Lab Overview

In this lab assignment, you will do the following:

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

This lab assignment is due by Saturday, April 2, 2005.
The deadline for this lab is Wednesday, April 6, 2005.

This lab is weighted as 17% 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).

Lab Details

1. 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.

2. [Optional, but recommended] Review Microchip app notes AN536, AN572, AN614 and AN709.

3. Review the data sheets for the Optrex DMC 20434 LCD and the Hitachi HD44780U LCD controller.

4. Refer to the EEPROM Guide and LCD Guide available on the course web site for further ideas and information on interfacing.

5. [Required Element] Design and implement your EEPROM circuit. Your EEPROM should be connected to two unused port pins on Port 1 or Port 3. Verify that you can write and read data from the EEPROM and verify the stored data after cycling power. Note that since you are connecting to the EEPROM using port pins, the EEPROM does not consume any 8051 address space.

6. [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.

7. [Required Element] Design and implement your LCD circuit. Your LCD should be memory mapped in the 8KB of address space reserved for peripherals. 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 has 14 lines which must be connected to your board. One option is to use a 14-pin strip header or SIPP wire wrap socket. Another option is a 14-pin or 16-pin DIP socket connected to a ribbon cable. (Note: LCDs with a backlight will have 16 pins, two of which control the backlight.)

NOTE: It may take you a little time to devise and implement a good physical interface between the LCD and your board so don’t wait too long before getting started on this interface. Wire can be used to easily attach your LCD to your board without requiring precise drilling of mounting holes.

Alternatively, standoffs can be used to mount your LCD above circuitry on your board. A limited supply of #2 machine screws and standoffs are available from the instructor.

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.
8. [Required Element] 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. For example, you might implement the following two functions.

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

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

**NOTE:** The MICRO-C compiler accepts two types of comment syntax: Traditional K&R C style comments, such as ‘/*’ and ‘*/’, and C++ style comments, such as ‘’.’’. Use a logic analyzer to prove that your byte write function sends the correct signals and has the correct I²C timing.

9. [Required Element] 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.
  void lcdgotoaddr(unsigned char addr)

- // Name: lcdgotoxy()
  // Description: Sets the cursor to the LCD DDRAM address corresponding to the specified row and column. Must call lcdgotoaddr().
  void lcdgotoxy(unsigned char row, unsigned char column)

- // Name: lcdputch()
  // Description: Writes the specified character to the current LCD cursor position.
  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)

10. [Required Element]

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 should be turned in with your lab.

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 on E going high. If E goes high during this test, then your implementation is incorrect.
11. [Required Element\(^1\)]

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. Do not make the user type in "0x" before the address hex value.

- **Swap Bytes**: Enter two EEPROM addresses in hex. If either address is invalid or if the addresses are equal to each other, display a descriptive error to the PC screen. If both addresses are valid, exchange the two bytes in the EEPROM.

- **Store String**: Enter and store two separate 24-character null-terminated ASCII user strings in EEPROM. Your program must prompt the user for a string number (1-2). As the user enters the string, echo the characters to the PC screen. Your program must support the backspace key, to allow the user to correct an input mistake. Be sure that you do not allow your string to overflow the array you allocated in your C code.

- **LCD Display**: Select a user string number (1-2) and display the string on the LCD. This function must utilize the `lcdgotoxy()` and `lcdputstr()` device driver functions. Strings exceeding the width of the LCD Display should wrap to the next line.

- **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:

  
  
  
  AAA: DD 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 AAA is the starting address (in hex) for each block of 16 data values DD (in hex). The first memory cell in the EEPROM is assigned address 000.

12. [Supplemental Element\(^1\), 7 points max]:

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, 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".

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

  ```c
  void eereset()
  
  // Name: eereset()
  // Description: Performs a software reset of the I2C EEPROM using an algorithm that conforms to Microchip application note AN709.
  
  Use a logic analyzer to prove that eereset sends the correct signals and has the correct I2C timing.
  ```

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

**NOTE:** Remember not to use any local variables from within the context of an ISR (a MICRO-C constraint). This includes any functions that your ISR calls.

**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.
13. **[Supplemental Element\(^1\), 5 points max]**:

**NOTE:** The following routines can be implemented as a stand-alone program, or can be integrated into the previous C programs above. Either way is acceptable.

Design and implement C routines which allow the user to create custom LCD characters using CGRAM. Explore other features of the HD44780 LCD controller.

- Implement the following function:
  ```c
  void lcdcreatechar(unsigned char ccode, unsigned char row_vals[])
  ```

- Write a program to provide a way for the user to interactively input (in hex) a custom character from the PC terminal emulator and to display the character on the LCD. Provide a way for custom characters to be stored in EEPROM and to be displayed after a power cycle of your board.

- 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 to control the spin rate. Also note that MICRO-C provides a `delay()` function.

- [Optional] In the same program, implement other features of the HD44780 LCD controller, including display shifting and cursor shifting and blinking. You may also choose to implement some other special effect, such as having each character of the string to be displayed 'slide' in from one side of the screen.

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

---

\(^1\) 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 an '88' (out of 100). ECEN 4613 students can earn full credit for this lab assignment by completing only the required elements.
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, April 2, 2005. 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 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 (.LST 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)
- Printout of fully, neatly, clearly commented source code. Ensure your printout is easy to read.

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."

Student Signature: ______________________________________

Checklist

Required Elements
- Pins and signals labeled and decoupling capacitors present on board
- Serial EEPROM functional, contents present after power cycle:
- C code for EEPROM hex byte reads, writes, and swaps functional, \( \text{I}^{2}\text{C} \) timing correct:
- LCD functional, C code for basic LCD routines functional:
- LCD control signal timing meets specifications:
- EEPROM user string storage and display:
- Hex dump of EEPROM functional: ___________________ __________[88/100]

Instructor/TA signature and date

Supplemental Element (Not required for ECEN 4613. Qualifies ECEN 5613 students for higher grade.)
- Elapsed time display (accurate 1 second resolution):
- Good integration with previous code, all functions work with no irregularities:
- Elapsed time stop, restart, reset to "00:00", up/down:
- Correct eereset() implementation (prove with LA): _____________________________[7]

Supplemental Element (Not required for ECEN 4613. Qualifies ECEN 5613 students for higher grade.)
- Support for user-defined LCD characters:
- Spinner symbol:
- (optional) Display/cursor shifting, blinking, others: _____________________________[5]

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