California State University, Channel Islands (CSUCI)
Department of Computer Science

COMP-462: Embedded Systems
Lab Report
Fall 2019

Lab Number: Lab 5

Lab Topic:

Stepper Motor Control Using
Linked Data Structures and Finite State Machines

Student Name: Marc Julian Jamerlan
Student ID: 001956436
Student Major: B.S. Computer Science

I. Objectives

The objective of this lab is to learn how to use finite state machines, linked data structures, and delay timers using SysTick in order to control a stepper motor's motion. The goal of this lab is to be able to create a system that acts much like a windshield wiper in most cars, wherein the windshield wiper and washer can be controlled with button inputs.

II. Introduction

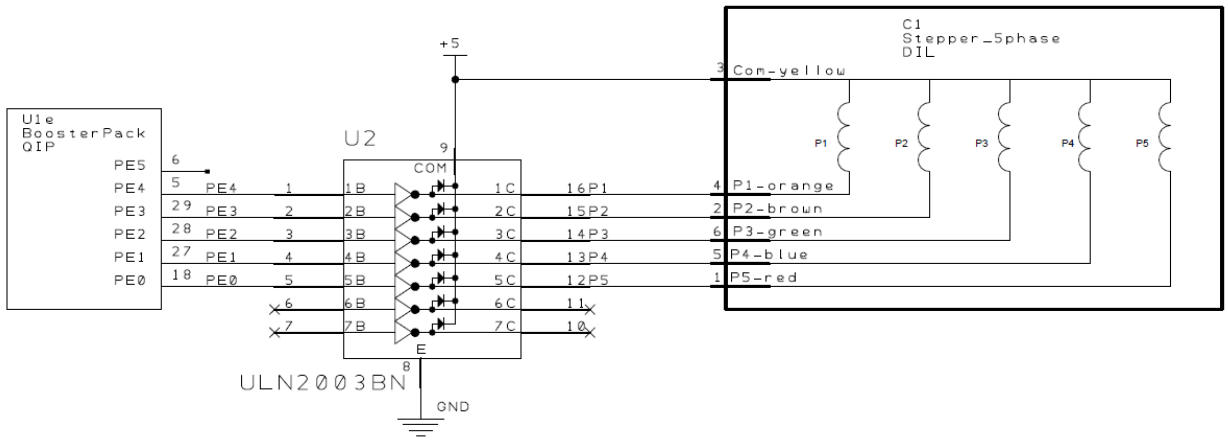


Figure 2.1. Stepper motor interface

For this lab, the task was to implement a simple stepper motor system that acts much like a windshield wiper:

- Two button inputs will control the system: One button for the windshield wiper, and another button for the windshield washer. Both inputs will be connected to PA4 and PA5 on the microcontroller, respectively.
- Ports PE0-4 act as outputs to the stepper motor, and depending on the data sent to Port E, the motor will move by a certain amount of degrees in one direction.
- Port PE5 acts as the output for the washer. The washer is indicated by an LED that turns on when the washer is activated and off otherwise.
- The system flow is as such:
    1. When no buttons are pressed, the motor is stopped. The washer LED is turned off.
    2. When the wiper button is pressed, the motor will move 40 degrees in one direction and then 40 degrees back, constituting one cycle. When held down, the wiper will move back and forth continuously.
    3. When the washer button is pressed, the washer LED will light up and then will turn off after the end of one cycle. Holding down the washer button will keep the LED lit.

The code to be modified includes `Systick.c` and `StepperMotorController.c`, a file for implementing the system delay timers and the main body of the stepper motor state machine, respectively.

III. Procedure

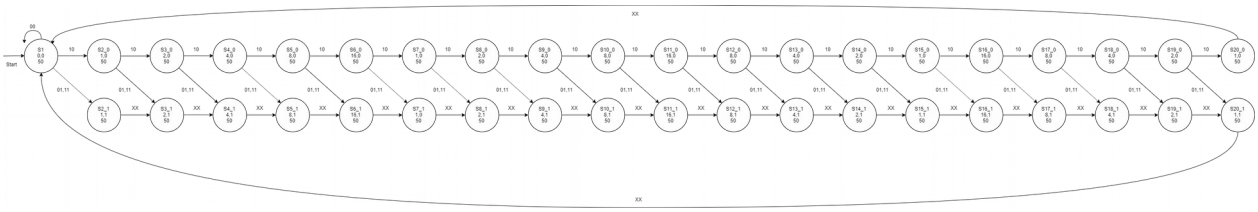


Figure 3.1. Moore state machine of the stepper motor controller

1. Designed a finite state machine that controls the stepper motor as described in the system flow.
    i. Four inputs are considered in this state machine, but for simplicity's sake having both buttons pressed will be the same as having the washer button pressed.
    ii. All states have a wait time of 50 milliseconds, and all wiper states will be able to transition to washer states at any time should the washer button be pressed during the operation of the wiper.
2. Implemented the linked data structure and loop that will control the stepper motor.
    i. All states are defined as pointers to an array that holds the necessary data for the state machine.
    ii. The data contains outputs for the wiper and the washer to send to Port E, wait times, and an array that holds the next states, depending on input received.
3. Tested the logic of the system in the simulation and logic analyzer in Keil.
4. Built the circuit for the stepper motor, LED, and switch interfaces.
    i. Both the switch and the LED interfaces are connected to the 3.3V pin on the microcontroller. The motor interface is powered by a 5V power supply.
    ii. An extra LED connected to PF1 will serve as a status LED for checking if the stepper motor interface is in operation or not.

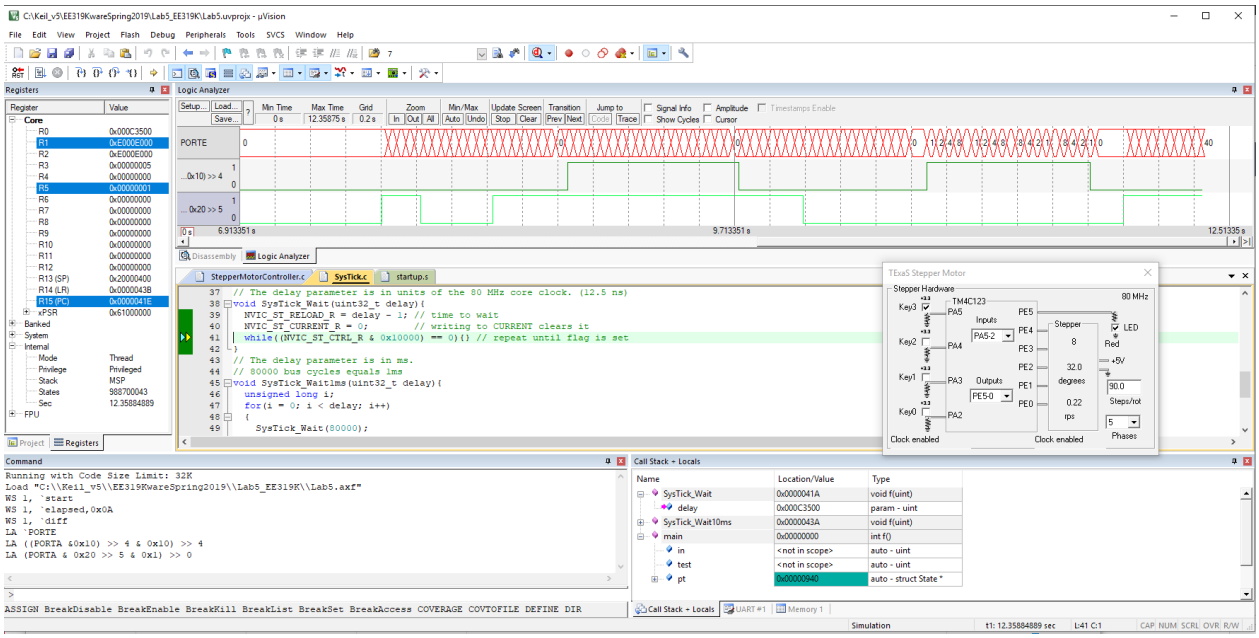
IV. Problems

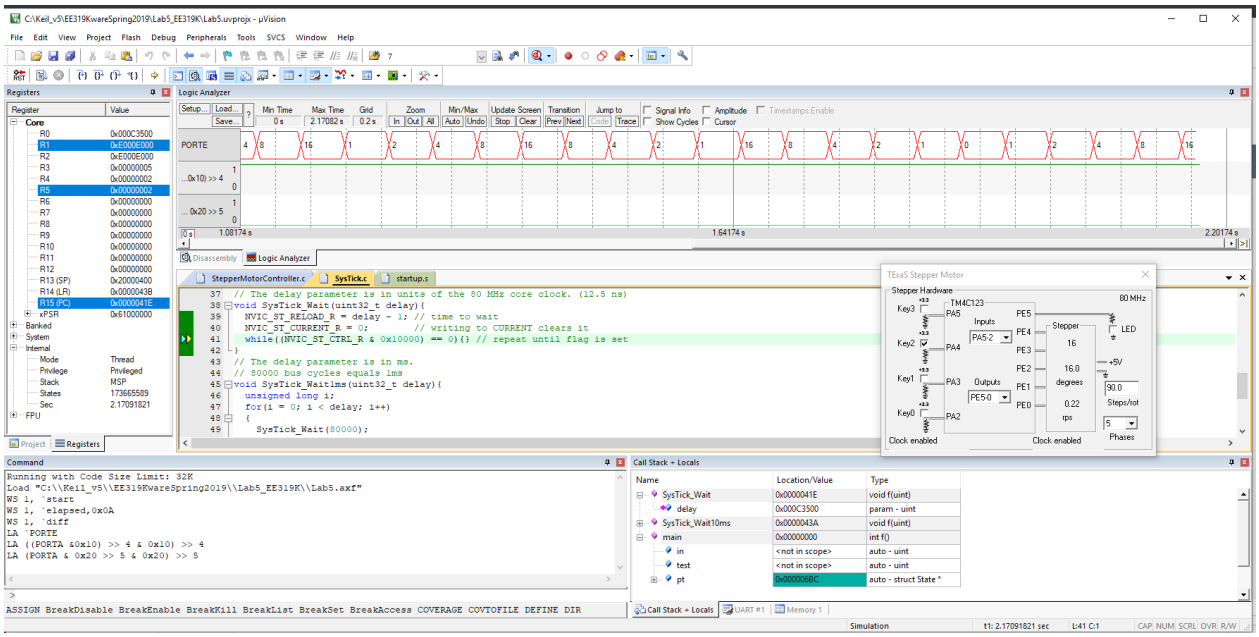
A few problems were encountered during the implementation phase of the state machine, which included 'off-by-one' errors in bit shifting and problems that occurred in trying to define the data structures within the `main()` function, but these were easily remedied. However, a larger problem was discovered when the state machine and the ports used were swapped. In this case, while the wiper port used PA4, my state machine assumed that the wiper input was PA5 instead. The solution to this was having to implement a `swapBits()` function that swapped the PA4 and PA5 bits around so the FSM could work properly.

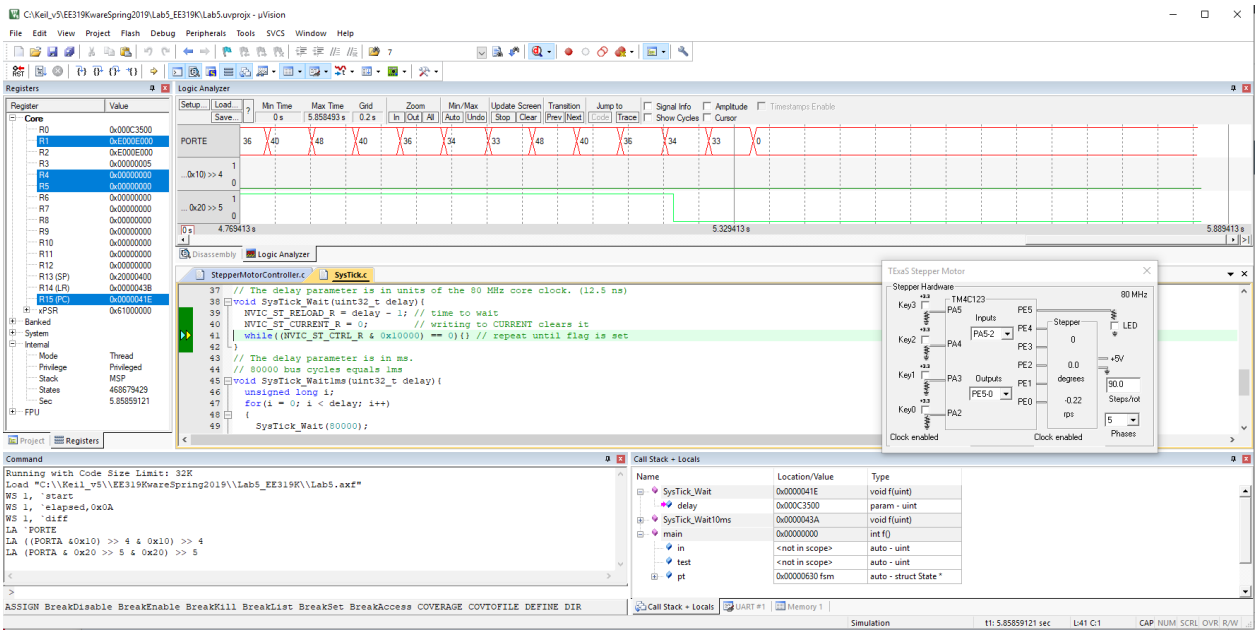
Another problem that arose during the building of the actual circuit was simply that the switches were placed improperly. The tabs used to connect the switch were found to be on the same side and always connected, and thus the system would always be powered on and the motor always moving. This was remedied simply by using the right tabs on the switch.

V. Results

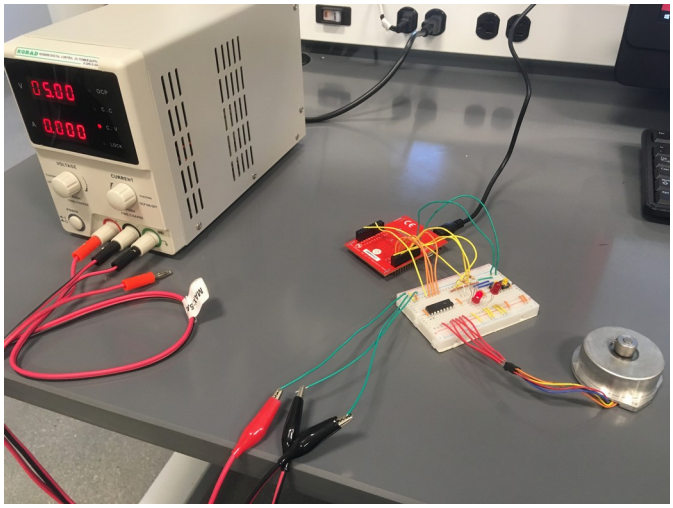
Screenshots of the Logic Analyzer, showing the change in states within Port E and the buttons being pressed:

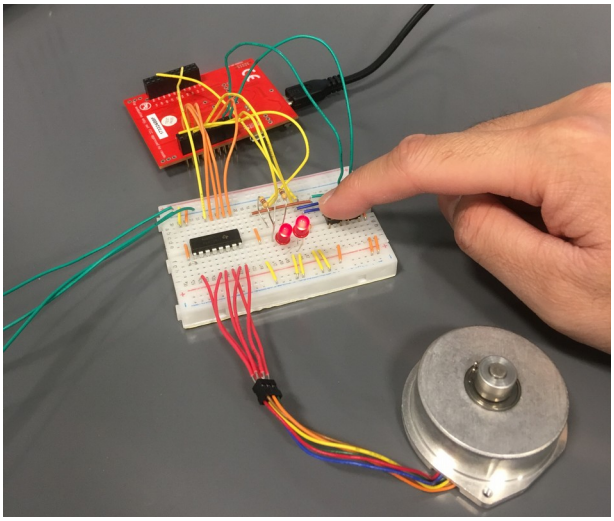






Screenshots of the stepper motor circuit:





Video of the stepper motor circuit in action: