Lab Overview

In this lab assignment, you will do the following:

- Add decode logic, an NVRAM (EPROM substitute), and a status LED to the hardware developed in Lab #1. **In this lab an NVRAM, instead of an EPROM, will be used for code storage.** Note that the same NVRAM will be used in Lab #3 as a standard SRAM, for both data and code storage.
- Write simple assembly programs to test NVRAM 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.
- Learn how to use a logic analyzer to capture state and timing information.

This lab assignment is due by **Saturday, February 21, 2009.**

The submission deadline for this lab is **Wednesday, February 25, 2009.**

This lab is weighted as 10% 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. Solder in the 28-pin wire wrap sockets for both the EPROM and SRAM. **Do not solder in the ZIF socket that is included in the tool kit - that ZIF socket needs to be returned at the end of the semester.**
3. Design and implement your decoding circuitry so that your memory map looks like the following: Your NVRAM (EPROM) must be located starting at address 0000h and must occupy 32KB of address space (addresses 0000h-7FFFh). Note that in future lab assignments, you will be adjusting your memory map. The last 32KB of address space (addresses 8000h-FFFFh) should be reserved for use later in the semester. Options for your decode logic include the Atmel ATF16V8C SPLD (the SPLD is the preferred solution), discrete logic, a 74LS138, or a 74LS156. Use a 74LS373 to demultiplex the 8051 address/data bus.
4. Design and implement your NVRAM (EPROM) circuit. Your NVRAM (EPROM) must drive the data bus only during a microcontroller program read cycle. The NVRAM (EPROM) must not drive the bus during a microcontroller write cycle or data read cycle. Learn how to use a digital logic probe to verify basic control signal operation. **If using the Atmel SPLD for decode logic, make sure you have a way to remove the chip from your system. You may need to reprogram it.**
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 NVRAM using one of the device programmers in the lab. Choose the correct NVRAM type (**for the TI BQ4011YMA NVRAM, choose NOVSRAM as the device type, Benchmarq as the manufacturer, and BQ4011Y as the device**). Be able to verify that a device is blank before programming. Be able to verify that the contents of the NVRAM match the contents of the buffer on the PC after the NVRAM is programmed. **Be careful to insert your device into the device programmer correctly.** Do not solder near the device programmer, as solder can easily damage the programmer electronics. When using the external device programmer (parallel port or USB), use only the power adapter specifically made for that particular programmer. Use of the wrong power adapter could damage the programmer.
8. Carefully insert the 28-pin ZIF socket into the "EPROM" wire wrap socket (either orientation of the ZIF socket is fine - choose an orientation that eliminates any interference between the ZIF lever and any components on your board). Do not solder in the ZIF socket that is included in the tool kit - that ZIF socket needs to be returned (in good condition) at the end of the semester.

9. [Optional] For initial hardware bring up, write a simple 'NOP CPL AJMP' infinite loop in assembly as shown on the hex record handout. 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 NVRAM. This will allow you to verify that fetches from the NVRAM are happening correctly and that the 8051 is correctly executing instructions. Your code should toggle an disconnected 8051 port pin to help you verify that your code is running properly. A logic probe can be used to check the pin output.

10. 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.

11. For source code control, the free Subversion (SVN) version control system is recommended: http://subversion.tigris.org.

12. [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 potentially debug your code using Emily52 so that you reduce the time you spend programming NVRAMs, 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 data memory (e.g. XRAM, NVRAM, or SRAM) and RS-232 interface working in the next lab assignment, you will be able to use a monitor program so that you won't have to go through so many NVRAM/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. You will likely have some conditional jumps in your ISR code, so make sure to calculate both the longest and shortest time it takes the ISR to execute.

- 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.

13. [Required Element] Hook up the logic analyzer to the address bus (all 16 signals A[15:0]), data bus (all 8 signals D[7:0]), the control lines on the 8051, and the chip selects from the decode logic and capture fetches of instructions from the NVRAM. Be able to decode the data shown on the logic analyzer and prove that the fetched instructions match the contents of the NVRAM. 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 PSEN signal is activated during an instruction fetch.

- Prove that your measured time meets the C501 data sheet value for tLLPL.
14. [Supplemental Element\(^1\), 10 points max]: Design and implement a debug circuit using a 74LS374 latch which allows values to be written to the 74LS374 chip whenever a write cycle is performed in "EPROM" address space (0000h–7FFFh). This latch can be used to help debug firmware by tracing function calls - each function could write a unique value to the latch, and the sequence of latch values could be seen using a logic analyzer. The latch must **not** activate for writes in the higher address range of 8000h–FFFFh.

Devise a good method for proving that your debug latch works correctly and that you can change its value under software control. One method is to have a value written to the latch at the beginning of your ISR - that value should be incremented each time the ISR is executed, and will repetitively cycle through values from 80–FFh. A second value will be written to the latch inside of the main loop (non-ISR) - that value should be incremented each time the main loop is executed, and will repetitively cycle through values from 00–7Fh. Note that the two sections of code will always generate debug codes that are unique to their section of code (e.g. the ISR will never output debug codes in the range of 00–7Fh).

The outputs of the 74LS374 should be **constantly enabled** and can be left unconnected on your board. The outputs can be monitored using a logic analyzer, or they can be hooked up to LEDs, since the 74LS374 can drive more current than other ICs. 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.

- **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?

Submission Checklist

In addition to the items listed on the signoff checklist, be sure to review the lab for additional requirements for submission, including:

- ISR timing measurements and calculations shown on listing printout; \( t_{LPL} \) timing analysis.
- All code commented (both assembly and SPLD code); printout neat and easy to read.

**NOTE:** Make copies of your code, SPLD code, and schematic files and save them as an archive. You will need to submit the Lab #2 files electronically at the end of the semester.

**NOTE:** Students are highly encouraged to start Lab #3 as soon as they finish Lab #2. Lab #3 is more complicated than Lab #2 and will take more time to complete. Students should also be making progress on their other assignments (e.g. student presentation and final project) in parallel.

**NOTE:** When used as an NVRAM as the only non-volatile code storage device in the system, the NVRAM must be removed from its socket and reprogrammed on a device programmer after each code change. Erasing and programming a parallel UV EPROM takes even more time and is tedious. After Lab #2, students will substitute the Atmel AT89C51RC2 processor for the existing C501. This Atmel part has built in Flash memory and will enable in-system updates to program code via the serial port on your embedded system.

---

\(^1\) 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 a '90' (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 21, 2009**. 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 full copy of complete and accurate schematic of acceptable quality (all components shown), 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. No cover sheet, please.

- Signed and dated signoff sheet (No cover sheet please)
- Full copy of complete and accurate schematic of acceptable quality (all old/new components shown).
  - Include programmable logic source code (e.g. .PLD file).
- Printout of fully, neatly, clearly commented code in .LST file. 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. I have clearly acknowledged work that is not my own."

**Student Signature:** ______________________________________

**Signoff Checklist**

**Required Elements**

- Schematic of acceptable quality, correct memory map
- Pins and signals labeled, decoupling capacitors, and two 28-pin wire wrap sockets present on board
- NVRAM (EPROM substitute), 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 NVRAM. Shows detailed knowledge of both state and timing modes. Captures latched address lines A[15:0], data lines D[7:0], ALE, /PSEN, and NVRAM chip select signal on the logic analyzer display.
- Measured and compared $t_{LLPL}$ to data sheet: __________
- Assembly program and timer ISR functional: _____________________________ [90]

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

- 74LS374 debug port and timing analysis _____________________________ [10]

**Instructor/TA Comments:** □ □ □