CS/ECE 5780/6780
Spring 2007
Myers

Handout #9

LAB #9: Stepper Motors

Lab writeup is due to your TA at the beginning of your next scheduled lab. Don't put this off to the last minute! There is pre-lab work to complete before the start of the next lab. **NO LATE LAB REPORTS WILL BE ACCEPTED.**

1 Objectives

- Gain experience with stepper motor control.
- Demonstrate a practical application of the 6812 timer system.

2 Reading

- Read Chapter 8 on stepper motors.

3 Background

In this lab, you will be designing an interface to a stepper motor. The stepper motor kit is available in the lab for checkout, but you should not need to check it out until you are ready to demonstrate your program. To control the stepper motor, you will be generating a pulse-width modulated output using output compare 7 (OC7), and commands will be input from your keypad. Therefore, you likely will want to start with your Lab 7 code.

Your program should support six commands.

1. 1*nnnnn# - set period for step or half-steps
2. 2*nnnnn# - set number of steps
3. 3* - forward or clockwise mode
4. 4* - backward or counter-clockwise mode
5. 5* - full-step iterations
6. 6* - half-step iterations

We will be using OC7 because you can tie OC7 to other port T pins (PT7 through PT3) using the OC7M control register. You will control the stepper motor using PT7 through PT4. To control the stepper motor, you should output the following patterns in sequence to create half-steps:

- %1010
- %1000
- %1001
- %0001
- %0101
- %0100

- %0110
- %0010

To move forward in half-steps, you will cycle through these patterns in the order shown. To move backward, you will cycle through these patterns in the reverse order. To move in full-steps, you will output every other pattern. PT3 should simply alternate from 0 to 1 to behave like a "heartbeat" to show your device is working.

4 Prelab

1. Write C code for your main program which reads the keyboard command and updates any global variables as necessary.

2. Write C code for your output compare interrupt handler. Here is pseudocode for this handler:

```
TC7 += step_period
if (step_count == 0)
  OC7D = 0
else
  choose next step pattern index i (for half or full, fwd or backward)
  update OC7D with next pattern
  step_count--
```

5 Lab Tasks

1. Add LEDs with resistors to pins PT7 through PT3.

2. Test each of the six commands, and verify that your code produces the patterns expected on the LEDs.

3. Capture the patterns that you see on the oscilloscope and include these plots in your writeup.

4. After you demonstrate your stepper motor controller to your TA, check out a stepper motor kit and D25 connector cable. Plug the D25 connector into the stepper motor kit and connect pins 2-5 to PT7, 6, 5, 4 (respectively) of your 6812 circuit. You can leave the LEDs in place to verify operation. Connect one of the D25 ground pins (11, 12, or 25) to the ground rail of your circuit. Hook up 12V across the +12V and GND terminals of the stepper motor kit. Specify a long period (65536) and verify that the 12V (24 step) motor steps appropriately each time a new pattern is driven on PT4-7.

5. Demonstrate all six commands. Demonstrate clockwise and counterclockwise rotations of precisely 90 degrees. Experiment with the motor control in order to answer the writeup questions.

6. Remove the 12V power and connect pins 6-9 of the D25 connector to PT7,6,5,4 of your 6812 circuit. Supply 9V across the +9V and GND terminals of the stepper motor kit. Specify a long period (65536) and verify that the 9V (400 step) motor steps appropriately each time a new pattern is driven on PT4-7. (It may be difficult to see the individual steps.)

7. Demonstrate the six commands. Demonstrate clockwise and counterclockwise rotations of precisely 90 degrees. Experiment with the motor control in order to answer the writeup questions. With the motor turning at its fastest speed, try to stop the motor by hand.

8. Modify your circuit in order to drive the motor with a constant pattern %1000. Try to turn the motor by hand.

6 Writeup

1. Include a printout of your final program and your HP Benchlink capture file.
2. Describe any problems you encountered.
3. Answer the following questions.
   (a) What is the fastest update frequency each of the motors could support without dropping steps?
   (b) How does the motor behave when driven faster than it can handle?
   (c) Can the motors support faster halfstep update rates than full-step rates? If so, why would this be the case?
   (d) When driving the motors at their highest speeds, do you notice any effects of inertia when switching the direction of rotation?
   (e) What is the effect of reversing the order of the stepper motor control pins (i.e., swapping PT7 with PT4, and PT6 with PT5)? You can answer this by experiment, or by analyzing the stepper patterns.
   (f) Is it harder to stop the 9V stepper motor when it's turning, or to turn it when it's holding its position? Suggest a reason why this would be the case.