Interfacing with an HD44780 based LCD with a PIC18F4520 microcontroller

Matthew Marcantonio

3/28/2012

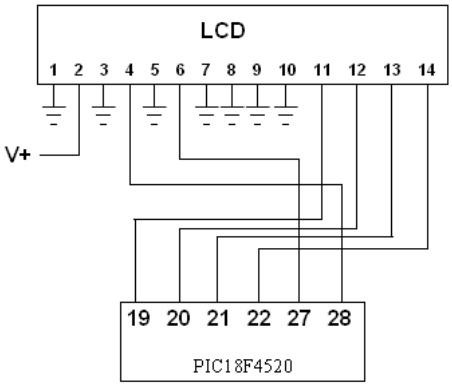
Abstract: Being able to visually display data from a microcontroller is very useful, a LCD can be programmed, but several steps must be taken to ensure proper operation. In this application note how to wire the LCD to the PIC18F4520, how to initialize the LCD, and how to send data to be displayed on the LCD will be discussed.

Keywords:
- PIC: Microchip's microcontroller line
- LCD: Liquid crystal display
- I/O: Input/Output of the microcontroller
- Nibble: 4 bits

Wiring the HD44780

HD44780 based LCDs have 14 pins, 11 of them are for interfacing with the LCD and the others for power and screen contrast. To save I/O pins on the microcontroller only 6 pins to successfully interface with the LCD, as opposed to 11

Since the LCD will be used in 4-bit mode, pins DB0 to DB3 can be grounded. We only wish to write to the microcontroller so the R/W pin can also be grounded, that leaves us with the enable pin, the register select pin, and DB4 to DB7 pins needed for the PIC.



The LCD interface will be set up on portD of the microcontroller, wire the LCD DB4 to DB7 on portD_0 to PortD_3, the enable pin to portD_4 and the register select to pin PORTD_5.

Initializing the Display

To send instructions to the LCD the data bits of the LCD are set and a pulse on the enable pin allows the LCD to read the data. The register select pin also plays a role, when it is low, the LCD interprets the data sent as an instruction to execute. When the register select pin is high, the LCD interprets the data pins as data.

Before the LCD can be used as a display, it needs to be initialized. The datasheet for the HD44780 has a process for initializing the LCD and it involves configuring the type of display attached to the HD44780. A flowchart of this initialization can be followed below

RS	RW	DB7	DB6	DB5	DB4
0	0	0	0	1	1
Enable Pulse					
0	0	0	0	1	1
Enable Pulse					
0	0	0	0	1	1
Enable Pulse					
0	0	0	0	1	0
Enable Pulse					
0	0	0	0	1	0
Enable Pulse					
0	0	N	F	X	X
Enable Pulse					
0	0	0	0	0	0
Enable Pulse					
0	0	1	D	C	B
Enable Pulse					
0	0	0	0	0	0
Enable Pulse					
0	0	0	0	0	1
Enable Pulse					
0	0	0	0	0	0
Enable Pulse					
0	0	0	1	ID	S

X = Don't Care

N = # Display lines; 0 = 1 line; 1 = 2 lines

F = font type; 0=5x7 display; 1 = 8x10 display

D = Display status; 0 = off ; 1 = on

C = Cursor status; 0 = off; 1 = on

B = Cursor Blinking; 0 = off; 1 = on

ID = Character Increment; 0 = off; 1 = increment +1 after character

S = Increment; 0 = off; 1 = on

Printing to the Display

Once the LCD has be initialized, characters can be printed the LCD using standard ASCII codes. Because characters are stored in ASCII code in C, passing a character to a function and reading it as an integer allows easy passing to the data ports.

Because a character is 8 bits and the LCD is configured in a 4 bit, the LCD reads data 4 bits at a time and combines the two 4 bit data nibbles into one full byte. The LCD will interpret the first nibble as the upper bits and the second nibble as the lower bits of the full byte. Since data is being sent, and not instructions the register select pin needs to be set high.

Appendix

Data sheets used:

http://www.sparkfun.com/datasheets/LCD/HD44780.pdf

http://ww1.microchip.com/downloads/en/DeviceDoc/39631a.pdf

Source Code:

```
#include <p18cxx.h>
#include <string.h>
#include <stdio.h>

#pragma config WDT=OFF
long int count;
char c; //Character to be printed to LCD

#define LCD_D4          PORTDbits.RD0
#define LCD_D5          PORTDbits.RD1
#define LCD_D6          PORTDbits.RD2
#define LCD_D7          PORTDbits.RD3
#define LCD_EN          PORTDbits.RD4
#define LCD_RS          PORTDbits.RD5


// Functions
void LCD_Init ( void );
void LCD_SetPosition ( unsigned int c );
void LCD_PutCmd ( unsigned int c );
void LCD_PulseEnable ( void );
void delay(void);
void upper (unsigned int c);
void lower(unsigned int c);
void LCD_PutChar ( unsigned int c );



void LCD_Init ( void ) //Initialize display
   {
            PORTD = 0;
            TRISD = 0x00;
            delay ();       /* wait enough time after Vdd rise */
            delay ();
            delay ();
            delay();
            LCD_RS =0 ;
            PORTD = 0x03;
            LCD_PulseEnable();
            delay ();
             LCD_PulseEnable();
            delay ();
              LCD_PulseEnable();
             PORTD =  0x02 ;   /* set 4-bit interface */
             LCD_PulseEnable();
             LCD_PutCmd ( 0x2C );    /* function set (all lines, 5x7 characters) */
            LCD_PutCmd ( 0x0C );    /* display ON, cursor off, no blink */
            LCD_PutCmd ( 0x01 );    /* clear display */
            LCD_PutCmd ( 0x06 );    /* entry mode set, increment & scroll left */
   }

void LCD_SetPosition ( unsigned int c )
   {
   /* this subroutine works specifically for 4-bit Port A */
   upper ( c | 0x08 );
```

```
    LCD_PulseEnable();
    lower ( c );
    LCD_PulseEnable();
    }



void LCD_PutCmd ( unsigned int c )
    {
    /* this subroutine works specifically for 4-bit Port A */
    upper ( c );     /* send high nibble */
    LCD_PulseEnable();
    lower ( c );      /* send low nibble */
    LCD_PulseEnable();
    }

 void LCD_PulseEnable ( void )
    {
    LCD_EN = 1;
    delay();         // was 10
    LCD_EN =0;
    delay();         // was 5
    }

void delay(void)
{
for(count = 1; count < 10000; count++);
}

void upper (unsigned int c)
{
                if(c & 0x80) LCD_D7=1; else LCD_D7=0;
                if(c & 0x40) LCD_D6=1; else LCD_D6=0;
                if(c & 0x20) LCD_D5=1; else LCD_D5=0;
                if(c & 0x10) LCD_D4=1; else LCD_D4=0;
}

void lower(unsigned int c)
{
                if(c & 0x08) LCD_D7=1; else LCD_D7=0;
                if(c & 0x04) LCD_D6=1; else LCD_D6=0;
                if(c & 0x02) LCD_D5=1; else LCD_D5=0;
                if(c & 0x01) LCD_D4=1; else LCD_D4=0;
}

void LCD_PutChar ( unsigned int c )
    {
    /* this subroutine works specifically for 4-bit Port A */

        LCD_RS =1;
        upper ( c );     /* send high nibble */
        LCD_PulseEnable();
        lower ( c );     /* send low nibble */
        LCD_PulseEnable();
        LCD_RS =0;

    }
```