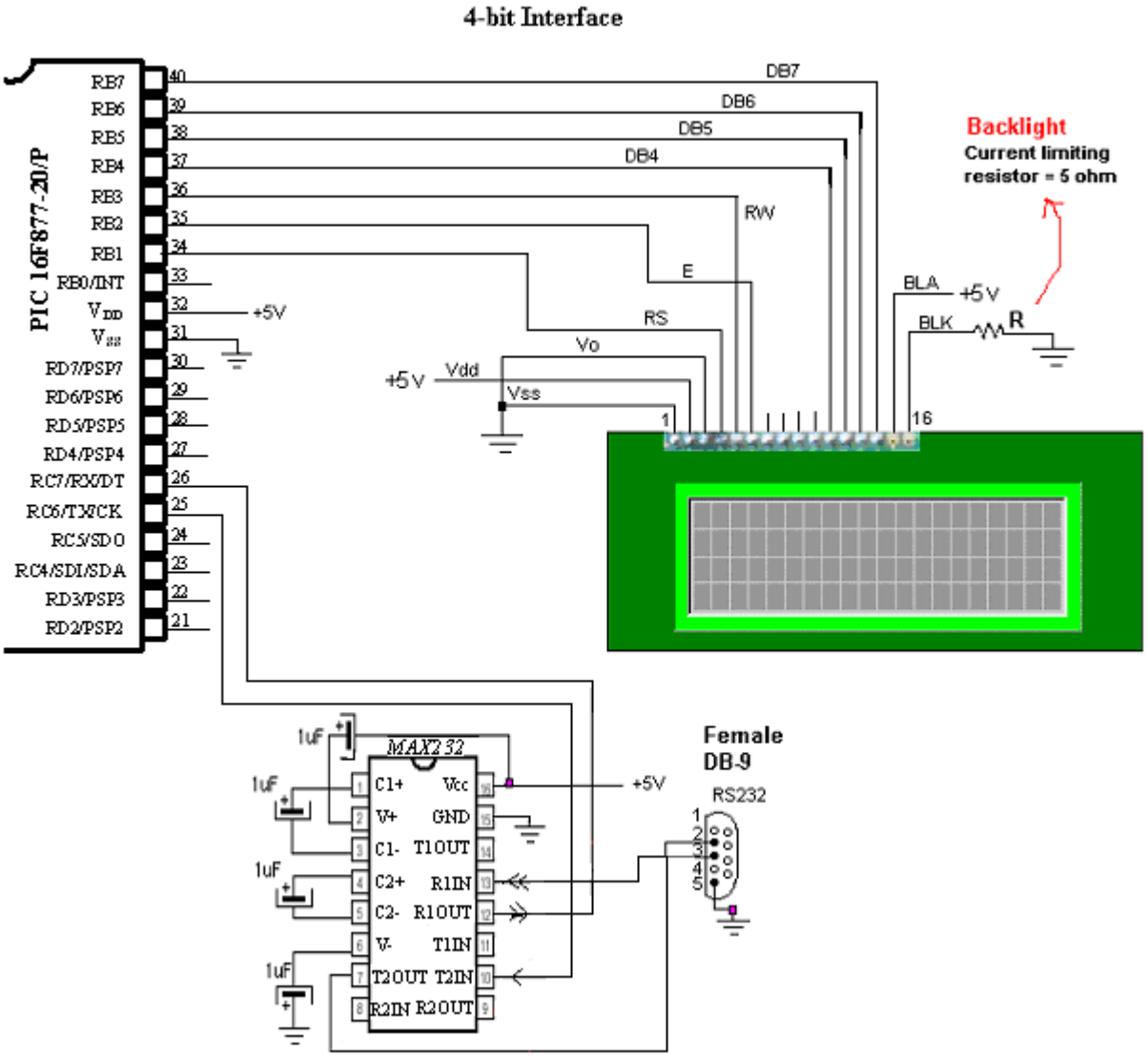


Fig. 26 4-bit Interface Illustration

Special Character Display using Character Generator ROM (CGROM)

The character generator ROM generates 5x8 dot or 5x10 dot character patterns from 8-bit character codes (See the CGROM character codes of HD44780 manual). It can generate 208 5x8 dot character patterns and 32 5x10 dot character patterns. User-defined character patterns are also available by mask-programmed ROM. So we can display even some weird characters. Let's add a few lines of instructions, then, to write a line of Alphabet and a line of symbol (or Greek) equivalent. From the CGROM map, we found that α, ρ, and μ are at E0, E6, and E4,

respectively. So by the following instruction should display the example display illustrated after the code.

```
;display a, r, m at line 1
;alpha, rho, and mu at line 2
;Set DDRAM ADDRESS for line 1
      movlw 0x80        ;00
      call  instw4
      movlw 'a'
      call  dataw4
      movlw 'r'
      call  dataw4
      movlw 'm'
      call  dataw4
;Set DDRAM ADDRES for line 2
;CGROM address for alpha, rho, and mu are E0, E6, and E4, respectively
      movlw 0xC0        ;00
      call  instw4
      movlw 0xE0
      call  dataw4
      movlw 0xE6
      call  dataw4
      movlw 0xE4
      call  dataw4
```