Nathan Isaman ME 477: Lab 4 14 Feb 17

1. Description

The objective of this lab is to implement a finite state machine to produce a pulse-width modulated (PWM) signal to drive a DC motor. Below is a high level hierarchy for the main function and subsequent function calls; all while and if statements are omitted for brevity, excluding the while loop in *main()*.

Hierarchy:

```
main()
|_MyRio_Open()                                  // Opens a session with the MyRio
|_double_in()                                   // Requests M and N values from user
|_initializeSM()                                // Initializes state machine (SM) and encoder count
|       |_EncoderC_initialize()                 // Function used to initialize the DC motor encoder
|       |_Dio_WriteBit()                        // Ch0 "Run" bit set to "off" (True)
|_while (exitflag==0)                           // SM loop with exit condition from STOP state
|       |_state_table[curr_state]()             // Calls the current state function via a state table
|       |       |_low()                         // SM LOW state
|       |       |       |_Dio_Writebit()        // Ch0 "Run" bit set to "off" (True)
|       |       |_high()                        // SM HIGH state
|       |       |       |_Dio_ReadBit()         // Ch6 and Ch7 read for STOP and SPEED states
|       |       |       |_Dio_WriteBit()        // Ch0 "Run" bit set to "on" (False)
|       |       |_speed()                       // SM SPEED state, outputs RPM on LCD
|       |       |       |_vel()                 // sub function of SPEED, obtains encoder count
|       |       |       |       |_Encoder_Counter() // function used to obtain encoder count
|       |       |_stop()                        // SM STOP function
|       |       |       |_Dio_WriteBit()        // Ch0 "Run" bit set to "off" (True)
|       |       |       |_printf_lcd()          // "Motor Stopping…" message sent to LCD
|       |       |       |_openmatfile()         // Beginning of data logging sequence
|       |       |       |_printf()              // Error message printed on LCD if MATLAB failed
|       |       |       |_matfile_addmatrix()   // Multiple calls to add matrices to Lab.mat file
|       |       |       |_matfile_close()       // MATLAB file closed
|       |_wait()                                // Kills time between states
|_MyRio_Close()                                 // Closes the session with the MyRio
```

Note 1: Some functions are called multiple times per hierarchy tier, but only one call is shown above.

2. Testing

Much of the testing procedure was outlined in the laboratory experiment's prompt:

A. **Use oscilloscope to view waveform produced by program N=5, M=3**

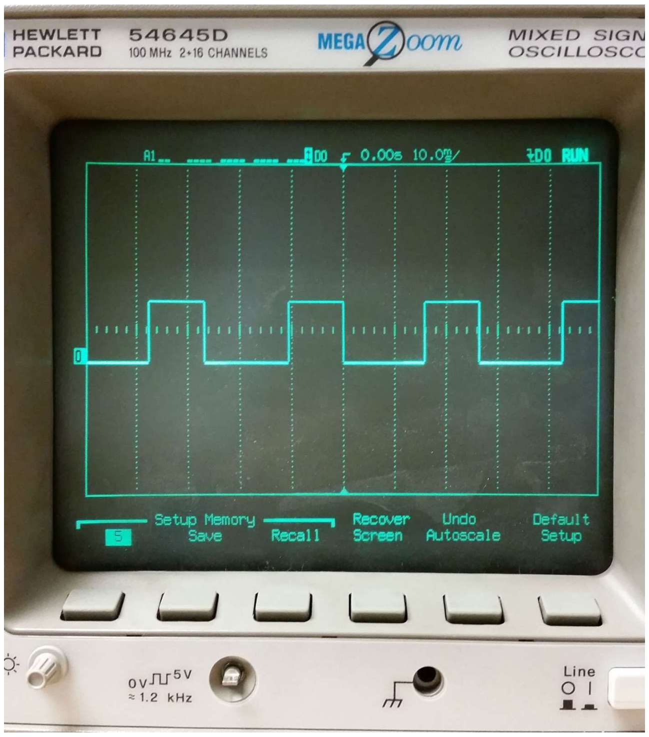


Fig 1: Standard Oscilloscope

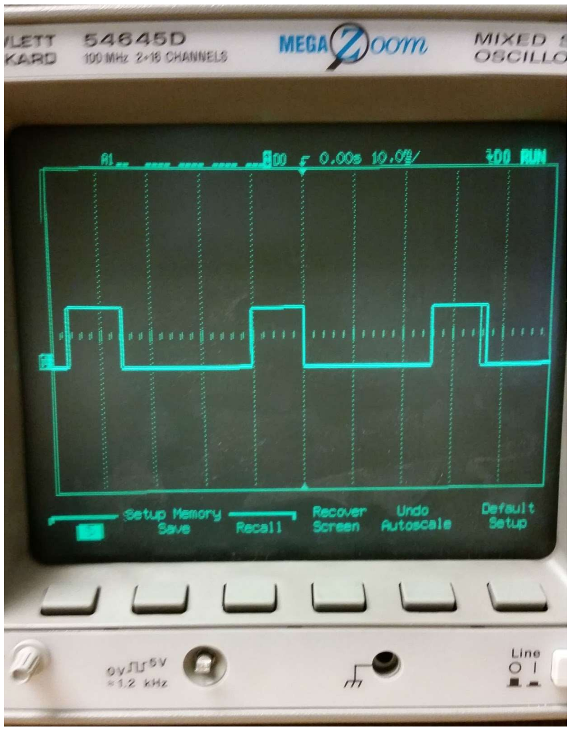


Fig 2: Speed button pressed Oscilloscope

B. **Compare program's printed output to tachometer**

The tachometer output and the LCD's printed output were quite different. The speed was tested about 12 seconds after the motor's start and the printed speed output read around 1100 RPM, but the tachometer would fluctuate between 600 and 1800 RPM at that time. The reading stayed mostly above the printed RPM output. The 12 second delay was an attempt to land inside the steady-state region of the motor's response so that the rise in motor speed would not contribute as much error.

C. **Use oscilloscope to view start/stop waveform and measure the actual length of a BTI;**
   a. **i. Is it what you expected, if not why not.**

The actual length of a BTI is 26ms based on Fig 1 above. The expected BTI is equal to cycle_time*N (5ms*5), which would be 25ms. The difference in the actual and expected BTI is the result of more operations occurring than those in the *wait()* function. These additional functions consume processor time and therefore increase the BTI.

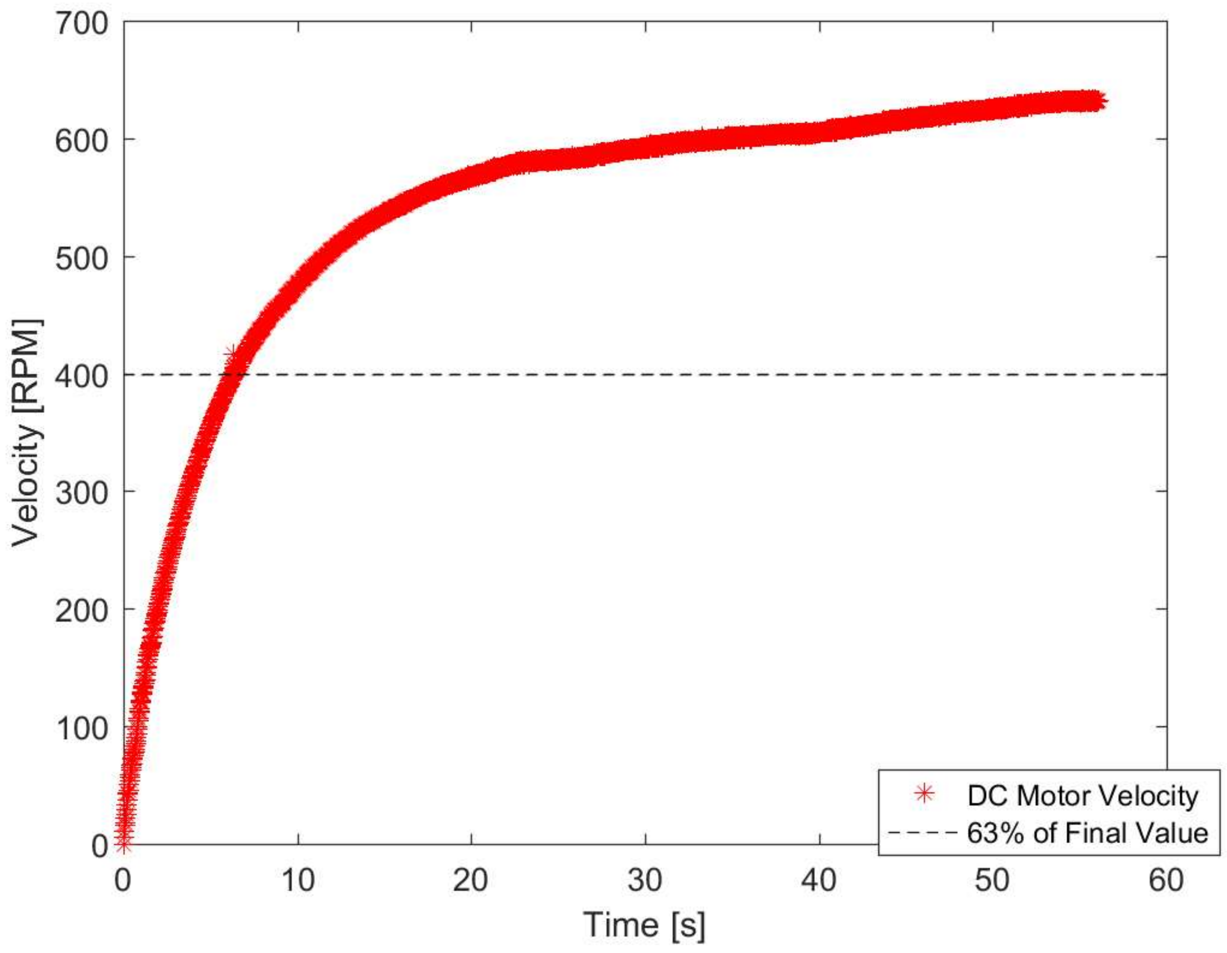
D. **Repeat previous step while printing the speed (hold switch)**
   a. **What does oscilloscope show has happened to the length of the BTI**

The BTI was increased by holding down the speed button (Fig 2). The BTI was increased by 10ms to a total of 36ms. This increase in BTI is, like above, the result of additional operations being conducted by the processor; in this instance this increase in BTI is due to *printf_lcd()* being called every cycle.

E. **Describe how you made this measurement and discuss any limitations in accuracy**

This measurement was done using the oscilloscope reading of the PWM waveform. The initial BTI length was taken from Fig 1 and the increase in BTI due to holding down the Speed button was obtained by finding the increase in length in the BTI between Fig 1 and Fig 2. The accuracy of this measurement is limited by the precision of the oscilloscope and the consistency of the processor in executing the program (i.e. no other processes running).

F. Record a Step Response



Note: The test was stopped before the vector was filled due to hardware complications. The program/hardware seemed to time out for runs longer than 70 seconds so the above test was stopped prior to reaching 60 seconds for safety margin. This will introduce error into the steady-state velocity and time constant calculations.

a. **Estimate time constant**
   The time constant obtained from the above graph was T=6.325 seconds.
b. **What is the steady-state velocity in RPM?**
   The steady-state velocity from the data set was V=633 RPM
   Note: the test was re-run as the RPM value seemed low, but the tachometer also read RPM values in this range. The motor remaining in the lab seems to be slower than the other ones tested by my peers.

Other questions asked throughout the prompt:

A. **Determine the exact number of clock cycles for the (wait()) code to execute, accounting for all instructions. From that, calculate the delay interval in milliseconds.**
   The delay interval calculated, given the command latencies, was 0.00500151424 seconds.
B. **Determine a reasonable value for N, the number of delay intervals in a BTI**

A reasonable value for N seemed to be 5 as this would allow the motor to reach approximately 1500 RPM +/- 300 RPM.

a. **What inaccuracies or programming difficulties are there in using a delay routine for control and time measurement**

One of the issues with programming a delay routine would be the variability in execution time between platforms. The MyRio has a 667MHz clock, but a computer or device with a different clock frequency would have a different delay time.

3. Results

All of the above tests and discussion points have been addressed above. Most results were expected, or can be rationalized given command latencies and processor usage.