Chapter 7

Alpha Numeric Display

Using an Alpha Numeric Display or Liquid Crystal Display (LCD) helps to create a much better user interface for our projects. With the LCD we can give messages or instructions, show the results of calculations or measurements, display time and lots of other information.

In this chapter we will be looking at employing a 16 character by two line display, which incorporates an HITACHI HD44780 LCD Controller Driver Chip. The HD44780 is an industry standard also compatible with displays other than the Hitachi (fortunately).

The code we are developing can easily be used in other display formats such as 8 × 1, 16 × 1, 20 × 2, 20 × 4, 40 × 2, and so on.

The display is operated by three control lines and eight data lines.

- A Register Select which selects if a control or a character is written to the display.
- An Enable which when pulsed high will clock the data into the display.
- A Read/Write line. So we can read from the display or write to it.
- Data lines D0–D7.

Instead of using 11 I/O lines to communicate with the display we can operate with 6. Thus saving valuable I/O pins.

Since we are only going to write to the display we can tie the Read/Write pin low, Writing, so we do not require the line.

Instead of using eight data lines sending 8 bits of data, we can use four lines and send the data twice. So we only require four data lines.

The circuit diagram is shown in Figure 7.1.

CONFIGURING THE DISPLAY

Before writing to the display you first of all have to configure it. That means tell it if you are:

- using a 4 bit or 8 bit Microcontroller,
- using a 1 or 2 line display,
- using a character font size of 5 × 10 or 5 × 7 dots,

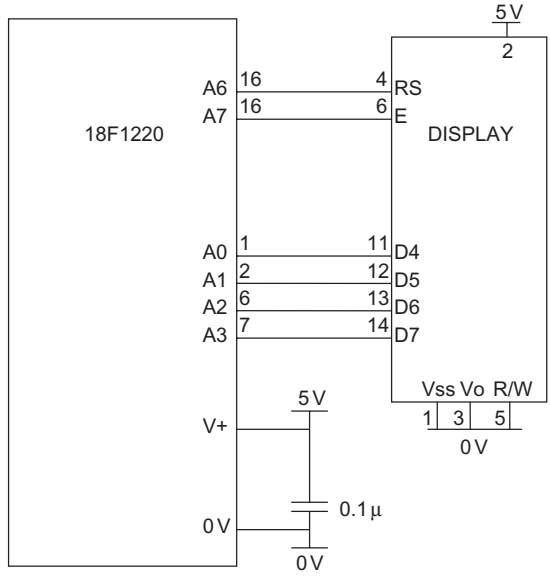


FIGURE 7.1 Circuit for the microcontroller driving the LCD with six lines.

- turning the display on or off,
- turning the cursor on or off,
- incrementing the cursor or not. The cursor position increments after a character has been written to the display.

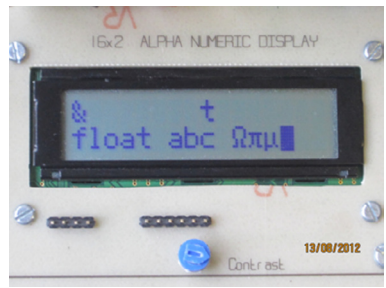
When configuring the display I have configured it for 4 bit operation, 2 lines, font size 5 × 7, turned the display and cursor on and incremented the cursor.

Microchip has written several routines to program the various display functions. They can be found in **Microchip\mplabc18\v3.40\h\xlcd.h** after you have installed MPLAB C18.

I have taken and modified these routines in order to make them easier to understand and use.

The program **LCD.C** written below gives an indication of the operation of the display by writing:

```
&       t
float abc Ωπμ
```



PROGRAM LCD.C

```
1.  // LCD.C by DW Smith, 25 January, 2012
2.  //E on A7, RS on A6, D4-7 on A0-3.
3.  #include <p18f1220.h>
4.  #pragma config WDT=OFF, OSC=INTIO2, PWRT=ON, LVP=OFF, MCLRE=OFF
5.  #include <delays.h>
6.  #define CLEAR_SCREEN          0b00000001
7.  #define FOUR_BIT              0b00101100
8.  #define LINES_5X7             0b00111000
9.  #define CURSOR_BLINK          0b00001111
10. #define CURSOR_RIGHT          0b00000110
11. #define DATA_PORT             PORTA
12. #define RS_PIN                PORTAbits.RA6
13. #define E_PIN                 PORTAbits.RA7
14. void Delay20TCY()             //20 clock cycles, 2.5 ms
    {
15. Delay10TCYx(2);
    }
16. void SetAddr(unsigned char DDaddr)
17. {
18. DATA_PORT &= 0xf0;            // Write upper nibble
19. DATA_PORT |= (((DDaddr | 0b10000000)»4) & 0x0f);
20. RS_PIN=0;                     // Set control bit
21. Delay20TCY();
22. E_PIN=1;                      // Clock the cmd and address in
23. Delay20TCY();
24. E_PIN=0;
25. DATA_PORT &= 0xf0;            // Write lower nibble
26. DATA_PORT |= (DDaddr&0x0f);
27. Delay20TCY();
28. E_PIN=1;                      // Clock the cmd and address in
29. Delay20TCY();
30. E_PIN=0;
31. }
32. void WriteCmd(unsigned char cmd)
33. {
34. DATA_PORT &= 0xf0;
35. DATA_PORT |= (cmd»4)&0x0f;
36. RS_PIN=0;                     // Set control signals for command
37. Delay20TCY();
38. E_PIN=1;                      // Clock command in
39. Delay20TCY();
40. E_PIN=0;
41. // Lower nibble interface
42. DATA_PORT &= 0xf0;
43. DATA_PORT |= cmd&0x0f;
44. Delay20TCY();
45. E_PIN=1;                      // Clock command in
46. Delay20TCY();
47. E_PIN=0;
48. }
49. void WriteChar(char data)
50. {
```

```
51.   DATA_PORT &= 0xf0;
52.   DATA_PORT |= ((data»4)&0x0f);
53.
54.   RS_PIN=1;                          // Set control bits
55.   Delay20TCY();
56.   E_PIN=1;                           // Clock nibble into LCD
57.   Delay20TCY();
58.   E_PIN=0;
59.   DATA_PORT &= 0xf0;                 // Lower nibble interface
60.   DATA_PORT |= (data&0x0f);
61.   Delay20TCY();
62.   E_PIN=1;                           // Clock nibble into LCD
63.   Delay20TCY();
64.   E_PIN=0;
65.   }
66.   void WriteString(const rom char *buffer)
67.   {
68.   while(*buffer)                     // Write data to LCD up to null
69.   {
70.   Delay20TCY();
71.   WriteChar(*buffer);                // Write character to LCD
72.   buffer++;                          // Increment buffer
73.   }
74.   return;
75.   }
76.   void main (void)
77.   {
78.   //SET UP
79.   // OSCCON defaults to 31 kHz. So no need to alter it.
80.   ADCON1=0x7F;//all IO are digital or 0b01111111 in binary
81.   TRISA=0b00100000;                  //sets PORTA
82.   PORTA=0b00000000;                  //turns off PORTA outputs
83.   TRISB=0b01000000;                  //sets PORTB as all outputs
84.   PORTB=0b00000000;                  //turns off PORTB outputs, good start position
85.   // this code configures the display
86.   WriteCmd (0x02);          // sets 4bit operation
87.   WriteCmd (FOUR_BIT & LINES_5x7);   // sets 5x7 font and multiline operation.
88.   WriteCmd (CURSOR_BLINK);           // blinks cursor
89.   WriteCmd (CURSOR_RIGHT);           // moves cursor right
90.   // Start of user program.
91.   WriteChar('&');
92.   SetAddr (0x88);
93.   WriteChar('t');
94.   SetAddr (0xC0);
95.   WriteString("float");
96.   WriteChar(' ');                    //space.
97.   WriteChar('a');
98.   WriteChar('b');
99.   WriteChar('c');
100.  WriteChar(' ');
101.  WriteChar(0xDE);                   //this is the omega sign.
102.  WriteChar(0xED);                   //this is the pi sign.
103.  WriteChar(0xEA);                   //this is the micro sign.
104.  while (1);                        //stop
105.  }
```

The Explanation of the Code

- Line 2 A comment indicates where the E, RS, and data lines are connected to the micro.
- Lines 3–5 are as previously discussed.
- Lines 6–10 define the display functions in terms of their code. Instead of writing 0b00000001 the code for CLEAR_SCREEN once it has been defined we can use CLEAR_SCREEN instead of 0b00000001 in our code.
- Lines 11–13 define where the DATA_PORT is connected, in this case PORTA and that the RS pin is on PORTAbits.RA6 and the E pin is on PORTAbits.RA7.
- Lines 14–15 is a 2.5 ms delay.
- Lines 16–31 is the code to set the character address on the screen.
- Lines 32–48 is the code for writing a command to the display.
- Lines 49–65 is the code which writes a character to the screen.
- Lines 66–75 is the code which writes a string of characters to the screen.
- Lines 76–84 should be familiar to you as the mico controller setup.
- Lines 86–89 configure the display using the WriteCmd function.

Lines 1–90 can be copied and pasted as the header for your display. The code writing to the display starts on line 90.

- Line 91 writes an **&** to the screen with the write character function.
- Line 92 moves the cursor to position 88 (eight places to the left on the top line) with the set address function, see Figure 7.3.
- Line 93 writes the **t** to the screen, at address 88.
- Line 94 moves the cursor to the beginning of the second line, the C0 position.
- Lines 95–103 writes **float abc Ωπμ** to the screen at the C0 position and
- Line 104 stops the program!!!

NB!!!!!!!!

If you do not want RS or E on the pins I have used that is ok, just change lines 12 and 13 to show where they are connected. If you do not want the data lines on PORTA you can put them on another PORT, in line 11, but be aware they have to go on bits0–3 on that PORT.

The display is capable of displaying 240 characters as shown in Figure 7.2.

Other Define Statements for the LCD

Some other defines that are useful in your display routines are:

```
#define LINE_5x7        0b00110011      // single line display
#define DISPLAY_ON      0b00001111
#define DISPLAY_OFF     0b00001011
#define CURSOR_ON       0b00001111
#define CURSOR_OFF      0b00001101
#define BLINK_ON        0b00001111
#define BLINK_OFF       0b00001110
```

FIGURE 7.2 The LCD character set.

For a more in-depth description of the LCD codes see:

Microchip\mplabc18\v3.40\h\xlcd.h
On your C: drive after installing MPLAB.

USING HEADER: dwsLCD.h

Just as we did with the scan routine in the keypad chapter we can include the LCD routines in a header file dwsLCD.h (lines 6–7 of the Program **LCD.C**) and then include the LCD file in line 6:

#include <dwsLCD.h>
Shown below in **LCDh.C**

PROGRAM LCDH.C

```
1.    // LCD.c by DW Smith, 25 January 2012
2.    //E on A7, RS on A6, D4-7 on A0-3.
3.    #include <p18f1220.h>
4.    #pragma config WDT=OFF, OSC=INTIO2, PWRT=ON, LVP=OFF, MCLRE=OFF
5.    #include <delays.h>
6.    #include <dwsLCD.h>
7.
8.    void main (void)
9.    {
10.   //SET UP
11.   // OSCCON defaults to 31 kHz. So no need to alter it.
12.   ADCON1=0x7F;                          //all IO are digital or 0b01111111 in binary
13.   TRISA=0b00100000;                     //sets PORTA
14.   PORTA=0b00000000;                     //turns off PORTA outputs
15.   TRISB=0b01000000;                     //sets PORTB as all outputs
16.   PORTB=0b00000000;                     //turns off PORTB outputs, good start position
17.   // this code configures the display
18.   WriteCmd (0x02);                      // sets 4bit operation
19.   WriteCmd (FOUR_BIT & LINES_5x7);      // sets 5x7 font and multiline operation.
20.   WriteCmd (CURSOR_BLINK);              // blinks cursor
21.   WriteCmd (CURSOR_RIGHT);              // moves cursor right
22.   // Start of user program.
23.   WriteChar('&');
24.   SetAddr (0x88);
25.   WriteChar('t');
26.   SetAddr (0xC0);
27.   WriteString("float");
28.   WriteChar(' ');                       //space.
29.   WriteChar('a');
30.   WriteChar('b');
31.   WriteChar('c');
32.   WriteChar(' ');
33.   WriteChar(0xDE);                      //this is the omega sign.
34.   WriteChar(0xED);                      //this is the pi sign.
35.   WriteChar(0xEA);                      //this is the micro sign.
36.   while (1);                            //stop
37.   }
```

This makes the program easier to read and debug.

CHARACTER DISPLAY ADDRESS

The address of the 16×2 characters on the display is shown in Figure 7.3.

- If you wish to write Hello World in the middle of the top line the code is:
  SetAddr (0x83);
  WriteString("Hello World");
- You can print out the word Hello with a delay between the letters so it looks like it is being typed with:
  SetAddr (0x85);
  WriteChar('H');
  Delay1KTCYx(4); //0.5 s delay
  WriteChar('e');
  Delay1KTCYx(4);
  WriteChar('l');
  Delay1KTCYx(4);
  WriteChar('l');
  Delay1KTCYx(4);
  WriteChar('o');

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

FIGURE 7.3 Character address.

DISPLAYING MEASUREMENTS

Suppose we have made a temperature measurement with our A/D convertor, as in the program **Temperature.C** and the temperature is stored in the variable TEMPERATURE.

Suppose the TEMPERATURE is 32, we need to separate the 3 and 2 to display both characters. We must declare two integer variables (or char variables) as:

```
int TEMPERATURE_TENS;
int TEMPERATURE_UNITS;
```

The TEMPERATURE_TENS value will be 3 and the TEMPERATURE_UNITS value will be 2.

To find the TEMPERATURE_TENS value do TEMPERATURE/10 and use the integer value:

```
TEMPERATURE_TENS=(int) TEMPERATURE/10;
```

To find the TEMPERATURE_UNITS value do TEMPERATURE%10 and use the integer value:

TEMPERATURE_UNITS=(int) TEMPERATURE%10; the % is dividing by 10 but using the remainder. 32/10 = 3 remainder 2.

So to display The TEMPERATURE
is 32°C

the code is:

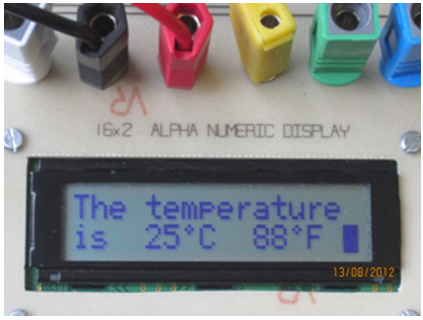
```
SetAddr (0x81);
WriteString("The TEMPERATURE");
SetAddr (0xC4);
WriteString("is");
WriteChar(0x30 + TEMPERATURE_TENS);      //code for 3 is 0x33
WriteChar(0x30 + TEMPERATURE_UNITS);     //code for 2 is 0x32
WriteChar(0xB2);                         //this is the degree sign.
WriteChar('C');
```

If the temperature is changed there is no need to rewrite all of the screen, move the cursor to the TEMPERATURE_TENS position, i.e.,C7 and just write the new temperature, i.e.:

```
SetAddr (0xC7);
WriteChar(0x30 + TEMPERATURE_TENS);
WriteChar(0x30 + TEMPERATURE_UNITS);
```

DISPLAYING ROOM TEMPERATURE

An example to display the room temperature using the LM335 temperature controller IC and the Alpha Numeric Display is shown below in Figure 7.4. The temperature is displayed in Centigrade and Fahrenheit.



The code for this is shown in **RoomTemperature.C** below. The Display routines have been included in the file dwsLCD.h.

ROOMTEMPERATURE.C

```
1.  // RoomTemperature.C by DW Smith, 8 March 2012
2.  //E on A7, RS on A6, D4-7 on A0-3.
3.  #include <p18f1220.h>
4.  #pragma config WDT=OFF, OSC=INTIO2, PWRT=ON, LVP=OFF, MCLRE=OFF
```

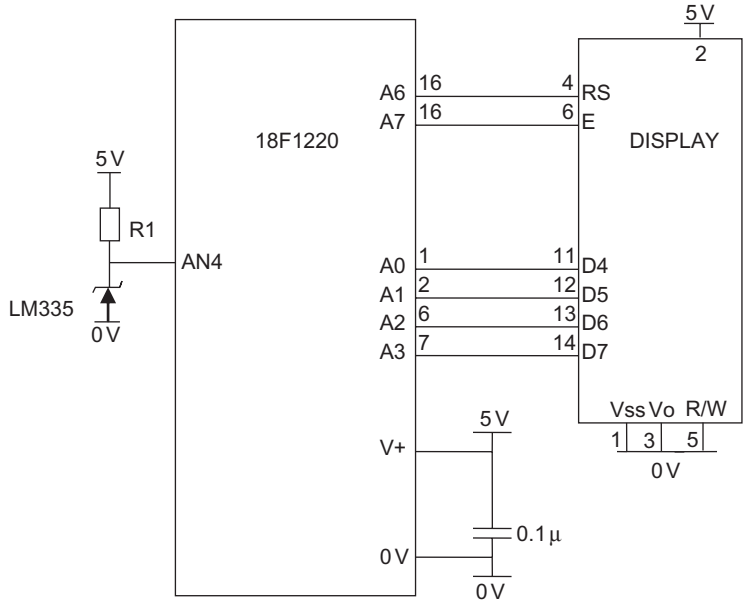


FIGURE 7.4 The temperature display circuit.

```
5.   #include <delays.h>
6.   #include <dwsLCD.h>
7.   int          TempC_TENS;
8.   int          TempC_UNITS;
9.   int          READING;
10.  float        TempC;
11.  float        TempF;
12.  int          TempF_TENS;
13.  int          TempF_UNITS;
14.  int          TempF_HUNS;
15.  int          TempF_Remain;
16.  void main (void)
17.  {
18.  //SET UP
19.  //OSCCON defaults to 31 kHz. So no need to alter it.
20.  ADCON1=0x6F;                    //AN4 set, rest are digital or 0b01101111 in binary
21.  TRISA=0b00110000;               //sets PORTA
22.  PORTA=0b00000000;               //turns off PORTA outputs
23.  TRISB=0b00000001;               //sets PORTB
24.  PORTB=0b00000000;                  //turns off PORTB outputs, good start position
25.  ADCON0bits.ADON=1;                 //turn on A/D
26.  ADCON2=0b10000000;                 //right justified, acquisition times are ok at 0
                                        with 31 kHz
27.  // this code configures the display
28.  WriteCmd (0x02);                   // sets 4bit operation
29.  WriteCmd (FOUR_BIT & LINES_5x7);   // sets 5x7 and multiline operation.
```

```
30.   WriteCmd (CURSOR_OFF);                  // cursor off
31.   WriteCmd (CURSOR_RIGHT);                // moves cursor right
32.   WriteCmd (CLEAR_SCREEN);
33.   WriteString("The temperature");
34.   SetAddr (0xC0);
35.   WriteString("is");
36.   SetAddr (0xC6);
37.   WriteChar(0xB2);                        //this is the degree sign.
38.   WriteChar('C');
39.   while (1)
40.           {
41.   ADCON0bits.CHS0=0;                      //selects CH.4 (100)
42.   ADCON0bits.CHS1=0;
43.   ADCON0bits.CHS2=1;
44.   ADCON0bits.GO_DONE=1;                   // do A/D measurement
45.   while (ADCON0bits.GO_DONE == 1);        //wait until bit=0 for measurement completed
46.   READING=ADRESL+(ADRESH * 256);
47.   TempC=READING/ 2.046−273.0;             //temperature in Centigrade
48.   TempC_TENS=(int) TempC/10;
49.   TempC_UNITS=(int) TempC%10;
50.   TempF=TempC*1.8 +32;                    //temperature in Fahrenheit
51.   TempF_HUNS=(int) TempF/100;
52.   TempF_Remain=(int) TempF%100;
53.   TempF_TENS=(int)TempF_Remain%10;
54.   TempF_UNITS=(int) TempF%10;
55.   SetAddr (0xC4);
56.   WriteChar(0x30 + TempC_TENS);
57.   WriteChar(0x30 + TempC_UNITS);
58.   SetAddr (0xCA);
59.   if(TempF_HUNS!=0) WriteChar(0x30 + TempF_HUNS);
60.   WriteChar(0x30 + TempF_TENS);
61.   WriteChar(0x30 + TempF_UNITS);
62.   WriteChar(0xB2);                        //this is the degree sign.
63.   WriteChar('F');
64.   WriteChar(' ');
65.         }
66.   }
```