Lab Overview

In this lab assignment, you will do the following:
- Add decode logic, an EPROM, and a status LED to the hardware developed in Lab #1.
- Write simple assembly programs to test EPROM accesses and perform user I/O.
- Learn how to use the 8051 timers and write ISRs in assembly.
- Learn how to use a device programmer and EPROM eraser.
- Learn how to use a logic analyzer to capture state and timing information.
- Continue learning how to use the ASM51 assembler and Emily52 simulator.

This lab assignment is due by **Saturday, February 26, 2005**.
The deadline for this lab is **Wednesday, March 2, 2005**.

This lab is weighted as 17% of your course grade.

Lab Details

1. Refer to Lab #1 regarding layout considerations, labeling, etc. All signals on all ICs must be labeled.
2. Design and implement your decoding circuitry so that your memory map looks like the following:
   Your EPROM must be located starting at address 0000h and must occupy 24KB of address space (addresses 0000h-5FFFh). Your SRAM must be located starting at address 6000h and must occupy 32KB of address space (addresses 6000h-DFFFh). The next 8KB of address space (addresses E000h-FFFFh) should be reserved for peripherals, which you will add later in the semester. Note: The top 8KB of the 32KB EPROM will not be used. Use a combined program and data space by ANDing the \( \overline{PSEN} \) and \( \overline{RD} \) lines to create a \( \overline{READ} \) signal. Options for your decode logic include discrete logic, a 74LS138, a 74LS156, or a PAL (if you know how to use PALs). Use a 74LS373 to demultiplex the 8051 address/data bus. If you choose to use a '245 transceiver to buffer Port 0, you can use the \( \overline{READ} \) signal to control the direction of data flow through the transceiver.
3. Design and implement a circuit which will allow you to drive an LED using one of the 8051 Port 1 or Port 3 pins. You may want to use a transistor and current limiting resistor in your design. Good choices for port pins are P1.1–P1.7.
4. Design and implement your EPROM circuit. Your EPROM must drive the bus only during a microcontroller read cycle. The EPROM must not drive the bus during a microcontroller write cycle. **Note:** During the next lab, you will wire up the SRAM chip. Its wiring will be very similar to the EPROM's wiring, except that the SRAM will be active during both read and write cycles. Learn how to use a digital logic probe to verify basic control signal operation.
5. Obtain a copy of the document which compares the Intel hex record format and the Motorola S-record format, and make sure you understand how hex records are used.
6. Read the document "Device Programmer Presentation Handouts" available on the course web site.
7. Learn how to generate Motorola S-records and Intel hex records with the software tools in the lab, and how to program your EPROM using one of the device programmers in the lab. Be able to choose the correct EPROM type and to verify that an EPROM is blank before programming. Be able to verify that the contents of the EPROM match the contents of the buffer on the PC after the EPROM is programmed. Learn how to erase an EPROM. Be careful to insert your EPROMs into the device programmer correctly. Do not solder near the device programmer, as solder can easily damage the programmer electronics. When using the external parallel port programmer, only use the power adapter specifically made for that particular programmer. Use of the wrong power adapter could damage the device programmer.
8. [Optional] For initial hardware bring up, write a simple ‘NOP CPL AJMP’ infinite loop in assembly. Your code should start at address 0000h and then jump to the loop, which loops at address 0021h. Verify that the microcontroller correctly executes this code out of the EPROM. This will allow you to verify that fetches from the EPROM are happening correctly and that the 8051 is correctly executing instructions. Your code should toggle an unconnected 8051 port pin to help you verify that your code is running properly. A logic probe can be used to check the pin output.

9. [Required Element] Write an assembly program which contains two parts; a main loop and an interrupt service routine (ISR). The main assembly code should first initialize the 8051 registers and then enter an infinite loop. An ISR triggered by Timer 0 must blink an LED (by toggling a port pin) at about 1 Hz (on for ~0.5 seconds and off for ~0.5 seconds). A second unused port pin must be toggled each time the ISR executes (set the port pin to a logic high as the first instruction in your ISR, and clear the port pin to a logic low immediately before you execute the RETI instruction).

You can debug your code using Emily52 so that you reduce the time you spend programming and erasing EPROMs, but note that full timer and interrupt support is not present in our version of Emily52. The 'V' command allows you to vector to an ISR in Emily52.

[Note: After you get your SRAM and RS-232 interface working, you will be able to use a monitor program so that you won't have to go through so many EPROM program/erase cycles.]

- Using the instruction set summary tables (available in the programmer’s guide or instruction set documents), calculate how long the ISR takes to execute once, assuming a clock frequency of 11.0592 MHz.
- Compare the calculated ISR time to the time measured with the second port pin, which toggles at the beginning and at the end of each ISR execution. Do the two times match? Explain any differences you see.

10. [Required Element] Hook up the logic analyzer to the address bus (at least A[7:0]), data bus (all 8 signals D[7:0]) and the control lines on the 8051 and capture fetches of instructions from the EPROM. Be able to decode the data shown on the logic analyzer and prove that the fetched instructions match the contents of the EPROM. Learn how to use both the state and timing modes of the logic analyzer (you may be quizzed on this, so practice this until you're good at using the logic analyzer). Using the state mode, capture a sequence of instructions and compare the sequence to the listing file for the code being executed. For the state clock, you can investigate using PSEN, READ, or ALE. Using the timing mode, measure the time which elapses from when the 74LS373 latches the address supplied by the 8051 to when the READ logic input is activated during an instruction fetch.

- Prove that your measured time meets the C501 data sheet value for tLLPL.
11. [Supplemental Element, 10 points max]: Design and implement a circuit using a 74LS374 which allows values to be written to the 74LS374 chip whenever a write cycle is performed in EPROM address space (0000h–5FFFh).

- **Perform a timing analysis to prove that your design satisfies the setup and hold requirements for the 74LS374. Your timing analysis should consist of two parts. First, calculate your circuit's minimum setup and hold time using the data sheets for the logic chips used in your design. Second, use a logic analyzer to measure the setup and hold time as seen at the '374 chip. Does your measured time satisfy the setup and hold time requirements of the '374?**

The outputs of the 74LS374 should be constantly enabled and can be left unconnected. The outputs can be monitored using a logic probe or logic analyzer, or they can be hooked up to LEDs, since the 74LS374 can drive more current than other ICs. This piece of hardware can be used to help debug firmware. Some students may choose to hook up the '374 to a 7-segment LED display. Try to minimize power usage if you choose to use LEDs.

Hook up a momentary pushbutton switch to one of the unused Port 1 pins and configure the pin as an input. Debounce the switch in firmware (debounce both the switch press and release). Your debounce algorithm should correctly handle a button bounce time of at least 10ms, even if you do not observe any switch bounce with the pushbutton included in your parts kit. Each time the switch is pressed, increment the value written to the 74LS374. Demonstrate your solution and prove that each button press is interpreted as a single event. Submit a listing of your commented firmware.

12. [Supplemental Element, 5 points max]: Due to how the human eye and brain respond to light, it is possible for an LED to appear to be continuously lighted, even if it isn't. This phenomenon can be exploited to reduce overall power consumption in an embedded system. By driving the LED with a pulsed signal of appropriate duty cycle and frequency, the power consumed can be reduced while maintaining the appearance of a steady light. Design a firmware module which uses a square wave or pulse width modulated (PWM) signal to drive the LED and which reduces overall power consumption in the circuit. Implement this device driver and by experimentation, determine the characteristics (duty cycle, frequency, etc.) of the driving signal which minimizes power usage in the LED driver circuit while maintaining a steady light. Implement a firmware module which demonstrates at least four different combinations of frequency and duty cycle for at least 3 seconds each. Choose combinations which provide visual differentiation so that the effects of frequency and duty cycle can be distinguished with the human eye. Consider how much power your solution takes as compared to driving the LED with a 100% duty cycle signal.

---

1 Required elements are necessary in order to proceed to the next lab assignment. 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 '85' (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, February 26, 2005. Labs completed after the due date or submitted after the deadline date will receive grade reductions.

Print your name below, sign the honor code pledge, circle your course number, 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 copy of your schematic, a printout of your full listing file (.LST file, printed legibly and complete with comments), and the answers to any applicable lab questions to the instructor in order to receive credit for your work.

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

☐ Schematic of acceptable quality, correct memory map
☐ Pins and signals labeled on board
☐ EPROM, decode logic, and LED functional
☐ Knows how to use device programmer with correct settings
☐ Knows how and when to use a digital logic probe
☐ Demonstrated ability to use logic analyzer to capture bus cycles and view fetches from EPROM. Shows detailed knowledge of both state and timing modes. Captures latched address lines A[7:0], data lines D[7:0], ALE, /PSEN, and EPROM chip select signal on the logic analyzer display.
☐ Measured and compared t\text{LLPL} to data sheet: ________
☐ Assembly program and timer ISR functional, code commented
☐ ISR timing calculations shown on listing printout [85]

Instructor or TA signature and date

Supplemental Elements (Not required for ECEN 4613. Qualifies ECEN 5613 students for higher grade.)

☐ 74LS374 debug port and timing analysis
☐ Pushbutton switch hardware
☐ Firmware debounce functional and code commented [10]
☐ Reduced power LED device driver and demo firmware [5]

Instructor/TA Comments: □ □ □