EECE 344: Digital Systems Design
Lab 6
Stepper Motor Finite State Machine

1. Pre-Lab

- Download, unzip, open, compile and run **StepperMotor_Lab6** into your computer.
- Read section 8.7 in your book.
- Read the whole lab procedure before you start building your circuit. Please follow the steps as prescribed in this lab manual.

2. Objective

The objective of this lab is to:

- Design, build and test a stepper motor controller
- Learn how to design stepper motor controller using a finite state machine
- Learn how to perform state machine input/output in the background.

3. Overview

You will be building a stepper motor controller system using the Tiva LaunchPad development board on a breadboard. The system may use on-board switches, off-board switches or a combination of both. You will be designing the software/hardware interface to control the motor. The status of the motor will be displayed on the LCD. There will be at least three hardware/software modules: switch input, motor output, and the finite state machine. Start by building and testing each module separately, then build and test the final system. You can include and use the Nokia5110 files (Nokia5110.h and Nokia5110.c) to output to the LCD.

4. Background

Stepper motors are popular with digital control systems because of their inherent digital interface. A microcontroller can control both the speed and position of a stepper motor in an open-loop fashion. Stepper motors are used in many applications such as: fuel mixtures in automobiles and controlling joints in robotics. In this lab you will be controlling a stepper motor using finite state machine. The state machine must be implemented as a linked data structure. Three switches will allow the operator to control the motor. An edge-triggered interrupt will be enabled on the switches. SW1 will be used to turn on and off the motor, SW2 will turn the motor clockwise CW, and SW3 will turn the motor counter clockwise CCW. There will be two hardware/software modules: the Control module and the Driver module. The finite state machine will be defined in main.

To spin a stepper motor at a constant speed the software must output the sequence 5-6-10-9 separated by a fixed time between outputs. This is an example of blind-cycle interfacing because there is no feedback from the motor to the software giving the actual speed or position of the motor. To make the motor spin in one direction, the microcontroller must output the sequence 5,6,10, 9 over and over. If the microcontroller outputs the sequence backwards then the motor will spin in the other direction, e.g. outputting 9, 10, 6, 5 will make the motor spin on the other direction. To ensure proper operation this sequence must be followed. For example, assume the microcontroller outputs …,9,5,6,10 and 9 then reverse direction, since the output was already in 9, the next output should be 10, 6, 5, 9, 10, 6,…. In other words if the output is 9 the two other possible next outputs should be 5 if spinning in one direction or 10 if spinning in the other direction.

To configure the motor speed we need to consider the stepper motor specifications. If a motor has n steps per revolution, let Δt be the time between outputs in seconds, the motor speed will be:

RPM = 60/(n*Δt),   where RPM is rounds per minute

The time between states Δt determines the rotational speed. Allowing more time between rotations will slow down the motor.

You will need an H-bridge driver to control the motor and provide the required current for the coil. This lab uses IC driver L293 to drive the motor as shown in Figure 1. A number of diodes is required to protect the components from emf. All Diodes are IN914.

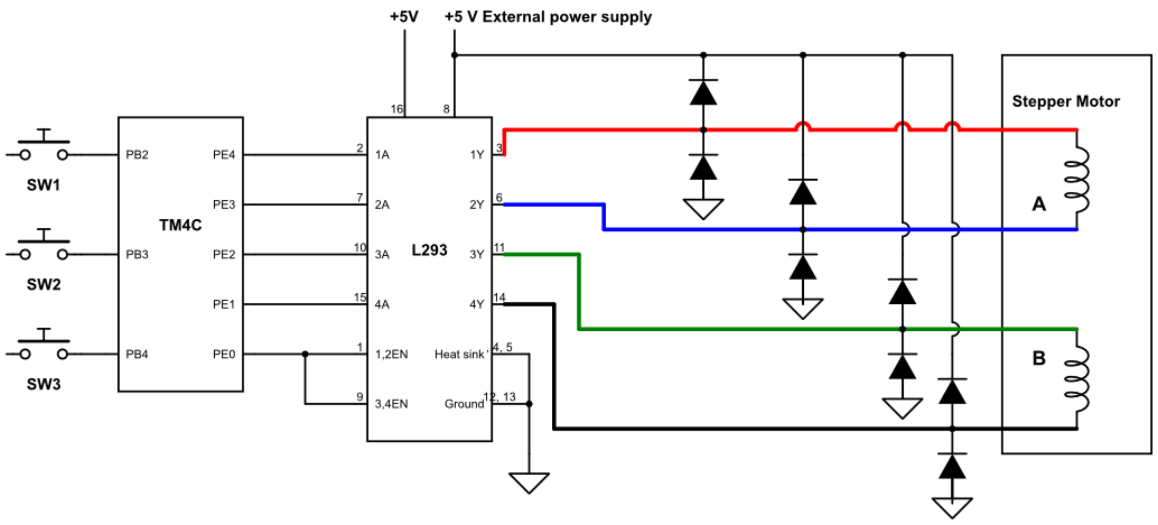

Figure 1

5. Function Description

- If all switches are released, the motor should stop.
- If switch 1 SW1 is pressed the motor is enabled waiting for input from SW2 or SW3 to start rotating but should not rotate.
- If SW1 and SW2 are both pressed the motor will spin in one direction, e.g. CW. The motor speed should be 30 RPM.
- If SW1 and SW3 are both pressed the motor will spin in the other direction, e.g. CCW. The motor speed should be 30 RPM.

- If SW1, SW2 and SW3 are all pressed, the motor will spin half a rotation in slower speed then stop. Each team should decide which direction they want the motor to spin in when all switches are pressed. The motor speed should be 5 RPM.
- Any other combination of switches input should not result in any motion.

6. Procedure

Part 1: Control Module

1. Create two functions Control_Init and Contorl_In and place them in control.c file
2. Create control.h file and place all function definitions and descriptions in it.
3. Control_Init function will initialize the input pins PB4-PB2 and enable edge triggered interrupts on both edges for SW1 only.
4. SW1 is used as the ON/OFF switch for the motor.
5. Control_In function will read and return the switches input when invoked.

Part 2: Driver Module

1. Create one function Driver_Init and place it in a Driver.c file
2. The function will initialize the output PINs PE4-PE0
3. PE0 will be used to enable/disable the motor driver
4. PE4-PE1 will be used to output the motion sequence.
5. Place function definitions in Driver.h file, make sure to document your code.

Part 3: SysTick Delay

1. Create SysTick.c file, feel free to use the SysTick files from previous assignments. Initialize SysTick without interrupts in this file.
2. Create a function SysTick_Wait1ms function that takes an unsigned input and invokes a delay of input*1ms.
3. Include the function definitions in the SysTick.h file
4. Use SysTick_Wait1ms to implement the delay in each state

Part 4: Main Function

1. Develop a finite state machine graph and table to control the motor before you start coding
2. There will be a 1-1 mapping from the state table to the C data structure.
3. There is a delay for each state, which saved in the FSM structure.
4. The motor output is always one of these values 5, 6, 10 or 9.
5. The motor moves such that the output never skips from 5 to 10, 10 to 5, 6 to 9, or 9 to 6.
6. There are at least 13 states to implement the motor control.
7. Place your ISR inside Stepper.c file

8. The interrupt ISR should read the input and writes to a mailbox or a flag indicating that SW1 was pressed.
9. The main program will initialize input, outputs and LCD, then loop around reading the switches input and waiting for the interrupt to set the flag, and then move to the required state.
10. The LCD should output “Stopped, Turn Left, Turn Right, or Half Round” reflecting the status of the motor.
11. Create your finite state machine structure inside the Stepper.c file outside main as a global constant structure.
12. It is very important to minimize the time to execute the ISR, measure the time to execute the ISR using heartbeat debugging.

Part 5: Safety Measures

- While developing software that doesn’t involve the motor:
    1. Disconnect the USB cable
    2. Disconnect the +5V power to the board/motor
    3. Reconnect the USB cable
    4. Edit/debug software.
- When developing software that does involve the motor
    1. Disconnect the USB cable
    2. Edit and compile software
    3. Reconnect the USB cable
    4. Download/debug software.
- Connecting and disconnecting wires on the board while power is applied may damage the board.
- Start by building the digital interface separate from the microcontroller, as shown in figure 2. Use switches as inputs for the stepper interface.
- Generate the sequence 5,6,10,9 through the switches.
- Using a scope, look at the voltages across the coils to verify the diodes are properly eliminating the back EMF.
- The fast turn-off times of the digital transistors inside the L293 driver can easily produce 100 to 200 V of back EMF, so please test the hardware before connecting it to the microcontroller.
- The switches in figure 2 will be eventually replaced by the microcontroller outputs. Once you are sure the hardware is operating properly, run a simple software function that rotates the motor at a slow constant velocity to verify hardware/software interface. Make sure to debug the Control and Driver modules separately.

- When you are ready to connect your motor, remove the USB cable and carefully power your stepper motor system using a lab power supply directly to the +5V pin marked as external power supply in the figure.

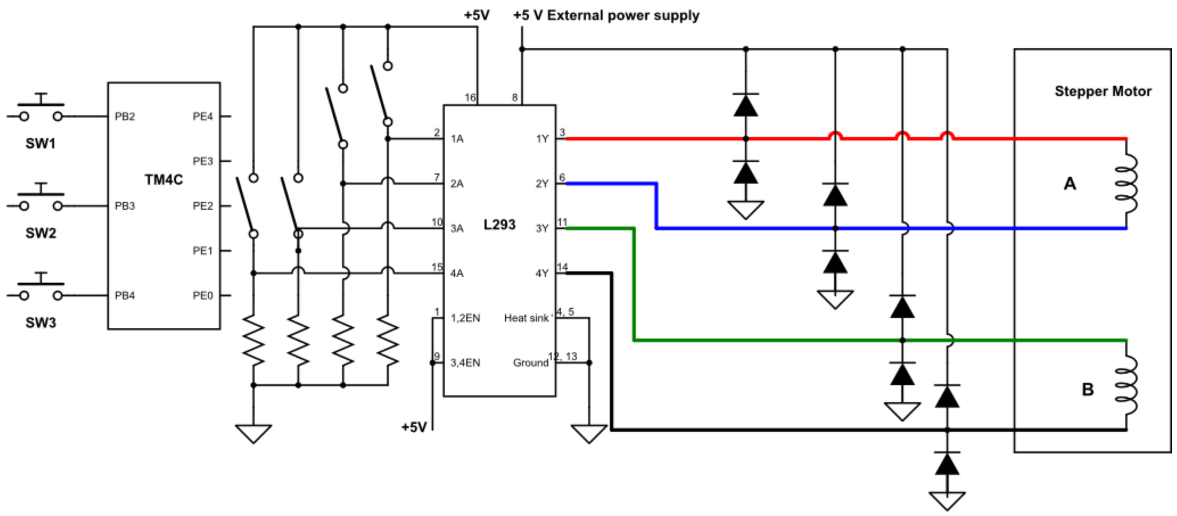

Figure 2

7. Demonstration

When demonstrating the program, you are expected to explain each line of code if asked. Each student in a team will be asked a different question to demonstrate his or her understanding of the project at hand.

Deliverables

1. Software design flowchart
2. All software design files
3. Specify the maximum time to execute one instance of the ISR
4. Analysis, discussion and suggestions for extra credit.