18-348 Lab #11

Spring 2016

This lab is is intended to be a two-week effort for a two-person team.

BOTH pre-lab design and lab implementation are to be done with your partner! Each partner should turn in a copy of the files. File names shall include group name in them.

Equipment hand-in:

- Review the equipment hand-in checklist to make sure you turn in everything
- Return your equipment after the demo (when you don't need it for your final write-up anymore)
- We must check you off for equipment hand-in
- See blackboard for a schedule for equipment drop-off

Navigation

Project Requirements:

These are the overall project requirements. These requirements will be fulfilled as parts of the pre-lab and lab below.

1. Your project shall include at least three of the following items:
   - D/A Converter
   - A/D
   - PWM
   - CAN
   - SPI
   - Serial Communications
   - Counter / timer
   - Fixed point multiply
   - Mutex
   - Priority Inheritance or priority ceiling
2. Your project shall use at least one interrupt service routine.
3. Your project shall use the watchdog/COP (that interrupt does not count for the ISR requirement)
4. Your project shall use parts in the lab or inexpensive parts (< $5 total) we can get via Tech (requires staff approval)
5. Your design shall include:
   - High level requirements
   - State-chart
   - Schematic showing any wiring and hardware
   - Well defined scheduling
   - Test plan
   - Traceability tables
6. Your testing shall include worst case timing analysis

Final projects can NOT involve any of the following:

- IRB approval for use of human subjects
- Use of hazardous or controlled chemicals

Note about RTOS projects: An example of an RTOS project would be to implement a mutex with priority ceiling protocol and use the counter/timer to initiate periodic tasks. Note that all other requirements for the project still hold (correct watchdog usage, interrupt, etc). An example of an interrupt includes: using SWI to implement yield. There are other possibilities (such as serial) but we point these out because to show that they are OK to count

---

Pre-Lab:

Goal:

- Pull together various design aspects throughout the course for your project

Discussion:

Each part of this pre-lab will draw upon some design aspect of the course lectures (or labs). We realize that many of these parts will be relatively short and simple for a two-week project.

The point of the pre-lab questions is to get your project design down on paper ***before*** you start hacking coding.

Procedure:

Part 1 - Overall project description

Write a ***short*** overall description of your project.

- Your description shall not exceed 200 words.
- You may include a graphic or two to help illustrate your idea.
- Include a list of at least three items from requirement 1 above.
- Include a list of at least one interrupt the ISRs will service in your project

Part 2 - Requirements

In this section, you will define the high level inputs, outputs, and state variables needed for requirements and state-chart design. Your requirements shall use these inputs and state variables to define changes in other state variables and outputs. You will use your requirements to develop the state-chart for your project.

The number of inputs, outputs, and internal states will depend on the complexity of your system. Define each as follows:

1. Define system inputs. Each input shall have a simple descriptive name such as "Off_Button", a single sentence description of its purpose, and the possible values (or ranges). These include any buttons, switches, potentiometers, serial interface inputs, or other user inputs to your system. You may define a set of inputs as an array if that is more convenient.

   Example: Off_Button - Indicates user preference to turn the wipers off. Values are PRESSED or UNPRESSED.

2. Define system outputs. Each output shall have a simple name, a single sentence describing the output behavior, and the possible values (or ranges). These may include any externally visible behaviors of LEDs (or LED arrays), LCD display, serial outputs, or other outputs.

   Example: Motor_Speed - Output sets the speed of the motor. Values are OFF, SLOW, and FAST.

3. Define internal state variables that are needed to track the current system state. Each internal state variable shall have a simple descriptive name, a single sentence description of its purpose, and possible values (or ranges).

   Example: Current_Floor - Current floor the elevator is stopped at. Value range from 1 (MIN_FLOOR) to 8 (MAX_FLOOR)

4. Define your high level requirements. Your requirements shall describe the overall behaviors of the system. We expect you will have less than 10 requirements for a project of this complexity. Refer to the Lab 3 wiper controller requirements for a good example.
   - Each requirement shall have a distinct number (we leave the numbering method to you).
   - Each requirement shall use the term "shall" or "should".
   - Each requirement shall be a single sentence in length.
   - Each requirement shall be in terms of defined inputs, outputs, and state variables (So, "It shall work" is not an acceptable requirement).

Part 3 - State-Chart Design

Develop one or more state-charts based upon your high level requirements. Follow the procedure set forth in Pre-lab 3 - Part B.

Your state-chart(s) must include:

- State numbers for each state.
- State names for each state.
- A "Do:" section with any outputs or internal state values that get set in that state.

- Outputs shall only occur within states, not on transitions.

- Transitions which are numbered and arrowed to clearly indicate the flow from one state to another.
- We do not expect you to include your ISRs into your state-chart(s).
- Your state-chart shall include ***at least*** two states.

Part 4 - Requirements / State-Chart Traceability Table

Create a traceability table which shows which states and transitions correspond to which requirements. Place an X in each row for the states and transitions which implement a behavior described by the requirement. Every requirement should correspond to a state or transition (forward traceability). Every state and transition should trace to at least one requirement (backward traceability).

You must have 100% traceability to your requirements. If you find detailed requirements that were left out of the state-chart, you may want to consolidate them into higher level requirements.

Example traceability matrix:

State / Transition -->	S1: STATE_OFF	S2: STATE_PARK	....	T1	T2	...
Requirement						
R1.	**X**					
R1.A.						
....						

Table 1 - Example Requirements to State-Chart Traceability Table

Part 5 - Scheduling

Define your plan for scheduling tasks in your system. You may use any scheduling method discussed in lecture.

Your schedule design shall:

- Include a scheduling method.
- List each task along with any priorities and periods / deadlines; state how you determined each one.
  Note: Your times and periods should be estimates at this point in the design.

Part 6 - Watchdog / COP

Define how your project will use the watchdog timer.

- Describe this in a paragraph (no more than 50 words) with respect to the tasks in your schedule.
- Define the time-out period and how you arrived at it based on the tasks in your schedule.
- Describe what actions your program shall take in the event that the watchdog resets your program.

Part 7 - Develop a Test Plan

In this section, you will develop a series of tests to ensure your design follows your state-chart and meets your requirements. Your test plan should include white box and black box testing methods. This part only includes test plans and traceability tables, you do not have to include any test results.

Part 7.1 - White Box Testing

Your white box tests should test the transitions of your state-chart. Create a sufficient number of white box tests to ensure 100% coverage of your state-chart transitions.

- Each white box test shall include a test number, initial state, and a series of inputs and the expected states after inputs (Table 2).
- Your test plan shall include a traceability table showing which state transitions are exercised by each white box test (Table 3).
- Your test plan shall have 100% coverage of all state-chart transitions.

	Pass/Fail	Initial State	Input 1 (State after input)	Step 2 (State after input)	Step 3 (State after input)
Test 2.1	PASS	1	N (2)	N (3)	N (4)

Table 2 - Example White Box Tests

	T1	T2	T3	T4	T5	T6	T7	....
Test 2.1	X			X			X	

Table 3 - Example White Box Test Traceability Table

Part 7.2 - Black Box Testing

Your black box tests should test that the behavior of your system conforms to your requirements. Create a sufficient number of black box tests to ensure 100% coverage of your requirements.

- Each black box test shall include a test number, and test description (Table 4).
- Your test plan shall include a traceability table showing which requirements are exercised by each black box test (Table 5).
- Your test plan shall have 100% coverage of all requirements.

	Pass/Fail	Test Description	Result
Test 3.1	PASS	Verify the timing of FAST MODE using a stopwatch to measure average cycle time over 10 cycles.	--

Table 4 - Example Black Box Tests

	R1	R2	R3	R4	R5	R6	R7
Test 3.1						X	

Table 5 -Example Black Box Test Traceability Table

Refer to Pre-lab 6 and Lab 6 - Part B for a refresher. Also refer to Lecture 11 - Debug and Test.

Part 8 - Hardware

Create a list of the components that your project will incorporate. If you need hardware other than what is provided in the lab, you must receive approval from the course staff.

Draw a simple block diagram of the hardware components of your system. You do not need to indicate specific ports and pin numbers.

Part 9 - Acceptance Test Plan

State the procedure that the TA can execute to determine if your project is functioning and has met the requirements of this project.

You should include enough detail that a TA can perform the acceptance test without either of the partners present.

Make sure these are the ***simplest criteria*** to prove that you met the requirements above. If your project includes complex hardware or PC side program, make sure that you have a simplified version of your project to demo to the TAs that does not rely on those complex components.

For example: If your project includes a complicated PC-side program that communicates with the module over serial, be able to demonstrate that the module can send and receive data over the serial port by manually typing queries and getting responses using HyperTerminal on the PC.

Hand-in Checklist: (210 points)

All non-code submissions shall be in a single PDF document.

Part 1

- (10 points) Short overall project description (may include graphics)
- (5 points) List of at least three items from requirement 1.
- (5 points) List of at least one interrupt the ISRs will service.

Part 2

- (10 points) List of system inputs.
- (10 points) List of system outputs.
- (10 points) List of system state variables.
- (10 points) List of high level requirements.

Part 3

- (20 points) One or more state-chart that shows implementation of the majority of project requirements.

Part 4

- (20 points) State-chart to requirements traceability table.

Part 5

- (20 points) Scheduling method, priorities, and periods / deadlines; state how you arrived at these numbers.

Part 6

- (20 points) Watchdog use definition, timeout period, and how you determined the period based on scheduled tasks.

Part 7

- (10 points) List of white box tests.
- (10 points) White box tests to state-chart traceability table.
- (10 points) List of black box tests.
- (10 points) Black box tests to requirements traceability table.

Part 8

- (10 points) List of hardware components.
- (10 points) Hardware block diagram.

Part 9

- (10 points) Acceptance test plan.

Refer to the LAB FAQ for more information on lab handin procedures and file type requirements. You MUST follow these procedures or we will not accept your submissions.

Lab:

Goal:

- Implement your design from pre-lab

Discussion:

In the pre-lab, you did a majority of the design work needed for the project. In the lab portion, you will implement your design.

After implementation, you will carry out your tests, record bugs, and fix your code.

Once you have completed the implementation and test, update your documentation to accurately reflect what you did in the implementation.

Procedure:

Part 1 - Software Implementation

Part 1.1 - Coding

Implement the code for your design according to your state-chart and scheduling. Make sure your code uses the watchdog. You may write your code in either C or assembly language (whichever is more convenient). You shall follow good coding practices set forth in the coding style guide. Name your final project folder lab_11_gXX for submission.

If you implement your code differently than your original state-chart, schedule, or watchdog use, update those documents to reflect your actual implementation.

Part 1.2 - Code to State-Chart Traceability

The comments of your code shall contain notation showing where the states and transitions have been implemented. Create a traceability table which shows the line number of your software which corresponding to each state or transition in your state-chart. We recommend you add these comments as you implement your software to save yourself time later. Include this traceability table in your final write-up.

State or Transition	Implementing Code
S1	line XX
....	....

Table 6 - Example state-chart to code traceability table

Part 2 - Wiring

Wire your design according to your block diagram, making any adjustments that are necessary.

Once you have finished your wiring, record the wiring and associated pins for a schematic. You will include the schematic in your final write-up.

Tools such as Microsoft Visio can be used to make schematics. To produce schematics similar to the ones used in previous labs ExpressPCB can be used.

Part 3 - Testing

Part 3.1 - Code Review

Sit down with your partner and review each other's code. Follow the guidelines for code review in Lecture 7 - Coding Tricks; Multiprecision Math; Reviews.

Keep a table of the defects you find in the code for your write-up. Record the line number and a brief summary of the defect. Remember, restrain yourself and wait until you have finished reviewing the code to fix the bugs.

Defect #	Line #	Description of Defect
...		

Table 7 - Example defect table

Part 3.2 Worst Case Timing Analysis

Perform worst case timing analysis for each task (not each line of code) in your main loop and ISR. Include the maximum (worst case) execution time of each task and the maximum (worst case) latency for each task to begin. Also, note any sources of blocking time for each task.

You may perform these measurements in any way that is convenient. Previous labs have used the oscilloscopes (Lab 8) and simulator (Lab 9) to measure execution time and latency.

Use these measurements to fine tune your watchdog timeout periods. Be sure to update these values in your final write-up.

Part 3.3 Execute Test Plan (White Box and Black Box Testing)

Perform each white box and black box test and record the results (PASS/FAIL). If a test fails, record what happens in your pre-lab test tables. Keep a separate table for black box and white box test results. Turn in the test results in your write-up.

Bug #	Test #	Requirements(s)	Description
...			

Table 8 - Example bug list found through testing

Part 4 - Correct Documentation

Once you have completed your tests, fix your bugs. You may find that a bug was not a direct result of a coding mistake. It may be due errors or omissions in your requirements, state-chart(s), wiring, scheduling, watchdog use etc. Make sure you fix your documentation to reflect your actual design (yes, this includes your traceability tables too) for your final write-up. Include a change log in your final write-up.

Your change log should contain one line per section that you updated with the date of the change and a brief (single sentence) description of the change.

Change #	Section	Description
...		

Table 9 - Example change log

Part 5- Execute Acceptance Test

Once you have completed your project and debugged it, demonstrate it to the TA. They will use your acceptance test as a guideline to determine if your project works.

They may also ask you additional questions regarding bugs and corrections, traceability, scheduling, watchdog use, etc. We understand that at the point of demonstration your documentation may not be 100% up to date, but you should be able to explain the workings of your project.

Demo Checklist: (100 points)

1. (100 points) Demonstrate your project functionality according to your acceptance test. If you have trouble getting your demo to work after five minutes, scale back to a simplified version to prove you met all the requirements.

Hand-in Checklist: (200 points)

All non-code submissions shall be in a single PDF document.

1. (1 point) Tell us how many hours each team member spent from Friday 15 April to Friday 22 April, including class time. Do not include hours spent working on bonus problems.
2. (1 point) Tell us how many hours each team member spent from Friday 22 April to Friday 29 April, including class time. Do not include hours spent working on bonus problems.
3. (1 point) Tell us how many hours each team member spent from Friday 29 April to Thursday 5 May, including class time. Do not include hours spent working on bonus problems.
4. (80 points) Software implementation including all C and assembly files. Submit your entire project directory as lab_11_gXX. Code should be fully commented and follow the Coding style sheet for full credit.
5. (10 points) Code to state-chart traceability table.
6. (20 points) Wiring schematic.
7. (10 points) Code review bug list.
8. (10 points) White box test results and bug list.
9. (10 points) Black box test results and bug list.
10. (10 poitns) Worst case timing analysis.
11. (40 points) Corrected pre-lab documentation (consistent with your actual implementation) along with change log.
    - Project Title
    - Short description
    - List of at least three items from requirement 1
    - List of at least one interrupt
    - List of inputs
    - List of outputs
    - List of state variables

    - List of high level requirements
    - State-chart(s)
    - State-chart to requirements traceability table
    - Scheduling
    - Watchdog use
    - List of white box tests
    - White box test traceability table
    - List of black box tests
    - Black box test traceability table
    - Hardware components list
    - Hardware block diagram
    - Acceptance test plan
12. (6 points) Two photographs of at least 800x600 resolution
    - One photo showing system setup to see what the hardware looks like
    - One photo showing an "action shot" (e.g., user pressing buttons, etc) that is representative of the project demo.
13. (1 point) Selected project photos will be posted publicly on the course website to inspire future projects. Answer the following question: Do you want your names associated with your photo if it is selected for posting (yes/no)

Refer to the LAB FAQ for more information on lab handin procedures and file type requirements. You MUST follow these procedures or we will not accept your submissions.

Hints and Suggestions:

- None

FILES for this lab:

- None

Relevant reading:

- Lab 3 - Part B - Wiper controller inputs, outputs, state variables, requirements, state-chart, and traceability tables
- Lab 4 - Part B - Code review and traceability.
- Lab 6 - Part B - White box testing, black box testing, and testing traceability.
- Lab 8 - Oscilloscope measurements.
- Lab 9 - Task latency measurements with the simulator.
- ExpressPCB
- Coding style sheet

Also, see the course materials repository page.

Change notes for 2016:

- 4/25/16 - Edited hours logging to reflect 2016 dates -- Milda
- 3/30/16 - Added filename and title requirements -- Milda
- 3/30/16 - Added navigation -- Milda