
Lab Overview

In this lab assignment, you will do the following:

- Add an LCD and a serial EEPROM to the hardware developed in Labs #1, #2 & #3.
- Write device drivers for the LCD and EEPROM.
- Write assembly and C programs to perform user output to the LCD.
- Write simple assembly and C programs to test EEPROM accesses.
- Continue learning how to use SDCC and Eclipse.

This lab assignment is due by **Saturday, November 15, 2008**.

The deadline for this lab is **Wednesday, November 19, 2008**.

This lab is weighted as 20% of your course grade.

You should be working on your final project in parallel with this lab assignment.

NOTE: The quality of your user interfaces will impact your score on the lab. Your goal should be to ensure that the user has a successful and positive experience with your software. Your programs should handle error conditions gracefully (e.g. user input values outside the allowable range). Top scores will be reserved for those students who submit outstanding implementations.

Lab Details

1. Review the data sheets for the Optrex DMC 20434 LCD and the SED1278F (or Hitachi HD44780U) LCD controller. Make sure to review the document "SED1278F Technical Manual Errata".
2. Refer to the **LCD Guide** ("Adding an LCD (with an HD44780 LCD controller) to your board") available on the course web site for further ideas and information on interfacing to the LCD module. It contains some very important notes, including one regarding errors in the LCD data sheet.
3. **[Required Element¹]** Devise a way to securely mount the LCD **and** properly connect all of the data lines to your board.

Data Lines: Most LCDs will have only 14 pins. However, those with a backlight will have 16, the extra two of which control the backlight. One option for connecting data lines is to use a 14-pin strip header or SIPP wire wrap socket. You may also attach the LCD through a ribbon cable to a 14- or 16-pin DIP socket on your board. **Note:** A sturdy data line connection using a strip header can make it easy to mount the LCD.

Mounting: It may take you a little time to devise a good physical interface, so don't wait too long before getting started. Wire can be used to easily attach the LCD to your board without requiring any drilling (remember the previous warnings against drilling holes in the PCB).

4. **[Required Element¹]** Design and implement your LCD circuit. Your LCD should be memory mapped in the address space reserved for peripherals, and will be accessible using the MOVX instruction. The LCD contrast (V_{EE}) can sometimes be grounded, but you can use a potentiometer or resistor divider to control the contrast if necessary. The LCD in the parts kit has 14 lines which must be connected to your board.

The eight data signals on the LCD must be connected to the data lines on Port 0 of the 8051. Ensure that the E signal on the LCD is high **only** when you're reading from or writing to the LCD.

5. **[Required Element¹]** Implement an LCD device driver with the following C functions:

- `// Name: lcdinit()
// Description: Initializes the LCD (see Figure 25 on page 212
// of the HD44780U data sheet).
void lcdinit()`
- `// Name: lcdbusywait()
// Description: Polls the LCD busy flag. Function does not return
// until the LCD controller is ready to accept another command.
void lcdbusywait()`
- `// Name: lcdgotoaddr()
// Description: Sets the cursor to the specified LCD DDRAM address.
// Should call lcdbusywait().
void lcdgotoaddr(unsigned char addr)`
- `// Name: lcdgotoxy()
// Description: Sets the cursor to the LCD DDRAM address corresponding
// to the specified row and column. Location (0,0) is the top left
// corner of the LCD screen. Must call lcdgotoaddr().
void lcdgotoxy(unsigned char row, unsigned char column)`
- `// Name: lcdputch()
// Description: Writes the specified character to the current
// LCD cursor position. Should call lcdbusywait().
void lcdputch(char cc)`
- `// Name: lcdputstr()
// Description: Writes the specified null-terminated string to the LCD
// starting at the current LCD cursor position. Automatically wraps
// long strings to the next LCD line after the right edge of the
// display screen has been reached. Must call lcdputch().
void lcdputstr(char *ss)`

NOTE: I prefer you to write your own code for these routines. However, a variety of LCD routines and libraries suitable for SDCC (and MICRO-C) are available on the web. You may use these libraries as long as your code contains clear documentation of how you obtained, utilized and/or modified them. (You already know this is the standing expectation in this class with regard to borrowed code.) **You must have a complete understanding of how all the code works.**

6. **[Required Element¹]** Write a simple program that uses your LCD driver to prove that the six required functions are implemented correctly. Choose the sequence carefully so that it is easy for the TA to see that each function did its job correctly during the demonstration.
7. **[Required Element¹]** Using a logic analyzer, prove that your LCD control signal timing is correct. Show the timing relationships between the E, RS, R/W*, and data signals as measured at your LCD interface.

- **A simple hand sketch or a logic analyzer screen capture of these timing relationships and values must be turned in with your lab, along with your timing analysis.**

You should also be able to prove that the LCD E control signal goes high only when the LCD is being accessed. You can verify this by running code which does not access the LCD and by triggering the logic analyzer on E going high. If E goes high during this test, then your implementation is incorrect. You may also be able to test this by using Paulmon2.

8. Read the **EEPROM Guide** "Adding an NM24C04 (or NM24C16) EEPROM to your board", available on the course web site. It has ideas and information on interfacing to the I²C EEPROM.
9. Read the data sheet for the Fairchild (National Semiconductor) NM24C16 or the actual serial EEPROM included in your parts kit. You may also want to read Fairchild Application Note AN-794.
10. [Optional, but recommended] Review Microchip app notes AN536, AN572, AN614 and AN709.
11. **[Required Element¹]** Design and implement your EEPROM circuit. Your EEPROM should be connected to two unused port pins on Port 1 or Port 3. Note that since you are connecting to the EEPROM using port pins, the EEPROM does not consume any 8051 address space.

NOTE: In the next step, your EEPROM driver code may require use of specific port pins

12. **[Required Element¹]** Implement an EEPROM I²C device driver with the ability to write and read a byte at any EEPROM I²C address using function calls from C. The underlying drivers may be in assembly if you wish, but **the functions must be accessible from C**. It does not matter what you name the functions. For example, you might implement the following two functions.

```
int eebytew(addr, data) // write byte, returns status
int eebyter(addr)      // read byte, returns data or status
```

NOTE: A variety of I²C routines and libraries suitable for SDCC (and MICRO-C) are available on the web, including those in Microchip AN614. You may use these libraries as long as your code contains clear documentation of how you obtained, utilized and/or modified them. (You already know this is the standing expectation in this class with regard to borrowed code.)

MICRO-C Users: It is acceptable to use the MICRO-C `ee_read()` and `ee_write()` functions instead of writing your own. If you use these MICRO-C functions, you must use the I/O pins defined in the MICRO-C I²C .ASM library code.

13. **[Required Element¹]** Verify that you can write data to and read data from the EEPROM using your I²C device driver and **verify the stored data is correct after cycling power**.
14. **[Required Element¹]** Use a logic analyzer to prove that your byte write function sends the correct signals and has the correct I²C timing.

- **A simple hand sketch or a logic analyzer screen capture of these timing relationships and values must be turned in with your lab, along with your timing analysis. You can use the floppy diskette from the tool kit for the logic analyzer screen capture.**

15. [Optional] Use the I²C triggering program on the Agilent 54622D oscilloscope to trigger on a write or read frame on the bus. Display SCL and SDA on the oscilloscope screen and verify that the transaction is for the address you intended. Verify that your rise and fall times fall within the limits given in the I²C specification. Alternatively, use a logic analyzer to trigger on a bus transaction.

16. [Required Element¹]

Provide a well-designed menu on the PC terminal emulator screen which allows the user to:

- **Write Byte:** Enter an EEPROM address and a byte data value in hex. If the address and data are valid, store the data into the EEPROM. The program must allow any hex value from 0x00 to 0xFF to be programmed into **any** location in the EEPROM. Do not make the user type in "0x" before the address or data hex value.
- **Read Byte:** Enter an EEPROM address in hex. If the EEPROM address is valid, display on the PC screen in hex the contents of the EEPROM address, using the format "AAAA: DD". Do not make the user type in "0x" before the address hex value.
- **Block Fill:** Enter a start address, end address and a fill value in hex. If the entered values are valid, the EEPROM contents from the start address to end address are written with the fill value.
- **LCD Display:** Enter an EEPROM address in hex. If the EEPROM address is valid, display on the LCD display in hex the EEPROM address and the contents of the EEPROM address, using the format "AAAA: DD", positioned on the LCD at (row,column) = (Y,0). 'Y' is the row number and cycles through the values {0,1,2,3,0,1,2,3,0...} Each time this function is called, the EEPROM cell content is printed, and then 'Y' is incremented according to the sequence shown above. Data from up to four EEPROM addresses can be seen on the LCD screen at any one time, depending on how many times the user has selected **LCD Display**. Do not make the user type in "0x" before the address hex value. This function must utilize the lcdgotoxy() device driver function.
- **Hex Dump:** Read the entire contents of the EEPROM and display the data on the PC screen in hexadecimal, with 16 bytes of data per line, in the following format:
AAAA: DD DD DD DD DD DD DD DD DD DD DD DD DD DD DD DD
This format is similar to what you see when using the EPROM programmer or when dumping memory contents using PAULMON2, where AAAA is the starting address (in hex) for each block of 16 data values DD (in hex). The first memory cell in the EEPROM is address 0000. You should be able to leverage code from Lab #3.

17. [Supplemental Element¹, 12 points max]:

NOTE: The following routines must be integrated into the previous C programs above. Modify your previous C program to do the following additional things:

- In the bottom right corner of the LCD, continuously display the elapsed time since your program started running using the format "MM:SS", where MM is the number of elapsed minutes and SS is the number of elapsed seconds. For example, 5 seconds would be displayed as "00:05" and 64 seconds would be displayed as "01:04".
- Provide additional **Clock** menu options to stop the elapsed time clock, to restart the clock, to reset the clock back to "00:00", and to change the direction of counting (up/down). If the clock is counting down, **it must stop when it reaches "00:00"**.

NOTE: Make sure that the cursor location is correctly stored before and restored after any ISRs.

NOTE: If using MICRO-C, remember not to use any local variables from within the context of an ISR. This includes any functions that your ISR calls. If using SDCC, read the interrupt sections of the SDCC user manual carefully, and remember the correct use of 'volatile'.

- Design and implement C routines which allow the creation of custom LCD characters using CGRAM. Implement the following function:

```
// Name: lcdcreatechar()  
// Description: Function to create a 5x8 pixel custom character with  
// character code ccode (0<=ccode<=7) using the row values given in  
// the 8-byte array row_vals[].  
void lcdcreatechar(unsigned char ccode, unsigned char row_vals[])
```

Provide a way for the user to enter and display their own customer characters.

- In the same program, continuously animate a spinner arrow symbol on the LCD using four characters: '↑', '→', '↓', '←'. By sequentially displaying each of these characters in the same position on the screen, you can make these look like a spinning arrow. You will have to create some of the characters, since they are not all built into the LCD. You may want to use a timer interrupt or a delay function to control the spin rate.
- Provide options for the user to **Save LCD Screen** and **Restore LCD Screen**. When the user wants to save the LCD screen, the screen is first frozen (turn off any updates to the LCD), and then the screen contents are saved to EEPROM. Note that if any custom characters are used, then the custom character CGRAM contents must also be saved to EEPROM. When the user wants to restore the LCD screen, then the contents are read back out of the EEPROM and written to the LCD controller. Verify that you can save the screen, turn power off and on, and then restore the previous screen contents correctly.

NOTE: This supplemental element is an addition to the previous required element. **The required and supplemental code must be integrated together.** The elapsed timer must work correctly while simultaneously allowing all the menu options in the previous C program to work correctly.

NOTE: If you get this supplemental element signed off, don't turn in separate printouts of code for both the required part and the supplemental part - just turn in one printout of the integrated version.

¹ Required elements are necessary in order to meet the requirements for the lab. Supplemental elements of the lab assignment may be completed by the student to qualify for a higher grade, but they do not have to be completed to successfully meet the requirements for the lab. The highest possible grade an ECEN 5613 student can earn on this assignment without completing any of the supplemental elements is a '78' (out of 100). The highest possible grade an ECEN 4613 student can earn on this assignment without completing any of the supplemental elements is an '88' (out of 100). ECEN 4613 students can earn full credit for this lab assignment by completing the required elements and the **first** supplemental element.

18. [Supplemental Element¹, 10 points max]:

- Read the PCF8574 I²C I/O expander data sheets and application notes available from the course web site.

Integrate the chip into your embedded system, and prove that you can configure some of its I/O pins as inputs and others as outputs at the same time. Your parts kit already included a 16-pin wire wrap socket that could be used with the I²C expander chip. You can purchase another wire wrap socket if necessary.

Provide a user interface that allows you to configure the pins individually as inputs or outputs, and also to check the status of the pins and to write to the pins that are outputs.

Nice enhancements would be to use the interrupt signal from the I/O expander to notify the processor of an event, or to have at least one of the I/O pins drive an LED directly.

- Modify your EEPROM I²C device driver to include a new function named `eereset()`:

```
// Name: eereset()
// Description: Performs a software reset of the I2C EEPROM using an
// algorithm that conforms to Microchip application note AN709.
void eereset()
```

Use a logic analyzer to prove that `eereset` sends the correct sequence and has the correct I²C timing. Show the trace on the logic analyzer to the TA during signoff. You do **not** need to print/submit the trace with your report.

19. Demonstrate your hardware/software and get your lab signoff sheet signed by the TA or instructor.

NOTE: Make copies of your code, SPLD code, and schematic files and save them as an archive. You will need to submit the Lab #4 files electronically at the end of the semester.

NOTE: You can save paper by using the features in the printer driver installed on the computers in the lab. Select duplex printing to print on both sides of the paper. Select "2 pages per sheet" to get two pages of your code on each side of the paper. Do not print more than 2 pages per sheet, as the print will be too small to grade easily.

You will need to obtain the signature of your instructor or TA on the following items in order to receive credit for your lab assignment. This assignment is due by **Saturday, November 15, 2008**. Labs completed after the due date will receive grade reductions.

Print your name below, sign the honor code pledge, and then demonstrate your working hardware & firmware in order to obtain the necessary signatures. All items must be completed to get a signature, but partial credit **is** given for incomplete labs. [Separate this sheet from the rest of the lab assignment](#) and turn in this signed form, a full copy of your updated and complete schematic, and a printout of your fully and neatly commented source code (**not .LST or .RST listing files**) in order to receive credit for your work.

- Signed and dated signoff sheet (No cover sheet please)
- Full copy of complete and accurate schematic of acceptable quality (all components shown). Include programmable logic source code (e.g. .PLD file), if using an SPLD.
- Printout of fully, neatly, clearly commented source code. Ensure your printout is easy to read.
- Timing diagrams for the LCD and EEPROM interfaces.

Make sure your name is on each item and staple the items together, with this signoff sheet as the top item.

Student Name: _____ **4613 or 5613** (circle one)

Honor Code Pledge: "On my honor, as a University of Colorado student, I have neither given nor received unauthorized assistance on this work. **I have clearly acknowledged work that is not my own.**"

Checklist

Student Signature: _____

Required Elements

- Pins and signals labeled and decoupling capacitors present on board
- LCD functional, C code for basic LCD routines functional:
- LCD control signal timing meets specifications (diagram):
- Serial EEPROM functional, contents present after power cycle:
- C code for EEPROM functional, I²C timing correct:
- LCD Display and hex dump of EEPROM: _____ [88/78]

Instructor/TA signature and date

Supplemental Elements (Qualifies undergraduate/graduate student for higher grade.)

- Elapsed time display (accurate 1 second resolution):
- Elapsed time stop, restart, reset to "00:00", up/down:
- Support for custom LCD characters:
- Spinner symbol:
- Save LCD Screen/Restore LCD Screen:
- Good integration with previous code, all functions work with no irregularities: _____ [12]

Supplemental Elements (Qualifies graduate student for higher grade.)

- PCF8574 I²C I/O Expander:
- EEPROM eereset () functional and correct: _____ [10]

Instructor/TA Comments (e.g. user interface quality/issues):