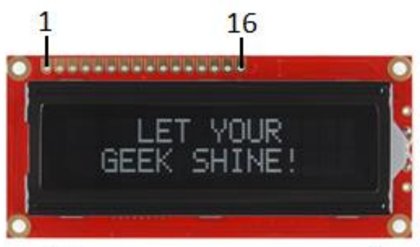
ECE 218
EXERCISE 6. INTERFACING LCD DISPLAY

In this exercise you will interface a 16x2 character LCD display to the PIC24 on the Microstick II board. You will make use of existing header and source library files that contain functions to initialize and write characters to the display. The LCD display interface includes a fair amount of wiring, so you will also practice your hardware prototyping skills.

HARDWARE SETUP

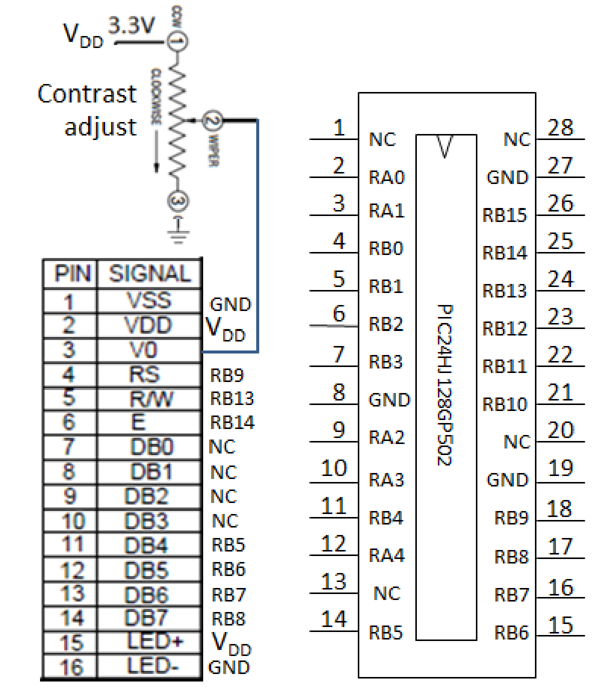
The LCD display has 4 bits of parallel data, (DB7-DB4) and 3 control signals (RS, R/W, E) that will be connected to 7 I/O pins on the PIC 24. The display has an 8-bit parallel data option as well, but the 4-bit version will be used to minimize the PIC24 I/O pins used. Figure 1a shows that the LCD display pins are numbered from left to right. Figure 1b defines the LCD display pins and gives the connections to the PIC24 and to the potentiometer that will provide a variable voltage for the Vo contrast input. For a more detailed description of the LCD pins, see Table 1 in the Appendix of this document.

The LCD display has two power inputs. One is the controller power connection (pin 2, labeled VDD) and the other powers a backlight LED that requires a positive voltage on pin 15 (LED+). We will use the 3.3V (V_{DD}) on the Microstick II board for both. Likewise, pin 1 and pin 16 of the LCD display are the negative voltages for the controller and the backlight, and we will connect both to GND.



Sparkfun AMD1602K-FSY-YBS/3.3V

a. LCD Display Pin Organization



PIN	SIGNAL	
1	VSS	GND
2	VDD	V_{DD}
3	V0	
4	RS	RB9
5	R/W	RB13
6	E	RB14
7	DB0	NC
8	DB1	NC
9	DB2	NC
10	DB3	NC
11	DB4	RB5
12	DB5	RB6
13	DB6	RB7
14	DB7	RB8
15	LED+	V_{DD}
16	LED-	GND

LCD Display Pins

1	NC	NC	28
2	RA0	GND	27
3	RA1	RB15	26
4	RB0	RB14	25
5	RB1	RB13	24
6	RB2	RB12	23
7	RB3	RB11	22
8	GND	RB10	21
9	RA2	NC	20
10	RA3	GND	19
11	RB4	RB9	18
12	RA4	RB8	17
13	NC	RB7	16
14	RB5	RB6	15

PIC24 Pins on Microstick II

b. LCD and PIC24 Connections

Figure 1. LCD Display Interface Circuit

Place the display on your breadboard so that you have some holes available for connections. Use the power and ground rails on the breadboard to keep your wiring neat. Make all of the connections shown in Figure 1b. Study the orientation of the LCD display and the orientation of the PIC24 on the Microstick II.

The 4-bits of parallel data will be used to send the LCD display control and data instructions. **Control instructions** include commands such as clearing the display, returning the cursor to the home position, turning the display or cursor on or off, etc. **Data instructions** are the characters that will be printed on the display.

Software Setup

You will use functions in a new library file. To gain access to these functions you will need to install the library files. Download the *lcd4bit_lib.c* and the *lcd4bit_lib.h* files from Nexus and install in the *pic24_code_examples/lib/src* and *pic24_code_examples/lib/include* folders, respectively. By placing the file here, and setting up your project as usual, these files will all be visible to your program. You will find the following functions from the library files useful:

configControlLCD() – this function configures the RS, RW and E control lines as outputs and initializes them low. It requires no parameters and returns no values.

initLCD() – this function executes the initialization sequence specified in the Hitachi HD44780 LCD controller datasheet, clears the screen and sets the cursor position to upper left (home). It requires no parameters and returns no values.

*writeLCD (uint8_t u8_**Cmd**, uint8_t **u8_DataFlag**, uint8_t **u8_CheckBusy**, uint8_t **u8_Send8Bits**)* – this function writes a command or data to the display. It requires 4 parameters as follows:

u8_Cmd: 4-bit value (bits DB3-DB0 of Table 2 in Appendix) to send to LCD
u8_DataFlag: 1 for sending data, 0 for sending a command
u8_CheckBusy –0 to use delay between transfers, 1 to check the busy flag
u8_Send8Bits – 0 for sending 4bits only, 1 for sending 8 bits

*outStringLCD(char *psz_s)* – this function sends out a string to the LCD display. The parameter should be a string.

Once the files are in place, start a new project (I suggest naming it Ex6_LCD) using the textbook library, following the steps from Exercise 3. Open the main_template_library.c file and save it as your main_LCD.c file for this exercise.

In the main part of the program, we will first write the following message to the LCD display, and then modify the program to scroll the message across the display.

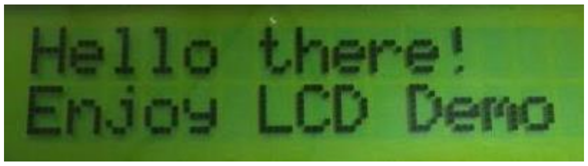


Figure 2. LCD Display Results

After the usual clock configuration, we will configure the control lines, and initialize the LCD display using the appropriate functions from the library. Then we will write the message in Figure 2 to the screen and create an empty infinite loop.

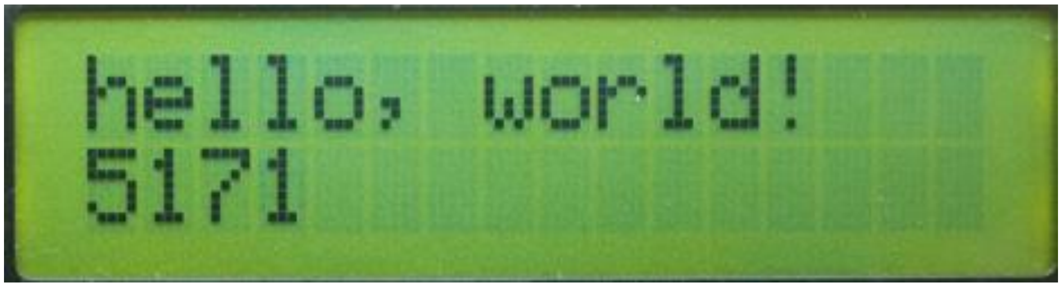
To write to the display, you will use the two functions: *writeLCD()* and *outStringLCD()*. *writeLCD()* sends commands to the LCD to do things like set the cursor move direction and set the cursor position. *outStringLCD()* sends an array of characters to be displayed.

The possible commands for the LCD display are given in Table 2 in the Appendix. We will use this table to understand what parameters to use for the *writeLCD* function. Further explanation is in the datasheet.

To set the cursor position, we will use the "Set DDRAM address in address counter" command. This is an 8-bit command, with the following format:

Command								**Description**	**Delay**
1	AC 6	AC 5	AC 4	AC 3	AC 2	AC 1	AC 0	Set DDRAM address in address counter	37 us

Notice that the first bit of the command is 1, and the other 7 bits, AC6 AC5 AC4 AC3 AC2 AC1 AC0, is the 7-bit address of the 32 character positions (2 rows of 16) on the screen, coded as in Figure 3.



00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
40	41	42	43	44	45	46	47	48	49	4A	4B	4C	4D	4E	4F

Figure 3. LCD Position Display Data RAM (DDRAM) Addresses

For example, to put the cursor at the top left, (position 0x00, where the 'h' character is), we write the command 0b10000000 to the LCD controller, with the instruction

writeLCD(0x80, 0, 1, 1); // write an 8-bit command, 0x80 (Set DDRAM address to 0x00)

To place it on the beginning of the second line, (position 0x40, where the '5' character is) we would write 0b11000000 to the LCD controller with instruction

writeLCD(0xC0, 0, 1, 1);

Once the cursor is placed, we can use the *outStringLCD()* command to write characters to the screen. If we want to send the special characters, we look up the CGRAM address in section 12 (Standard character pattern) in the datasheet, and send it as data. For example to display a ° we would use the instruction:

writeLCD(0xDF, 1, 0, 1);

Program 1 – Hello Message

Starting with the library template file, you will need to include the lcd4bit_lib.h file in your code to access the LCD library functions:

```
#include "lcd4bit_lib.h"
```

Here is some pseudocode for the main part of the first program to simply display a 2-line message on the screen.

```
Main program{
  Configure the clock and LCD control lines.
  Initialize LCD
  Write string “Hello there!”
  Write command to position cursor at 0x40
  Write string “Enjoy LCD Demo”
   Do forever {
   }
}
```

RUN PROGRAM

Make and program the device, or use the debugger to run the code using breakpoints to better understand it. Demonstrate the working program to your instructor.

Program 2 – Scrolling Hello Message

Next, in the main loop we will send the command to set the cursor moving to the right (see Table 2 in Appendix and the LCD controller datasheet on Nexus) and then delay about 200ms so that it does not scroll too quickly and appear to be overwriting itself.

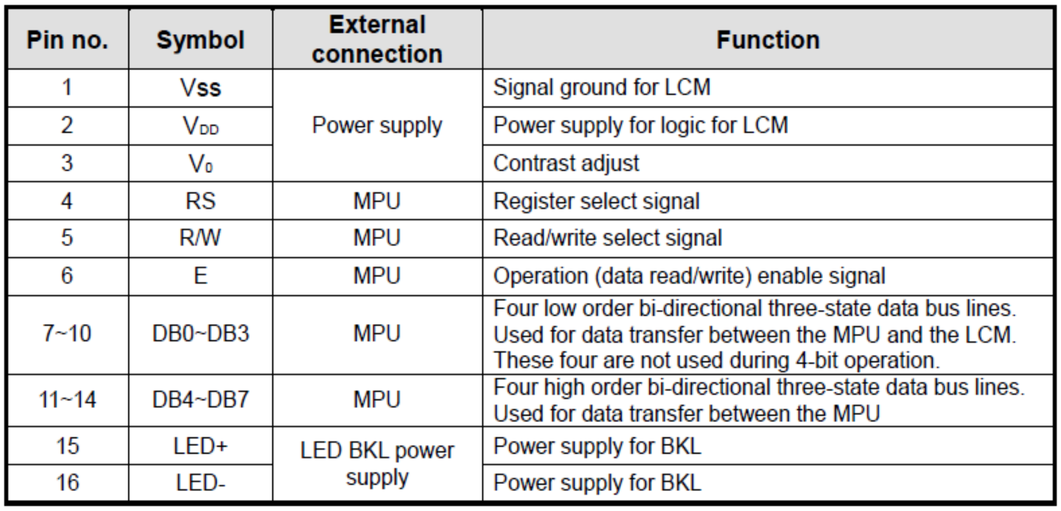
CHECK FOR UNDERSTANDING

How would we display a centered greeting such as:

ECE-218
Embedded Systems

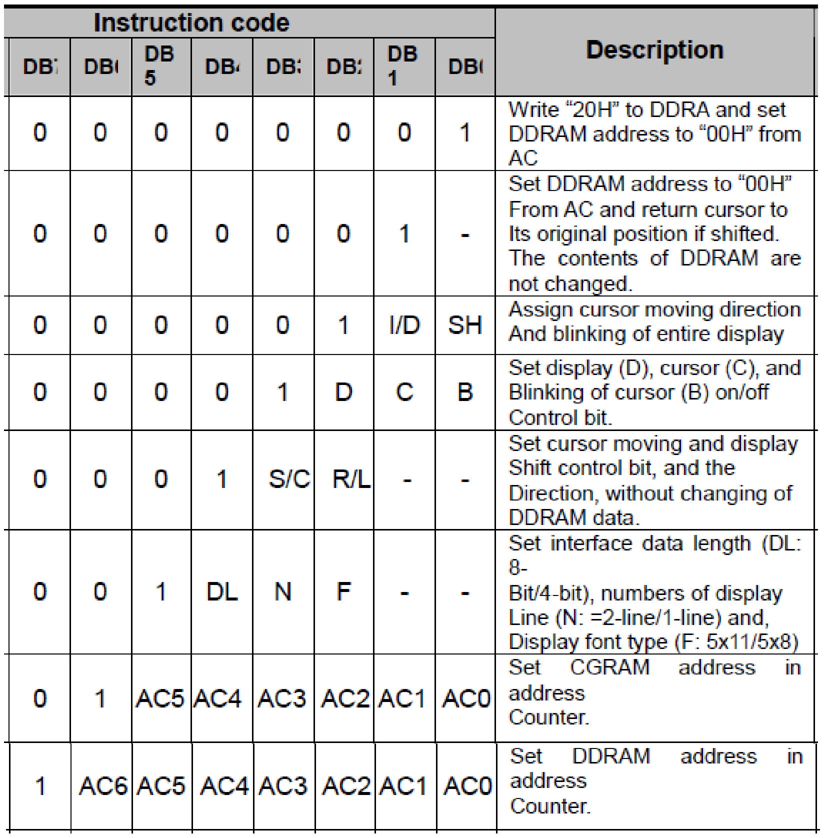
Upload your code to Nexus when you have completed the exercise.

APPENDIX.



Pin no.	Symbol	External connection	Function
1	Vss	Power supply	Signal ground for LCM
2	V_{DD}		Power supply for logic for LCM
3	V_0		Contrast adjust
4	RS	MPU	Register select signal
5	R/W	MPU	Read/write select signal
6	E	MPU	Operation (data read/write) enable signal
7~10	DB0~DB3	MPU	Four low order bi-directional three-state data bus lines. Used for data transfer between the MPU and the LCM. These four are not used during 4-bit operation.
11~14	DB4~DB7	MPU	Four high order bi-directional three-state data bus lines. Used for data transfer between the MPU
15	LED+	LED BKL power supply	Power supply for BKL
16	LED-		Power supply for BKL

Table 1. LCD Interface Pin Description[1]



Instruction code								Description
DB	DB	DB 5	DB	DB	DB	DB 1	DB	
0	0	0	0	0	0	0	1	Write "20H" to DDRA and set DDRAM address to "00H" from AC
0	0	0	0	0	0	1	-	Set DDRAM address to "00H" From AC and return cursor to Its original position if shifted. The contents of DDRAM are not changed.
0	0	0	0	0	1	I/D	SH	Assign cursor moving direction And blinking of entire display
0	0	0	0	1	D	C	B	Set display (D), cursor (C), and Blinking of cursor (B) on/off Control bit.
0	0	0	1	S/C	R/L	-	-	Set cursor moving and display Shift control bit, and the Direction, without changing of DDRAM data.
0	0	1	DL	N	F	-	-	Set interface data length (DL: 8-Bit/4-bit), numbers of display Line (N: =2-line/1-line) and, Display font type (F: 5x11/5x8)
0	1	AC5	AC4	AC3	AC2	AC1	AC0	Set CGRAM address in address Counter.
1	AC6	AC5	AC4	AC3	AC2	AC1	AC0	Set DDRAM address in address Counter.

Table 2. Command Instructions for LCD Display[1]

[1] From LCD_Sparkfun_3.3V.pdf datasheet available on Nexus