Lab 23 Microcomputer-Based Motor Controller

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

Goals
- Use the 6812 Output Compare for generating variable pulse widths,
- Use the 6812 input capture for measuring period,
- Interface a DC motor and tachometer,
- Write real time digital control software.

Review
- Valvano Section 6.1.4 concerning period measurement,
- Valvano Section 6.2.5 concerning pulse width modulation,
- Valvano Chapter 13,
- Chapter on the input capture/output compare in the Motorola MC68HC812A4 Manual,
- Chapter on Fuzzy Logic in the Motorola CPU12 Reference Manual,
- Checkout the Fuzzy logic CD and see a professional development platform.

Starter files
- PWMOD.C, INTERP12.C, SCI12H, SCI12A.C

Web Research go to the Motorola site and look up and read
http://www.mcu.motsps.com/lit/
- AN1215 PID routines for the MC68HC11
- AN1120 Fuzzy Logic Character Recognition

Background

The objective of this problem is to design a microcomputer-based motor controller. The desired rotation speed **x*** will be selected interactively by the operator typing on the PC. The interrupt driven serial drivers SCI12A.C will be used. You will power the motor with a fixed frequency (e.g., 100 Hz) variable duty-cycle waveform. The speed will be defined as the frequency of the tachometer output in Hz. With a 12-volt motor power, the range of **x*** is 0 plus 100 to 800 Hz with a resolution of 1 Hz. If you power the motor using a 5-volt supply then the maximum speed will be less than 800 Hz. You will estimate the motor speed (**x'**) by measuring the period of a squarewave coming from the tachometer. The 6812 will measure the period using input capture and convert to frequency. Your first control software will implement an incremental control algorithm. Your second system will implement a proportional/integrator (PI), proportional/integrator/differential (PID), or Fuzzy Logic feedback control system. The goal of the control software is to maintain the motor speed as close to **x*** as possible. Your system will use a fixed frequency variable duty-cycle squarewave to control the electrical power applied to the permanent-magnet DC motor.

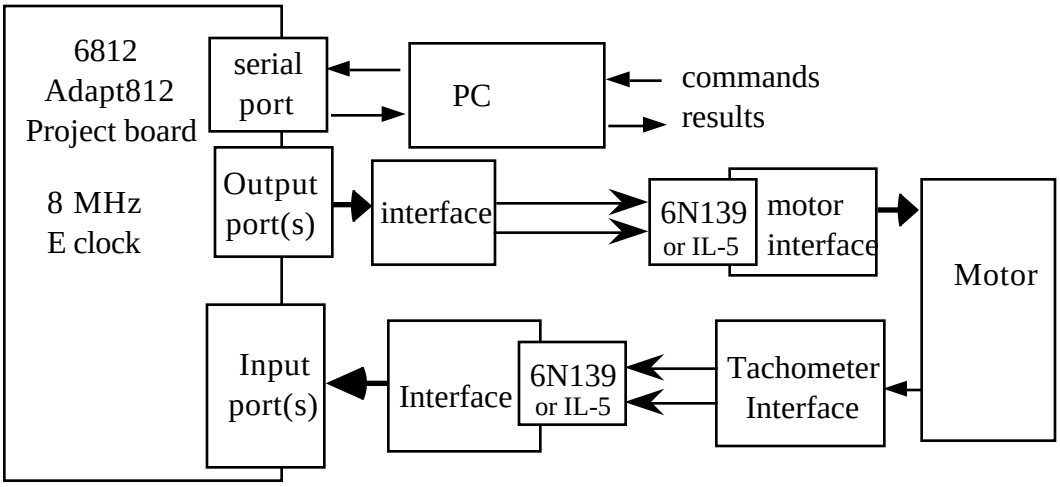

Figure 23.1. Block diagram of the motor control system.

The Buehler motor coil resistance is about 12 Ω, costs about $5.00, and is rated at 6,800 rpm at 12 V. The red/blue wires are the motor power, and the yellow/green wires are the tachometer output.

Typical Static Characteristics

Voltage (V)	Current (A)	Power (W)	Tach Freq (Hz)
0	0.000	0.000	0
1	0.100	0.100	28
2	0.120	0.240	125
3	0.140	0.420	196
4	0.150	0.600	266
5	0.165	0.825	336
6	0.180	1.080	424
7	0.195	1.365	494
9	0.200	1.800	636
11	0.215	2.365	785
12	0.230	2.760	858

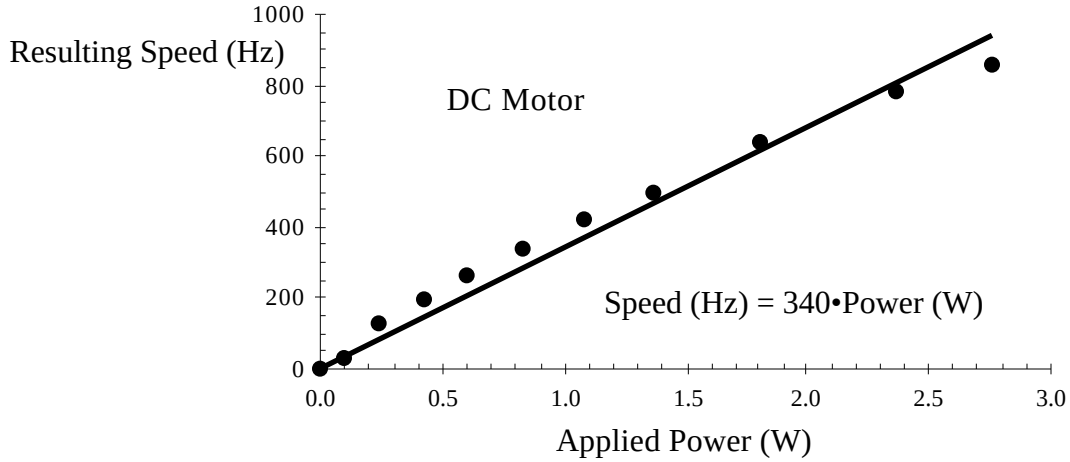

Figure 23.2. The speed of the motor is a function of the applied power (at 100% duty cycle). Notice how nonlinear this motor is!

Simple Controller

You will implement two control systems. The first one will be a simple incremental controller. Let **u** be the duty cycle of a fixed 100 Hz squarewave controlling the motor. The power to the motor is directly related to the duty cycle. The duty cycle is increased by a fixed amount (e.g., 1%) if it is spinning too slowly, and decreased by a fixed amount (e.g., -1%) if it is spinning too fast. The incremental control algorithm executes the following at a regular rate:

u = min($\mathbf{u_{max}}$,**u+1**) if **x*>x'** (too slow)

or **u** = max($\mathbf{u_{min}}$,**u-1**) if **x*<x'** (too fast)

The min and max operations maintain the duty cycle within the valid range of $\mathbf{u_{min}}$ to $\mathbf{u_{max}}$. The incremental control will be used to test the sensor and actuator subsystems. The disadvantage of incremental control is that it has a very slow response. The second system you will implement will be a PI, PID or Fuzzy Logic controller.

Linear Control Theory

We can use linear control theory to develop the digital controller. Since the tachometer frequency is the motor shaft speed we will define **x(t)** as the actual tachometer frequency, and **x'(n)** as the measured tachometer frequency. Since period measurement on the 6812 is so accurate we can assume **x'(n)** correctly estimates motor speed. I.e.,

x'(n) = **x(t)** at $\mathbf{t = n \cdot \Delta t}$

Any error in the state estimator will lead to a nonremovable controller error. Just like the data acquisition and digital filter situations, **t** is the continuous time and **n** is the discrete time. We will assume the controller is executed at a fixed interval, Δ**t**. Next, let **c** be the linear gain between the actuator command **u(n)** (duty cycle of the pulse width modulated wave) and the applied power **p(t)**.

$$p(t) = c \cdot u(n) \qquad \text{for } n \cdot \Delta t \le t < (n+1) \cdot \Delta t$$

We will analyze the system in the frequency domain. Let **X(s)** be the Laplace transform of the state variable **x(t).** Let **X*(s)** be the Laplace transform of the desired state variable **x*(t).** Let **E(s)** be the Laplace transform of the error

$$E(s) = X^*(s) - X(s)$$

Let **G(s)** be the transfer equation of the PI or PID linear controller (we can not write a transfer equation for the incremental or Fuzzy Logic controller.)

$$G(s) = c\left(k_P + k_D s + \frac{k_I}{s}\right)$$

Let **H(s)** be the transfer equation of the DC motor. If we assume the DC motor has a simple single pole behavior, then we can specify its response in the frequency domain with two parameters. The motor is certainly not linear, but we will assume it to be. **m** is the DC gain and τ is its time constant. The transfer function of the motor is

$$H(s) = \frac{m}{1 + \tau s}$$

The overall gain of the control system is

$$\frac{X(s)}{X^*(s)} = \frac{G(s)H(s)}{1 + G(s)H(s)}$$

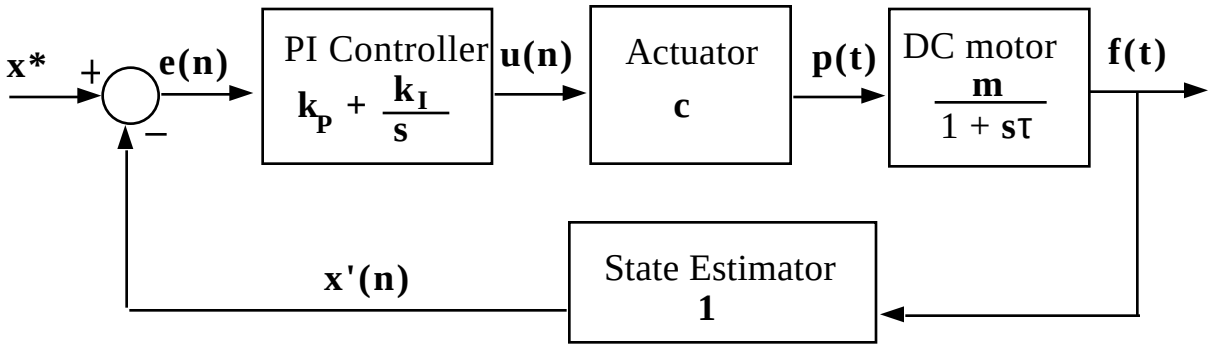

Figure 23.3. Block diagram of a linear control system in the frequency domain.

Theoretically we can choose controller constants, k_P k_I and k_D, to create the desired controller response. Unfortunately it can be difficult to estimate **c**, **m** and τ. If a load is applied to the motor, then **m** and τ will change. In addition, we can tell from Figure 23.3 that the motor does not follow a simple single pole relationship.

Nevertheless, we can still implement a very effective control system using PI or PID equations. A simple empirical method can be used to determine the controller constants. This empirical approach starts with just a proportional term (k_P). This proportional controller will generate a smooth motor speed (actuator output achieves a constant value), but the speed will not be correct. Try different k_P constants until the response times are fast enough. The response time is the delay after **x*** is changed for the motor to reach a new constant speed. k_P is too big if the actuator saturates both at the maximum and minimum after **x*** is changed. Steady state controller accuracy is defined as the average difference between **x*** and **x'**. The next step is to add some integral term (k_I) a little at a time to improve the steady state controller accuracy without adversely affecting the response time. Don't change both k_P and k_I at once. Rather, you should vary them one at a time. Overshoot is defined as the maximum positive error that occurs when **x*** is increased. Similarly, undershoot is defined as the maximum negative error that occurs when **x*** is decreased. If the response time, overshoot, undershoot and accuracy are within acceptable limits, then a PI controller is adequate. On the other hand, if accuracy and response are OK but overshoot and undershoot are unacceptable, add a little derivative term (k_D) to reduce the overshoots/undershoots in the step response.

Instead of PI or PID control, you have the option of implementing the motor controller using Fuzzy logic. Full credit will be given to a fuzzy logic solution with at least 10 Fuzzy Membership input sets and 5 Fuzzy Membership output sets.

Previous Labs (this section was previously completed)

Show the interface between tachometer and the Adapt812 project board. You may select either period or frequency measurement mode. Frequency measurement affords a direct calculation of shaft speed while the period measurement is faster. Frequency measurement also has the advantage of returning a reasonable result when the motor is stopped. Period measurement must handle the *no period* situations as special case. You should be using input capture to measure period, then performing a division in software to calculate frequency. Be careful to convert the AC voltage from the tachometer into a CMOS compatible digital signal. Use hysteresis to reject jitter noise. You must use the electrical isolation circuits developed in previous labs. You will extend the 6N139 isolation so that the signals are isolated in both directions.

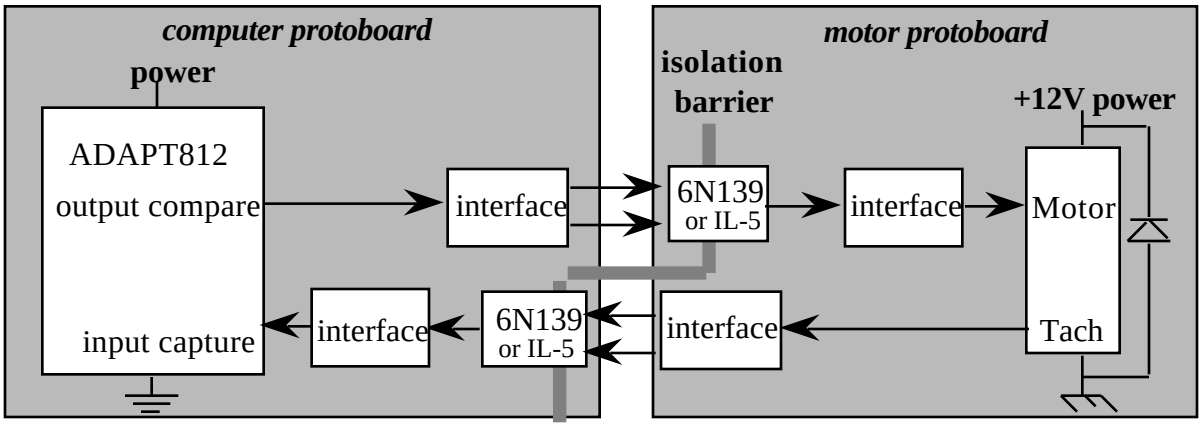

Figure 23.4. Hardware block diagram showing the isolation barrier.

Show the interface between the Adapt812 board and the DC motor. Use either the +5 V or +12 V supply to power the motor. You will use pulse-width modulation using a variable duty-cycle wave. If you were to use a DAC method to implement the actuator, then you would need expensive analog isolation, and a power amplifier to convert the DAC voltage into motor current. Since your actuator is either on or off, you will need a high current driver like the ULN2074 or IRF540 MOSFET to switch the motor current. PLEASE DO NOT CONNECT THE MOTOR DIRECTLY TO THE OUTPUT OF THE 6812, TTL CHIPS, OP AMPS, or 6N139 (or IL-5).

You will need to measure the static response of the motor: the tachometer output, **x(t)**, (in Hz) versus applied power, **p(t)**, (in watts) response of the motor under a no load condition (nothing attached to the shaft.) Measure the shaft speed with both your 6812 system and with a calibrated frequency meter. Show the electrical circuit used to make the measurements. Take 10 measurements with voltages ranging from 0 to 12 volts. Show both the raw measurements (voltage, current, period), and the calculated data (motor impedance, applied motor power, tachometer frequency) in a table. Fit the data to the linear equation:

$$\mathbf{x(t) = m\,p(t)}$$

Include a plot of the tachometer frequency versus applied power data with the linear fit.

Preparation (do this before your lab period)

Include software listing in the preparation. Organize into modules, so that the debugging can be done piece by piece. A "syntax-error-free" hardcopy listing for the software is required as preparation. The TA will check off your listing 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.

Procedure (do this during your lab period) (even if you are doing Fuzzy Logic)

Next you will measure the dynamic response of the motor. We will assume the motor is a first order (single pole) system. Measure the shaft speed with your 6812 system during a step change in the motor power. Show the electrical circuit used to make the measurements. Take multiple tachometer measurements under during three transient conditions:

full stop to half speed,
half speed to full speed,
full speed to half speed (you will get three different time constants because the motor is not linear).

Fit the data to the first order exponential equation:

$$\mathbf{x(t) = A + (B-A)e^{-t/\tau}}$$

where **B** is the initial speed and **A** is the final speed. Calculate the time constant τ. Include semilog plots of the transient data with the regression fits. The following figure shows a simulated response when the duty cycle is changed from 10 % to 50%.

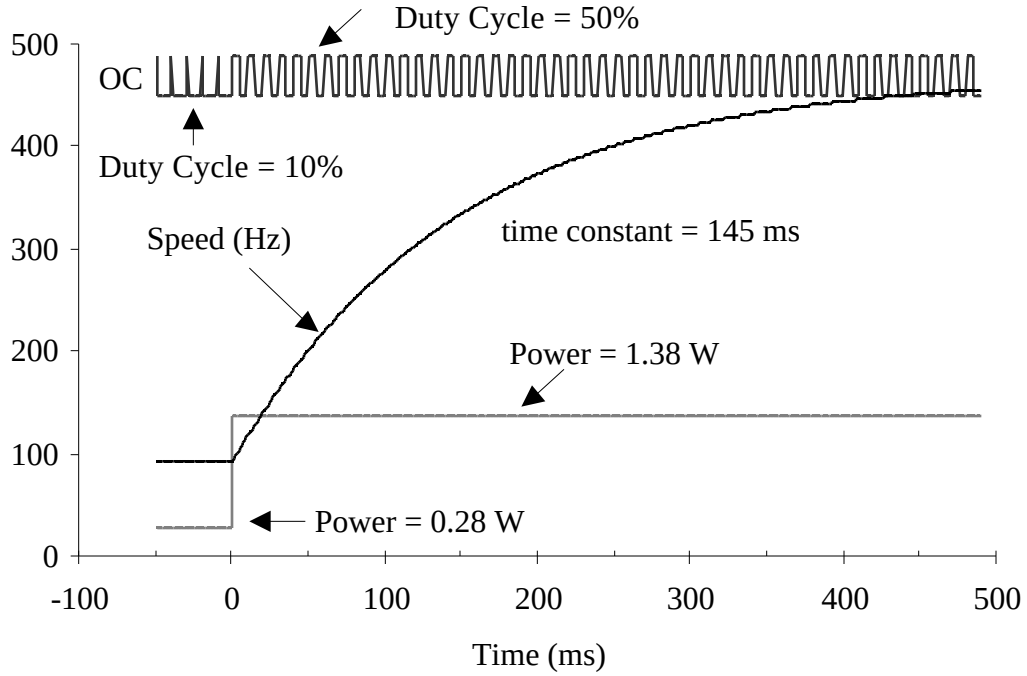

Figure 23.5. Simulated transient response of a DC motor. Your measurement data will fit an exponential equation as well as this simulated response.

You will run the PI control system 2 to 10 times faster than τ. This combination gives us a model of the motor:

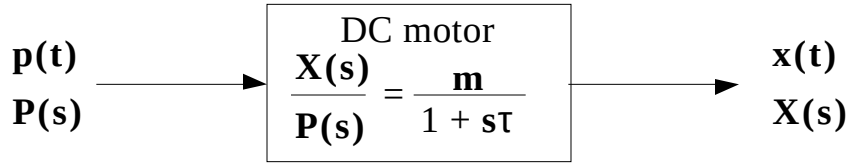

Figure 23.6. Simple model of a DC motor.

The software should have no foreground (main) process and four or more background (interrupt) processes. Please consider priority among the various tasks. One or more of the background processes may be executed from the same interrupt, but please keep the source code of processes logically separate. Choose appropriate data structures to pass parameters. Please specify in the comments the UNITS of each software variable.

The foreground (main) process:

initializes I/O ports and data structures
explanations of the various interpreter commands
executes a do-nothing loop

The interpreter process (using periodic polling):

can specify the desired motor speed, $0 \le \mathbf{x^*} \le 500$ Hz with a resolution of 1 Hz

The Motor Speed Estimator (interrupt) process:

begins with a frequency (or period) measurement
let **x'(n)** be the sequence of estimated frequency values in Hz
the 1 Hz resolution is required for the 100 to 500 Hz range

the sampling rate is 2 to 10 times faster than τ.

The Digital Controller (interrupt) process:

the controller rate is 2 to 10 times faster than τ.
update the actuator control depending on the error, $\mathbf{e(n) = x^*\text{-}x'(n)}$
let **u(n)** be the sequence of actuator commands (duty-cycles for squarewave)
implement a PID control system (the PI system has $\mathbf{k_D}$=0)

Proportional part is	$\mathbf{P(n)}$	$= \mathbf{k_P \cdot e(n)}$	
Integral	$\mathbf{I(n)}$	$= \mathbf{I(n\text{-}1) + k_I \cdot e(n) \cdot \Delta t}$	
Derivative	$\mathbf{D(n)}$	$= \mathbf{k_D \cdot [e(n) + 3e(n\text{-}1) - 3e(n\text{-}2) - e(n\text{-}3)]/(6\Delta t)}$	
PID	$\mathbf{u(n)}$	$= \mathbf{P(n)+I(n)+D(n)}$	
Final output		$\mathbf{u_{max}}$	if $\mathbf{u(n) \geq u_{max}}$
	$\mathbf{u(n)}$	$= \mathbf{u(n)}$	if $\mathbf{u_{max} \leq u(n) \leq u_{min}}$
		$\mathbf{u_{min}}$	if $\mathbf{u(n) \leq u_{min}}$

The Display (interrupt) process:

maintain a flicker-free display of **x(n)** on the PC terminal with units of Hz.

You will first implement the Incremental controller, then you will implement PI, PID or Fuzzy. Please compare and contrast the two approaches.

Checkout (show this to the TA)

1. Shaft speed estimator accuracy and control system accuracy. The TA will attach a load to the shaft. TA will select desired speeds in the range of 100 to 500 Hz. After 10 to 15 seconds, the TA will record the speed as estimated by your system (the three LEDs) and as measured by a calibrated frequency meter.
2. The TA will measure the stability by recording five measurements at the same **x***.
3. The TA will evaluate the quality of the user interface:
   on-line documentation, simple self-explanatory operation,
   error handling of illegal commands and illegal input numbers,
   jitter-free display,
   no crashes.

Hints/clarifications

1. Be very careful to include the snubber diode to remove back EMF. Please observe the motor voltages on the scope before connecting to the microcomputer.

2. Some PI and PID controllers limit the magnitude of the I(n) term, so that long term controller errors do not saturate the system (called anti-reset-windup).

3. If you are have trouble getting anything to work, demonstrate the simple incremental controller for partial credit.

4. The derivative term in both the PID and Fuzzy can be very unstable, so use it in small doses.

5. See the Fuzzy Logic examples included with the TExaS simulator.

6. For this lab you will use the pulse width modulation (PWM) hardware and software that you created in a previous lab. You will also need to design an analog interface circuit to convert the tachometer output to a CMOS compatible digital signal (square wave). Note that the tachometer output at 1200 Hz can reach 40 volts peak to peak. Please add optical isolation between the tachometer circuit and the 6812.

7. Measure the dynamic response of the motor. For this exercise it is important to measure frequency as fast as you can. You may want to use the period technique because it will automatically go faster as the motor goes faster. Take enough data points to draw a smooth curve. You may want to save the frequency values in a table, and after completing the transient response, print them to the screen. The ICC12 terminal window can 'log to file' what you print to the screen. If you format the data appropriately you can directly import the file into EXCEL.

8. The control software will need to run 2 to 10 times faster than the time constant (τ) that you calculated in the previous step. This is VERY important, the most likely reason for a poorly performing controller is executing the control interrupt handler at too slow a rate. The software will have no foreground (main) thread and four or more background (interrupt) threads. You must be able to input a new desired speed at any time while the motor is running. The motor speed will be displayed on the screen at a reasonable display rate. Both the speed input and the speed display routines will be handled by periodic interrupt handlers. See Chapters 4 and 6 concerning periodic interrupts.

9. You must implement the incremental controller. You must also implement either the PI controller or the Fuzzy logic controller. If you do both PI and Fuzzy logic, you may get bonus points. Compare and contrast the incremental controller results with the results for your other controller. You may add the derivative term to make a PID controller, but it is not necessary to make the controller work. The derivative term for both PID and Fuzzy can be very unstable, so use it in small doses.

10. Desired speed range: 100 to 800 Hz in 1 Hz increments. This is easy, since you are using 12-volt motor power.

11. Design your software so that it is easy to change control gains (both PI and Fuzzy) because you will need to play with them in order to make you motor respond properly. Your TA would like to see what happens if you make the gains too large. It is important to understand under what conditions a controller is unstable. Document your development results. Tell your TA what you learned.

12. A good controller (PI or Fuzzy) should be able to make a large step change in speed (100 Hz to 700 Hz) in approximately 1 second. Your TA will test your controller by making step changes in speed and by applying a disturbance or a load to your motor. Your displayed tachometer speed must be very close to the desired speed and must be equal to the tachometer speed as measured by a calibrated frequency meter.

13. You will probably have trouble with noise impulses on the tachometer signal especially at 100 Hz. The period measurement method is extremely sensitive to noise, the frequency measurement method is less sensitive. Consider the combined frequency/period method described in Problem 6.3 of the book Embedded Microcomputer Systems by Jonathan W. Valvano. You will likely have trouble controlling the motor speed precisely at 100 Hz, do your best. You may want to display the tachometer speed and error during debugging to see if your software is working OK. If you get unreasonably large errors due to noise, you may want to limit the error to some maximum and minimum value to keep your control system from going crazy. Consider changing you duty cycle resolution from 1% to 0.1% (equivalent to changing from 0-100 to 0-1000 based system) because changing duty cycle by +/- 1% at 100 Hz changes the motor speed significantly and makes the control job more difficult.