Micro Design 2

Lab #4

* Liquid Crystal *
*   Displays   *

Interfacing, Programming
and
Applications

Introduction

An embedded system is generally limited by constraints such as size, supply power, or the necessity of operating in harsh environments. In these systems it would be unfeasible to incorporate a CRT. Quite often a decision must be made between alternate display technologies such as LEDs (single indicators, 7-segment or character), Vacuum Flourescent Displays, or Liquid Crystal Displays. Each of these has advantages and drawbacks. Some are easy to view in bright environments (daylight) while others are easy to view in the dark. As display brightness goes up so does power consumption. The displays which are capable of displaying more complex information are more expensive. Does the final product have to operate in bright condtions? Is it battery powered? Does it convey simple on/off information or is it necessary to display words and/or graphics?

You have already had a fair bit of experience with CRTs and LEDs. The LCD will present a different kind of challenge. It is an “intelligent” peripheral, but the rules for it's operation are quite strict and a little peculiar. By adding menu software, one can create a final product that is completely self-contained, easy to use and professional.

In this lab we will look at the communication that goes on between the microprocessor and an intelligent peripheral like an LCD. Once the hardware is up and some basic routines have been estabished, we can put the LCD through it's paces. LCDs lend themselves naturally to embedded systems that are menu driven (meaning that the input device is likely only a few buttons rather than a complete keyboard). The adventerous programmer can build a computer system that packs a fair deal of power, is “bullet-proof” (idiot proof), and requires only minimal input hardware (even the A/D can be employed in the menu operation).

The LCD controller chosen for this course allows for a limited amount of custom character design. The last section of the lab will explore these character generation facilities.

Lab Outline

PRE-LAB: LCD Circuit and Address Decoding Design

You will prepare the schematic for your LCD interface circuit in order to place it in the 0x8000 block of the 68HC11's memory map.

SECTION 1: Wiring, Testing and Simple Data Displays

Section 1 will consist of the wiring and testing of your circuit. You will then develop one or two simple functions which will be used to verify LCD initialization. Following that, two programs which are very similar will serve to demonstrate effective use of the LCD's display RAM and a few LCD operating modes.

SECTION 2: Digital Voltmeter

Your A/D voltmeter lab can be adapted to work with the LCD. A simple rewrite of Lab #3 will be required in order to “port” the SCI/CRT version of the voltmeter for embedded (stand-alone) operation via the LCD. Adherence to proper program structure and rules will allow for a simple conversion to EPROM operation later.

SECTION 3: Character Generator RAM

The above voltmeter can be adapted to the LCD equivalent of an “LED Bar Graph” by first creating a few custom characters and then applying a little math to scale the range to fit within the limited resolution of the LCD (mind you the result will have much greater resolution than actual LED bar graphs of home stereo equipment).

SECTION 4: EEPROM VERSION

The final version of the program is burned into EEPROM and becomes a stand-alone device.

Pre-Lab

LCD Circuit and Address Decoding Design

NOTE!

Do NOT deviate from the standards and procedures detailed here. If you destroy your LCD it will be your responsibility to acquire a replacement!

( ) Step 1: HC11/LCD Interface Socket Schematic

The following page shows a diagram of the 14 pin DIP plug - to - LCD schematic which will be the standard interface for this course. Your task for the Pre-Lab is to prepare a schematic of the mate socket and decoding logic for the 68HC11 - to - 14 pin DIP socket connection. The actual wiring and assembly will be covered in the lab procedure (next section) and will be supplemented by class notes.

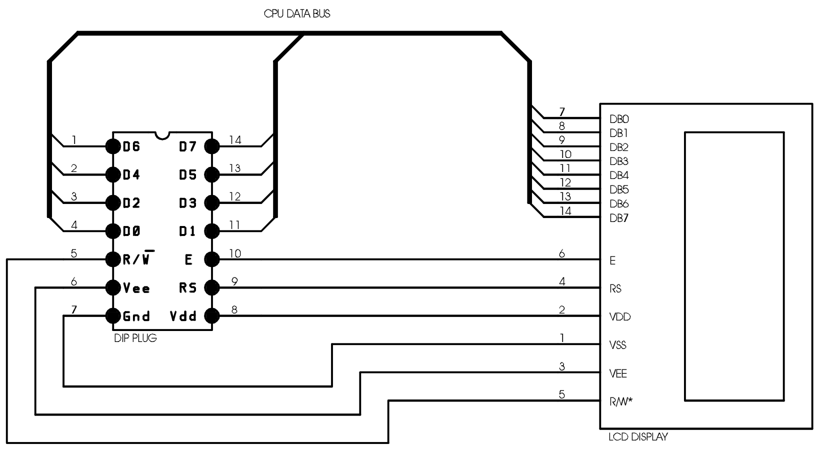
After considering the timing for the LCD and HC11 you should have a truth table which will satisfy the requirements for two-way communication between the LCD and HC11 via the HC11's data bus. E_{LCD} should be decoded to appear in the 0x8000 block of the 68HC11's memory map.

On your schematic, include the following:

- The 14 pin socket (you might want to add this schematic to your LED socket page).
- Power and contrast connection including the 10k trim-pot used for contrast control.
- The decoding logic (or you may do this on your decoding pages) and the other control signals ($R/\overline{W}$, RS, etc.).
- The part numbering and title block usual to all schematics.

You do NOT need to reproduce the drawing on the next page (unless you really want to)!

*** Have your schematic checked off at this point!!! ***



NAIT Computer Engineering Technology Drawn by Grant Beattie		
Title LIQUID CRYSTAL DISPLAY & RIBBON CABLE		
Size A	Document Number Unit3b.sdt	REV C
Date: January 24, 1993	Sheet 1 of 1	

Procedure

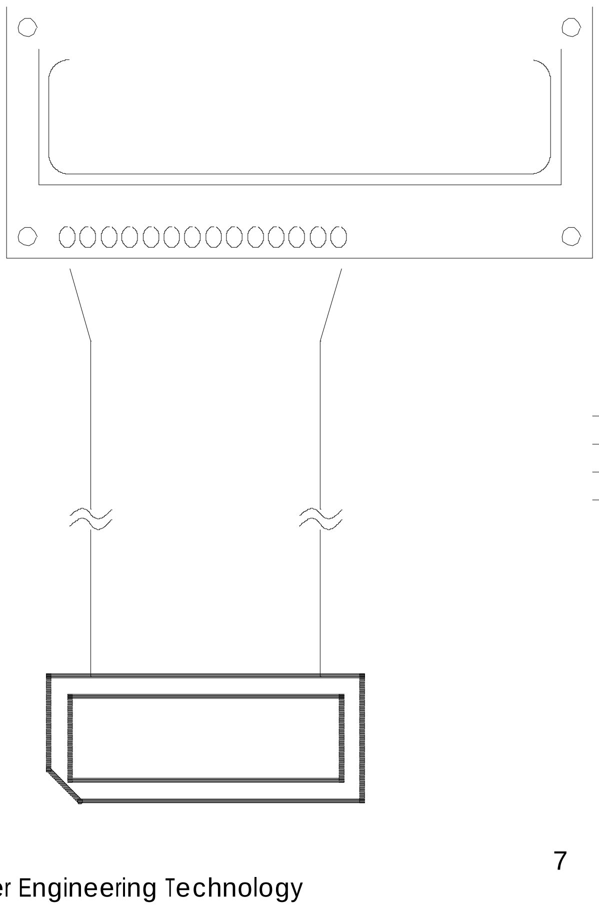
SECTION 1: WIRING, TESTING AND SIMPLE DATA DISPLAYS

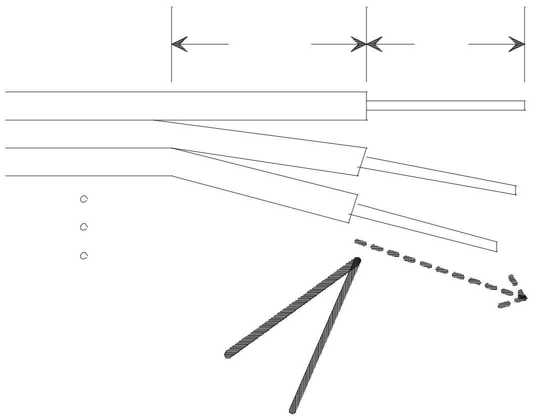
( ) Step 1: LCD Cable Wiring

WARNING: The LCD display is a static sensitive device! Do NOT remove it from the protective bag until you are at an ESD safe workbench.

Your first task is to attach the ribbon cable to the LCD module. Referring to the diagrams below, separate the ribbon cable into individual wires and tin the wires carefully. Solder the wires one at a time from pin 1 through pin 14. Push the wires through the holes from the bottom of the LCD and solder them on top. This will allow you to probe the signals later if necessary. Trim all excess wire protruding from the holes after the soldering is complete.

NOTE: In the diagram below, the dip plug pins go INTO the page! Also, ignore the red stripe. It cannot be used to locate pin 1 as not all student cables are created equal.





(   )   Step 2: LCD Placement

Locate the LCD, it's socket and the contrast control in the space remaining on your board. The precise location is not critical, but you should consider the wiring for the socket before you begin drilling the board.

You will need to drill 4 holes for the standoffs which support the LCD display. The dimensions for the LCD are supplied in the data sheets at the back of this lab. MEASURE these holes. If your drilled holes do not exactly match the holes on the LCD, carefully enlarge the holes on the HC11 vectorboard (NOT the LCD) until they do. The holes on the LCD will probably need to be enlarged very slightly to accomodate the screws. Do this using a drill bit and your fingers (NOT THE DRILL!). Be careful of the fine traces which run on the LCD module near these holes. Use the plastic washers included with your kit!

(   )   Step 3: LCD Wiring

Wire up the 14 pin socket and your decoding logic according to your schematic. The LCD is very ESD sensitive. Do NOT wire the 14 pin socket with the LCD plugged in! Follow all color codes as per usual and do not forget to include the contrast control. It can be held in place with a dab of clear RTV silicone.

When you have completed your wiring, install the 14 pin LCD dip plug into your socket BUT DO NOT APPLY POWER. Using an ohmmeter, do a continuity test between ALL 14 connections on the LCD and the circuit board.

EXAMPLE: Pin 1 on the LCD circuit board is nearest the center. This connection is ground. Verify that it is connected to the ground rail on the HC11 board.

(   )   Step 4: LCD Test Initialization

When you power up your HC11 board with the LCD installed there will be no instant gratification. The LCD must be sent several initialization commands before anything will be visible in the display. Using the Memory Modify facility in Micro11 follow the steps below to initialize the LCD. Note that ALL OF THE DATA BELOW MUST BE SENT TO THE LCDCR:

- FUNCTION SET - Set the interface data length (DL) to 8 bits. Set the display for (N) 2 lines of (F) 5 x 7 characters.
- DISPLAY ON/OFF CONTROL - Turn on the display (D), turn on th cursor underline (C) and turn on the blinking cursor block (B).
- ENTRY MODE SET - Set the cursor to move to the right (Inc/Dec) but do not shift the screen (S).
- CLEAR DISPLAY - Issue a Clear Display command. This will set all locations to blank spaces and will return the cursor to the “home” position.

You should see a flashing cursor at this point. If you do not see anything adjust your contrast control. If the display is blank ask your instructor for help.

( ) Step 5: Subroutine LCDWAIT

Write a short subroutine or function which polls the LCD busy flag until clear. (The assembly version can be accomplished in three lines and without using any CPU registers!) See the bottom of page 10 for more information about LCD_LIB.ASM and LCD_LIB.C.

The subroutine header for LCDWAIT might look like this:

```
;LCDWAIT        Reads the LCD busy flag and stays in the subroutine until the flag is clear.
;               Subroutine time will be approx. 40us or 1.64ms depending on which command
;               the LCD is executing.
;REQUIRES -     Nothing
;RETURNS -      Nothing
;REGS AFFECTED - None
```

The C prototype might look like:

```
void LCDWait(void);
```

( ) Step 6: Data Display Text

The two short programs below will display the same text information in two different ways. Make sure your program includes the LCD equates, LCD initialization and the appropriate use of LCDWAIT.

Compose some text which can be placed into the LCD's display ram. Your text must be between approximately 40 and 64 characters (enough for two full screens). For example:

NAIT Computer Engineering Technology. World Domination by Nikola Tesla.

This text will be displayed in two formats. This means it must be placed in display RAM in two different ways:

1. Two screens or pages:

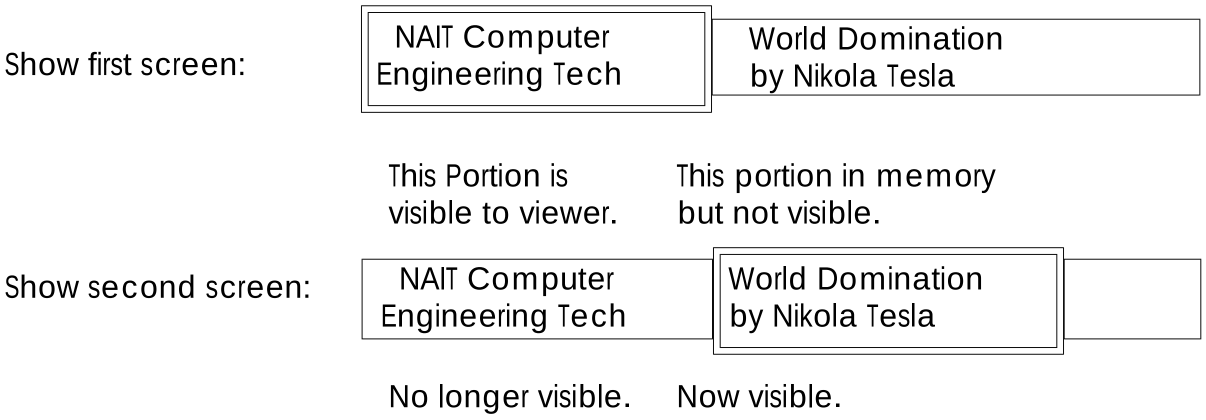
```
 NAIT Computer                    World Domination
Engineering Tech                  by Nikola Tesla
```

2. Walking or scrolling text:

```
NAIT Computer                     Engineering Technology
World Domination                  by Nikola Tesla
```

( ) Step 7: Display Program #1 - Paging

Since the LCD can display only two lines of 16 characters each at any one time, the above text will be placed in LCD display RAM at one time, but actually shown to an observer in two “pages”. The text shown is only an example, be creative!



PROGRAM #1 DEFINITION:

i. Initialize the LCD for two lines of 5 x 7 display.

ii Turn the display on, but do not enable the cursor underline or blink. (Until your program is debugged, you may want to leave the cursor visible.)

iii. Transfer all of the text to the LCD display RAM. Remember, after 40 characters the LCD will automatically move to the next line.

iv. Return the window to the home position. Enter a delay loop for a second or two. The first screen or “page” should be visible to the viewer.

v. “Shift Once” sixteen times so that the second screen is visible to the viewer. Enter a delay loop for a second or two.

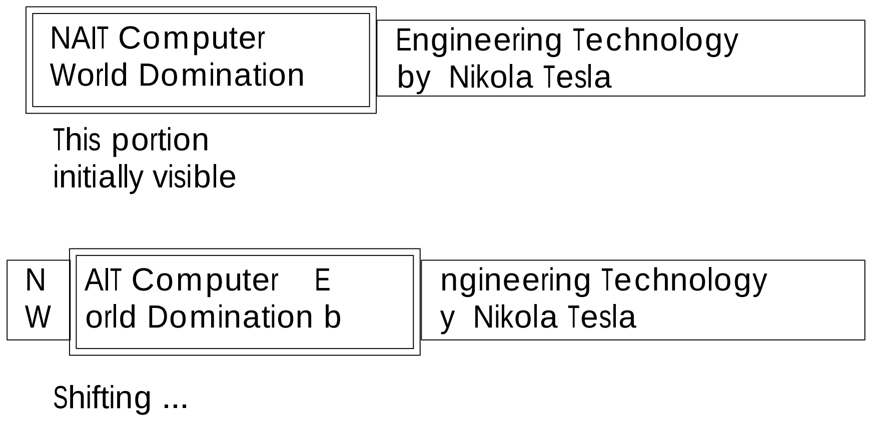
vi. Return to step iv in an endless loop.

Create an LCD_LIB.ASM or LCD_LIB.C library file in the usual manner. In it, create the following subroutine or functions:

i. LCDdata( ) - Sends a byte to the LCD data register.
ii. LCDctrl( )- Sends a byte to the LCD control register.
iii. LCDstring( ) - Sends a string to the LCD data register.

( ) Step 8: Display Program #2 - Shifting

This time you are to place the text in LCD display RAM in a different fashion and then scroll the text horizontally. By continually shifting the display the text will continue to rotate.



PROGRAM #2 NOTES:

i. For each shift of the display pause for approximately two-tenths of a second..

ii Shift the display continuously. The LCD controller will take care of the wrap-around.

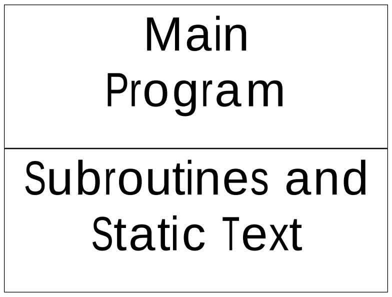
*** Have your programs checked off at this point! ***
Turn in only ONE of these two programs (but include the library file).

SECTION 2: LCD Digital Voltmeter Program

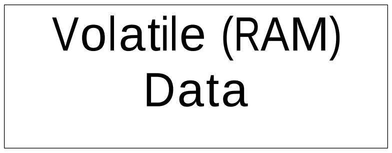
We can use the LCD display to make the voltmeter from Lab #3 more compact and self-contained. If the program was ROMed there would be no need for the PC or it's umbilical cord.

Port your program from Lab #3 to operate with the LCD in place of the SCI/terminal combination. Do NOT incorporate terminal output at all in your program. Keep your source file logically partitioned into three sections for a simple move to EEPROM in the future:

```
.if eeprom
   .ORG  0xC000
   LDS   #0x00FF
.else
   .ORG  0x1040
.endif
```



```
.if eeprom
   .ORG  0x0000
.endif
```



```
.if eeprom
   .ORG  0xFFFE
   .DW   0xC000
.endif
```

The C version cannot take advantage of conditional assembly, so again be prepared to modify your CRT.S file when it comes time to ROM the code (Section 4).

( ) Step 1: Voltmeter Program Specification

i. Place a static message and the voltage reading all on the first line. The second Line should remain blank for now.

ii. Leave the cursor underline and blink off.

iii. This program need not be blasted into EPROM, but it must be EPROM-ready. That is, it must be logically partitioned and it must contain conditional assembly Directives (for those using assembly).

iv. Sample display:

```
DC Voltage: 1.92
```

*** Have your program checked off when it is working! ***

SECTION 3: Character Generator RAM

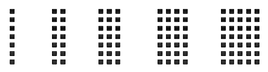
The first eight ASCII characters (0x00 thru 0x07) can be user defined to provide system developers with the ability to tailor the LCD output for specific needs. Quite often characters are designed which can display two numbers with only one 5 x 7 character. For instance, a single character can be built to display the number “10" in a single 5 x 7 space. Admittedly, the number 10 would be a little scrunched, but this kind of trick is very common.

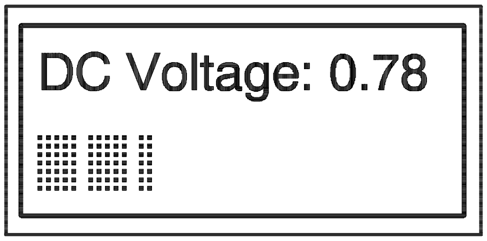
Other obvious uses would be to show symbols common to a particular product. Electronic musical instruments which employ an LCD must display a wide assortment of symbols common to written music. Graphic LCDs are also available ($100+), but then it is up to the programmer to create the necessary algorithms for drawing lines, circles, and so on.

The objective of this lab section is to create a series of characters which can be used to add a “bar graph” feature to the previous voltmeter.

( ) Step 1: Creating the Characters

You will need to create five characters callable via ASCII 0x00 through 0x04. The five characters appear below individually and in a finished example:





For character 0x00, all seven of the leftmost dots that make up the left column are “on”. For character 0x01, the two left columns are “on”. Therefore you can create tables using the .db directive and transfer the CG RAM data to the LCD after it is initialized. Another alternative is to use a series of nested loops (in either language). Since the characters will be built at the same time as the LCD is initialized, speed will not be an issue.

( ) Step 2: Scaling Algorithm

The range of values returned from the SAMPLE subroutine is 0x00 to 0xFF, 256 values in all. Since the LCD is 16 characters of 5 pixels width, the number of horizontal pixels is 80. Obviously we cannot create a horizontal bar-graph of 1 pixel = 1 step since we would need 255 horizontal pixels (excluding zero). Therefore, we must “scale” the SAMPLE value into something 80 or below.

In order to keep the math reasonable, scale the SAMPLE value by 80/255. Therefore an input of 255 (0xFF) results in 80 pixels. Note that all values must be properly rounded (not simply truncated). For the example on the previous page SAMPLE returned 0x27 (39 decimal) which translates into 0.78 volts. In order to change it to pixels multiply by 80/255: (39x80)/255 = 12. As you can see, the bargraph shows 12 horizontal pixels. If the returned sample value had been 0x28 (40 decimal) then the LCD should show (40x80)/255 = 12.55 rounded to 13 pixels.

The problem now comes down to finding out how many 5-pixel-wide characters to send before dropping down to the final smaller character. There are a few different ways to do this, but an iterative solution is shown below. Another twist is to take advantage of the fact that all of the characters are five_bar_chars except the very last one.

```
/*  Scale SAMPLE by 80/255 then ...  */
while(sample)
    {
    if (sample >= 5)
        {                           /* If SAMPLE is bigger than 5 show a */
        sample -= 5;                /* complete 5 pixels, take 5 off of   */
        LCDdata(five_bar_char);     /* SAMPLE, and go around again.   */
        }
    else
        {
        if (sample == 4)
            {                       /* If SAMPLE is 4, the last char is 4 */
            sample -= 4;            /* pixels (then exit).                */
            LCDdata(four_bar_char);
            }
        else
            {
            if (sample == 3);
                {

            :      etc.

    }
```

*** Have your program checked off when it is working! ***

SECTION 4: EEPROM Version

( ) Step 1: ROMing the code

The final operation is to verify that the conditional assembly code works by burning a final stand-alone EEPROM version. Re-read the discussion from SECTION 2 to make sure that your code is ROMable. Re-assemble the code with EEPROM = 1 and then burn the EEPROM using Bootload.exe

If you are using C, modify your CRT.S file and re-compile. Check the .LST file to make sure that everything is where it should be.

Note the following additions to the specifications:

i. Add an opening display screen which SCROLLS by once at power-up. This screen must contain at minimum the following information:

```
CNT-442 Lab#4  Version 1.00
Copyright 1997 by Nikola Tesla
```

When you have verified the operation of your program, have it checked off. You will only need to hand in ONE set of files (the EEPROM version only) for sections 2 through 4.

*** Have your program checked off when it is working! ***

LAB #4: LABCHECK SUMMARY

NAME: ______________________________

SECTION: ____________________________

PRE-LAB: Schematic

Make sure you hand in the LCD schematic and the decoding logic (if it is on a separate page).

Date:__________________	Instructor: ____________________	/5

LABCHECK #1: Simple Data Displays

Wiring (workmanship, color codes, etc.)		/5
Program #1: Paging		
Date:__________________	Instructor: ____________________	
Program #2: Shifting		
Date:__________________	Instructor: ____________________	/10
Program Documentation (Hand in one program only)		/10

LABCHECK #2: Digital Voltmeter

Operation (This is a milestone checkoff)		/5
Date:__________________	Instructor: ____________________	

LABCHECK #3: Character Generator RAM

Operation (This is a milstone checkoff)	/5

Date:_________________ Instructor: ___________________

LABCHECK #4: EEPROM Version

Operation	/5
Documentation (Final EEPROM version)	/15

Date:_________________ Instructor: ___________________

YOU MUST HAND IN ONE LIST FILE

TOTAL: /60

APPENDIX

Stanley LCD Module GMD-16202
(Equivalent to LM016L)

Hitachi HD44780A00
Liquid Crystal Display Controller
Data Sheets