LABORATORY ASSIGNMENT NUMBER 3 FOR CMPE 118/218

Due by 5:00pm on Wednesday, February 8, 2012
Pre-Lab Due by 5:00pm on Thursday, February 2, 2012

Purpose:
This lab is intended to acquaint you with:

- Controlling a DC motor.
- Controlling a stepper motor.
- Controlling an R/C servo.
- Finding information in data sheets.
- Pulse width modulation.
- Phenomena encountered with a DC motor.
- Limitations of techniques implemented purely in software.

Minimum Parts Required:
1 each: DC motor, R/C servo, and stepper motor (provided at each lab station), 1N4744A 15 Volt zener diode, 50K potentiometer. CMPE-118 I/O Board, CMPE-118 Dual H-Bridge, DS3568 High Current Drive, and CMPE-118 Stepper Motor Driver. Ribbon cable connectors and stake headers for the I/O board. See a TA/tutor for any parts you need that are not in your parts kit.

Pre-Lab:
Complete the following exercises AFTER you have read through the lab assignment and BEFORE coming in to complete the lab.

- 0.1) Decide which port and bit(s) you will use to control the motors (all parts of lab).
- 0.2) Decide what, if any, initialization is required for that port.
- 0.3) Describe the mode in which the A/D converter will be operating.
- 0.4) Determine which pins on which cables/connectors you need to hook up.
- 0.5) Determine which ribbon connectors you will need, and how to wire them in.
- 0.6) Draw a fully detailed state diagram for a PWM software driver. Include an example plot of the expected output signal for a given frequency and duty cycle. HINT: using timer events may assist you getting the proper duty cycles for a given frequency.

In the report:
Include a description of which of the port lines on the Uno32 you used to implement the control as well as a justification for why you chose to use that (those) particular line(s). Include the answer to question 0.3.

Part 1 Interfacing to a DC Motor and Potentiometer

Reading:
CKO Ch. 8. CMPE118_AD Library and CMPE118_PWM reference documentation, the CMPE118 Uno32 IO Board documentation, the CMPE118 DS3658 Module board documentation, the MPLAB-X Quickstart manual.
Assignment: You are to design the software necessary for the PIC32 to drive the supplied DC motor. Motor drive should use Pulse Width Modulation using the provided PWM software module. The speed at which you run the motor should be determined by the setting of an external potentiometer, which your software should read using the A/D converter on the PIC32. Your program should continuously read the voltage set by the potentiometer and set the motor speed directly proportional to the voltage (full scale A/D is full speed). You should be able to exit the program by pressing any key. When you have completed this task, you will be prepared to continue with the laboratory exercises in Part 2. At this point find a tutor or TA and demonstrate Part 1. **BE CAREFUL HERE:** Do NOT hook power and ground up with the power supply turned on. It is the easiest way to fry microcontrollers, followed by reversing power and ground. Always hook up wires with power supply **TURNED OFF!**

**NOTE:** In this lab, we are using the ATX power supplies (and will be for the rest of the quarter). They are robust, clean, and supply a great deal of current (~16Amps). While they will save themselves from a short circuit, they will **NOT** save your board. Be careful with these—they will smoke chips easily.

Set-Up:

1.1) For this, and the next, part you will be using the CMPE118 DS3658 Module board to drive the DC motor. Since the DS3658 has open collector outputs, you will need to supply the power to the motor separately. To bring the power to your proto-board for connection to the motor, use your banana plug jumpers to bring the power from the positive adjustable outputs to your proto-board.

1.2) Use the +12V voltage source from the power supply for the positive output. Now connect the supply (at the proto-board) to one lead of the DC motor using the connector from the motor assembly. Also, you will need to supply power to the CMPE118 DS3658 Module board, so connect the +12V supply to J4, pin 1 on the DS3658 module, and connect J4, pin 2 to the GND terminal on the power supply. Since you will be using this kind of connector everywhere, take the time to solder one up with red and black wire, and use heat shrink to ensure a nice robust insulation.

1.3) Make sure you have a jumper across J3 on the DS3658. Now connect the DS3658 output for Channel A to the remaining lead of the DC motor. The motor should not be rotating at this point, right? If the motor is rotating, this is due to the fact that you haven’t hooked up anything to the input of the DS3658, so the state of the input is not guaranteed – it’s floating!

1.4) Hook up the PIC32 output(s) you selected in the pre-lab to the input of the channel you are using on the DS3658 (use ribbon cable connectors between the IO board and the protoboard).

1.5) Hook the two outer leads from the 50K potentiometer to the **power supply’s +3.3V & GND terminals. Be very careful NOT to use the power going to the motor, since that is +12V and will destroy the PIC32 once you connect it.** After doing this, check to see that the voltage at the wiper (the remaining terminal on the potentiometer) swings between 0 and +3.3V. You will also use this circuit in Part 5. Check the voltage levels **BEFORE** you hook up the IO board.

1.6) Using one of the jumpers that goes from a wire to a push-on connector, make the connection from the wiper to one of the analog pins on the IO Board (on either Port V or Port W). The ADC pins are clamped with a TL7726 Hex Clamp at 5V, and additional 200Ω Resistor and Diode to 3.3V. What does this do? What limitations does it impose of the micro?

1.7) It’s finally time to try out your code. In MPLAB-X, build your project into an executable .hex file. Then, with the ds30Loader window active, turn on (or reset) the PIC32 I/O board and click on the **<Check for BL>** button. You should get the connection message. If not, go back through the MPLABX_NewProject handout from Lab0 and check all of your settings. At this point you are ready to download your code. Downloading and testing your code should proceed exactly as it did in Lab 0. Once your code is finished downloading, the ds30Loader
will switch directly to its terminal window. If you want to do more complicated things using the Serial port, use TeraTerm or RealTerm. The proper settings are at 115.2Kbaud, no flow control, and be sure to init the serial interface by calling Serial_Init() in your main.c

**In the report:** You must include the wiring diagrams of the circuits you used, noting which bits were used and what connections were made on the PIC32 IO Board and the CMPE118 DS3658 Module board, the sign-off sheet, the answers to the questions in part 1.6, as well as a listing of the software used to control the motor. All wiring diagrams and schematics should be completed using the OrCAD Capture program (or Eagle) provided on each workstation.

---

**Part 2 Exploring DC Motors**

**Reading:**  
CKO Ch. 22, 23, 24, 26 and 27

**Assignment:** Complete the following exercises:

**Basic Waveforms**

2.1) Set the O'scope up for dual trace and examine the waveforms at the input and output of the DS3658. Draw a neat and readable representation of the two waveforms. Include at least two full cycles. Label the parts of the waveform to show the active (driven) portion of the waveform, the decay period as the motor field collapses, the peak of the inductive kickback, and the back-EMF generated by the motor. Do this at 2 duty cycle levels (i.e. 20% and 80%). Make a separate scope face drawing for each. Note the different back-EMF values at the different motor speeds. To best see the back-EMF, you might want to draw a composite of two waveforms, one with the motor stalled (hold the shaft to keep it from turning) and one with the motor turning.

**Using a Zener in lieu of the kick-back diode**

2.2) Disconnect the clamp diode from the circuit by removing the jumper across J3 on the DS3658 and replace it with the 15 volt zener in a configuration like the one shown below.  
**Note:** the band on the zener diode body corresponds to the horizontal bar in the zener symbol. Repeat the labeled waveform drawing from Part 2.1. Contrast the decay times for the two techniques; be specific about durations. To do this you will probably need to use a different time scale on the 'scope. This portion will be most clear if you examine the alternatives at a very low duty cycle (i.e. active for < 1% of the time). This way the effects of back EMF do not obscure the decay time. From this exercise you can see how a zener protects the output transistor by limiting the peak voltage to the zener voltage, and at the same time minimizes the decay time of the collapsing field. It is important to minimize this decay time in order to make the duty cycle response of the motor as linear as possible. At very high duty cycles it would be possible for a long decay time to completely overlap the un-driven portion of the waveform.
The Limits of Software control

2.3) Using the oscilloscope and/or the counter-timer, determine the frequency of the duty cycle waveform that you are generating. What is the frequency?

2.4) Test the range of frequencies for the PWM module. What happens at very low frequencies (ie: < 200Hz), and happens at high frequencies (ie: >20KHz).

2.5) What happens to the upper frequency limit if we only have 10 bits of resolution? How about 16 bits? We are making a trade-off between resolution and frequency of operation.

2.6) Given a resolution, what is the key parameter in determining the total period of the PWM signal?

2.7) Can your solution achieve a full range of operation? (i.e. What happens if you try to get 0% or 100% duty cycle). If it can't, don't try and fix it, just explain why it won't work.

2.8) For a given duty cycle, what are the sources of error in the approach you have chosen? (i.e. if you ask for 50% duty cycle do you get it? ...theoretically, that is, you probably will not be able to measure the error.)

In the report: Several scope face drawings are requested; please make your drawings legible. Using different colors for different waveforms is helpful. Include the discussion requested in Part 2.2. Be sure also to include in the lab report your answers to the all the questions in Parts 2.3-2.8. To ease the grading task please quote the question you are answering before your answer. This way the grader can read the question you think you are answering followed by your answer.

Part 3 Interfacing to a Stepper Motor

Reading: CKO Ch. 26, the CMPE118 DRV8814 H-Bridge Driver Module documentation, CMPE118 STEPPER software module documentation.

Assignment: You are to design the necessary software to run on the PIC32 to drive the supplied stepper motor in Full Drive, Wave Drive, and Half-Step Drive modes. After demonstrating this to the TA or tutors, you will use a hardware stepper driver to drive the same motor.

Set-Up: 3.1) For this part, you should use the CMPE118 DRV8814 Dual H-Bridge Driver Module board.
3.2) You will need to connect power and ground to J3 of the CMPE118 H-Bridge Module, which will in turn supply current to the stepper motor. Note that you must supply it using the +12V source. J1 of the CMPE-118 H-Bridge Module will need to be interfaced to the Port Z on PIC32 I/O Board to work properly with the STEPPER module. Check in Lab for the proper connection diagram.

3.3) You will have to probe the motor wires to determine which wires correspond to which coils and then connect those wires to the proper motor outputs on the Drive Module. Use the STEPPER software module to control the Drive Module and ensure the motor rotates properly with the provided modules. You will need to experiment with the motor coil polarities to get the motor to rotate. If at first you don’t succeed, try, try again (there are only a few possibilities). Also, try changing the step rate in the software initialization.

3.4) Now, edit the STEPPER module to drive the motor using Wave Drive and Half-Step Drive. You will have to edit the state machine outputs and add additional states.

3.5) After verifying your motor can be driven all three ways, increase the frequency until it stops stepping. How fast can you go under each mode of operation? Note that having a load on the motor helps here (a finger should be fine). Does this change is you start at one frequency and then ramp up to another once the motor is moving?

3.6) Now run the motor forward a complete revolution then in reverse for a complete revolution—did you lose steps? How fast can you operate the motor without losing steps (for accurate positioning)? Again, check if this is affected by load (put a finger on the motor), and if it is affected by changing frequencies once started.

3.7) After demonstrating the above code to the TAs/Instructors and getting signed off; get one of the CMPE-118 DRV8811 Stepper Motor Driver boards. These boards have only two control signals, step and direction (plus an enable), and a jumper to select Full, Half-Step, Quarter Step, and Eighth Step Drive, as discussed in the description of the module. You will now have to write code to produce these signals.

3.8) Repeat parts 3.5 and 3.6 using the stepper driver under both Full and Half-Step operation. How did this compare to your software driver? Play with the current limiting jumpers, and see if increasing the current from 0.5 – 2Amps changes the behavior. Be careful here, you are dumping a whole lot of power through the coils, they are going to get hot. Demonstrate the working stepper module to a tutor or TA for check off.

In the report: Include the wiring diagram of the circuit you used, a copy of the code that drives the stepper motor, discussion of the performance differences between hardware and software stepper drivers, and the sign-off sheet from the coach. Be sure to indicate on the diagram which bits of which ports of the PIC32 I/O board you used.

Part 4 Interfacing to a DC Motor and H-Bridge

Reading: The CMPE 118 DRV8814 Dual H-Bridge Module documentation.

Assignment: You are to duplicate the functionality of Part 1, this time using the H-Bridge to drive the motor. You will also need to add a little complexity. You should include a provision to change the direction of rotation of the motor. When you have completed this task, get a tutor or TA to sign off on its function. For this section of the lab, you should use the supplied libraries providing PWM output.

Set-Up:

4.1) You will need to add a power connector to supply power to the H-bridge on the DRV8814 Dual H-Bridge Module. Connect power and ground to J3 of the module.
4.2) You will need to supply enable and direction inputs to the module. Remember to bring ground back to the power supply for both the module and I/O board.

4.3) Experiment and determine the maximum and minimum frequencies you can use to properly drive your motor at a given duty cycle.

**In the report:** Include the wiring diagram of the circuit you used, a copy of the code that drives the DC motor and the sign-off sheet from the coach. Be sure to indicate on the diagram which bits of which ports of the PIC32 I/O Board you used.

---

### Part 5 Driving an R/C Servo Motor

**Reading:** CKO Ch. 27 (especially section 27.3), RC_SERVO module documentation. RC Servo tutorial at: [http://www.societyofrobots.com/actuators_servos.shtml](http://www.societyofrobots.com/actuators_servos.shtml)

**Assignment:** You are going to drive an R/C servo with position feedback. By adjusting the duty cycle of a fairly slow PWM signal to the servo, you command a position for the head to go to and hold. Just as you read an analog value of 0-3.3V and adjusted the speed of the DC motor, you will now read that same voltage and position the servo motor to correspond to the voltage. When reading 1.65V on the A/D, the servo should be in the neutral position, with 0V positioning it at one end of the range and 3.3V positioning it at the other.

**Set-Up:**

5.1) **Important note:** the R/C servos can only take +5V, so DO NOT connect them to a higher power supply, you will kill them. Power the servos off of +5V from the power supplies, again making sure NEVER TO EXCEED THIS—you will break the feedback circuitry around the motor. Eventually (like in your final project) you may need to use a linear regulator to get +5V from your batteries, but you will be wasting lots of power as heat—you will need a heatsink.

5.2) Connect the servo on the motor drive lab board to +5 on the RED wire, and GND on the BROWN wire. Connect a wire from your I/O board to the ORANGE signal wire (your I/O board and the servo need to have a common ground connection back to the power supply).

5.3) The servo is a position feedback device. That is, it will go to and hold a specific position that ranges from about +90 degrees to -90 degrees, with a 1.5ms pulse being CENTER. We have developed a full module for driving up to 10 R/C Servos.

5.4) Experiment and determine the minimum change of the pulse high time that results in motion of the servo head. The module enables 1uS resolution of the pulse width.

5.5) Experiment and determine the minimum and maximum duty cycles that the servo is able to respond to. How do these limits and the neutral duty cycle correspond to head position?

5.6) Use the potentiometer from Part 1, after first checking that it stays within the range 0-3.3V, to command a position of the servo. When sweeping from 0-3.3V the servo head should follow from -90 degrees to +90 degrees. If the servo introduces noise on your power and this is affecting your A/D readings, bypass your potentiometer with a capacitor to smooth it out.

5.7) Show the completed servo control to the tutors or TA for your final checkoff.

**In the report:** Include the wiring diagram of the circuit you used, a copy of the code that drives the servo motor (your main.c), a discussion of the behavior of the servo as relating to questions 5.4 and 5.5, and the sign-off sheet from the coach. Be sure to indicate on the diagram which bits of which ports of the PIC32 I/O Board you used.
Hints on working this assignment

When you go into the lab, follow a well thought-out and systematic approach to testing both the hardware and the software. It is a good idea to get into the habit of testing your software, as much as possible, separately from testing your hardware. This makes trapping failures much easier (and debugging is ALWAYS the longest task).

If you really did your preparation properly, you should come into the lab ready to build and test the hardware in Part 1. If you have spent more than one hour on any single task, after coming in prepared as described above, something is wrong! STOP and ask a TA or your neighbor to take a look at what you are doing. Often a new look will spot simple problems that you’ve missed.

And, needless to say, better preparation for this lab (reviewing the reading, examining the data sheets, planning your software) will greatly cut down on the amount of time in the lab. It is very inefficient to try to do the prep at the same time as doing the lab.

Lastly, you are working with high currents in this lab. This means that your “electronic hygiene” had better be good, or you are going to blow stuff up. Never, NEVER, plug things in and out of your board with the power supply turned on! You have banana plugs, use them (we don’t want to see alligator clips hooking things up). Check your polarity several times before hooking up power to things. Be very careful about wires dangling from power supplies or O’scope leads shorting things out. You have 10-16 amps behind you—it will blow traces off your board (very hard to repair). Always inspect your boards for solder bridges and the like (these have just been made, and most have not been extensively tested).

A Final note: frustration rarely makes your decision process better—if you are having difficulty, ask for help, or at least take the time to walk away, get a breath of fresh air, and then return to see if you can determine what exactly was the problem. Good notes and communication with your lab partner will help quite a bit here.
Lab #3

Time Summary

Be sure to turn this in with your lab report

This information is being gathered solely to produce statistical information to help improve the lab assignments.

<table>
<thead>
<tr>
<th>Pre-Lab</th>
<th>Preparing Outside of the lab</th>
<th>In the lab working this part</th>
</tr>
</thead>
<tbody>
<tr>
<td>Part 1</td>
<td>Preparing Outside of the lab</td>
<td>In the lab working this part</td>
</tr>
<tr>
<td>Part 2</td>
<td>Preparing Outside of the lab</td>
<td>In the lab working this part</td>
</tr>
<tr>
<td>Part 3</td>
<td>Preparing Outside of the lab</td>
<td>In the lab working this part</td>
</tr>
<tr>
<td>Part 4</td>
<td>Preparing Outside of the lab</td>
<td>In the lab working this part</td>
</tr>
<tr>
<td>Part 5</td>
<td>Preparing Outside of the lab</td>
<td>In the lab working this part</td>
</tr>
<tr>
<td>Report</td>
<td>Preparing the Lab Report</td>
<td></td>
</tr>
</tbody>
</table>

Lab #3

Partner Evaluation

Email the Professor (only) at mailto:elkaim@soe.ucsc.edu with a subject line of: “CMPE118 Lab 3 Partner Evaluation” and include:

(1) Partner’s name:

Rate them on a scale of 1 (bad) to 5 (excellent) on the following areas:

(2) Knowledge:
(3) Prep:
(4) Work:
(5) Neatness/Cleanup:

This will be used to help us match partners for the project, and is extremely important. Failing to send this in will result in a 20% penalty on your lab grade.