

EE461 Microcomputer-Based Digital Design

Lab3

Stepper Motor Control

1 Specifications

In this lab you will implement a PIC16F874-based design that controls the rotational speed of a stepper motor. One switch will control the direction of rotation. Three more switches will be used to specify one of seven (not eight!) possible rotational speeds.

1. Use Timer 2 to generate an interrupt every 1 ms.
2. The Timer 2 interrupt service routine (ISR) is to count each Timer 2 interrupt as it occurs. After 5000 of them have elapsed, that is, after 5 s, the ISR is to read the three switches that specify motor speed and the single switch that specifies a new direction. It will then update internal general purpose registers so that the new commands will take effect. If it discovers an invalid switch combination, it must ignore it. The relationship between input switch settings and rotational speeds is shown in Table 1 on the following page.
3. The main program is to issue half-step control signals to the stepper motor at suitable intervals so that the commanded rotation speed is achieved. This requires additional help from the Timer 2 ISR, which must also count the required number of 1 ms interrupts between successive half-steps. When the right amount of time has elapsed, the Timer 2 ISR can let the main program know by setting a bit in a convenient general purpose register. The main program continually looks for this bit to be set and, when it sees it, issues a new control pattern and resets the bit.
4. The main program should also display the current motor speed range in binary using the LEDs on the proto-board. In effect, it reflects the positions of the switches, but not until after the Timer 2 ISR has read them. Thus, there could be up to 5 s between the time the switches are moved and the time the LEDs reflect the new positions.

2 Subdividing the Work

It is not feasible to try to implement a complicated system all at once and then test it. There are so many places where errors could creep in that you would find it difficult or even impossible to isolate them. It is much more effective to divide the project up into distinct subproblems and test each piece as you complete it. Here are some suggestions, but feel free to modify these suggested pieces as you see fit.

1. Program Timer 2 and make sure it generates interrupts at the right interval. Making the Timer 2 ISR issue a brief pulse every time it gets control will let you measure these intervals on the oscilloscope. The results of these measurements should be reported.
2. Use a table look-up scheme to obtain the correct number of 1 ms intervals needed between successive half-steps for any specified voltage range. Test it by giving it a switch combination value and verifying that the subroutine returns the correct value. Make sure that invalid switch combinations are ignored. Report on your tests.
3. Verify that the main program can respond at the appropriate intervals by having it issue a single pulse whenever the Timer 2 ISR informs it that another half-step is required. Use the oscilloscope to measure these intervals and report the measurements.
4. Modify the main program so that it issues stepper motor control signals in sequence. Make sure the main program responds correctly to the direction-switch inputs. Use a second lookup table to get the next half-step control pattern. You should be able to go either forward or backward through the table as specified by the direction switch. Use the oscilloscope to verify that the correct signals are issued at the appropriate times and report the results.
5. Build the stepper motor circuit. You will need transistors to provide enough current to make the stepper motor work. Test the circuit by

Required Speed Switch Setting	
10rpm	0
16rpm	1
25rpm	2
39rpm	3
61rpm	4
96rpm	5
150rpm	6

Table 1: The relationship between input switch positions and required rotational speeds for the stepper motor.

operating the transistors with digital switches. Make sure that the motor responds correctly to a sequence of half-step commands. Once the motor does respond correctly, connect it to the microprocessor.

6. At this point all subsystems will have been tested and you can test the complete system.

3 Other Items for Your Report

1. Report the rotational speeds actually achieved. For each speed range, report the relative error between the rotational speed desired and that achieved.
2. Include a properly drawn schematic diagram with your report. Naturally it needs to show all circuitry connected to the microprocessor.
3. Include the *code* for your program and include commentary.
4. Include oscilloscope displays that show the measurements you have made using the oscilloscope.
5. Be sure all enclosed diagrams, listings, and figures are described in the narrative of your report. Don't expect the reader to understand them without this assistance.