Laboratory Assignment Number 3 for ME 118

Due by 5:00 PM on February 17, 1995

Purpose:

This lab is intended to acquaint you with:

Controlling a dc motor.
Controlling a stepper motor.
Finding information in data sheets.
Pulse width modulation.
Phenomenon encountered with a DC motor.
Limitations of purely software techniques.

Minimum Parts Required:

1 each: DC motor and stepper motor (provided at each lab station), 1N5250 20 Volt Zener diode, 20K potentiometer, There will be a bag of the diodes in 577. See a TA 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 into SPDL to complete the lab.

- ❑ 0.1) Decide which port and bit(s) you will use to control the motors (all parts).
- ❑ 0.2) Decide what, if any, initialization is required for that port.
- ❑ 0.3) Decide what mode you intend to use for the A/D converter.
- ❑ 0.4) Determine which pins on which cables you need to hook up.

In the report:

Include a description of which of the port lines on the 68HC11 you used to implement the control as well as a justification for why you chose to use that (those) particular line(s). You should also include a description of the mode of A/D converter operation you chose, and why.

Part 1 Interfacing to a DC Motor and Potentiometer

Reading:

AD_LIB reference documentation, ME118 I/O Board, Digital Ouputs Documentation. The Ideal users manual

Assignment:

You are to design the software necessary for the 68HC11 NMI to drive the supplied DC motor. Motor drive should use Pulse Width Modulation with a minimum resolution of 1 part in 256 (i.e., 8 bits). This control is to be implemented completely in software without the use of the PWM libraries. The speed at which you run the motor should be determined by the setting of an external potentiometer which your software reads using the A/D converter in the 68HC11. 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 coach and get a sign-off sheet after demonstrating Part 1.

Set-Up:

- ❑ 1.1) For this, and the next part, you be using the power drivers from the I/O board to drive the DC motor. Since these are 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) Adjust the positive output to approximately 7.5V using the control dial on the power supply. Now connect the supply (at the proto-board) to one side of the DC motor using the connector from the motor assembly.
- ❑ 1.3) Now connect the clamp diode for drivers 1&2 to the motor supply, at the proto-board and the output for driver 1 to the remaining side of the DC motor. The motor should not be rotating.

❑ 1.4) Explain why the motor is rotating at this point.

❑ 1.5) Hook the two outer leads from the potentiometer to +5 & Gnd. These are most easily obtained using the 3 pin connector that you used in the last lab. Be **very** careful **NOT** to use the power going to the motor, that is 7.5V. After doing this, check to see that the voltage at the wiper(the remaining terminal on the potentiometer) swings between 0 and 5V.

❑ 1.6) Using one of the jumpers that goes from a wire to a push-on connector, make the connection from the wiper to the pin labeled E0 on the I/O board (it's part of JP2A1). We could have simply used one of the unity gain buffers to bring the signal to the board, much like you did in the last lab. Why would you not want to do this ? What are the limitations that this would impose ?

❑ 1.7) You are almost ready to fire up your code and test it out, but first you must adjust the Ideal configuration to suit the NMI board, rather than the Mini-Board. To do this you will need to change one of the parameters to the linker and redefine the build sequence. To change the linker option, select Configure from the main menu and Command Lines from the pull-down. In the resulting dialog box find the field labeled Linker. In that field, change the option '-gce2' to read '-gcb'. To adjust the build sequence, select Define build from the Build Menu. In the resulting dialog box, you must delete Debug from the build sequence, and replace it with Download.

❑ 1.8) It's finally time to try out your code. In Ideal, with the communications window active, turn on (or reset) the NMI board. You should get the Buffalo signon message. Press the return key, and you should see Buffalo's standard prompt '>'. At this point you are ready to download your code. Select Build from the build menu and, if you have no errors, your code should be compiled & downloaded (the download process looks different than it did for the Mini-Board). If this went smoothly, you should be back at a Buffalo prompt. To start you code, type '`c c000`' (those are zeros).

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 ME118 I/O board, the sign-off sheet, the answers to the questions in parts 1.4 & 1.6, as well as a listing of the software used to control the motor.

Part 2 Exploring DC Motors

Reading: Lecture Notes on DC motors.

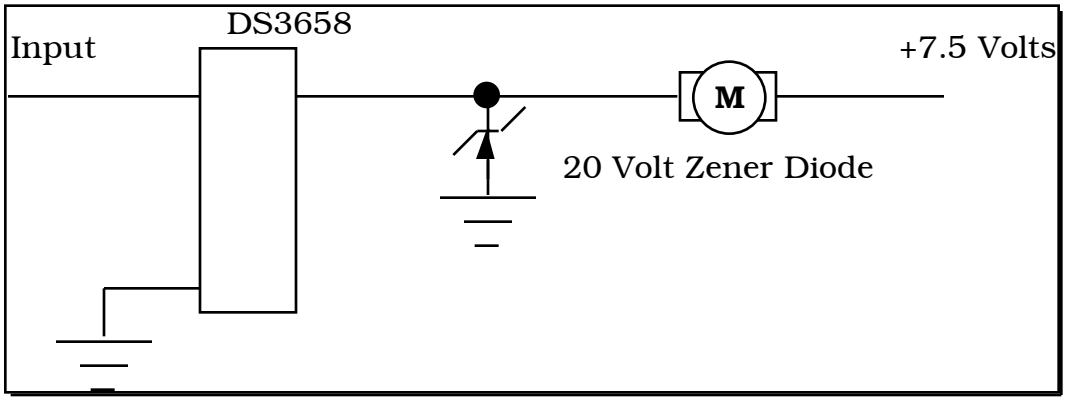
Assignment: Complete the following exercises:

Basic Waveforms

❑ 2.1) Set the '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 wave-forms. Include at least 2 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 kick-back, 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 and replace it with the 20 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 2 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 part in 256). 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 delay time to completely overlap the undriven 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) Can you find a way to increase the operating frequency? (Look at how to shorten your cycle time to the minimum.)

2.5) What happens to the upper frequency limit if we only have 7 bits of resolution? How about 10 bits? We are making a trade-off between resolution and frequency of operation.

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

2.6) 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.

2.7) 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 2.3-2.7. 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 and Potentiometer

Reading:

Stepper Motor Handbook.

Assignment:

You are to design the necessary software to run on the 68HC11 NMI to drive the supplied stepper motor. The speed at which you run the motor should be determined by the setting of an external potentiometer which your software reads using the A/D converter in the 68HC11. 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, get a coach to sign off on it's function. For this, and subsequent sections of the lab, you may use the supplied libraries. providing pulse generation and PWM output.

Set-Up:

3.1) For this part you should use the stepper motor drive chip whose outputs appear on JP43.

❑

❑ 3.2) You will need to add a power connector to supply power to the stepper motor. Use the connector labeled 'Motor Power' and connect it to JP58.

❑ 3.3) You will need to supply pulse and direction inputs to the chip. For this test, you may simply hook the Full/Half input to ground.

❑ 3.4) 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)..

In the report:

Include the wiring diagram of the circuit you used, a copy of the code that drives the stepper motor and the sign-off sheet from the coach. Be sure to indicate on the diagram which bits of which ports of the 68HC11 you used.

Part 4 Interfacing to a DC Motor and H-Bridge

Reading:

ME118 I/O Board Power Outputs 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 coach to sign off on it's 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. Use the connector labeled 'Motor Power' and connect it to JP54.

❑

❑ 4.2) You will need to supply enable and direction inputs to the chip.

In the report:

Include the wiring diagram of the circuit you used, a copy of the code that drives the stepper motor and the sign-off sheet from the coach. Be sure to indicate on the diagram which bits of which ports of the 68HC11 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.

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, 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.