Lab 5d Stepper Motor Finite State Machine

This laboratory assignment accompanies the book, Embedded Microcomputer Systems: Real Time Interfacing, by Jonathan W. Valvano, published by Brooks-Cole, copyright © 2000.

Goals
- To interface a stepper motor,
- To implement background processing with periodic interrupts,
- To develop a dynamic linked command structure.

Review
- Valvano Section 1.6 about open collector logic,
- Valvano Section 1.7 about initializing and accessing I/O ports,
- Valvano Section 2.4 about abstraction, linked lists and FSM's,
- Valvano Chapter 8, about stepper motor interfacing,
- Valvano Section 13.2.3, about an open loop stepper motor control system.

Starter files
- **`MOORE OC`** and **`OC3`** projects

Background

Stepper motors are popular with digital control systems because they can be used in an open loop manner with predictable responses. Such applications include positioning heads in disk drives, adjusting fuel mixtures in automobiles, and controlling robot arms in robotics. In this lab, you will control a stepper motor using a finite state machine. The finite state machine must be implemented as a linked data structure. Two switches will allow the operator to control the motor. A background periodic interrupt (either **`OC`** or **`RTI`**) thread will perform inputs from the switches and outputs to the stepper motor coils. The worm gear on the motor, as shown in Figure 5.1, produces a linear motion as the stepper motor turns. The operation of switch 1 has two modes: fast and slow. If switch 1 is momentarily pressed (longer than 20ms but less than 500ms), then the system should move as quickly as possible all the way to one end of the linear motion. If switch 1 is pressed and held for longer than 500ms, then the system should move slowly (4 to 10 times slower than maximum) in that direction as long as the switch is pressed. After being pressed for more than 500ms, if switch 1 is released the system should stop. The operation of switch 2 is simpler than switch 1. If switch 2 is pressed, then the system should move slowly in the other direction as long as the switch is pressed. If both switches are simultaneously pressed, then the system should stop, and wait for both switches to be released. In other words, you don't want the system to move after you've issued an emergency stop and you just happen to release the two switches as slightly different times. If the system is performing one operation (e.g., moving all the way to the end), and another command is issued (e.g., move slow the other way), then the first operation is terminated and the second command is performed. The system somewhat behaves like a car's power window.

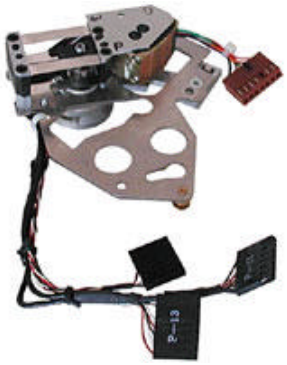

Figure 5.1. Stepper motor with worm gear.

Preparation (do this before your lab period)

Please use an external power supply to power the stepper at the voltage rating of the stepper. With an ohmmeter, measure the resistance of one coil. Apply the power across the coil and simultaneously measure using both a voltmeter and a current-meter the voltage and current required to activate one coil of your stepper motor. Do

this before your lab period, without any connections to the 6812 board. Design the hardware interface between the 6812, and prepare a circuit diagram labeling all resistors and diodes. An interface for a Singapore LB82246 stepper is shown in Figure 5.2. Include pin numbers and resistor/capacitor types and tolerances. Be sure the interface circuit you select can sink enough current to activate the coil. Make sure you have all the parts you need before lab starts.

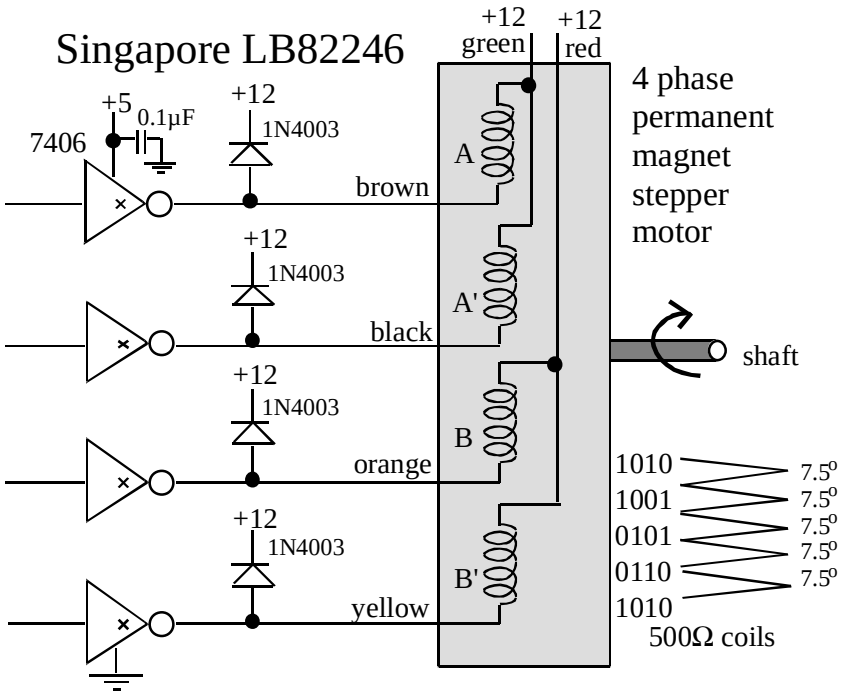

Figure 5.2. Singapore LB82246 Stepper Motor Specifications

A "syntax-error-free" hardcopy listing for the software is required as preparation. This will be checked off by the TA at the beginning of the lab period. You are required to do your editing before lab. The debugging will be done during lab. Document clearly the operation of the routines. The periodic interrupt-driven background thread will execute the finite state machine, while the foreground thread initializes the system then does nothing.

Procedure (do this during your lab period)

Make sure your TA checks your hardware diagram before connecting it to the 6812. We do not have extra boards, so if you fry your board, you may not be able to finish. First build the digital interface on a separate protoboard from the 6812, as shown in Figure 5.3. Use switches as inputs for the stepper interface. You should be able to generate the 5,6,10,9 sequence with your fingers. Using a scope, look at the voltages across the coils to verify the diodes are properly eliminating the back EMF. The very fast turn-off times of the digital transistors can easily produce 100 to 200 volts of back EMF, so please test the hardware before connecting it to the 6812.

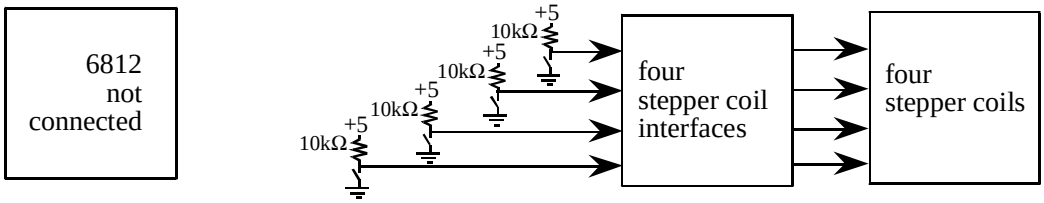

Figure 5.3. Hardware test procedure.

The switches will eventually be replaced by 6812 outputs, as shown in Figure 5.4. Once you are sure the hardware is operating properly, run a simple constant velocity software function to verify the hardware/software interface.

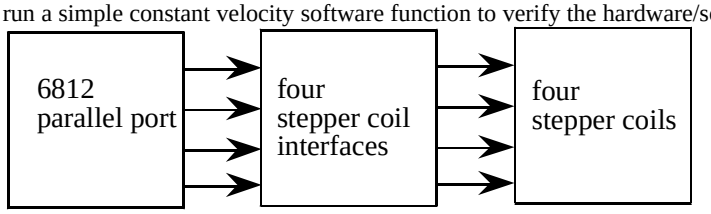

Figure 5.4. Hardware block diagram.

Deliverables (exact components of the lab report)

A) Objectives (1/2 page maximum)

B) Hardware Design

stepper motor interface, showing all external components

switch interfaces, showing all external components

C) Software Design (no software printout in the report)

Draw figures illustrating the major data structures used, e.g., the finite state machine

A call-graph illustrating the modularity of the software components

D) Measurement Data

Prep) Give the voltage, current, and resistance measurements

Determine the range of speed possible for the motor

E) Analysis and Discussion (1 page maximum)

Checkout (show this to the TA)

You should be able to demonstrate correct operation of the stepper motor system. Be prepared to describe how your stepper interface works. Explain how your system handles the switch bounce. Demonstrate debugging features that allow you to visualize the software behavior.

Your software files will be copied onto the TA's zip drive during checkout.

Hints

1. Stepper motors other than the *Singapore LB82246* will have different coil resistances and different coil voltages. Be sure the interface driver (e.g., the 7406 in Figure 5.2) has an I_{OL} large enough to deliver the needed coil current. Please do not use the Adapt812 +5V regulated supply to run external components that require more than 100 mA of current. Although many steppers will operate at voltages less than the rated voltage, the torque is much better when using the proper voltage. Remember to actually measure the coil current rather than dividing voltage by resistance.
2. Be sure to put an appropriate delay between each step to prevent the motor from burning up.
3. To prevent the power supply from overheating, make sure the supply can deliver the required current.
4. Debug the lab in small steps.
5. The **TExaS** simulator does include a stepper motor, so you can develop and test the software in parallel with the hardware development. In particular, the **OC** project can be run on the **TExaS** simulator.