Lab Overview

In this lab assignment, you will do the following:

- Add decode logic, an NVRAM (an 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 data storage.
- Write simple assembly programs to test NVRAM accesses and perform user I/O.
- Continue learning how to use the Siemens C501 processor.
- Learn how to use the 8051 timers and write ISRs in assembly.
- Learn how to use a device programmer for code storage.
- Learn how to use a logic analyzer to capture state and timing information.

Students must work individually and develop their own original and unique hardware/software.

The signature due dates for this lab assignment are **Friday, February 14, 2014 (Required Elements) and Tuesday, February 18, 2014 (Supplemental Elements)**.

The submission due date for this lab is **11:59pm Wednesday, February 19, 2014**.

The cutoff date for this lab is **Wednesday, March 05, 2014**.

Labs completed after the signature due date or submitted after the submission due date will be accepted, but will receive grade reductions. Labs will not be accepted after the cutoff date.

This lab is weighted as 10% of your course grade.

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 'B+'. ECEN 4613 students can qualify for full credit for this lab assignment by completing only the required elements.**

All items on the signoff sheet must be completed to get a signature, but partial credit is given for incomplete labs. Note that receiving a signature on the signoff sheet does not mean that your work is eligible for any particular grade; it merely indicates that you have completed the work at an acceptable level.

Note: Logic analyzers will be used for the rest of the semester starting with this lab assignment. In addition to the large HP/Agilent logic analyzers available in the lab, the professor has a number of 34-channel LA1034 Intronix LogicPort logic analyzers available for students to sign out for the semester. Students can download the LogicPort software and help files from http://www.pctestinstruments.com. Students will need to take financial responsibility for any equipment they sign out, and must return the items in good semester by the final project submission date. The LogicPort analyzers are PC-based, with a USB connection, and are quite portable.

Note: All of the NVRAM chips included in the tool kits were individually checked in the device programmers by the TA's to verify that the chips are in good working condition. Take good care of these chips, since they're expensive (~$25 each), and if you damage the chip you will need to pay for a replacement.

Lab Details

1. Review Homework assignments #3, #4, and #5, which are associated with Lab #2.
2. Refer to Lab #1 regarding layout considerations, labeling, etc. All signals on all ICs must be labeled.
3. 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.
4. 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.
5. Design and implement your NVRAM (EPROM) circuit. Your NVRAM (EPROM) must drive the data bus only during a microcontroller program read cycle. For this lab, the NVRAM should simulate an EPROM: it must not drive the bus during a microcontroller data read cycle, and it must not accept data during a microcontroller data write cycle. **If using the Atmel SPLD for decode logic, make sure you have an easy way to remove the SPLD chip from your system. You may need to reprogram it.**
6. 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.
7. Read the documents available on the course web site regarding the device programmers we use in our laboratory. You will need to use these programmers with your SPLD, NVRAM, and processor.
8. 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 **Benchmarq** as the manufacturer and **BQ4011Y** as the device) [and if using the Needham's programmer, select **NOVSRAM** as the device type and make sure the correct family module adapter is installed on the programmer]. 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.** Incorrect insertion will damage the (expensive) NVRAM. 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.
9. Carefully insert the 28-pin ZIF socket into the "EPROM" wire wrap socket on your board (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.
10. [**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 unconnected 8051 port pin to help you verify that your code is running properly. A probe can be used to check the pin output.
11. 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.
12. [Optional] For source code control, the free Subversion (SVN) version control system is recommended: **http://subversion.tigris.org**. Instructions for setting up a Subversion repository using ECES can be found at http://ecee.colorado.edu/~mcclurel/Subversion_Overview_10-12-2009.pdf.

Note: In this lab, continue using the same Siemens C501 processor that you used in Lab #1.

13. [**Required Element**[1]] 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 2.5 Hz +/- 0.2% (on for ~0.200 seconds and off for ~0.200 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 some of your code using EdSim51 or 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 RS-232 interface and flash-based processor working in the next lab assignment, you will be able to program your processor while it is in the system.]

    Optional: As a suggestion on how you might differentiate your work, you can consider implementing an option to generate a Morse Code SOS signal in addition to the basic blinking LED. This is completely optional. http://en.wikipedia.org/wiki/SOS

- **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 calculated and measured times match? Explain any differences you see.**

14. [**Required Element**[1]] 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 $\overline{PSEN}$, $\overline{READ}$, or ALE. **Before your demo to the TA, prepare one screen capture of the logic analyzer triggered on a fetch in state mode.**
- Using the timing mode, measure the time which elapses **from when the 74LS373 latches the address supplied by the 8051 to when the** $\overline{PSEN}$ **signal is** activated during an instruction fetch. **Before your demo to the TA, annotate a screen capture to show the measurement of t_{LLPL} in timing mode and prove that your measured time meets the C501 data sheet specification for t_{LLPL}.**
- **Show and discuss both of these screen captures with the TA during the signoff.**

15. **[Supplemental Element[1]]:** 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 "CODE / EPROM" address space (0000h–7FFFh). Note that for this lab the NVRAM will not be activated during a write cycle, since in this lab the NVRAM is simulating a non-writable EPROM. The latch must **not** activate for writes in the higher address range of 8000h-FFFFh. A debug latch like this can be used to help debug firmware by tracing function calls - each function could write a unique value to the 74LS374 latch via a standard Port 0 data write transfer, and the sequence of latch values could be seen using a logic analyzer.

    Devise a method for proving that your debug latch works correctly and that you can change its value under software control. One method is to write a value 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 codes in the range of 00-7Fh).

    The outputs of the 74LS374 should be constantly enabled (so that they're always driving valid data out) and these outputs can be left unconnected on your board. The outputs can be monitored using a logic analyzer. (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?**

[1] Required elements are necessary in order to meet the requirements for the lab. Supplemental and challenge 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 minimum requirements for the lab.

Embedded System Design Grading Criteria

CU grading criteria are used for this course.

A	Superior, outstanding
A-	
B+	
B	Above average
B-	
C+	
C	Average, has adequately met course requirements
C-	
D+	
D	Below average
D-	Minimum passing grade
F	Fail, has not met course requirements

Note: The minimum overall course grade that counts towards the embedded systems certificate is a "B-".

To be confirmed: The minimum overall course grade that counts towards a Master's degree is a "C".

To be confirmed: The minimum overall course grade that counts towards a Ph.D. degree is a "B-".

Characteristics of an 'A' grade

- All assigned work completed, including supplemental elements
- Work completed, signatures obtained, and report submitted on time
- Excellent quality, including schematics and code
- Submission is in top 20% of class regarding overall quality
- Exceeds expectations, student went above and beyond requirements in their assignment, and work stands out compared with the work of other students. Student meets all other requirements
- Detailed and relevant code comments, including header comments for every file
- Code follows a coding standard and is nicely formatted
- Student able to complete assignment independently without excessive help from TA and others
- Student signoff completed without excessive retries
- Student presentation during sign-off is superior; student well organized and efficient
- Student clearly communicates knowledge during sign-off
- Lab submission completely legible and makes grading easy to achieve
- Student attitude positive and cooperative
- Student helpful to other students, answers peer questions, etc.
- Student demonstrates complete understanding of the material
- Student follows instructions for lab submission

Approximately 10% to 30% of the class could expect this grade

Characteristics of a "B" grade

- Lower quality than an "A" submission
- Some elements may be incomplete or the submission lacks thoroughness and/or quality
- Some elements may have been completed after the due date
- Student's work and presentation of knowledge not outstanding as compared with others

Characteristics of a "C" grade

- Lower quality than an "B" submission
- Student has demonstrated basic competence with the assignment, and has met most core learning goals

Submission Preparation

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_{LLPL} timing analysis**.**
- All code is well commented (both assembly and SPLD code), including header comments; printout (PDF) neat and easy to read.

NOTE: Make and save archive copies of your assembly/SPLD code and schematic files. You need to submit the Lab #2 files electronically, now and at the end of the semester.

Submission Instructions

Instructions: Print your name, circle your course number, and sign the honor code pledge. Separate the signoff sheet from the rest of the lab and turn in a scan of the signed form, the items in the checklist below, and the answers to any applicable lab questions in order to receive credit for your work. No cover sheet please. **Submit as many items as possible electronically via Desire2Learn (D2L, https://learn.colorado.edu), to reduce paper usage.**

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

- Scan of signed & dated signoff sheet with honor code pledge as the top sheet (No cover sheet please)
- (Scan of) answers to applicable lab questions (e.g. ISR timing calculations and measurements shown on .LST file, timing proof for t_{LLPL}, setup/hold time timing analysis)
- Full copy of complete and accurate schematic of acceptable quality (all old/new components shown). Include programmable logic source code (e.g. .PLD file for the SPLD).
- Fully, neatly, and clearly commented code in .ASM and .LST file. Ensure your code is neat and easy to read, and that each source file has header comments that identify the author and any leveraged code the file contains.

Make copies of your code, SPLD code, and schematic files and save them as an archive.

If you submit any items in hard copy format, make sure your name is on each item and staple the items together, with the signoff sheet as the top item.

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. current topics presentation and final project) in parallel.

NOTE: When used 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.

ECEN 4613/5613 **Embedded System Design** **Spring 2014**

Lab #2 Signoff Sheet

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. Signatures are due by **Friday, February 14, 2014 (Required Elements) and Tuesday, February 18, 2014 (Supplemental Elements)**.

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.

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, SPLD .PLD file
- ❑ Pins and signals labeled, decoupling capacitors, and two 28-pin wire wrap sockets present on board
- ❑ NVRAM (as EPROM substitute), decode logic, and LED functional
- ❑ Understands device programmer.
- ❑ 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.
- ❑ Shows and discusses logic analyzer screen captures:
- ❑ Assembly program and timer ISR functional:

_____
TA signature and date

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

- ❑ 74LS374 debug port functional
- ❑ Understands timing analysis, setup/hold/propagation

_____
TA signature and date

Instructor/TA Comments: □ □ □

FOR INSTRUCTOR USE ONLY **Required Elements**	Not Applicable	Poor/Not Complete	Meets Requirements	Exceeds Requirements	Outstanding
Schematics, SPLD code	☐	☐	☐	☐	☐
Hardware physical implementation	☐	☐	☐	☐	☐
Required Elements functionality	☐	☐	☐	☐	☐
Sign-off done without excessive retries		☐	☐		
Student understanding and skills	☐	☐	☐	☐	☐
Overall Demo Quality (Required Elements)	☐	☐	☐	☐	☐

FOR INSTRUCTOR USE ONLY **Supplemental Elements**	Not Applicable	Poor/Not Complete	Meets Requirements	Exceeds Requirements	Outstanding
Schematics, SPLD code	☐	☐	☐	☐	☐
Hardware physical implementation	☐	☐	☐	☐	☐
Supplemental Elements functionality	☐	☐	☐	☐	☐
Sign-off done without excessive retries		☐	☐		
Student understanding and skills	☐	☐	☐	☐	☐
Overall Demo Quality (Supplemental)	☐	☐	☐	☐	☐

NOTE: This signoff sheet should be the top/first sheet of your submission.