UNIVERSITI MALAYSIA PAHANG

BORANG PENGESAHAN STATUS TESIS♦

JUDUL: **PID DIGITAL CONTROLLER FOR DC MOTOR SPEED USING MC68HC11 MICROCONTROLLER**

SESI PENGAJIAN: **2007/2008**

Saya **SHARON PETERUS (861116-13-5148)**
(HURUF BESAR)

mengaku membenarkan tesis (Sarjana Muda/~~Sarjana~~ /~~Doktor Falsafah~~)* ini disimpan di Perpustakaan dengan syarat-syarat kegunaan seperti berikut:

1. Tesis adalah hakmilik Universiti Malaysia Pahang (UMP).
2. Perpustakaan dibenarkan membuat salinan untuk tujuan pengajian sahaja.
3. Perpustakaan dibenarkan membuat salinan tesis ini sebagai bahan pertukaran antara institusi pengajian tinggi.
4. **Sila tandakan (√)

☐	**SULIT**	(Mengandungi maklumat yang berdarjah keselamatan atau kepentingan Malaysia seperti yang termaktub di dalam AKTA RAHSIA RASMI 1972)
☐	**TERHAD**	(Mengandungi maklumat TERHAD yang telah ditentukan oleh organisasi/badan di mana penyelidikan dijalankan)
☑ √	**TIDAK TERHAD**	

Disahkan oleh:

(TANDATANGAN PENULIS) (TANDATANGAN PENYELIA)

Alamat Tetap:

NO. 17 REJANG 21
JALAN SIBU, TAMAN TUNKU
98000 MIRI SARAWAK

Puan Haszuraidah binti Ishak
(Nama Penyelia)

Tarikh: **17 NOVEMBER 2007** Tarikh: **17 NOVEMBER 2007**

CATATAN: * Potong yang tidak berkenaan.
** Jika tesis ini SULIT atau TERHAD, sila lampirkan surat daripada pihak berkuasa/organisasi berkenaan dengan menyatakan sekali tempoh tesis ini perlu dikelaskan sebagai atau TERHAD.
♦ Tesis dimaksudkan sebagai tesis bagi Ijazah doktor Falsafah dan Sarjana secara Penyelidikan, atau disertasi bagi pengajian secara kerja kursus dan penyelidikan, atau Laporan Projek Sarjana Muda (PSM).

PID Digital Controller for DC Motor Speed Using MC68HC11 Microcontroller

SHARON PETERUS

This thesis is submitted as partial fulfillment of the requirement for the award of the Bachelor Degree of Electrical Engineering (Electronics)

Faculty of Electrical & Electronics Engineering
Universiti Malaysia Pahang

NOVEMBER 2008

"I hereby acknowledge that the scope and quality of this thesis is qualified for the award of the Bachelor Degree of Electrical Engineering (Electronics)"

Signature : _____

Name : MDM. HASZURAIDAH BINTI ISHAK

Date : 17 NOVEMBER 2007

Signature : _____

Author : SHARON PETERUS

Date : 17 NOVEMBER 2007

DEDICATION

To my beloved parents, sisters and brother.
May God bless all of you, always.

ACKNOWLEDGEMENT

I would like to take this opportunity to express my deepest gratitude to my project supervisor, Mdm. Haszuraidah Ishak who has persistently and determinedly assisted me along the progress of the project. It would have been difficult to complete this project without enthusiastic support, insight and advice given by her.

My outmost thanks to my family members, for their endless love, care and support throughout my academic years in Universiti Malaysia Pahang.

Special thanks to FKEE lecturers and staffs, for the help and advises in order for to complete my project. Suggestions and criticisms from my friends have always been helpful as well in finding solutions to my problems. Thank you all.

Finally, I would like to express my thanks and gratitude to those who involves directly or indirectly in the completion of my project. Unfortunately, it is not possible to list all of them in this limited space.

Thank you so much. Your sincere help will be remembered for life.

ABSTRACT

The proportional-integral-derivative (PID) controllers are widely used in many industrial control systems for several decades since Ziegler and Nichols proposed their first PID tuning method. This is because the PID controller structure is simple and its principle is easier to understand than most other advanced controllers. On the other hand, the general performance of PID controller is satisfactory in many applications. For these reasons, the majority of the controllers used in industry are of PI/PID type. PID controllers are widely used for process control applications requiring very precise and accurate control. The purpose of the motor speed controller is to take a signal representing the demanded speed, and to drive a motor at that speed. The controller does not actually measure the speed of the motor. Thus, it is called an Open Loop Speed Controller. Motors come in a variety of forms, and the speed controller's motor drive output will be different dependent on these forms. The speed controller presented here is designed to drive special dc motor which is not easily available anywhere in store, thus it is a good example to be used due to the special characteristics and parameters. Matlab Simulink® is an important tool used it this project, from designing the mathematical model of the dc motor, obtaining the transfer function, and designing the PID controller using both model and programming using m-files. The transfer function will be linearized and used for tuning the gain of PID controller like KP, KI, and KD. Simulink is chosen to simulate the performance of the control system.

ABSTRAK

Sistem pengawal PID digunakan dengan meluas dalam jangka masa beberapa puluh tahun kebelakangan ini, semenjak Ziegler dan Nichols memperkenalkan kaedah PID yang pertama. Ini kerana struktur sistem pengawal PID yang senang difahami berbanding dengan kaedah pengawalan yang lain. Banyak aplikasi menggunakan kaedah pengawalan ini kerana kestabilan system yang boleh diaplikasikan dalam pelbagai system. Pengawal PID amat berguna bagi aplikasi yang memerlukan sistem kawalan yang tepat dan stabil. Tujuan utama sistem pengawalan kelajuan motor adalah untuk mendapatkan signal bagi kelajuan yang dikehendaki dan mengurangkan kesalahan untuk mendapatkan sistem yang tepat dan stabil. Sistem kawalan yang akan dibuat ini tidak mengukur halaju motor tersebut, oleh itu ia dipanggil Sistem Kitaran Terbuka. Terdapat banyak jenis motor di dalam pasaran, di mana setiap system pengawalan kelajuan berbeza bergantung kepada jenis motor yang digunakan. Dari model yang dihasilkan, rangkap pindah boleh diperolehi. Ragkap pindah ini dilinearkan dan digunakan untuk talaan gandaan pengawal PID. Perisian Simulink dipilih untuk mengkaji prestasi Sistem Kawalan Kelajuan Motor ini.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

P	Proportional
PI	Proportional-Integral
PID	Proportional-Integral-Derivative
DC	Direct Current
DSP	Digital Signal Processing
MCU	Microcontroller
EMF	Electromagnetic Field
PC	Program Counter
RAM	Random Access Memory

LIST OF APPENDIXES

CHAPTER 1

INTRODUCTION

1.1 Introduction

The proportional-integral-derivative (PID) controllers are widely used in many industrial control systems for several decades since Ziegler and Nichols proposed their first PID tuning method. This is because the PID controller structure is simple and its principle is easier to understand than most other advanced controllers. On the other hand, the general performance of PID controller is satisfactory in many applications. For these reasons, the majority of the controllers used in industry are of PI/PID type. PID controllers are widely used for process control applications requiring very precise and accurate control. Unlike on/off controls, the smooth and steady state control is achievable using these controllers. Various models are available featuring single loop with universal input, two to eight loop with eight independent inputs and sixteen control outputs. All types of digital and analog outputs are available to operate final controlling devices such as Solid State Relays, Contactors, Solenoid valves, Modulating motorized valves, thyristorized power packs etc.

The purpose of a motor speed controller is to take a signal representing the demanded speed, and to drive a motor at that speed. The controller may or may not actually measure the speed of the motor. If it does, it is called a Feedback Speed Controller or Closed Loop Speed Controller, if not it is called an Open Loop Speed Controller. Feedback speed control is better, but more complicated, and may not be required for a simple robot design. Motors come in a variety of forms, and the speed controller's motor drive output will be different dependent on these forms. The speed controller presented here is designed to drive a special dc motor which is suitable for education purposes.

In this project, Motorola 68HC11 processor board will be used for all decentralized processing duties. This Motorola boards provide high-speed analog to digital conversion along with high reliability and robustness. This makes the 68HC11 is well suited for sensor translation as well as motor control.

Rapid progress in microelectronics and microcontrollers in recent years has made it possible to apply modern control technology to automobiles that need real-time control. DC motors control, many of these operations and therefore there is a need for implementing effective control strategies for digital control of these motors. Therefore, it is quite important to develop real-time DC motor control strategies such that these devices are effectively integrated with their control electronics using the MC68HC11 microcontroller. Realizing the fact that large number of motors are utilized in modern vehicles and there is especially a need to control these small motors using a common bus with interrupt priorities, such real-time control is extremely essential.

1.2 Project Objective

The objectives of this PID Digital Controller for DC Motor Speed using MC68HC11 microcontroller are to design a closed-loop controller, a very common means of keeping motor speed at the required "setpoint" under varying load conditions, to implement a PID controller system onto a MC68HC11 microcontroller and to designate in advance the mathematical model of the system to be controlled with the purpose of simulating its dynamic behavior in open-loop mode.

In general, the main purpose of this project is to propose a DC motor control design approach using the PID algorithm and MC68HC11 microcontroller that contains an embedded closed loop digital controller for the motor speed correction.

1.3 Project Scopes

The scopes of this project include deriving a transfer function for a DC motor and simulating the results in Simulink® MATLAB application. A model of a PID controller algorithm is then designed and simulated in Simulink® MATLAB application. The usge of the PID controller is then implemented based on the configuration of the MC68HC11 microcontroller.

1.4 Thesis Outline

The thesis is orderly organized into 6 chapters and they are outlined as below:

Chapter 1 explains the proportional-integral-derivative (PID) controllers and essential concepts which guide to the development of the digital controller system. It also outlines objective and scope of this project.

Chapter 2 describes the overview of the project elaborately. The literature review will be focused on hardware and software design. Explanation will be based on theory and conceptual ideas. Some practical approach in this project will also be discussed.

Chapter 3 discusses the derivation of the transfer function and development of the mathematical model of the DC motor.

Chapter 4 provides description and discussion on the design of the PID Controller. It also indicates the development of the controller and system operation.

Chapter 5 presents various testing and results that are conducted.

Lastly, Chapter 6 summarizes the overall conclusion for this thesis and a few suggestion and recommendation for future development.

CHAPTER 2

LITERATURE REVIEW

There are many ways to control DC motors. Open-loop current control acts directly on torque and thus protects the electronics, the motor and the load. Open-loop variable voltage control makes sense if the motor and electronics are not overloaded when the motor stalls. Open-loop variable voltage control with a current limiting circuit constitutes the simplest way of varying speed. However, a closed-loop system is needed if precision is called for in selecting speeds. In order to be able to build a closed loop controller, we need some mean of gaining information about the rotation of the shaft like the number of revolutions executed per second, or even the precise angle of the shaft. This source of information about the shaft of the motor is called "feed-back" because it sends back information from the controlled actuator to the controller.

In a closed loop speed controller, a signal proportional to the motor speed is fed back into the input where it is subtracted from the set point to produce an error signal. This error signal is then used to work out what the magnitude of controller output should be to make the motor run at the required set point speed. For example, if the error speed is positive, the motor is running too fast so that the controller output should be reduced and vice-versa. The clever part is how the output drive is worked out. [12] In a closed-loop configuration, a portion of the information is fed back from the process and subtracted from the reference signal in order to calculate the error signal. This error signal is used by the PID to adjust the control input such that the process output can reach the given reference. [15] In other words, a closed loop controller will regulate the power delivered to the motor to reach the required velocity or speed. If the motor is to turn faster than the required velocity, the controller will deliver less power to the motor. Controlling the electrical power delivered to the motor is usually done by Pulse Width Modulation.

The aim of a control circuit is to keep the permanent magnet dc motor running at a constant speed, set externally. To do this, the current through, and the voltage across, the brushes of the motor are monitored. The voltage consists of two components: First, a back-EMF generated by the windings of the armature moving through the magnetic field of the motor. Secondly, there's a voltage caused by the current passing through the real resistance of the windings and the brushes.

Most of real plant operates in a wide range of operating conditions; the robustness is then an important feature of the closed loop system. When this is the case, the controller has to be able to stabilize the system for all operating conditions. To this end, it is possible to employ an internal-model-based PID tuning method [9 and 3]. However, this method gives very slow response to load disturbance for lag-dominant processes because of the pole-zero cancellations inherent in the design methodology [2]. Another popular approach with similar emphasis is the tuning of PI or PID controller by the gain and phase margin specifications [7 and 1]. Gain margin and phase margin have always served as important measures of robustness. It is well known that phase margin is related to the damping of the system, and can therefore also serve as a performance measure [4]. In this way, numerous progresses have been made to improve the performances of the PID control [6]. In particular, tuning methods based on optimization approach have recently received more attention in the literature, with the aim of ensuring good stability robustness of the controlled system [8 and 5]. However, these new methods are not easy to use for the operating engineer who is the main user of the PID controller.

PID controllers are widely used in the process-control industry, mainly because of their effectiveness and simple structure. A feedback-control scheme can be implemented in a simple 8-bit microcontroller (MCU). Despite the fact that the MCU is among the simpler ones available, the time it requires to compute the whole program and bring the motors to their maximum reference speed (9,100RPM) was only 866ms in the worst case. So it is believed that using the newer MCUs in a closed-loop configuration can improve the control of some electro-mechanical actuators such as valves or motors and improve the performance of some processes with slow dynamic behavior. [15]

The main PID controller routine was designed to be fairly general purpose and hence modular. Whilst here it is used to control a DC motor, it could be re-deployed to other situations where some parameter has to be controlled to a set value under varying conditions. The actual control software is located in a single function and its major inputs and output are held in a structure. Although it was designed originally for a specific job it is really only intended as an example of the basic techniques involved and to allow those with no control system knowledge to experiment with a simple PID system.

Currently, several manufacturers make 16- and 32-bit microcontrollers (MCUs) with features that enable easy control of almost any process of medium complexity. Eight-bit microcontrollers still dominate the market, however, because of their small size, low cost, and simple programming. Because of these advantages, 8-bit MCUs are found in process control, automotive, industrial, and appliance applications, among many others. Some of the newer MCUs provide clock speeds from 4 to 40MHz; 64KB of internal flash memory and 1KB of RAM in some models; on-chip analog-to-digital converters (ADCs), digital-to-analog converters (DACs), or pulse-width modulator (PWM) outputs; a watchdog timer; 16-bits timers; and serial or USB ports.

Although the features of 8-bit MCUs are continually improving, in most cases these new features are ignored by designers because they're using the chips for the control of states, which don't require the newer features. [10 AND 11] Recently a novel method has been used to exploit these kinds of MCUs by using them in a closed-loop configuration aided with the well-known classical control theory. Examples of work on this topic are available in the literature. [12, 13, and 14] In such applications the authors demonstrated that feedback control improves the control of some DC motors. It's important to mention that, in the examples, except in Johnston, [13] the realization of the PID (proportional-integral-derivative) controllers were implemented using 16-bit MCUs in Hitex's paper [12] or, as in Neary,[14] where an integrated data acquisition system particularly the model ADuC845 by Analog Devices, was used.

Until few years ago, these kinds of tasks (micro-positioning or servo control) had been addressed using digital signal processors (DSPs), mainly because such devices are faster and have higher precision than the 8-bits MCUs. However, some applications don't require high precision or the team simply can't justify the cost of a DSP. It's usually cheaper to use an 8- or 16-bit MCU without diminishing performance.

Generator represents the back electromotive force (BEMF) generated by the motor's rotation and which opposes the electromotive force of the supply. The value of the BEMF is a function of the motor's angular velocity. If the motor has no external load and its velocity is not limited, it will accelerate up to the velocity w such that V(w) equals the supply voltage Vs. [10] In this situation the two EMF's cancel each other and thus the motor torque responsible for acceleration will go away. In reality V(w) is always slightly less than Vs in which case a small motor torque is necessary to compensate resistive torque due to internal friction. Thus it can be seen that the motor's BEMF can reach elevated values which in some cases can create application problems due to a certain type of stress.

Driving DC motors with integrated circuits seems at first to be rather simple. Yet by analyzing the actual application it is possible to see if there exist conditions causing stresses to the IC during operation which in the end can cause failure. With proper design and analysis in critical applications it is possible to avoid conditions which lead to IC damage. A closed loop controller can be an analog circuit, a digital circuit made of logic gates, or a micro controller. Generally, a micro controller is the option that will provide more design flexibility. Recent microcontrollers running at very high clock rates can completely replace similar analog controllers, and can even be cheaper.

In a closed loop system, a microcontroller will have two main tasks:

- Constantly adjust the average power delivered to the motor to reach the required speed.
- Precisely calculate the position/angle of the motor's output shaft.

At first sight it might be imagined that something simple like "if the error speed is negative, multiply it by some scale factor (usually known as "gain") and set the output drive to this level", i.e. the voltage applied to the motor is proportional to the error speed. In practice, this approach is only partially successful for the following reason: if the motor is at the setpoint speed under no load there is no error speed so the motor free runs. If a load is applied, the motor slows down so that a positive error speed is produced. The output increases by a proportional amount to try and restore the speed. However, as the motor speed recovers, the error reduces and so therefore does the drive level. The result is that the motor speed will stabilise at some speed below the setpoint at which the load is balanced by the error speed x the gain. If the gain is very high so that even the smallest change in motor speed causes a significant change in drive level, the motor speed may oscillate or "hunt" slightly . This basic strategy is known as "proportional control" and on its own has only limited use as it can never force the motor to run exactly at the setpoint speed.

The next improvement is to introduce a correction to the output which will keep adding or subtracting a small amount to the output until the motor reaches the setpoint, at which point no further changes are made. In fact a similar effect can be had by keeping a running total of the error speed speeds observed for instance, every 25ms and multiplying this by another gain before adding the result the proportional correction found above. This new term is based on what is effectively the integral of the error speed.

Thus far we have a scheme where there are two mechanisms trying to correct the motor speed which constitutes a PI (proportional-integral) controller. The proportional term is a fast-acting correction which will make a change in the output as quickly as the error arises. The integral takes a finite time to act but has the ability to remove all the steady-state speed error.

A further refinement uses the rate of change of error speed to apply an additional correction to the output drive. This means that a rapid motor deceleration would be

counteracted by an increase in drive level for as long as the fall in speed continues. This final component is the "derivative" term and it is a useful means of increasing the short-term stability of the motor speed. A controller incorporating all three strategies is the well-known Proportional-Integral-Derivative, or "PID" controller.

For best performance, the proportional and integral gains need careful tuning. For example, too much integral gain and the control will tend to over-correct for any speed error resulting in oscillation about the setpoint speed. Several well-known mathematical techniques are available to calculate optimal gain values, given knowledge of the combined characteristics of the motor and load, i.e. the "transfer function". However, some simple rules of thumb and a little experimentation can yield satisfactory results in practical applications.

CHAPTER 3

METHODOLOGY
(TRANSFER FUNCTION DERIVATION AND MATHEMATICAL MODEL DEVELOPMENT)

3.1 Introduction

The mathematical model for the transfer function of the DC motor is derived and simulated in the Simulink® MATLAB application to observe its dynamic behavior. The transfer function model of the motor is then used to design the digital controller.

3.2 Proportional-Integral-Derivative Controller

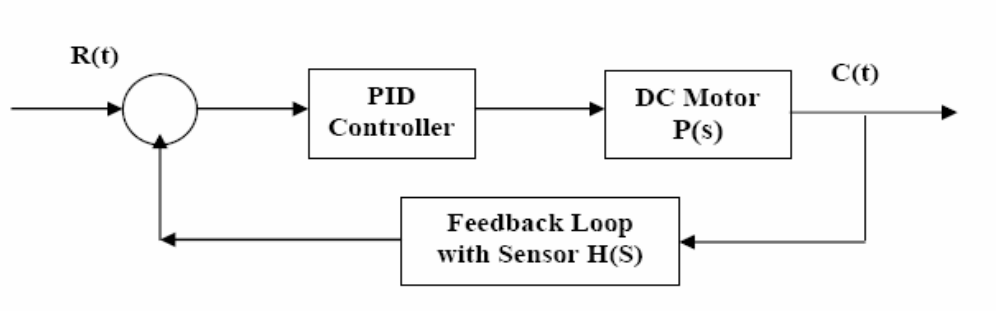


Figure 3.1: DC Motor PID Controller Model

The controller model shows a closed-loop controller, where the feedback from the output of the motor, which is speed, is analyzed again at the set-point. This ensures that the error is reduced from time-to-time, reducing the error each time.

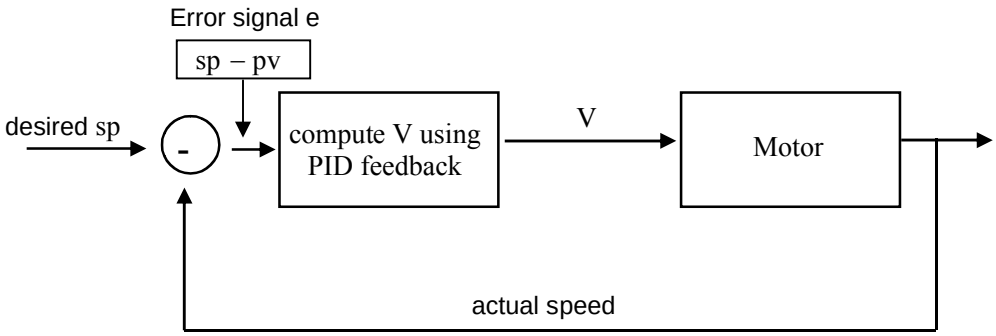


Figure 3.2: DC Motor PID Controller Model 2

A more detailed model is shown in Figure 3.2. The error signal calculates the current process variable, deducting it from the set point value to obtain the value of the error signal.

3.2.1 Derivation of the Transfer Function

The transfer function of the DC motor (CLIFTON 2250 SERVO MOTOR) is derived by using the values obtained from the datasheet of the motor and can be proved by using MATLAB SIMULINK® application.

The first step of this project is modeling the DC servo motor. Motor modeling is required in order to obtain the transfer function of the motor which is providing the open loop system of this project. Then PID controller is added to change the system to closed loop system. Below is the step of the motor modeling.

DC Servo Motor Parameters:

Electric Resistance, $R = 2.7\ \Omega$

Electric inductance, $L = 0.004$ H

Electromotive force constant ($K = Ke = Kt = 0.105$ Vs rad^{-1})

Moment of inertia of the rotor, $J = 0.0001$ Kg m^2

Damping ration, $B = 0.0000093$ Nms rad^{-1}

3.2.1.1 Manual Calculation of the Transfer Function

$$\frac{di_a}{dt} = \frac{R}{L} i_a - \frac{K}{L} \omega_r + \frac{1}{L} V_a$$

$$\frac{d\omega_r}{dt} = \frac{K}{J} i_a - \frac{B}{J} \omega_r \tag{3.1}$$

$$\begin{bmatrix} \frac{di_a}{dt} \\ \frac{d\omega_r}{dt} \end{bmatrix} = \begin{bmatrix} -\frac{R}{L} & -\frac{K}{L} \\ \frac{K}{J} & -\frac{B}{J} \end{bmatrix} \begin{bmatrix} i_a \\ \omega_r \end{bmatrix} + \begin{bmatrix} \frac{1}{L} \\ 0 \end{bmatrix} V_a \tag{3.2}$$

$$y = \begin{bmatrix} 0 & 1 \end{bmatrix} \begin{bmatrix} i_a \\ \omega_r \end{bmatrix} + [0] V_a \tag{3.3}$$

$$\begin{bmatrix} \frac{di_a}{dt} \\ \frac{d\omega_r}{dt} \end{bmatrix} = \begin{bmatrix} -\frac{2.7}{0.004} & -\frac{0.105}{0.004} \\ \frac{0.105}{0.0001} & -\frac{0.0000093}{0.0001} \end{bmatrix} \begin{bmatrix} i_a \\ \omega_r \end{bmatrix} + \begin{bmatrix} \frac{1}{0.004} \\ 0 \end{bmatrix} V_a$$

$$A = \begin{bmatrix} -675 & -26.25 \\ 1050 & -0.093 \end{bmatrix} \qquad B = \begin{bmatrix} -250 \\ 0 \end{bmatrix}$$

$$C = \begin{bmatrix} 0 & 1 \end{bmatrix} \qquad D = [0]$$

$$[sI - A] = \begin{bmatrix} s & 0 \\ 0 & s \end{bmatrix} - \begin{bmatrix} -675 & -26.25 \\ 1050 & -0.093 \end{bmatrix} \tag{3.4}$$

$$= \begin{bmatrix} s + 675 & 26.25 \\ -1050 & s - 0.093 \end{bmatrix}$$

From $$[sI - A]^{-1} = \frac{adj(sI - A)}{def\,(sI - A)}$$

(3.5)

If $$A = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \quad ; \quad A - 1 = \frac{1}{ad - bc}\begin{pmatrix} d & -b \\ -c & a \end{pmatrix}$$

(3.6)

$$\begin{aligned} \text{ad-bc} &= (s\text{-}675)(s\text{+}0.093)\text{-}(26.25)(1050) \\ &= s^2 + 0.093s - 675s + 62.775 + 27562.5 \\ &= s^2 + 675.093s + 27625.275 \end{aligned}$$

$$[sI - A]^{-1} = \frac{1}{s^2 + 675.093s + 27625.275}\begin{bmatrix} s - 0.093 & -26.25 \\ 1050 & s - 675 \end{bmatrix}$$

$$= \frac{\begin{bmatrix} s - 0.093 & -26.25 \\ 1050 & s - 675 \end{bmatrix}}{s^2 + 675.093s + 27625.275}$$

$$T(s) = \frac{Y(s)}{U(s)} = C[sI - A]^{-1}B + D \qquad (3.7)$$

$$= \begin{bmatrix} 0 & 1 \end{bmatrix}\frac{\begin{bmatrix} s - 0.093 & -26.25 \\ 1050 & s - 675 \end{bmatrix}}{s^2 + 675.093s + 27625.275}\begin{bmatrix} 250 \\ 0 \end{bmatrix} + [0]$$

We get,

$$= \frac{262500}{s^2 + 675.093s + 27625.275}$$

$$T(s) = \frac{Y(s)}{U(s)} = \frac{262500}{s^2 + 675.093s + 27625.275}$$

Thus, the transfer function of the DC Motor, TF = $\dfrac{262500}{s^2 + 675.093s + 27625.275}$

3.2.2 Mathematical Model of the DC Motor

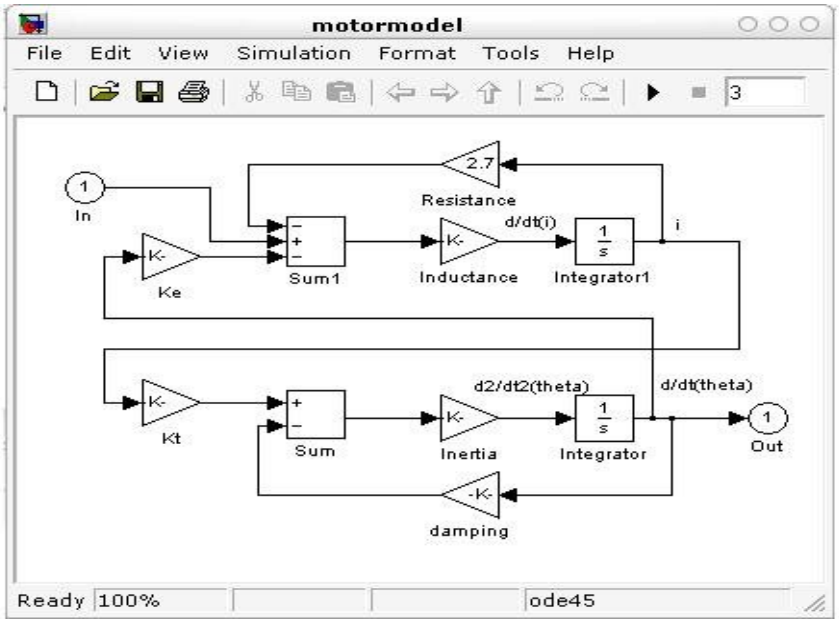


Figure 3.3: Linear Model of the DC motor built in MATLAB

This system is modeled by summing the torque acting on the rotor inertia and integrating the acceleration to give the velocity. Kirchoff's law is applied to the armature circuit. First, the integrals of the rotational acceleration and the rate of change of the arm,ature current is modeled.

An Integrator block (from the Linear block library) is inserted and lines are drawn to and from its input and output terminals. The input line "d2/dt2(theta)" and the output line "d/dt(theta)" are labeled as shown above. Another Integrator block is inserted above the previous one and lines are drawn to and from its input and output terminals. The input is "d/dt(i)" and the output line "i" are labeled.

The angular acceleration is equal to 1/J multiplied by the sum of two terms (one pos., one neg.). Similarly, the derivative of current is equal to 1/L multiplied by the sum

of three terms (one pos., two neg.). Insert two Gain blocks, (from the Linear block library) one attached to each of the integrators. The gain block corresponding to angular acceleration is edited by double-clicking it and changing its value to "1/J". The label of this Gain block is changed to "inertia" by clicking on the word "Gain" underneath the block. Similarly, the other Gain's value is edited to "1/L" and it's label is changed to Inductance. Two Sum blocks (from the Linear block library) were then inserted, one attached by a line to each of the Gain blocks. The signs of the Sum block corresponding to rotation to "+-" is changed since one term is positive and one is negative. The signs of the other Sum block is also changed, to "-+-" to represent the signs of the terms in Kirchoff's equation.

The torqueses which are represented in Newton's equation are then added. First, the damping torque is added. A gain block is inserted below the inertia block, selected by single-clicking on it, flipping it left-to-right. The gain value is then set to "b" and renamed to "damping". A line (hold Ctrl while drawing) is tapped off the rotational integrator's output and connected it to the input of the damping gain block. A line is drawn from the damping gain output to the negative input of the rotational Sum block.

The torque from the armature is added by inserting a gain block attached to the positive input of the rotational Sum block with a line. It's value is then edited to "K" to represent the motor constant and it is labeled with "Kt". The line leading from the current integrator is drawn and connected to the Kt gain block.

The voltage terms which are represented in Kirchoff's equation is then added. First, the voltage drop across the coil resistance is added in. A gain block above the inductance block is added and flipped left-to-right. The gain value is set to "R". A line (hold Ctrl while drawing) is then tapped off the current integrator's output and connected to the input of the resistance gain block. A line is drawn from the resistance gain output to the upper negative input of the current equation Sum block.

The back emf from the motor is added by inserting a gain block which is attached to the other negative input of the current Sum block with a line. It's value to

"K" to represent the motor constant and labeled with "Ke". A line is tapped off the rotational integrator output and connected to the Ke gain block.

The third voltage term in the Kirchoff equation is the control input, V. A step input is applied into the model. A Step block (from the Sources block library) is inserted and connected with a line to the positive input of the current Sum block.

To view the output speed, a Scope (from the Sinks block library) is inserted anc connected to the output of the rotational integrator.

To provide an appropriate unit step input at t=0, the Step block double-clicked and the Step Time is set to "0".

The output response:-

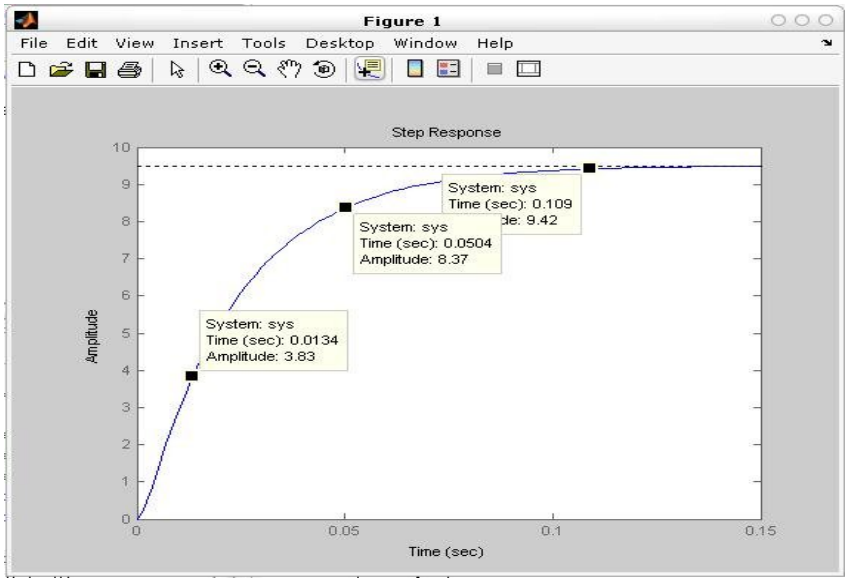


Figure 3.4: Open-loop step response of the extracted transfer function

CHAPTER 4

CONTROLLER DEVELOPMENT AND SYSTEM OPERATION

4.1 Microcontroller System Board Module

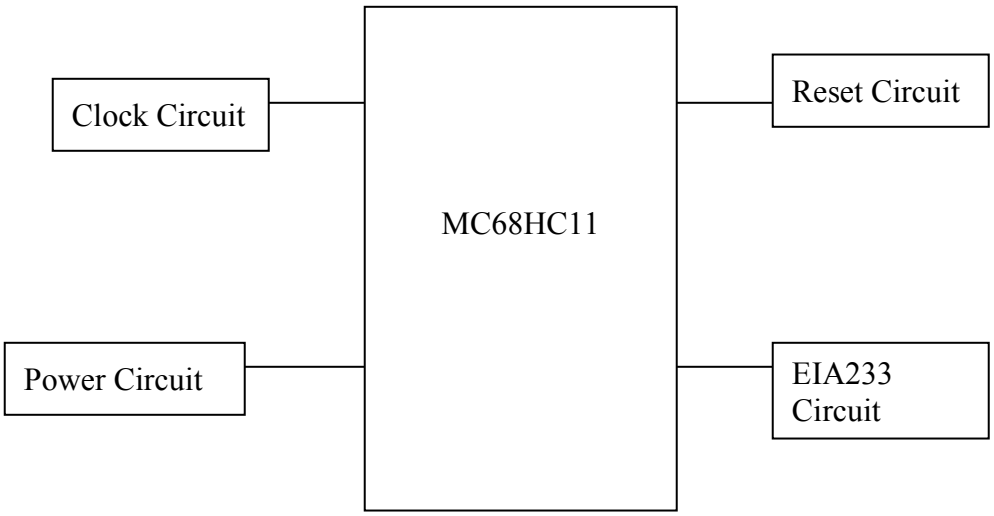


Figure 4.1 : Microcontroller System Board Module

MC68HC11 is chosen to be implemented in this project due to its high performance, high speed, low power consumption, various function and features. Bootstrap mode is chosen to operate in this project because its does not require extra input and output ports. Bootstrap mode allows special purpose programs to be loaded into internal RAM. The system board consists of power circuit, reset circuit, clock circuit and EIA233 module. Power circuit is needed to provide constant 5V voltage to the system. Reset circuit is used to reset microcontroller process. Clock circuit is required to supply constant 2MHz clock speed. EIA 233 module is important to transfer program between PC and micro controller .

One of the advantages of the system is that the MC68HC11 Development Board offers user to use the system in various ways. One of them is to use only system board as shown in Figure 3.7. The user can also use the system board and interface with its own application as shown in Figure 3.8 or to the some extent, user can use both application board and system board to interface with its own application..

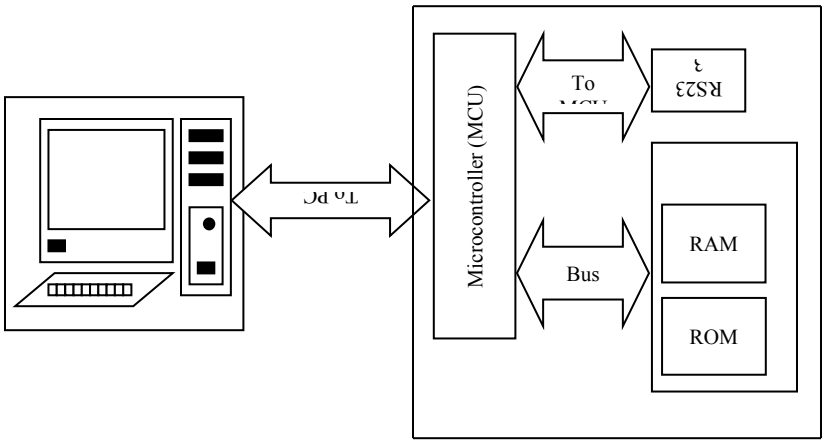


Figure 4.2: System Board

4.2 Direct Current Motor

The model of the DC motor used in this project is CLIFTON PRECISION SERVO MOTOR MODEL JDH-2250-HF-2C-E. Various types of motor are available in the market now, such as DC motor, stepper motor and servo motor. In this project, the speed of DC servo motor is an importance. The speed of a DC servo motor is directly proportional to the supply voltage. The speed controller works by varying the average voltage sent to the motor. Therefore the DC servo motor (Clifton Precision JDH 2250-HF-2C-E) was selected because servo motors have low-inertia armatures that respond quickly to excitation-voltage changes. Servomotors have three wires; usually red, black and white. The red wire is for +VDC, the black for ground and the white is for position control.

Figure 4.3: The DC Motor

4.3 The PID Controller

The derivation of the transfer function of the PID controller looks like the following:

$$K_p + \frac{K_i}{s} + K_d s = \frac{K_d s^2 + K_p s + K_i}{s} \qquad (4.1)$$

K_p = Proportional gain

K_i = Integral gain

K_d = Derivative gain

The PID controller routine is designed by simulating the parameters in the Simulink® MATLAB software.

C++ language MC68HC11 is used to program the MC68HC11 microcontroller. Bootstrap mode is used thus the address must in between $B600 and $B7FF. If there is more program to write and address must be added, then expanded mode will be used in which the address is from $E000 to $FFFF.

4.3.1 PID Routine in MC68HC11

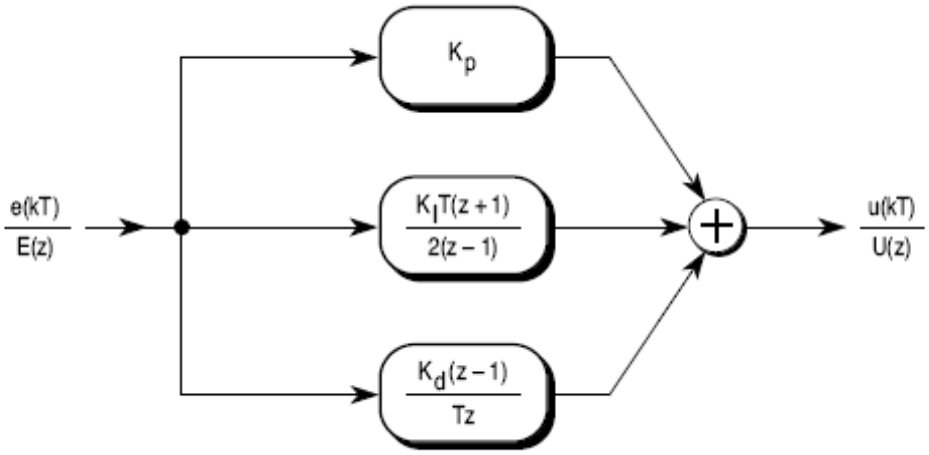


Figure 4.4: PID Flow Diagram

There is a desired setpoint in our process (Gd) and a measurement of the actual value G(t) in time.

Error is: $e(t) = Gd - G(t)$; as e = sp – pv (4.2)

Output correction x(t) for the PID controller is:

$$x(t) = K_P\, e(t) + K_I \int e(t) + K_D \frac{de(t)}{dt} \Big| t = T \quad (4.3)$$

where KP, KI, and KD are constants.

Now, rewriting the integral:

$$x(t) = K_P\, e(t) + K_I \int_{t=0}^{t} [Gd - G(t)]dt + K_D \frac{de(t)}{dt} \Big| t = T \qquad (4.4)$$

Let $A0 = Kp + Ki\,\Delta t + \frac{Kd}{\Delta t}$ Δt = sampling time interval (4.5)

e = set point – process variable

$$A1 = Kp + 2\frac{Kd}{\Delta t} \qquad (4.6)$$

$$A2 = \frac{Kd}{\Delta t}$$

Thus, $\mathbf{c_n = c_{n-1} + A_0 e_n - A_1 e_{n-1} + A_2 e_{n-2}}$ (4.7)

From Equation

We get the PID Digital Equation;

$$C_0 = C_1 + A_0E_0 - A_1E_1 + A_2E_2 \qquad (4.8)$$

Where C_0 = present output (**cn**)

C_1 = previous output (**cn-1**)

E_0 = present error (**en**)

E_1 = previous error (**en-1)**

E_2 = error preceding previous error (**en-2**)

To introduce discrete time, let t = kT where k = 1,2,...,n and T = the sampling and control update period. Now, $t_0 = (k - 1)T$. The integral evaluated from
(k – 1)T to kT can be approximated using the trapezoidal integration rule. The derivative of the error term is simply the rate of change of error.

The form which can be executed directly on the microprocessor is:

$$x(t) = K_P e(t) + K_I \left(Gdt - \frac{T}{2}(G(Kt) + G[(k-1)T])\right) + \frac{K_D}{6T}((e(kT) - e(k-3) + 3(e(k-1) - e(k-2)))$$

This term is added to the current output and put into the PWM control register at the beginning of the next calculation cycle. Substituting the microcode labels for constants and variables into the equation and using C language operator notation (refer appendix) gives:

```
NEWDTY = KP * (ERRX) + KI * PERDT * (CMNDX - (ADRCX + ADRCXM1) / 2) +
         (KD / (6 * PERDT)) * ((ERRX - ERRM3X) + 3 * (ERRM1X - ERRM2X)) +
         OLDDTY
```

(4.9)

The function of the proportional term is clear, but the derivative and integral terms may need a brief explanation. When a system with only proportional control is off the specified setpoint, the controller will increase the control voltage until the error signal is zero, and the system thus returns to the setpoint with more applied voltage than is required for maintaining equilibrium. This causes overshoot and, as the process continues, under-damped ringing. The derivative term contributes proportionally to the error rate of change, but with the opposite sign of the proportional term. If the proper constants are chosen, critical damping can be achieved. The role of the integral term is to eliminate steady state error. A system that has a steady state error when tracking a ramping input function can use an integral term to integrate the error over time and compensate for it.

4.4 **C Language Implementation** (refer to Appendix B for a complete C listing).

The equation (C Language operator notation) :-

```
NEWDTY = KP * (ERRX) + KI * PERDT * (CMNDX - (ADRCX + ADRCXM1) / 2) +
         (KD / (6 * PERDT)) * ((ERRX - ERRM3X) + 3 * (ERRM1X - ERRM2X)) +
         OLDDTY
```

(4.10)

After necessary files are included and floating point variables are declared, a prototype is given to define an assembly language function that is used later. The main program initializes constants and variables, sets up the required on-chip peripherals, and waits for interrupts to occur. The M68HC11 real-time interrupt (RTI) is used to establish a precise time base for performing PID compensation. The period (T or PERDT) is determined by the RTI rate.

The RTI_interrupt function does the PID loop PWM duty cycle calculations, performs I/O using the DOIO assembly language function, and checks the results against the usable PWM output range of $00 to $FF. If a floating-point result is out of range, the closest limit is substituted. This “saturation arithmetic” prevents out-of-range results from causing sign-reversals in the PWM output.

When the RTI_interrupt function returns control to the C routine, the last task the routine must perform is preparation for the next period. A two-element pipeline of the A/

D reading and a four-element error pipeline are updated. Finally the “old” duty cycle value is copied into OLDDTY.

4.5 Pseudocode for the PID Controller Algorithm

```
        define constants A0, A1, A2
        define SP
        initialize E1, E2 = 0
        input PV from A/D subsystem
        C1 =  factor pv
        sample-time delay

Loop
        output C1 to D/A or PWM subsystem
        input PV from A/D subsystem
                        ; calculate PID sample output
        E0 = SP – PV  ; sample error
        C0 = C1 + A0 E0 – A1 E1 + A2 E2
                        ; now reassign variables in scratchpad RAM
        C1 = C0
        E2 = E1
        E1 = E0
        sample-time delay
        repeat loop
```

Note that the time delay can be triggered using an output-compare interrupt or real-time interrupt. The service routine can include the rest of the loop. The pseudocode above does not take into account the bit size limits of the numbers. Controller output must fall between maximum and minimum limits to avoid overflow and underflow resulting from the calculations. In other words, the outputs are clamped. This is analogous to the analogue PID, whose outputs are naturally clamped due to the high and low limits of the power supply. The C code will include clamping. Refer to Appendix B for full program listing.

CHAPTER 5

RESULT AND TESTING

5.1 PID Tuning

When the system dynamics are not precisely known, we must resort to experimental approaches. Ziegler-Nichols Rules for Tuning PID Controller:
Using only Proportional control, the gain of the system is turned up until the system oscillates without dying down, i.e., is marginally stable. Assume that K and P are the resulting gain and oscillation period, respectively.

Then, use

for P control	for PI control	for PID control
Kp = 0.5 K	K_p = 0.45 K Ki = 1.2 / P Ki = 1.2 / P	Kp = 0.6 K Ki = 2.0 / P Kd = P / 8.0

Table 5.1: The PID tuning equation

Controller Type	Kp	Ki	Kd
P	8	0	0
PI	7.2	400	0
PID	9.6	666.67	0.000375

Table 5.2: Ziegler-Nichols Tuned Parameter, Frequency Response.

A proportional controller (Kp) will have the effect of reducing the rise time and will reduce, but never eliminate, the steady-state error. An integral control (Ki) will have the effect of eliminating the steady-state error, but it may make the transient response worse. A derivative control (Kd) will have the effect of increasing the stability of the system, reducing the overshoot, and improving the transient response. The transient responses obtained are shown in this chapter.

5.2 Model of the PID Controller

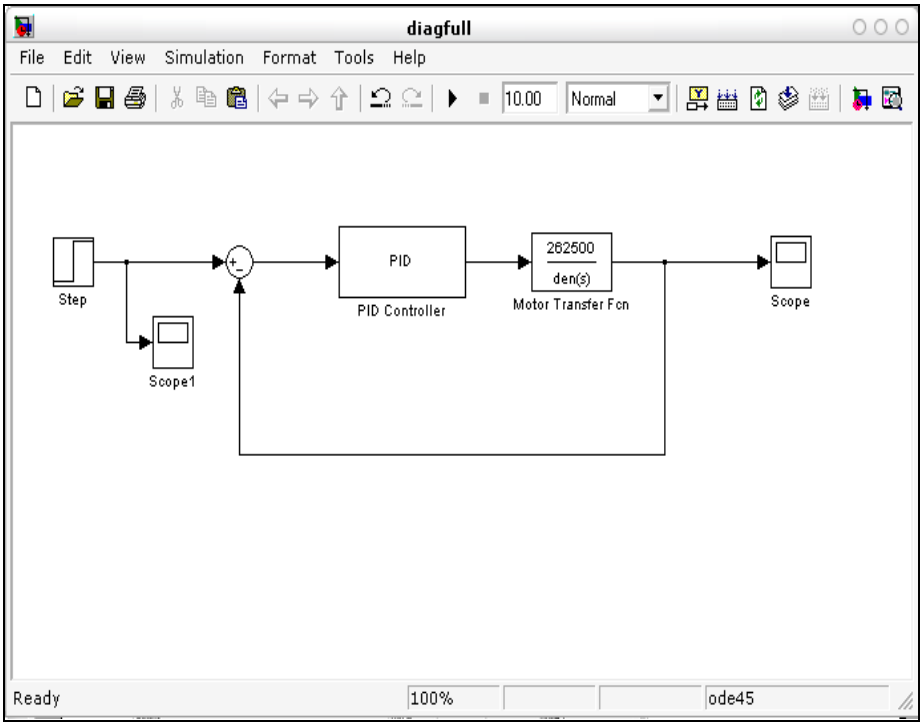


Figure 5.1: Model of PID Controller in MATLAB

The model of the PID controller is design by using the transfer function obtained earlier. The PID controller block is available in the MATLAB library, and the values obtained from the PID tuning are inserted when the PID block is double-clicked.

5.3 The Transient Responses

5.3.1 Response of Speed without any Controller

(m/s)

(s)

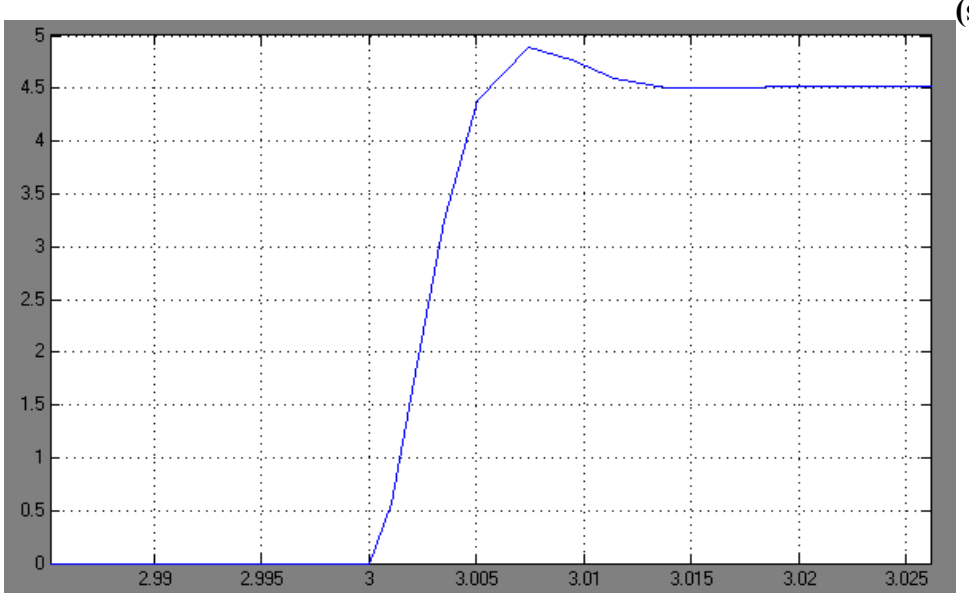


Figure 5.2: Transient Response of DC Motor Speed without Controller

5.3.1.1 Graph Analysis:

Percentage of Overshoot, %OS is 2%, which is obtained from the graph. Percentage of overshoot is the amount that the waveform overshoots the steady state, or final, value at the peak time, expressed as a percentage of the steady state value. The Peak Time, Tp is

3.0075 seconds, which is the time required to reach the first, or maximum, peak. The Rise Time, Tr is 0.0040 seconds, which is the time required for the waveform to go from 0.1 of the final value to 0.9 of the final value. The Settling Time, Ts, is 3.0130 seconds, which is the time required for the transient's damped oscillation to reach and stay within 2% of the steady-state.

5.3.2 Response of Speed with a Proportional Controller

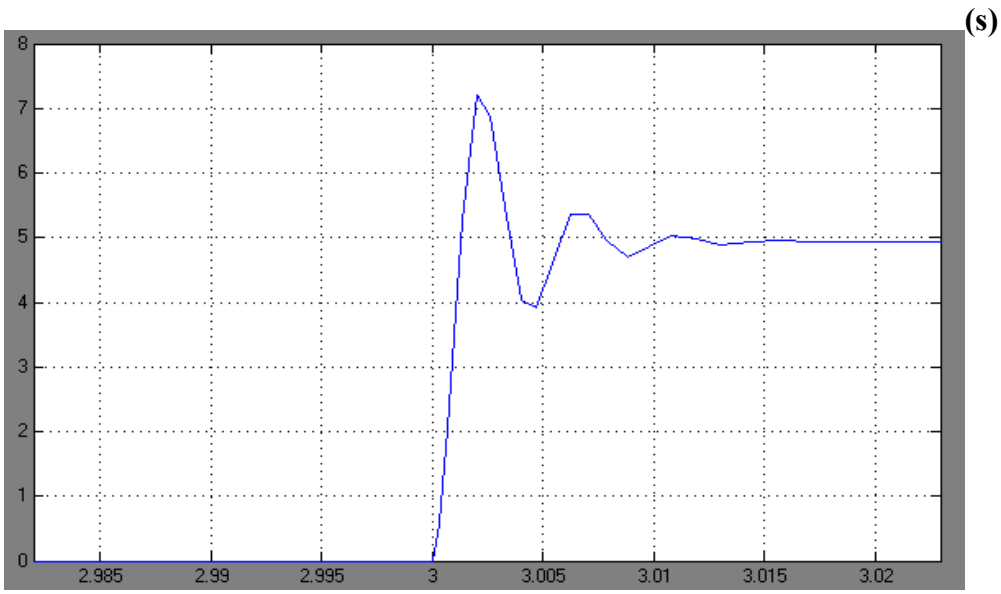


Figure 5.3: Transient Response of DC Motor Speed with a P Controller

5.3.2.1 Graph Analysis:

Percentage of Overshoot, %OS is 47%, which is obtained from the graph. Percentage of overshoot is the amount that the waveform overshoots the steady state, or final, value at the peak time, expressed as a percentage of the steady state value. The Peak Time, Tp is 3.0025 seconds, which is the time required to reach the first, or maximum, peak. The

Rise Time, Tr is 0.0010 seconds, which is the time required for the waveform to go from 0.1 of the final value to 0.9 of the final value. The Settling Time, Ts, is 3.0135seconds, which is the time required for the transient's damped oscillation to reach and stay within 2% of the steady-state.

5.3.3 Response of Speed with a Proportional-Integral Controller

(m/s)

(s)

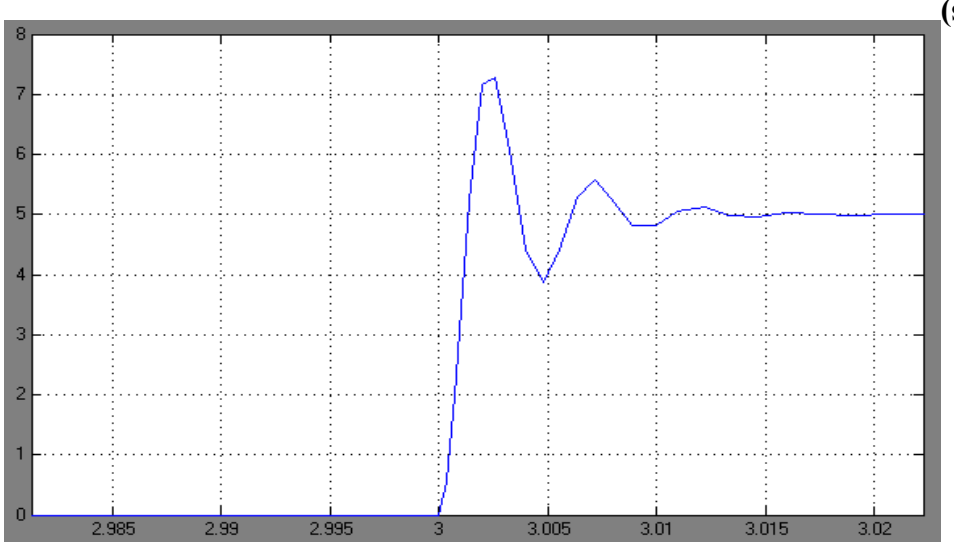


Figure 5.4: Transient Response of DC Motor Speed with a PI Controller

5.3.3.1 Graph Analysis:

Percentage of Overshoot, %OS is 46%, which is obtained from the graph. Percentage of overshoot is the amount that the waveform overshoots the steady state, or final, value at the peak time, expressed as a percentage of the steady state value. The Peak Time, Tp

is 3.0030 seconds, which is the time required to reach the first, or maximum, peak. The Rise Time, Tr is 0.0009 seconds, which is the time required for the waveform to go from 0.1 of the final value to 0.9 of the final value. The Settling Time, Ts, is 3.0120 seconds, which is the time required for the transient's damped oscillation to reach and stay within 2% of the steady-state.

5.3.4 Response of Speed with a Proportional Integral Derivative Controller

Figure 5.5: Transient Response of DC Motor Speed with a PID Controller

5.3.4.1 Graph Analysis:

Percentage of Overshoot, %OS is 48%, which is obtained from the graph. Percentage of overshoot is the amount that the waveform overshoots the steady state, or final, value at the peak time, expressed as a percentage of the steady state value. The Peak Time, Tp

is 3.0025 seconds, which is the time required to reach the first, or maximum, peak. The Rise Time, Tr is 0.0007 seconds, which is the time required for the waveform to go from 0.1 of the final value to 0.9 of the final value. The Settling Time, Ts, is 3.0090 seconds, which is the time required for the transient's damped oscillation to reach and stay within 2% of the steady-state.

5.4 Discussion

In summary;

Type of Controller	%OS	Peak Time, Tp	Rise Time, Tr	Settling Time, Ts
Uncontrolled	2%	3.0075s	0.0040s	3.0130s
P	47%	3.0025s	0.0010s	3.0135s
PI	46%	3.0030s	0.0009s	3.0120s
PID	48%	3.0025s	0.0007s	0.0090s

Table 5.: Summation of Graph Analysis

PID is the best controller to be used for this system. The response shows a better performance where the rise time (Tr) and the settling time (Ts) are lower than other controllers. The %OS is a little bit higher than other controller, as the Ziegler-Nichols Method produce more overshoot compared to other methods.

Chapter 6

CONCLUSION AND FUTURE RECOMMENDATION

6.1 Conclusion

The mathematical model of the DC Motor system to be controlled with the purpose of simulating its dynamic behavior in closed-loop mode can be developed in advanced using the Matlab Simulink®.

A PID Controller algorithm has developed using the same method, by means of keeping the speed of DC motor under control, at the same time reducing the error that may occur.

6.2 Suggestion

6.2.1 Microcontroller

Microcontrollers that had full-duplex CAN communication capability may be chosen for real-time observation. The designed program used in this project is based on the functions and addresses of the K series. Thus, for real time observation, the K series of the MC68HC11 may be used, due to the PWM, and which has more than 1Mbyte memory space. Large memory size will assist in the programming of a better PID algorithm into the microcontroller.

6.2.2 Tuning Method

For future advancements of this project, other PID tuning methods may also be used, to compare the outcome of the controlled speed.

6.3 Costing and Commercialization

This project focused on designing the program for the PID Controller. Thus it does not involve any hardware, except for the DC motor, as the parameter is needed for the derivation of the transfer function in the beginning of the project. The datasheet for the microcontroller is needed as well to know the configuration which is used in the program. These datasheet are available online, from the Motorola official online resource. Therefore, there is no cost involve. This project is considered an initial research, and it is recommended that any future project is based on this research, to apply it onto real observation. When applied onto real application, this project has a good potential to be commercialized in the future.

REFERENCES

[1] K.J. Astrom and T. Hagglünd, . (1988). Automatic tuning of simple regulators with specification on phase and amplitude margin, pp. 645–651.

[2] K.J. Astrom and T. Hagglünd, (1995). PID Controller. (2nd ed.), Instrument of Society of America, Research Triangle Park, NC

[3] I.-L. Chien and P.S. Fruehauf, (1990) Consider IMC Tuning to improve controller performance. *Chem. Eng. Progr.* 86, pp. 33–41.

[4] G.F. Franklin, J.D. Powell and A.E. Naeini, (1986). Feedback Control of Dynamic Systems. , Addison-Wesley, Reading, MA

[5] M. Ge, M.-S. Chiu and Q.-G. Wang, (2002). Robust PID controller design via LMI approach. *J. Process Contr.* **1** 12 , pp. 3–13

[6] C.C. Hang, K.J. Astrom and Q.G. Wang, (2002). Relay feedback auto-tuning of process controllers-a tutorial review. *J. Process Contr.* **1** 12, pp. 143–162

[7] W.K. Ho, C.C. Hang and L.S. Cao, (1995). Tuning of PID controller based on gain and phase margin specification, pp. 497–502.

[8] C. Hwang and C.-Y. Hsiao, (2002). Solution of a non-convex optimization arising in pi/pid control design. pp. 143–162.

[9] M. Morari and E. Zafirou, (1989). Robust Process Control. , Prentice-Hall, Englewood Cliffs, NJ.

[10]Maurice, B. (1998)."ST62 microcontrollers drive home appliance motor technology, AN885/1196," Application Note, ST Microelectronics, www.st.com.

[11]Katausky, J., I. Horder, and L. Smith. "Analog/Digital Processing with Microcontrollers," AR-526 Applications Engineers, Intel Corporation, www.intel.com.

[12]Hitex. "Basic DC Motor Speed Control With The Infineon C167 Family." Hitex: UK. www.hitex.co.uk/c166/pidex.html.

APPENDIX A

Data Sheets

MOTOROLA
SEMICONDUCTOR
TECHNICAL DATA

Order this document
by MC68HC11A8TS/D

MC68HC11A8
MC68HC11A1
MC68HC11A0

Technical Summary
8-Bit Microcontrollers

1 Introduction

The MC68HC11A8, MC68HC11A1, and MC68HC11A0 high-performance microcontroller units (MCUs) are based on the M68HC11 Family. These high speed, low power consumption chips have multiplexed buses and a fully static design. The chips can operate at frequencies from 3 MHz to dc. The three MCUs are created from the same masks; the only differences are the value stored in the CONFIG register, and whether or not the ROM or EEPROM is tested and guaranteed.

For detailed information about specific characteristics of these MCUs, refer to the *M68HC11 Reference Manual* (M68HC11RM/AD).

1.1 Features

- M68HC11 CPU
- Power Saving STOP and WAIT Modes
- 8 Kbytes ROM
- 512 Bytes of On-Chip EEPROM
- 256 Bytes of On-Chip RAM (All Saved During Standby)
- 16-Bit Timer System
  - — 3 Input Capture Channels
  - — 5 Output Compare Channels
- 8-Bit Pulse Accumulator
- Real-Time Interrupt Circuit
- Computer Operating Properly (COP) Watchdog System
- Synchronous Serial Peripheral Interface (SPI)
- Asynchronous Nonreturn to Zero (NRZ) Serial Communications Interface (SCI)
- 8-Channel, 8-Bit Analog-to-Digital (A/D) Converter
- 38 General-Purpose Input/Output (I/O) Pins
  - — 15 Bidirectional I/O Pins
  - — 11 Input-Only Pins and 12 Output-Only Pins (Eight Output-Only Pins in 48-Pin Package)
- Available in 48-Pin Dual In-Line Package (DIP) or 52-Pin Plastic Leaded Chip Carrier (PLCC)

Figure 3 48-Pin DIP Pin Assignments

2 Operating Modes and Memory Maps

In single-chip operating mode, the MC68HC11A8 is a monolithic microcontroller without external address or data buses.

In expanded multiplexed operating mode, the MCU can access a 64 Kbyte address space. The space includes the same on-chip memory addresses used for single-chip mode plus external peripheral and memory devices. The expansion bus is made up of ports B and C and control signals AS and R/$\overline{W}$. The address, R/$\overline{W}$, and AS signals are active and valid for all bus cycles including accesses to internal memory locations. The following figure illustrates a recommended method of demultiplexing low-order addresses from data at port C.

Features

- Fast Read Access Time - 120 ns
- Fast Byte Write - 200 µs or 1 ms
- Self-Timed Byte Write Cycle
  - Internal Address and Data Latches
  - Internal Control Timer
  - Automatic Clear Before Write
- Direct Microprocessor Control
  - READY/BUSY Open Drain Output
  - DATA Polling
- Low Power
  - 30 mA Active Current
  - 100 µA CMOS Standby Current
- High Reliability
  - Endurance: 10^4 or 10^5 Cycles
  - Data Retention: 10 Years
- 5V ± 10% Supply
- CMOS and TTL Compatible Inputs and Outputs
- JEDEC Approved Byte-Wide Pinout
- Commercial and Industrial Temperature Ranges

64K (8K x 8) CMOS E^2PROM

Description

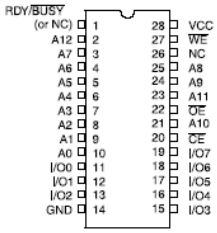
The AT28C64 is a low-power, high-performance 8,192 words by 8 bit nonvolatile Electrically Erasable and Programmable Read Only Memory with popular, easy to use features. The device is manufactured with Atmel's reliable nonvolatile technology.

(continued)

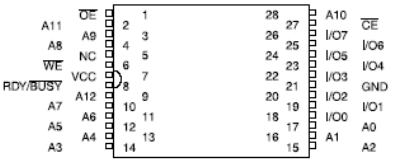
Pin Configurations

Pin Name	Function
A0 - A12	Addresses
$\overline{CE}$	Chip Enable
$\overline{OE}$	Output Enable
$\overline{WE}$	Write Enable
I/O0 - I/O7	Data Inputs/Outputs
RDY/$\overline{BUSY}$	Ready/$\overline{Busy}$ Output
NC	No Connect
DC	Don't Connect

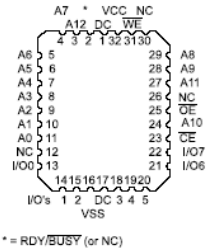
PDIP, SOIC
Top View



TSOP
Top View



LCC, PLCC
Top View



* = RDY/$\overline{BUSY}$ (or NC)

Note: PLCC package pins 1 and 17 are DON'T CONNECT.

0001G

MOTOROLA
SEMICONDUCTOR
TECHNICAL DATA

Order this document
by MC68HC11KTS/D

M68HC11 K Series

Technical Summary
8-Bit Microcontroller

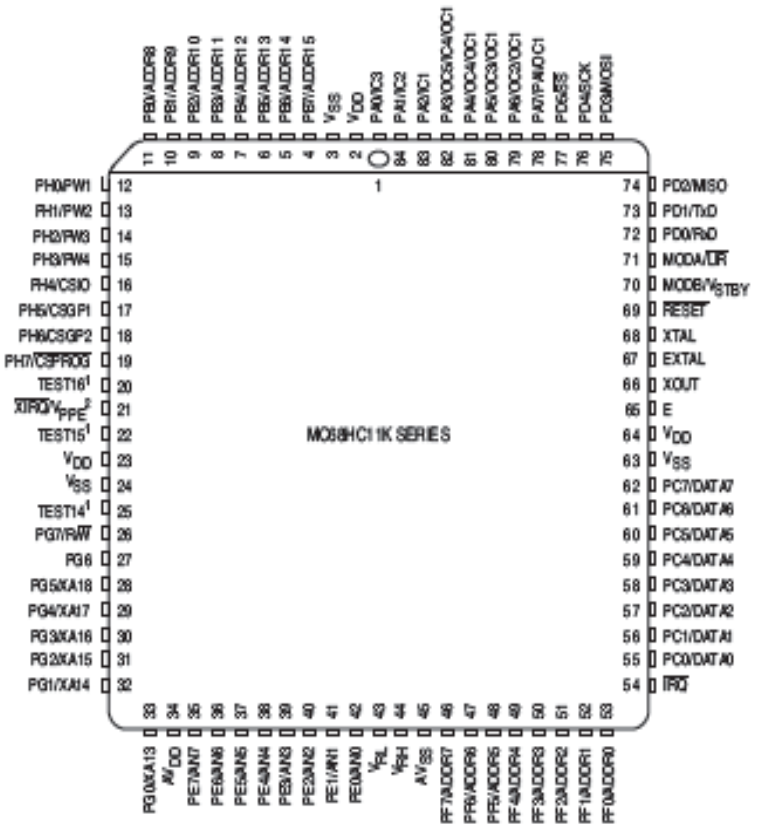
The M68HC11 K-series microcontroller units (MCUs) are high-performance derivatives of the MC68HC11F1 and have several additional features. The MC68HC11K0, MC68HC11K1, MC68HC11K3, MC68HC11K4 and MC68HC711K4 comprise the series. These MCUs, with a nonmultiplexed expanded bus, are characterized by high speed and low power consumption. Their fully static design allows operation at frequencies from 4 MHz to dc.

This document contains information concerning standard, custom-ROM, and extended-voltage devices. Standard devices include those with disabled ROM (MC68HC11K1), disabled EEPROM (MC68HC11K3), disabled ROM and EEPROM (MC68HC11K0), or EPROM replacing ROM (MC68HC711K4). Custom-ROM devices have a ROM array that is programmed at the factory to customer specifications. Extended-voltage devices are guaranteed to operate over a much greater voltage range (3.0 Vdc to 5.5 Vdc) at lower frequencies than the standard devices. Refer to the device ordering information tables for details concerning these differences.

1 Features

- M68HC11 CPU
- Power Saving STOP and WAIT Modes
- 768 Bytes RAM (All Saved During Standby)
- 24 Kbytes ROM or EPROM
- 640 Bytes Electrically Erasable Programmable Read Only Memory (EEPROM)
- Optional Security Feature Protects Memory Contents
- On-Chip Memory Mapping Logic Allows Expansion to Over 1 Mbyte of Address Space
- PROG Mode Allows Use of Standard EPROM Programmer (27C256 Footprint)
- Nonmultiplexed Address and Data Buses
- Four Programmable Chip Selects with Clock Stretching (Expanded Modes)
- Enhanced 16-Bit Timer with Four-Stage Programmable Prescaler
  - — Three Input Capture (IC) Channels
  - — Four Output Compare (OC) Channels
  - — One Additional Channel, Selectable as Fourth IC or Fifth OC
- 8-Bit Pulse Accumulator
- Four 8-Bit or Two 16-Bit Pulse Width Modulation (PWM) Timer Channels
- Real-Time Interrupt Circuit
- Computer Operating Properly (COP) Watchdog
- Clock Monitor
- Enhanced Asynchronous Nonreturn to Zero (NRZ) Serial Communications Interface (SCI)
- Enhanced Synchronous Serial Peripheral Interface (SPI)
- Eight-Channel 8-Bit Analog-to-Digital (A/D) Converter
- Seven Bidirectional Input/Output (I/O) Ports (54 Pins)
- One Fixed Input-Only Port (8 Pins)
- Available in 84-Pin Plastic Leaded Chip Carrier (PLCC), 84-Pin Windowed Ceramic Leaded Chip Carrier (CLCC), and 80-Pin Quad Flat Pack (QFP)

This document contains information on a new product. Specifications and information herein are subject to change without notice.



1. Pins 20, 22, and 25 are used only during factory testing and should not be connected to external circuitry.
2. V_{PPE} applies only to devices with EPROM.

Figure 1 Pin Assignments for 84-Pin PLCC/CLCC

APPENDIX B

Program Listing

```
/*this program initializes constants and variables,
  sets up required on-chip peripherals, and waits for interrupts to occur.*/
/*uses RTI to establish precise time base for performing the PID routine.*/
/*uses RTI to do PID loop PWM duty cycle calculations*/
/*uses RTI to to perform I/O using DOIO assembly language function*/



#include <stdio.h>              /*Provides the core input and output capabilities of the C
                                  language*/
#include <io6811k4.h>
#include <int6811k.h>
#include <math.h>                       /*to add floating point math, when
                                  compiling in any C compilers*/

zpage unsigned int TOFCOUNT;                    /* declare variables */
zpage float CMNDVX;
zpage float ADRCX;
zpage float ADRCXM1;
zpage float ADRCXM2;
zpage float ADRCXM3;
zpage float ERRX;
zpage float ERRM1X;
zpage float ERRM2X;
zpage float ERRM3X;
zpage float PERDT;
zpage float NEWDTY;
zpage float OLDDTY;
zpage float KP;
zpage float KD;
zpage float KI;
```

```
extern int ASSEM (void);              /* prototype for assembly routine.*/
                                      /* any void statements indicate that no arguments
are being
                                        passed to or from the interrupt routine.*/
                                      /* declaration that are not definition. */

void main()                           /* main program. void means function do not
return a value*/
{
CMNDVX = 1.5;
PERDT = 0.016383;   /* RTI and therefore PID loop period = 16.383 ms.*/
                              /* value is different if being used in real
                              applications,depends on performance. */
KP = 9.6;
KI = 666.67;
KD = 0.000375;
OLDDTY = 1.9;                 /* start out with pwm set fairly high */
PORTA = 0x00;                 /* this will be used for a scope trigger */
DDRA = 0xFF;                          /* set PORTA as output */
PACTL = 0x03;         /* Pulse Accumulator Control. set RTI to 16.383 ms  */
TMSK2 = 0x40;               /* Timer Interrupt Mask. enable RTI interrupts */
OPTION = 0x90;                        /* System Configuration Option */
                              /* enable A/D charge pump.The (A/D) converter
                system uses an all-capacitive charge-redistribution technique to convert
                analog signals to digital values. */
PWPER1 = 0xFF;       /* Pulse-Width Modulation Timer Period 1.Determines period of
                     associated PWM channel. set up PWM channel 1 at 15.625 kHz */
PWDTY1 = 0xFF;       /* Pulse-Width Modulation Timer Duty Cycle 1. Determines duty
                              cycle of associated PWM channel.*/
```

```
PWPOL = 0x01;                       /* Pulse-Width Modulation Timer Polarity
*/
DDRH = 0x00;
PWEN = 0x01;                        /* Pulse-Width Modulation Timer Enable
*/
TFLG2 = 0x40;                       /* Timer Interrupt Flag */
enable_interrupt();                 /* wait here for RTI to cause loop execution */
wait_for_interrupt();
for (;;) {
; }
}
interrupt void IRQ_interrupt(void)  /* Interrupt Request.should initialize all
                                    interrupts...refer to either the act of interrupting the
                                    bus lines used to signal an interrupt */
{
PORTA = 0xFF;
PORTA = 0x00;
}
interrupt void TO_interrupt(void)
{
TOFCOUNT++ ;
}

interrupt void RTI_interrupt(void)  /*PID LOOP/PWM routine. declare as an interrupt
                                    function istead of a normal subroutine. */
{
PORTA = 0xFF;                       /* scope strobe */
ASSEM ();                           /* read A to D and output the duty cycle calculated
                                    last period */
ADCTL = 0x10;                       /* begin new conversion cycle */
ERRX = (CMNDVX - ADRCX);            /* calculate current error */
```

```
/* The statement below is the entire floating point PID loop */

NEWDTY = KP*(ERRX) + KI*PERDT*(CMNDVX - (ADRCX + ADRCXM1)/2) +
(KD/(6*PERDT))*((ERRX - ERRM3X) + 3*(ERRM1X - ERRM2X)) + OLDDTY;

if (NEWDTY > 1.99609)                    /* test for result being in usable */
NEWDTY = 1.99609;                /* limits and set PWM duty cycle if */

else if (NEWDTY < 1.0)                   /* beyond saturation */
NEWDTY = 1.0;
TFLG2 = 0x40;                            /* clear RTI flag */
ADRCXM1 = ADRCX;                 /* update A/D result for next cycle */
ERRM3X = ERRM2X;                         /* update error pipeline */
ERRM2X = ERRM1X;
ERRM1X = ERRX;
OLDDTY = NEWDTY;        /* update old duty cycle for next calculation   period */
PORTA = 0x00;                            /* scope strobe. strobe is a spot of higher
                        than normal intensity in the sweep of an indicator  on a
                        scanning device (oscilloscope), as on a radar screen, used
                        as a reference mark for determining the position or
                        distance of the object scanned or detected. */
}
```

```
*******************************************************
* ASSEM assembly function                             *
* This routine handles the conversion between         *
* 8 bit register values and the C float variables     *
*******************************************************
        MODULE      ASSEM
        PUBLIC      ASSEM

        P68H11
        RSEG CODE

PWDTY1      set     $006c                   REGISTER LOCATIONS
ADDR1       set     $0031
            EXTERN      ADRCX:ZPAGE         EXTERNAL  VARIABLE
                                                        LOCATIONS
            EXTERN      NEWDTY:ZPAGE
ASSEM:
        LDAA  #$3F
        CLRB
        STD   ADRCX             INITIALIZE FLOAT LOCATION.
        LDAA  ADR1              GET CHANNEL 1 A/D RESULT.
        LSRD                    SHIFT TO FLOAT MANTISSA POSITION.
        ORAA  #$80              OR IN LEAST SIGNIFICANT EXP BIT
        STD   ADRCX+1           AND STORE IT IN FLOAT VARIABLE.
        CLRB                    CLEAR LEAST SIGNIFICANT
        STAB  ADRCX+3           FLOAT BYTE.
        LDD   NEWDTY+1          GET TWO BYTES OF FLOAT MANTISSA.
        LSLD                    SHIFT TO CORRECT REGISTER
                                                        POSITION.
        STAA  PWDTY1                OUTPUT TO PWM DUTY
                                                        REGISTER.
        RTS
        END
```

```
NEWDTY = KP*(ERRX) + KI*PERDT*(CMNDVX - (ADRCX + ADRCXM1)/2)
+ (KD/(6*PERDT))*((ERRX - ERRM3X) + 3*(ERRM1X - ERRM2X))
+ OLDDTY
```