EE/CS 120B: Introduction to Embedded Systems
University of California, Riverside
Winter 2016

Laboratory Exercise 4

This laboratory exercise will teach you how to use the LCD display.

Note: Some LCD displays may require soldering; to the best of our knowledge, the LCD screen that accompanies the IEEE kit **does**. If your LCD requires soldering, before using it, you will need to solder the display to the connector pins.

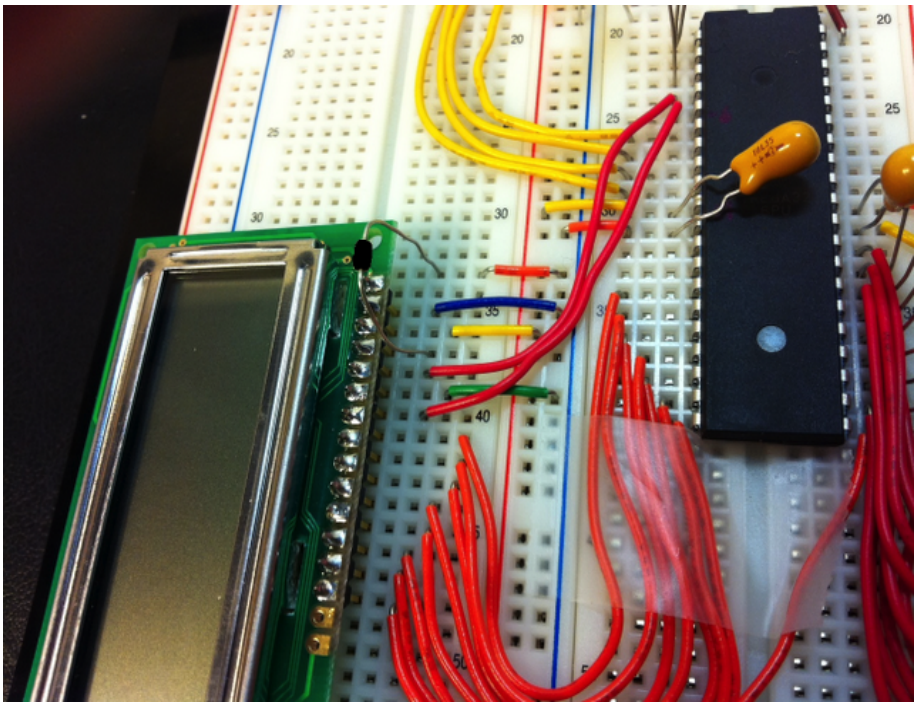
- Watch this YouTube video before attempting to solder any connections:

How and WHY to Solder Correctly.

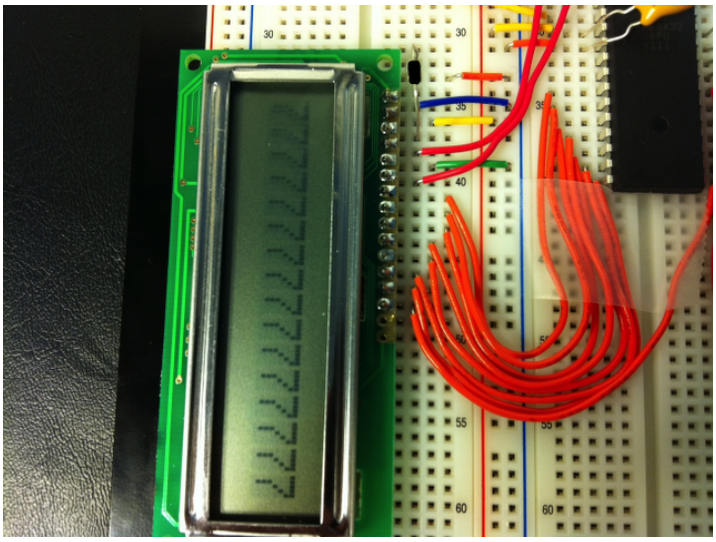
- It is also advisable to read the EE lab Rules and Policies before soldering.
http://www.ee.ucr.edu/labs/lab-rules-and-policies.html

Note: Email the TAs to set up an appointment for soldering in WCH 464. TAs will soon be providing dates and times to sign up for. If possible complete soldering before this lab. Here is a link to the LCD datasheet.

The following figures and tables illustrate the usage of the LCD display on a breadboard.



LCD Connections – The resistor is 10 kΩ; It **SHOULD BE** replaced with a 10 kΩ potentiometer for proper viewing of the LCD.



Functional LCD

NOTE: The above images use Port D on the microcontroller. Please use Port C as specified in the provided io.c and the following LCD wiring pin configuration:

LCD Display Pin Connections

LCD PIN #	**1**	**2**	**3**	**4**	**5**	**6**	**7-14**	**15-16**
Connection	GND	5 Volts	Potentiometer (10KΩ) thru. to GND	AVR PORT D6	GND	AVR PORT D7	AVR PORT C0-C7	Vcc-GND

The code provided below provides a number of simple functions that can be used to write characters on the LCD display. Create a new folder called “includes”. Download the files below and store them in the “includes” folder. You can add these files to your project by including the documents with a #include statement, and providing the entire path to your “includes” folder and file.

io.h
io.c

If the LCD display has been connected according to the pinout given above, and the “io” files have been added to the working directory of the project, then the sample code below should print “Hello World” to the LCD display.

Sample LCD code: Copy, Paste, Save, Build, and Program this code into the micro-controller.

```
#include <avr/io.h>
#include "/path/includes/io.c"

int main(void)
{
        DDRC = 0xFF; PORTC = 0x00; // LCD data lines
        DDRD = 0xFF; PORTD = 0x00; // LCD control lines

        // Initializes the LCD display
        LCD_init();

        // Starting at position 1 on the LCD screen, writes Hello World
        LCD_DisplayString(1, "Hello World");

        while(1) {continue;}
}
```

The following three functions from “io.c” suffice to operate the LCD display

LCD_ClearScreen(void): clears the LCD display
LCD_Cursor(unsigned char column): positions the cursor on the LCD display
LCD_WriteData(unsigned char Data): Writes a char to the cursor’s current position
LCD_DisplayString(unsigned char column, const unsigned char* string):
Writes a char* (string) to the LCD display starting at position ‘column’

Knowing the current state of your program when testing is a critical debugging tool. A useful way to debug your program is to write the current state of your synchSM onto LEDs attached to an unused port. This approach gives you a visual representation of where your synchSM is at any given point. Just as in Lab3 with the unused enum state variable.	

Exercise

1. Buttons are connected to PA0 and PA1. Outputs PORTC and PORTD drive the LCD display, initially displaying 0. Pressing PA0 increments the display (stopping at 9). Pressing PA1 decrements the display (stopping at 0). If both buttons are depressed (even if not initially simultaneously), the display resets to 0. Be sure to set a logical threshold for both buttons being pressed, so that both buttons can be used independently. ***If a button is held, then the display continues to increment (or decrement) at a rate of once per second.*** Use a synchronous state machine captured in C. Writing with LCD_WriteData() should only occur once pre increment/decrement/reset in order to avoid repeated writes, and maximize LCD readability.

   Note: The LCD display displays ASCII values so a char value of 9 (0x09) is not the same as ASCII character '9'.

   To display numbers 0 thru 9, a quick conversion technique is to add the character '0' to the number:

```
LCD_WriteData( 9 ); // will display nothing on the LCD
LCD_WriteData( 9 + '0' ); // will display '9' on the LCD
```

   Video Demonstration: http://youtu.be/cRNJc6lhcKs

   Why adding + '0' works: http://www.asciitable.com/
   Char value '0' adds an ascii offset of 30.

Challenge Problem

2. Extend Part2 of Lab3's light game to maintain a score on the LCD display. The initial score is 5. Each time the user presses the button at the right time, the score increments. Each time the user fails, the score decrements. When reaching 9, show victory somehow.

   Video Demonstration: http://youtu.be/r8yzRMPD3lE

Each student must submit their .c source files according to instructions in the lab submission guidelines. Post any questions or problems you encounter to the wiki and discussion boards on iLearn.