

UNIVERSITI MALAYSIA PAHANG

BORANG PENGESAHAN STATUS TESIS♦

JUDUL: **A DC MOTOR CONTROLLER USING PID ALGORITHM
IMPLEMENTATION ON PIC**

SESI PENGAJIAN: 2008/2009

Saya WAN ROBAH BINTI W AHMAD (860602-11-5164)
(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:

**LOT 3353 KAMPUNG FIKRI PERPINDAHAN
JALAN MAK LAGAM 24000 KEMAMAN
TERENGGANU**

HASZURAIDAH BINTI ISHAK
(Nama Penyelia)

Tarikh: **17 NOVEMBER 2008**

Tarikh: : **17 NOVEMBER 2008**

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).

“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 : HASZURAI DAH BINTI ISHAK

Date : 17 NOVEMBER 2008

A DC MOTOR CONTROLLER USING PID ALGORITHM
IMPLEMENTATION ON PIC

WAN ROBAAH BINTI W AHMAD

This thesis is submitted as partial fulfillment of the requirements for the award of the
Bachelor of Electrical Engineering (Hons.) (Electronics)

Faculty of Electrical & Electronics Engineering
Universiti Malaysia Pahang

NOVEMBER 2008

“All the trademark and copyrights use herein are property of their respective owner. References of information from other sources are quoted accordingly; otherwise the information presented in this report is solely work of the author.”

Signature : _____

Author : WAN ROBAAH BINTI W AHMAD

Date : 17 NOVEMBER 2008

Dedicated to my beloved parent, brothers, sisters,
niece and to all Malaysians.

ACKNOWLEDGEMENT

In the name of Allah, invocation and greetings to adoration of Nabi Muhammad (S.A.W.), thanks to God because giving me strength and patience in finishing this Project Bachelor. In particular, I wish to express my sincere appreciation to my supervisor, Madam Haszuraidah binti Ishak, for encouragement, guidance, critics and friendship. My fellow friends under the same supervisor should also be recognized for their support and ideas. In addition, my sincere appreciation also extends to all my colleagues and others who have provided assistance at various occasions. Their views and tips are useful indeed. Unfortunately, it is not possible to list all of them in this limited space. I sincerely appreciated all of the efforts and precious time to be spent together in making this final year project educational, enjoyable and memorable. And last but not least, I am grateful to all my family members for their moral and financial support. Thank you.

ABSTRACT

This project is about controlling the speed of DC servo motor by using Proportional-Integral-Derivative (PID) algorithm then implemented on Peripheral Interface Circuit (PIC) microcontroller. The main objective of this project is to control the speed of DC servo motor at the demanded speed or to drive the motor at that speed. The speed of a DC motor usually is directly proportional to the supply voltage. So, if we reduce the supply voltage from 12 Volts to 6 Volts, the motor will run at half the speed. It could be achieved by simply adjusting the voltage sent to the motor, but this is quite inefficient to do. So, A PID controller becomes the best way to overcome this problem. PID attempts to correct the error between a measured process variable and a desired setpoint by calculating and then outputting a corrective action that can adjust the process accordingly. In this project, the PID algorithm that is added to the system becomes a closed loop system. A simulation using MATLAB software is implemented to tune PID algorithm by changing the value of Proportional gain, K_p , Integral gain, K_i and Derivative gain, K_d to get a speed of the motor which is less overshoot and increase settling time. Then, a PIC microcontroller is programmed by adding the value of tuned PID algorithm to control the speed of DC servo motor. At the end of the project, the speed of the DC servo motor should be maintain even the supply voltage is varied.

ABSTRAK

Projek ini adalah mengenai mengawal kelajuan motor DC servo dengan menggunakan algorithm “Proportional-Integral-Derivative (PID)” yang kemudiannya diimplementasikan pada “PIC microcontroller”. Objektif utama projek ini ialah mengawal kelajuan motor DC servo pada kelajuan yang dikehendaki atau memandu motor tersebut pada kelajuan yang dikehendaki. Kelajuan motor DC biasanya berkadar langsung dengan bekalan kadar voltan. Oleh itu, jika kita mengurangkan bekalan kadar voltan daripada 12 voltan kepada 6 voltan, motor juga akan bergerak pada separuh daripada kelajuan tersebut. Kelajuan motor boleh dicapai dengan mengubah kadar voltan yang dihantar kepada motor tetapi ia tidak cekap untuk dilakukan. Oleh itu, pengawal PID merupakan jalan terbaik untuk mengatasi masalah ini. PID berusaha untuk memperbetul kesilapan diantara perubahan proses yang telah diukur dan titik rujukan yang dikehendaki dengan mengira dan kemudian mengeluarkan satu tindakan pembetulan yang boleh mengubah proses dengan sewajarnya. Dalam projek ini, algorithm PID yang ditambah pada sistem menjadi kitaran tertutup. Satu simulasi menggunakan MATLAB perisian yang diimplementasikan untuk menala algorithm PID dengan mengubah nilai pemalar “Proportional”, K_p , pemalar “Integral”, K_i dan pemalar “Derivative”, K_d untuk mendapat kelajuan motor yang mengurangkan “overshoot” dan meningkatkan “settling time”. Kemudian, “PIC microcontroller” diprogramkan dengan menambah nilai algorithm PID yang telah ditala untuk mengawal kelajuan motor. Pada akhir projek ini, kelajuan motor DC servo sepatutnya tetap walaupun bekalan voltan bertambah.

TABLE OF CONTENTS

CHAPTER	TITLE	PAGE
	DECLARATION	ii
	DEDICATION	iii
	ACKNOWLEDGEMENT	iv
	ABSTRACT	v
	ABSTRAK	vi
	TABLE OF CONTENT	vii
	LIST OF TABLES	x
	LIST OF FIGURES	xi
	LIST OF APPENDICES	xiii
 1	 INTRODUCTION	
	1.1 Background	1
	1.2 Problem Statement	2
	1.3 Project Objective	3
	1.4 Scope of Project	3

2**LITERATURE REVIEW**

2.1	Servo Motor	4
2.2	Microcontroller (PIC 16F84)	5
2.2.1	Pin description	7
2.2.2	Central Processing Unit	9
2.2.3	Status Register	9
2.2.4	Addressing Mode	10
2.2.5	Applications	12
2.3	Proportional-Integral-Derivative (PID) Controller	13
2.4	Ziegler-Nichols Method	14

3**METHODOLOGY**

3.1	Modeling Servo Motor	18
3.2	Ziegler Tuning Method	20
3.3	Hardware Development	21
3.3.1	DC Servo motor	21
3.3.2	PIC 16F84	23
3.3.3	Power Supply	24
3.4	Software Development	24
3.4.1	PID Method	25
3.4.2	Program the PIC Microcontroller	30
3.5	Circuit Diagram	34
3.6	Final Prototype	35

4	RESULT AND ANALYSIS	
4.1	No Controller	38
4.2	Proportional Controller	40
4.3	Proportional-Integral Controller	43
4.4	Proportional-Derivative Controller	46
4.5	Proportional-Integral-Derivative Controller	49
5	CONCLUSION AND RECOMMENDATION	
5.1	Conclusion	55
5.2	Recommendation	56
5.3	Commercialization	56
5.4	List and Cost of the Component	57
	REFERENCES	58
	APPENDIX A	60
	APPENDIX B	63
	APPENDIX C	65
	APPENDIX D	75
	APPENDIX E	77
	APPENDIX F	87

LIST OF TABLES

TABLE NO.	TITLE	PAGE
2.1	Pins on PIC16F84 microcontroller	8
2.2	Estimation value for gain, reset and derivative	15
3.1	Ziegler Tuning Table	20
3.2	Range value of controller	21
3.3	Parameters of motor	22
4.1	No controller	38
4.2	Comparison of Proportional controller	42
4.3	Comparison of Proportional-integral controller	45
4.4	Comparison of Proportional-derivative controller	48
4.5	Comparison of PID controller	53
5.1	List and cost of components	57

LIST OF FIGURES

FIGURE NO.	TITLE	PAGE
2.1	PIC 16F84 outline	7
2.2	Pin number of PIC 16F84	8
2.3	Status Register	9
2.4	Direct addressing	10
2.5	Indirect addressing	11
2.6	A block diagram of a PID controller	14
2.7	Oscillations with constant amplitude	15
3.1	Project procedure	16
3.2	Clifton Precision JDH 2250-HF-2C-E	22
3.3	PIC 16F84 microcontroller circuit	23
3.4	Power Supply circuit	24
3.5	System before using PID controller	25
3.6	System with PID controller	25
3.7	Designed using m-file	27
3.8	Typing program	27
3.9	Changing the value	28
3.10	Closed loop system	28
3.11	Save and run	29
3.12	Programming of the system	30
3.13	Select for PIC16F84A	31

3.14	Setup for PLL	31
3.15	Programming selected	32
3.16	Verifying the program	33
3.17	Circuit diagram for the whole system	34
3.18	Main Circuit	35
3.19	Motor and encoder	35
3.20	The whole system	36
4.1	No controller	38
4.2	Proportional controller $K_p=70$	40
4.3	Proportional controller $K_p=228$	40
4.4	Proportional controller $K_p=386$	41
4.5	Proportional controller $K_p=594$	41
4.6	Proportional controller $K_p=700$	42
4.7	Proportional-integral controller $K_p=70$ $K_i=0.518$	43
4.8	Proportional-integral controller $K_p=700$ $K_i=0.518$	44
4.9	Proportional-integral controller $K_p=70$ $K_i=51.8$	44
4.10	Proportional-integral controller $K_p=700$ $K_i=51.8$	45
4.11	Proportional-derivative controller $K_p=140$ $K_d=2.59$	46
4.12	Proportional-derivative controller $K_p=700$ $K_d=2.59$	47
4.13	Proportional-derivative controller $K_p=140$ $K_d=51.8$	47
4.14	Proportional-derivative controller $K_p=700$ $K_d=51.8$	48
4.15	PID controller $K_p=140$ $K_i=5.18$ $K_d=2.59$	49
4.16	PID controller $K_p=700$ $K_i=5.18$ $K_d=2.59$	50
4.17	PID controller $K_p=140$ $K_i=51.8$ $K_d=51.8$	50
4.18	PID controller $K_p=700$ $K_i=51.8$ $K_d=51.8$	51
4.19	PID controller $K_p=140$ $K_i=5.18$ $K_d=51.8$	51
4.20	PID controller $K_p=700$ $K_i=5.18$ $K_d=51.8$	52
4.21	PID controller $K_p=140$ $K_i=51.8$ $K_d=2.59$	52
4.22	PID controller $K_p=700$ $K_i=51.8$ $K_d=2.59$	53

LIST OF APPENDICES

APPENDIX	TITLE	PAGE
A	Project Programming	60
B	Whole Circuit Diagram	63
C	PIC 16F84A Datasheet	65
D	Regulator 7805 Datasheet	75
E	Servo Motor	
	Clifton Precision JDH 2250-HF-2C-E	77
F	CD Content	87

CHAPTER 1

INTRODUCTION

1.1 Background

This project is focusing on controlling the speed of DC servo motor using Proportional Integral-Derivative (PID) algorithm as a method to reduce overshoot, and settling time of the motor. Peripheral Interface Circuit (PIC) microcontroller is an implementation of the motor to control speed by programmed the PIC.

The PID algorithm that is added to the motor becomes a closed loop system. The system is implemented using MATLAB software and PID algorithm is tuned by changing the value of Proportional gain, K_p , Integral gain, K_i and Derivative gain, K_d to get a speed of the motor which is less overshoot and increase settling time.

The PIC microcontroller is programmed using PIC BASIC either using high language or assembler language.

1.2 Problem Statement

The speed of a DC motor is directly proportional to the supply voltage, so if we reduce the supply voltage from 12 Volts to 6 Volts, the motor will run at half the speed.

The speed controller works by varying the average voltage sent to the motor. It could do this by simply adjusting the voltage sent to the motor, but this is quite inefficient to do.

A better way is to switch the motor's supply on and off very quickly. However, if the switching is fast enough, the motor doesn't notice it, it only notices the average effect.

So, PID algorithm is the best way to overcome this problem without varying the voltage sent to the motor. PID will maintain the speed of motor even the voltage supply sent to the motor changed.

1.3 Project Objective

The objectives of this project are:

- i. To control the speed of DC servo motor using PID algorithm.
- ii. To compare the performance speed of the motor using Proportional (P) controller, Proportional-Integral (PI) controller, Proportional-Derivative (PD) controller and Proportional-Integral-Derivative (PID) controller.
- iii. To implement PID algorithm in PIC microcontroller.

1.4 Scope of Project

The scope of this project is concentrates on controlling the speed of DC servo motor using PID algorithm and to compare the performance using Proportional (P) controller, Proportional-Integral (PI) controller, Proportional-Derivative (PD) controller and Proportional-Integral-Derivative (PID) controller using simulation in MATLAB software.

CHAPTER 2

LITERATURE REVIEW

2.1 Servo Motor

DC servo motors are normally used as prime movers in computers, numerically controlled machinery, or other applications where starts and stops are made quickly and accurately. Servo motors have lightweight, low-inertia armatures that respond quickly to excitation-voltage changes. In addition, very low armature inductance in these servo motors results in a low electrical time constant (typically 0.05 to 1.5 msec) that further sharpens servo motor response to command signals. [1]

Servo motors include permanent-magnetic, printed-circuit, and moving-coil (or shell) dc servo motors. The rotor of a shell dc servomotor consists of a cylindrical shell of copper or aluminum wire coils which rotate in a magnetic field in the annular space between magnetic pole pieces and a stationary iron core. The servo motor features a field, which is provided by cast AlNiCo magnets whose magnetic axis is radial. Servo motors usually have two, four, or six poles. [1]

DC servo motor characteristics include inertia, physical shape, costs, shaft resonance, shaft configuration, speed, and weight. Although these dc servo motors have similar torque ratings, their physical and electrical constants vary. [1]

DC Servo Motor Selection: The first selection approach is to choose a servo motor large enough for a machine that has already been designed; the second is to select the best available servo motor with a specific feature and then build the system around it; and the third is to study servo motor performance and system requirements and mate the two. [1]

The final servo motor system design is usually the least sophisticated that meets the performance specifications reliably. Servo motor requirements may include control of acceleration, velocity, and position to very close tolerances. This says that the servo designer must define the system carefully, establish the servo motor's performance specifications, determine critical areas, and set up tolerances. Only then will the designer be able to propose an adequate servo system and choose a servo motor type. [1]

2.2 Microcontroller (PIC 16F84)

PIC16F84 belongs to a class of 8-bit microcontrollers of RISC architecture. Its general structure is shown on the following map representing basic blocks. Program memory (FLASH)- for storing a written program. Since memory made in FLASH technology can be programmed and cleared more than once, it makes this microcontroller suitable for device development. [2]

EEPROM is the data memory that needs to be saved when there is no supply. It is usually used for storing important data that must not be lost if power supply suddenly stops. For instance, one such data is an assigned temperature in temperature regulators. If during a loss of power supply this data was lost, we would have to make the adjustment once again upon return of supply. Thus our device loses on self-reliance. [2]

RAM is the data memory used by a program during its execution. In RAM are stored all intermediate results or temporary data during run-time. [2]

PORTA and PORTB are physical connections between the microcontroller and the outside world. Port A has five, and port B has eight pins. [2]

FREE-RUN TIMER is an 8-bit register inside a microcontroller that works independently of the program. On every fourth clock of the oscillator it increments its value until it reaches the maximum (255), and then it starts counting over again from zero. As we know the exact timing between each two increments of the timer contents, timer can be used for measuring time which is very useful with some devices. [2]

CENTRAL PROCESSING UNIT has a role of connective element between other blocks in the microcontroller. It coordinates the work of other blocks and executes the user program. [2]

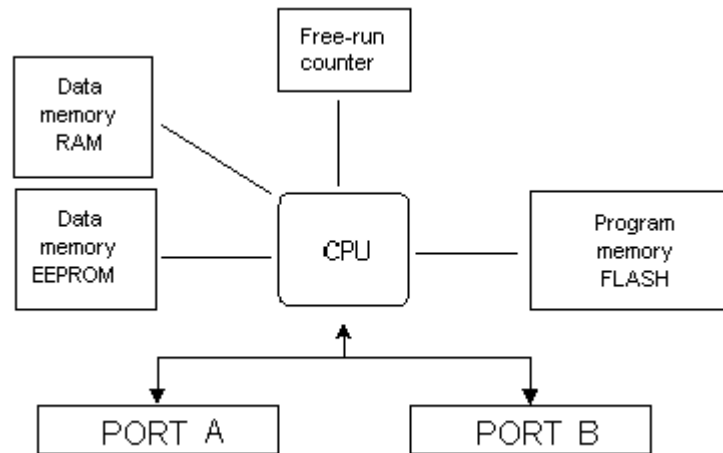


Figure 2.1 PIC 16F84 outline

2.2.1 Pin description

PIC16F84 has a total of 18 pins. It is most frequently found in a DIP18 type of case but can also be found in SMD case which is smaller from a DIP. DIP is an abbreviation for Dual In Package. SMD is an abbreviation for Surface Mount Devices suggesting that holes for pins to go through when mounting aren't necessary in soldering this type of a component. [2]

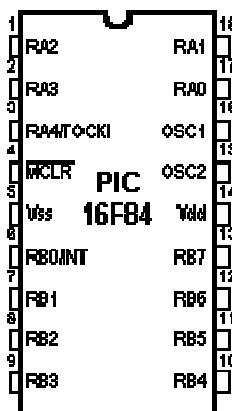


Figure 2.2 Pin number of PIC 16F84

Table 2.1 Pins on PIC16F84 microcontroller

PIN NO	PIN NAME	MEANING
1	RA2	Second pin on port A. Has no additional function
2	RA3	Third pin on port A. Has no additional function
3	RA4	Fourth pin on port A. T0CK1 which functions as a timer is also found on this pin.
4	MCLR	Reset input and Vpp programming voltage of a microcontroller
5	Vss	Ground of power supply
6	RB0	Zero pin on port B. Interrupt input is an additional function
7	RB1	First pin on port B. No additional function
8	RB2	Second pin on port B. No additional function
9	RB3	Third pin on port B. No additional function
10	RB4	Fourth pin on port B. No additional function
11	RB5	Fifth pin on port B. No additional function
12	RB6	Sixth pin on port B. 'Clock' line in program mode
13	RB7	Seventh pin on port B. 'Data' line in program mode

14	Vdd	Positive power supply pole
15	OSC2	Pin assigned for connecting with an oscillator
16	OSC1	Pin assigned for connecting with an oscillator
17	RA0	Second pin on port A. No additional function
18	RA1	First pin on port A. No additional function

2.2.2 Central Processing Unit

Central processing unit (CPU) is the brain of a microcontroller. This part is responsible for finding and fetching the right instruction which needs to be executed, for decoding that instruction, and finally for its execution. Central processing unit connects all parts of the microcontroller into one whole. [2]

2.2.3 Status Register

R/W-0	R/W-0	R/W-0	R/W-1	R/W-1	R/W-x	R/W-x	R/W-x
IRP	RP1	RP0	\overline{TO}	\overline{PD}	Z	DC	C
7							0

R = Readable bit **W** = Writable bit

U = Unimplemented bit, read as '0' - n = Value at power-on reset

Figure 2.3 Status Register

2.2.4 Addressing Mode

RAM memory locations can be accessed directly or indirectly.

i) Direct addressing

Direct Addressing is done through a 9-bit address. This address is obtained by connecting 7th bit of direct address of an instruction with two bits (RP1, RP0) from STATUS register as is shown on the following picture. [2]

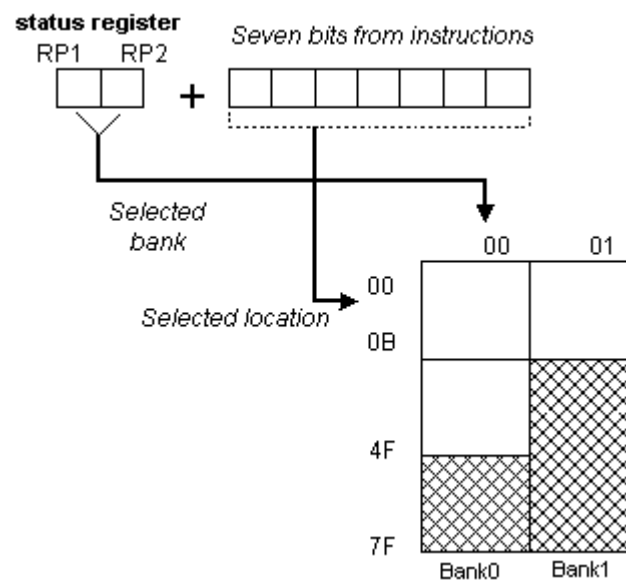


Figure 2.4 Direct addressing

ii) Indirect Addressing

Indirect unlike direct addressing does not take an address from an instruction but derives it from IRP bit of STATUS and FSR registers. Addressed location is

accessed via INDF register which in fact holds the address indicated by a FSR. In other words, any instruction which uses INDF as its register in reality accesses data indicated by a FSR register. [2]

Indirect addressing is very convenient for manipulating data arrays located in GPR registers. In this case, it is necessary to initialize FSR register with a starting address of the array, and the rest of the data can be accessed by incrementing the FSR register. [2]

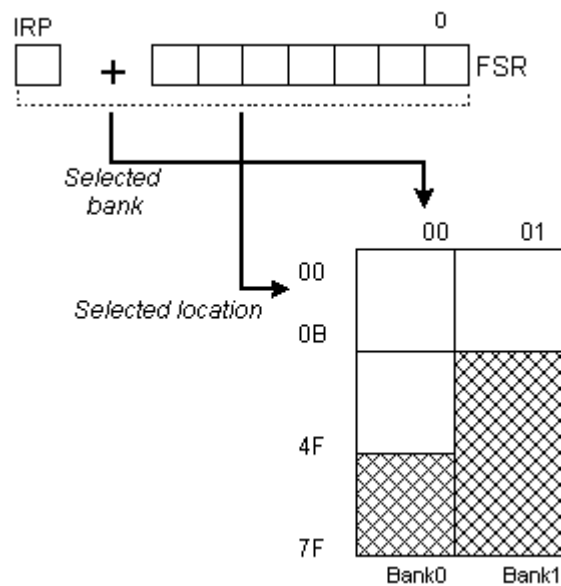


Figure 2.5 Indirect addressing

2.2.5 Applications

PIC16F84 perfectly fits many uses, from automotive industries and controlling home appliances to industrial instruments, remote sensors, electrical door locks and safety devices. It is also ideal for smart cards as well as for battery supplied devices because of its low consumption. [2]

EEPROM memory makes it easier to apply microcontrollers to devices where permanent storage of various parameters is needed (codes for transmitters, motor speed, receiver frequencies, etc.). Low cost, low consumption, easy handling and flexibility make PIC16F84 applicable even in areas where microcontrollers had not previously been considered (example: timer functions, interface replacement in larger systems, coprocessor applications, etc.). [2]

In System Programmability of this chip (along with using only two pins in data transfer) makes possible the flexibility of a product, after assembling and testing have been completed. This capability can be used to create assembly-line production, to store calibration data available only after final testing, or it can be used to improve programs on finished products. [2]

2.3 Proportional-Integral-Derivative (PID) Controller

A proportional–integral–derivative controller (PID controller) is a generic control loop feedback mechanism widely used in industrial control systems. A PID controller attempts to correct the error between a measured process variable and a desired setpoint by calculating and then outputting a corrective action that can adjust the process accordingly. [3]

The PID controller calculation (algorithm) involves three separate parameters; the Proportional, the Integral and Derivative values. The Proportional value determines the reaction to the current error, the Integral determines the reaction based on the sum of recent errors and the Derivative determines the reaction to the rate at which the error has been changing. The weighted sum of these three actions is used to adjust the process via a control element such as the position of a control valve or the power supply of a heating element. [3]

By "tuning" the three constants in the PID controller algorithm, the controller can provide control action designed for specific process requirements. The response of the controller can be described in terms of the responsiveness of the controller to an error, the degree to which the controller overshoots the setpoint and the degree of system oscillation. Note that the use of the PID algorithm for control does not guarantee optimal control of the system or system stability. [3]

Some applications may require using only one or two modes to provide the appropriate system control. This is achieved by setting the gain of undesired control outputs to zero. A PID controller will be called a PI, PD, P or I controller in the absence

of the respective control actions. PI controllers are particularly common, since derivative action is very sensitive to measurement noise, and the absence of an integral value may prevent the system from reaching its target value due to the control action. [3]

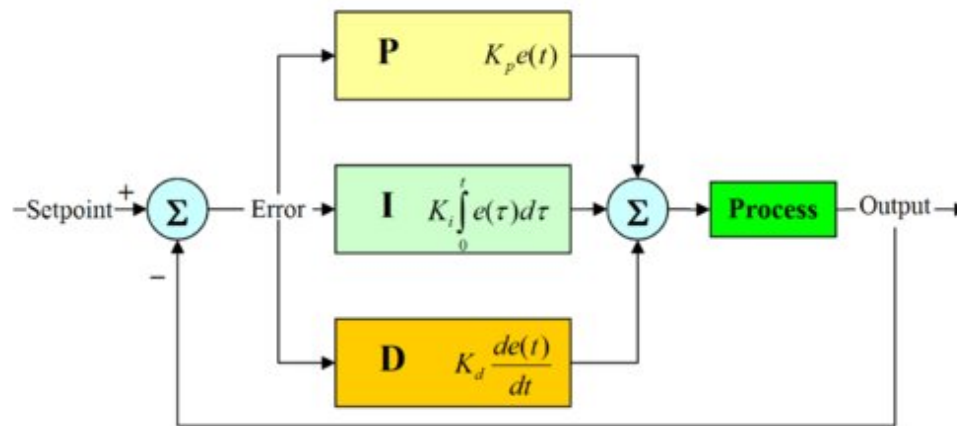


Figure 2.6 A block diagram of a PID controller

2.4 Ziegler-Nichols Method

The Ziegler–Nichols tuning method is a heuristic method of tuning a PID controller. It was developed by John G. Ziegler and Nathaniel B. Nichols. It is performed by setting the I and D gains to zero. The "P" gain is then increased (from zero) until it reaches the critical gain K_c , at which the output of the control loop begins to oscillate. K_c and the oscillation period T_c are used to set the P, I, and D gains depending on the type of controller used: [4]

Steps :

- i) Place controller into automatic with low gain, no reset or derivative.
- ii) Gradually increase gain, making small changes in the setpoint, until oscillations start.
- iii) Adjust gain to make the oscillations continue with a constant amplitude.
- iv) Note the gain (Ultimate Gain, G_u) and Period (Ultimate Period, P_u .)
- v) The Ultimate Gain, G_u , is the gain at which the oscillations continue with a constant amplitude.

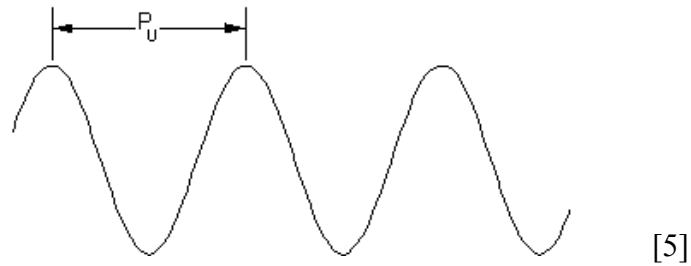


Figure 2.7 Oscillations with constant amplitude

Table 2.2 Estimation value for gain, reset and derivative

	Gain	Reset	Derivative
P	$0.5 G_u$	-	-
PI	$0.45 G_u$	$1.2u$	-
PID	$0.6 G_u$	$2/P_u$	$P_u/8$

CHAPTER 3

METHODOLOGY

Some methodologies are apply in order to control the DC servo motor control using PID method which is implement in PIC 16F84A microcontroller. The relationship of this project is shown below:

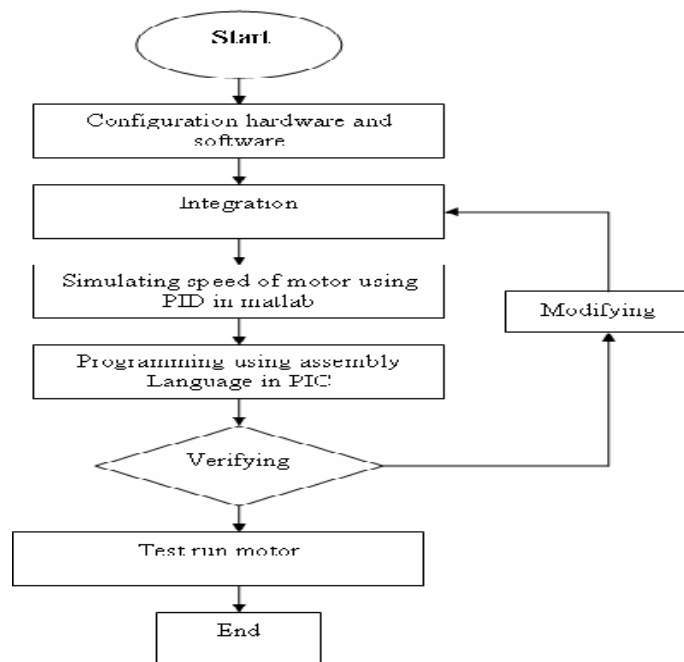


Figure 3.1 Project procedure

Configuration;

Hardware and software that involve in the system should be configuring to ensure that the system can run properly. The hardware of the system are PIC 16F84 and DC servo motor (Clifton Precision JDH-2250-HF-2C-E) while the software are PICbasic and matlab

Programming;

After configure the hardware and software, the second stage is designing the program. The program must be developing in order to ensure the DC servo motor is run as the needed position.

Verifying;

This project must be verified to ensure either the system is run or not. The program must be modified if the motor not run properly.

Integration;

This part of the project is explaining the background of the motor control deriving, PID method simulating in matlab and the integration of the DC servo motor, PID algorithm and PIC microcontroller.

3.1 Modeling DC Servo Motor

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 adding to changing the system to closed loop system. Below is the step of the motor modeling.

$$R = 2.7 \, \Omega$$

$$L = 0.004 \, \text{H}$$

$$K = 0.105 \, \text{Vs rad}^{-1}$$

$$K = 0.105 \, \text{Nm A}^{-1}$$

$$J = 0.0001 \, \text{Kg m}^2$$

$$B = 0.0000093 \, \text{Nms rad}^{-1}$$

$$\frac{di_a}{dt} = \frac{R}{L}i_a - \frac{K}{L}\omega_r + \frac{1}{L}V_a \quad (3.1)$$

$$\frac{d\omega_r}{dt} = \frac{K}{J}i_a - \frac{B}{J}\omega_r \quad (3.2)$$

$$\begin{bmatrix} \frac{di_a}{dt} \\ \frac{d\omega_r}{dt} \end{bmatrix} = \begin{bmatrix} -\frac{R}{L} & -\frac{K}{L} \\ \frac{K}{J} & -\frac{B}{J} \end{bmatrix} \begin{bmatrix} i_a \\ \omega_r \end{bmatrix} + \begin{bmatrix} \frac{1}{L} \\ 0 \end{bmatrix} V_a \quad (3.3)$$

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

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

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

$$\begin{aligned}
C &= [0 \quad 1] & D &= [0] \\
[sI - A] &= \begin{bmatrix} s & 0 \\ 0 & s \end{bmatrix} - \begin{bmatrix} -675 & -26.25 \\ 1050 & -0.093 \end{bmatrix} \\
&= \begin{bmatrix} s+675 & 26.25 \\ -1050 & s-0.093 \end{bmatrix}
\end{aligned} \tag{3.4}$$

$$\text{From } [sI - A]^{-1} = \frac{\text{adj}(sI - A)}{\text{def}(sI - A)} \tag{3.5}$$

$$\text{If } A = \begin{pmatrix} a & b \\ c & d \end{pmatrix} ; \quad A^{-1} = \frac{1}{ad - bc} \begin{pmatrix} d & -b \\ -c & a \end{pmatrix} \tag{3.6}$$

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

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

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

$$\begin{aligned}
&= [0 \quad 1] \frac{\begin{bmatrix} s-0.093 & -26.25 \\ 1050 & s-675 \end{bmatrix}}{s^2 + 675.093s + 27625.275} \begin{bmatrix} 250 \\ 0 \end{bmatrix} + [0] \\
&= \frac{262500}{s^2 + 675.093s + 27625.275}
\end{aligned}$$

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

3.2 Ziegler Tuning Method

Table 3.1 show the typical value of the Proportional, Integral and Derivative feedback coefficients for PID-type controllers.

Table 3.1 Ziegler Tuning Table

controller	Kp	Ki	Kd
PID	$K_p \in [0.1 \ 0.5]K_{pmax}$	$K_i \in [0.1 \ 10]x$ $K_{pmax}T_{osc}$	$K_d \in [0.05 \ 1]x$ $K_{pmax}T_{osc}$
PD	$K_p \in [0.1 \ 0.5]K_{pmax}$	0	$K_d \in [0.05 \ 1]x$ $K_{pmax}T_{osc}$
PI	$K_p \in [0.05 \ 0.5]K_{pmax}$	$K_i \in [0.01 \ 1]x$ $K_{pmax}T_{osc}$	0
P	$K_p \in [0.05 \ 0.5]K_{pmax}$	0	0

The value of K_{pmax} and T_{osc} are such as below:

$$K_{pmax} = 1400$$

$$T_{osc} = 0.037s$$

Table 3.2 show the range value of Proportional, Integral and Derivative gain after multiply to the K_{pmax} and T_{osc} value. The value of K_{pmax} can be get by setting the value of Integral gain, K_i and Derivative gain, K_d to zero. Then, tuning the value of Proportional gain, K_p until get an oscillation. The value of the best oscillation is the value of K_{pmax} . The value of T_{osc} is the different value of time for the two first of the oscillation.

Table 3.2 Range value of controller

controller	Kp	Ki	Kd
PID	140 - 700	5.18 - 518	2.59 – 51.8
PD	140 - 700	0	2.59 – 51.8
PI	70 -700	0.518 – 51.8	0
P	70 - 700	0	0

3.3 Hardware Development

This part is about implementation PID controller to the PIC microcontroller. In order to develop this part, its need hardware consist of some components. The best components must be choosing by considering some factors such as quality, reliability and cost effective. By choosing or using the wrong components will lead to much more problems to the hardware development and also escalating cost.

3.3.1 DC Servo motor

There are various types of motor 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. So, 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 3.2 Clifton Precision JDH 2250-HF-2C-E

Table 3.3 Parameters of motor

PARAMETER	SYMBOL	VALUE
Resistances of armature	R	2.7Ω
Inductances of armature	L	0.004 H
Inertia	J	0.0001 Kg m^2
Friction Coefficient	B	$0.0000093 \text{ Nms rad}^{-1}$
Torque	K	$0.105 \text{ Vs rad}^{-1}$

3.3.2 PIC 16F84

In this project, PIC 16F84 microcontroller was selected to drives the DC servo motor. The PIC microcontroller was programmed to give instruction to drive the DC servo motor. DC servo motor can be connected to output port which is PORTB, from RB1 to RB7. RB1 which is located at pin 6 was selected to connect DC servo motor in this project such as in Figure 3.3. 4-MHz oscillator is used because the circuit this time doesn't need high-speed operation.

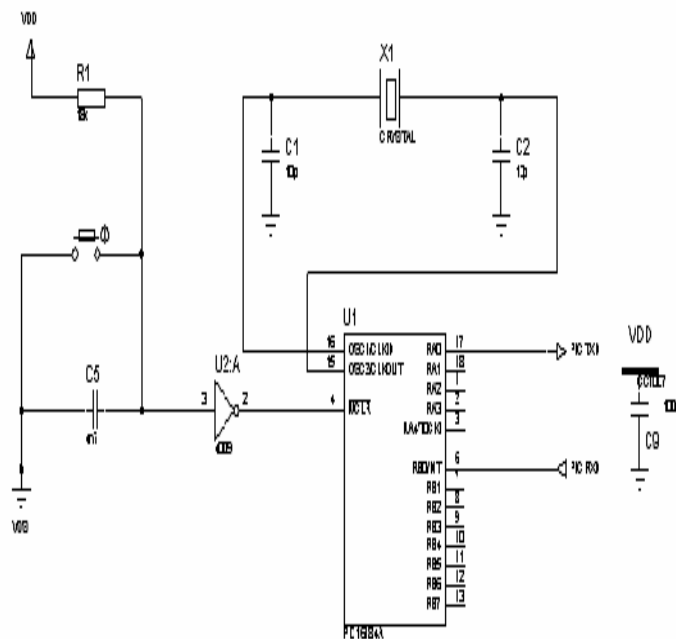


Figure 3.3 PIC 16F84 microcontroller circuit

3.3.3 Power Supply

The purpose of this circuit is to keep power supply voltage to PIC to 5V. In this case, the voltage which is applied to PIC becomes less than 5V because of the voltage drop (about 1V) of the regulator. In case of PIC16F84, the operation is possible even if the power falls to about 3V because the operating voltage range is from 2V to 5.5V. It is enough in the 100-mA type.

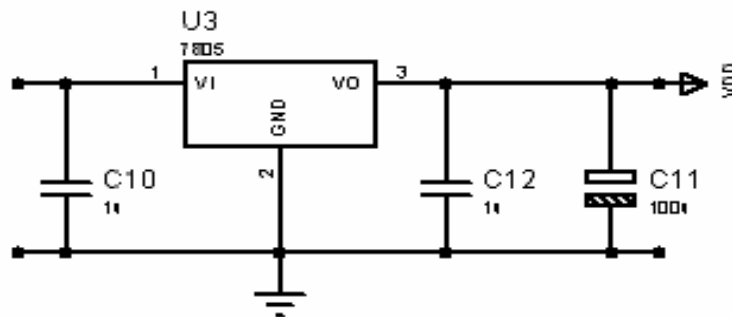


Figure 3.4 Power Supply circuit

3.4 Software Development

This topic is discussed the development of software in order to complete this project. There are MATLAB 7.0 to develop PID method and Melabs EPIC™ Programmer software to program the PIC microcontroller.

3.4.1 PID Method

From the modeling DC servo motor, the transfer function is

$$T(s) = \frac{Y(s)}{U(s)} = \frac{262500}{s^2 + 675.093s + 27625.275} \quad (3.8)$$

The system before using PID controller is looks like in Figure 3.5:

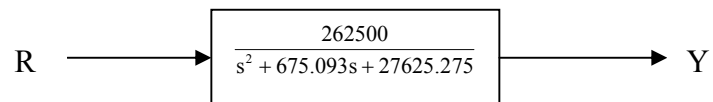


Figure 3.5 System before using PID controller

Then, PID controller is added to the system. Now, the system looks like in Figure 3.6:

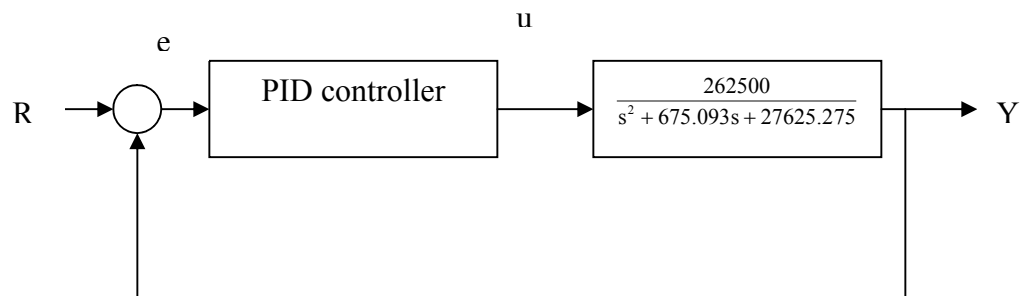


Figure 3.6 System with PID controller

In Figure 3.6, the variable (e) represents the tracking error which is the difference between the desired input value (R) and the actual output (Y). This error signal (e) will be sent to the PID controller, and the controller computes both the derivative and the integral of this error signal. The signal (u) just past the controller is now equal to the proportional gain (Kp) times the magnitude of the error plus the integral gain (Ki) times the integral of the error plus the derivative gain (Kd) times the derivative of the error (equation 3).

The transfer function of the PID controller is:

$$K_p + \frac{K_i}{s} + K_d s = + \frac{K_d s^2 + K_p s + K_i}{s} \quad (3.9)$$

So, the signal (u) that is past the controller is:

$$U = K_p e + K_i \int e dt + K_d \frac{de}{dt} \quad (3.10)$$

This signal (u) will be sent to the plant, and the new output (Y) will be obtained. This new output (Y) will be sent back to the sensor again to find the new error signal (e). The controller takes this new error signal and computes its derivative and its integral again. This process goes on and on.

In this project, the PID controller that was added into the system is designed using m-file in matlab software. (Refer Figure 3.7)

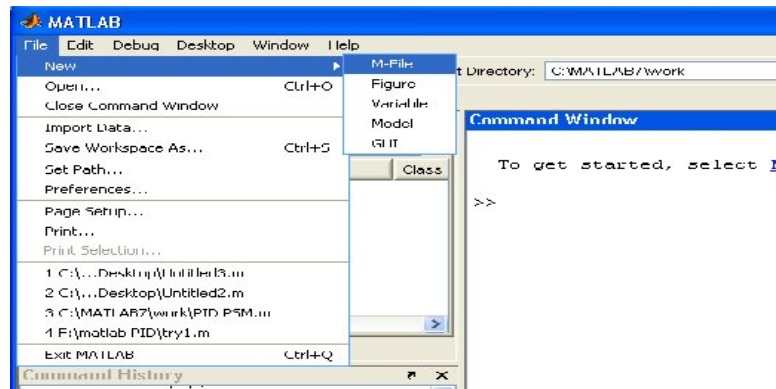


Figure 3.7 Designed using m-file

Then the following commands are typing into m-file. (Refer Figure 3.8)

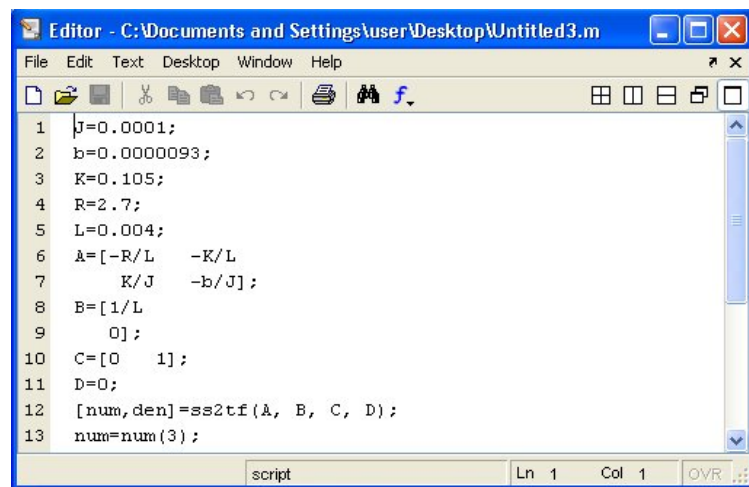
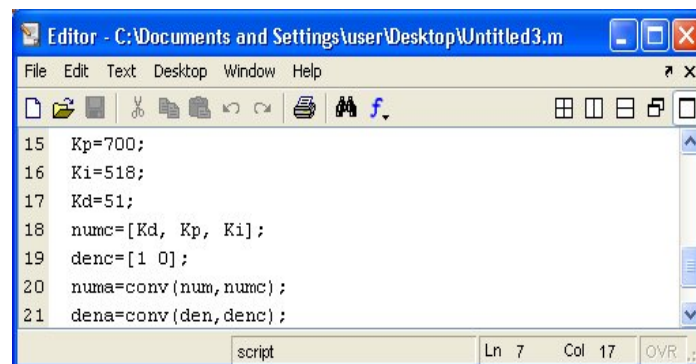


Figure 3.8 Typing program

In Figure 3.8, '[num,den] = ss2tf(A,B,C,D)' command creates the numerator and denominator of the transfer function of DC servo motor. This numerical

inconsistency can be eliminated by adding the following 'num=num(3)' command after the ss2tf command to get rid of the numbers that are not supposed to be there.

The transfer function of PID controller is recalled using following commands. The value of the proportional gain, Kp, integral gain Ki and derivative gain, Kd can be adjust by changing the value. (Refer Figure 3.9):



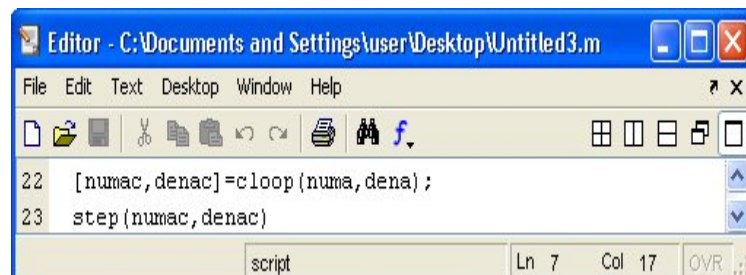
```

Editor - C:\Documents and Settings\user\Desktop\Untitled3.m
File Edit Text Desktop Window Help
Kp=700;
Ki=518;
Kd=51;
numc=[Kd, Kp, Ki];
denc=[1 0];
numa=conv(num,numc);
dena=conv(den,denc);
script Ln 7 Col 17 OVR

```

Figure 3.9 Changing the value

The closed loop of the system is determined by 'cloop' command and the command 'step (numac,denac)' is to see how the step response looks as in Figure 3.10.



```

Editor - C:\Documents and Settings\user\Desktop\Untitled3.m
File Edit Text Desktop Window Help
[numac,denac]=cloop(numa,dena);
step(numac,denac)
script Ln 7 Col 17 OVR

```

Figure 3.10 Closed loop system

Then, save and run it such in Figure 3.11.

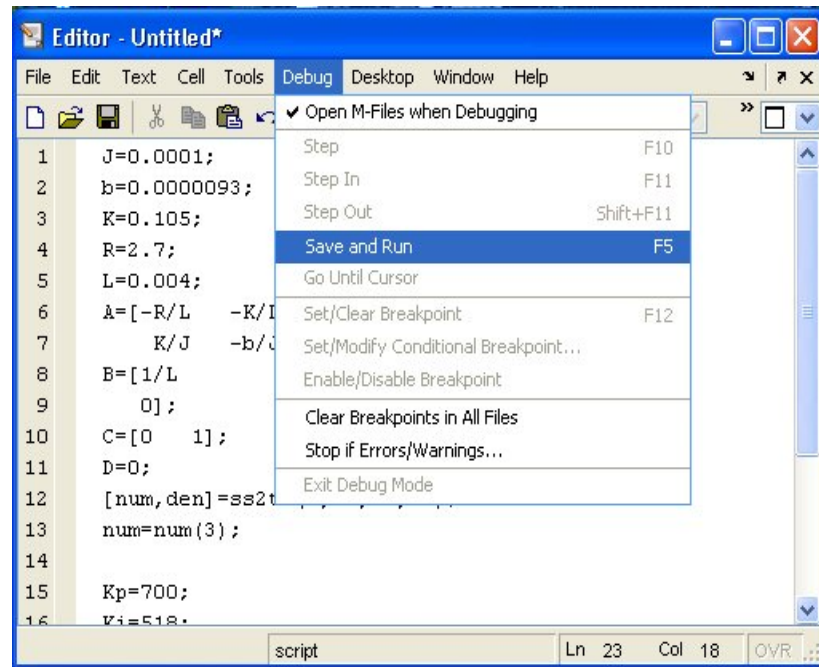


Figure 3.11 Save and run

The result of this system is obtained by changing the value of the proportional gain, K_p , integral gain K_i and derivative gain, K_d . The best five result for the proportional (P) controller, proportional-Integral (PI) controller, proportional-derivative (PD) controller and proportional-integral-derivative (PID) controller that is apply in this system is obtained.

(iii) Open the Melabs EPIC™ Programmer



Figure 3.13 Select for PIC16F84A

Open Melabs EPIC™ Programmer then select for PIC16F84A. The melabs EPIC™ Programmer connects to a PC compatible parallel printer port. The melabs Serial Programmer connects to a PC compatible serial port. The melabs USB Programmer and melabs U2 Programmer connect to a PC USB port or powered USB hub. Each programmer may be controlled by the melabs Programmer software.

(iv) Open melabs configuration

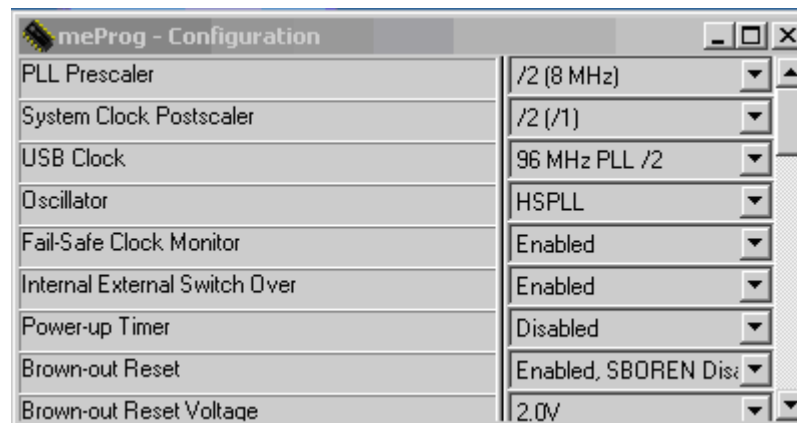


Figure 3.14 Setup for PLL

After the selection of the PIC type Figure 3.14 show the step to setup the melabs configuration for PLL application. Seen the 4 MHz crystal is used and the HS mode oscillator for frequencies is up to 48 MHz, the configuration must be fill correctly.

(v) Open the compile programming

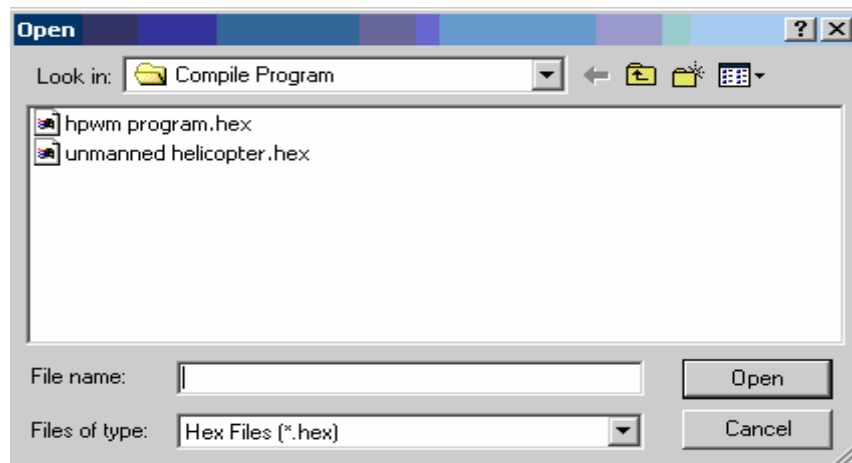


Figure 3.15 Programming selected

Figure 3.15 show the step for selecting the programming that had been saving in .HEX type documentation. This were done after the deleting the entire previous program in the PIC.

(vi) Verify the Program

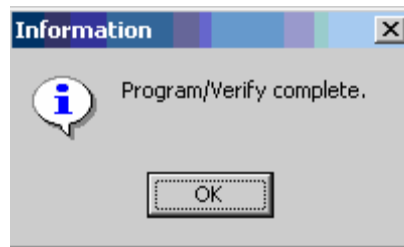


Figure 3.16 Verifying the program

Figure 3.16 shows the complete of verifying program and the PIC 16F84A is ready to be used.

3.5 Circuit Diagram

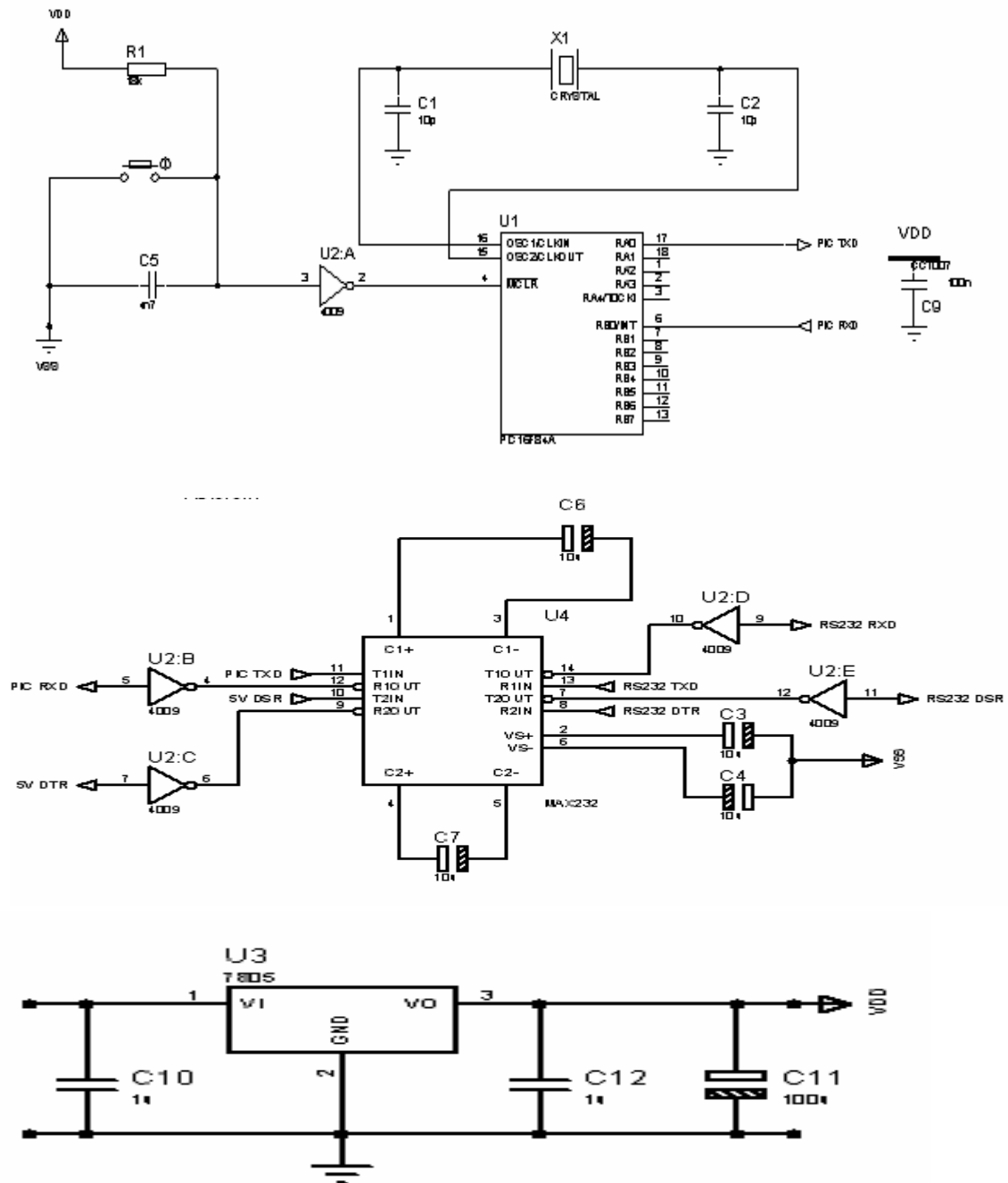


Figure 3.17 Circuit diagram for the whole system

3.6 Final Prototype

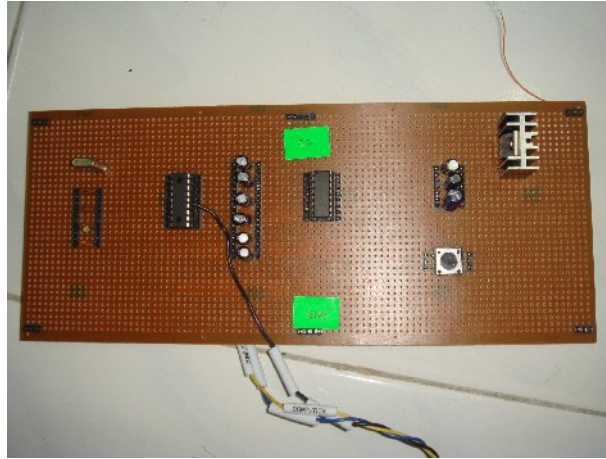


Figure 3.18 Main Circuit



Figure 3.19 Motor and encoder

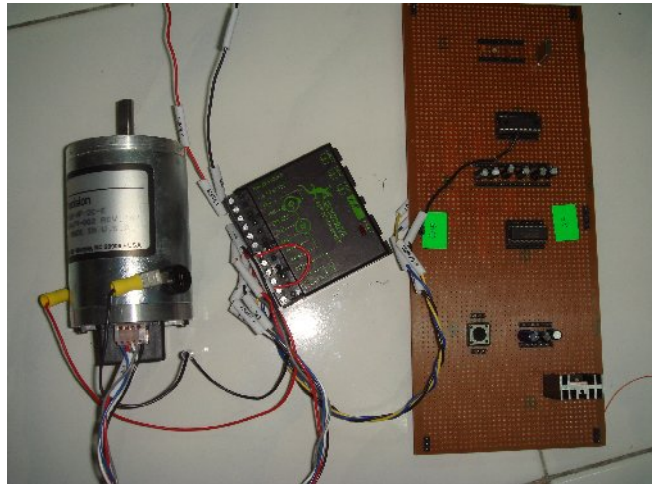


Figure 3.20 The whole system

CHAPTER 3

METHODOLOGY

Some methodologies are apply in order to control the DC servo motor control using PID method which is implement in PIC 16F84A microcontroller. The relationship of this project is shown below:

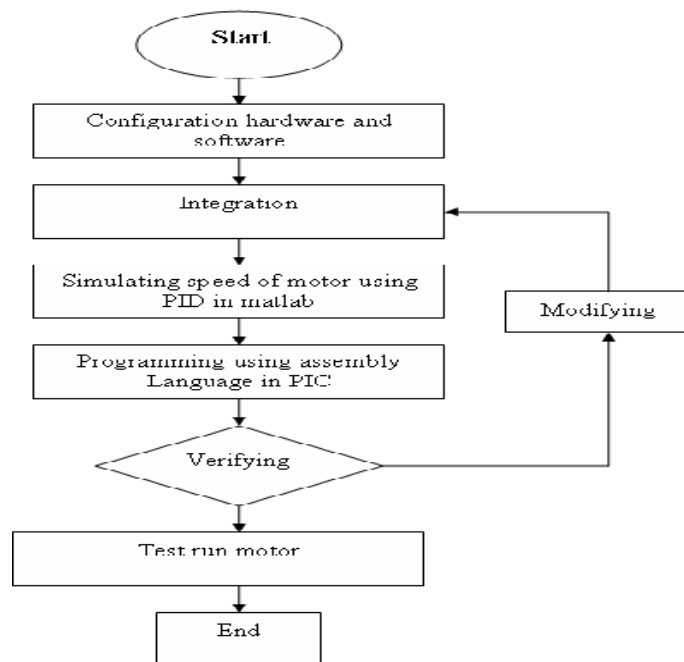


Figure 3.1 Project procedure

Configuration;

Hardware and software that involve in the system should be configuring to ensure that the system can run properly. The hardware of the system are PIC 16F84 and DC servo motor (Clifton Precision JDH-2250-HF-2C-E) while the software are PICbasic and matlab

Programming;

After configure the hardware and software, the second stage is designing the program. The program must be developing in order to ensure the DC servo motor is run as the needed position.

Verifying;

This project must be verified to ensure either the system is run or not. The program must be modified if the motor not run properly.

Integration;

This part of the project is explaining the background of the motor control deriving, PID method simulating in matlab and the integration of the DC servo motor, PID algorithm and PIC microcontroller.

3.1 Modeling DC Servo Motor

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 adding to changing the system to closed loop system. Below is the step of the motor modeling.

$$R = 2.7 \, \Omega$$

$$L = 0.004 \, \text{H}$$

$$K = 0.105 \, \text{Vs rad}^{-1}$$

$$K = 0.105 \, \text{Nm A}^{-1}$$

$$J = 0.0001 \, \text{Kg m}^2$$

$$B = 0.0000093 \, \text{Nms rad}^{-1}$$

$$\frac{di_a}{dt} = \frac{R}{L}i_a - \frac{K}{L}\omega_r + \frac{1}{L}V_a \quad (3.1)$$

$$\frac{d\omega_r}{dt} = \frac{K}{J}i_a - \frac{B}{J}\omega_r \quad (3.2)$$

$$\begin{bmatrix} \frac{di_a}{dt} \\ \frac{d\omega_r}{dt} \end{bmatrix} = \begin{bmatrix} -\frac{R}{L} & -\frac{K}{L} \\ \frac{K}{J} & -\frac{B}{J} \end{bmatrix} \begin{bmatrix} i_a \\ \omega_r \end{bmatrix} + \begin{bmatrix} \frac{1}{L} \\ 0 \end{bmatrix} V_a \quad (3.3)$$

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

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

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

$$\begin{aligned}
C &= [0 \quad 1] & D &= [0] \\
[sI - A] &= \begin{bmatrix} s & 0 \\ 0 & s \end{bmatrix} - \begin{bmatrix} -675 & -26.25 \\ 1050 & -0.093 \end{bmatrix} \\
&= \begin{bmatrix} s+675 & 26.25 \\ -1050 & s-0.093 \end{bmatrix}
\end{aligned} \tag{3.4}$$

$$\text{From } [sI - A]^{-1} = \frac{\text{adj}(sI - A)}{\text{def}(sI - A)} \tag{3.5}$$

$$\text{If } A = \begin{pmatrix} a & b \\ c & d \end{pmatrix} ; \quad A^{-1} = \frac{1}{ad - bc} \begin{pmatrix} d & -b \\ -c & a \end{pmatrix} \tag{3.6}$$

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

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

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

$$\begin{aligned}
&= [0 \quad 1] \frac{\begin{bmatrix} s-0.093 & -26.25 \\ 1050 & s-675 \end{bmatrix}}{s^2 + 675.093s + 27625.275} \begin{bmatrix} 250 \\ 0 \end{bmatrix} + [0] \\
&= \frac{262500}{s^2 + 675.093s + 27625.275}
\end{aligned}$$

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

3.2 Ziegler Tuning Method

Table 3.1 show the typical value of the Proportional, Integral and Derivative feedback coefficients for PID-type controllers.

Table 3.1 Ziegler Tuning Table

controller	Kp	Ki	Kd
PID	$K_p \in [0.1 \ 0.5]K_{pmax}$	$K_i \in [0.1 \ 10]x$ $K_{pmax}T_{osc}$	$K_d \in [0.05 \ 1]x$ $K_{pmax}T_{osc}$
PD	$K_p \in [0.1 \ 0.5]K_{pmax}$	0	$K_d \in [0.05 \ 1]x$ $K_{pmax}T_{osc}$
PI	$K_p \in [0.05 \ 0.5]K_{pmax}$	$K_i \in [0.01 \ 1]x$ $K_{pmax}T_{osc}$	0
P	$K_p \in [0.05 \ 0.5]K_{pmax}$	0	0

The value of K_{pmax} and T_{osc} are such as below:

$$K_{pmax} = 1400$$

$$T_{osc} = 0.037s$$

Table 3.2 show the range value of Proportional, Integral and Derivative gain after multiply to the K_{pmax} and T_{osc} value. The value of K_{pmax} can be get by setting the value of Integral gain, K_i and Derivative gain, K_d to zero. Then, tuning the value of Proportional gain, K_p until get an oscillation. The value of the best oscillation is the value of K_{pmax} . The value of T_{osc} is the different value of time for the two first of the oscillation.

Table 3.2 Range value of controller

controller	Kp	Ki	Kd
PID	140 - 700	5.18 - 518	2.59 – 51.8
PD	140 - 700	0	2.59 – 51.8
PI	70 -700	0.518 – 51.8	0
P	70 - 700	0	0

3.3 Hardware Development

This part is about implementation PID controller to the PIC microcontroller. In order to develop this part, its need hardware consist of some components. The best components must be choosing by considering some factors such as quality, reliability and cost effective. By choosing or using the wrong components will lead to much more problems to the hardware development and also escalating cost.

3.3.1 DC Servo motor

There are various types of motor 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. So, 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 3.2 Clifton Precision JDH 2250-HF-2C-E

Table 3.3 Parameters of motor

PARAMETER	SYMBOL	VALUE
Resistances of armature	R	2.7Ω
Inductances of armature	L	0.004 H
Inertia	J	0.0001 Kg m^2
Friction Coefficient	B	$0.0000093 \text{ Nms rad}^{-1}$
Torque	K	$0.105 \text{ Vs rad}^{-1}$

3.3.2 PIC 16F84

In this project, PIC 16F84 microcontroller was selected to drives the DC servo motor. The PIC microcontroller was programmed to give instruction to drive the DC servo motor. DC servo motor can be connected to output port which is PORTB, from RB1 to RB7. RB1 which is located at pin 6 was selected to connect DC servo motor in this project such as in Figure 3.3. 4-MHz oscillator is used because the circuit this time doesn't need high-speed operation.

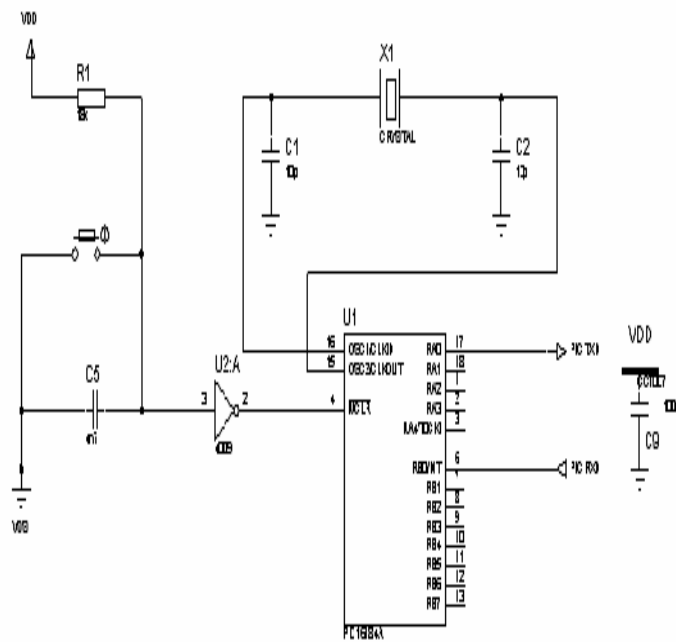


Figure 3.3 PIC 16F84 microcontroller circuit

3.3.3 Power Supply

The purpose of this circuit is to keep power supply voltage to PIC to 5V. In this case, the voltage which is applied to PIC becomes less than 5V because of the voltage drop (about 1V) of the regulator. In case of PIC16F84, the operation is possible even if the power falls to about 3V because the operating voltage range is from 2V to 5.5V. It is enough in the 100-mA type.

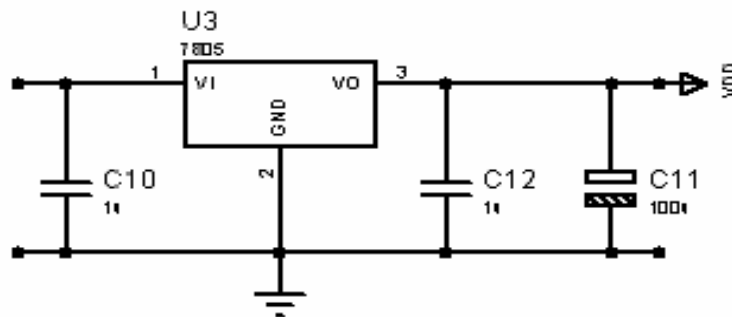


Figure 3.4 Power Supply circuit

3.4 Software Development

This topic is discussed the development of software in order to complete this project. There are MATLAB 7.0 to develop PID method and Melabs EPIC™ Programmer software to program the PIC microcontroller.

3.4.1 PID Method

From the modeling DC servo motor, the transfer function is

$$T(s) = \frac{Y(s)}{U(s)} = \frac{262500}{s^2 + 675.093s + 27625.275} \quad (3.8)$$

The system before using PID controller is looks like in Figure 3.5:

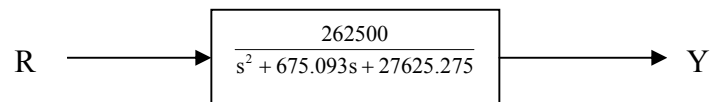


Figure 3.5 System before using PID controller

Then, PID controller is added to the system. Now, the system looks like in Figure 3.6:

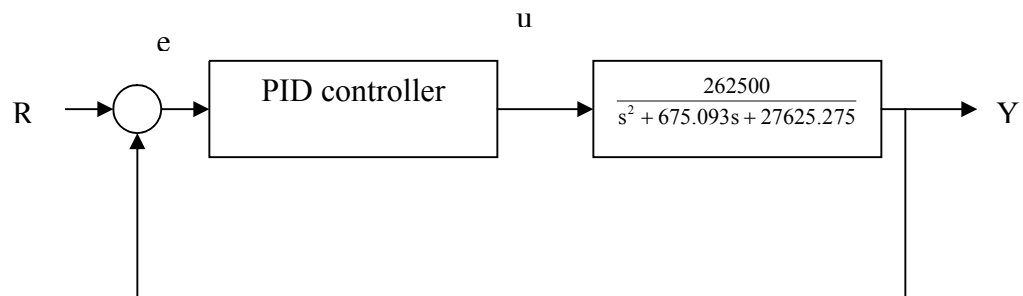


Figure 3.6 System with PID controller

In Figure 3.6, the variable (e) represents the tracking error which is the difference between the desired input value (R) and the actual output (Y). This error signal (e) will be sent to the PID controller, and the controller computes both the derivative and the integral of this error signal. The signal (u) just past the controller is now equal to the proportional gain (K_p) times the magnitude of the error plus the integral gain (K_i) times the integral of the error plus the derivative gain (K_d) times the derivative of the error (equation 3).

The transfer function of the PID controller is:

$$K_p + \frac{K_i}{s} + K_d s = + \frac{K_d s^2 + K_p s + K_i}{s} \quad (3.9)$$

So, the signal (u) that is past the controller is:

$$U = K_p e + K_i \int e dt + K_d \frac{de}{dt} \quad (3.10)$$

This signal (u) will be sent to the plant, and the new output (Y) will be obtained. This new output (Y) will be sent back to the sensor again to find the new error signal (e). The controller takes this new error signal and computes its derivative and its integral again. This process goes on and on.

In this project, the PID controller that was added into the system is designed using m-file in matlab software. (Refer Figure 3.7)

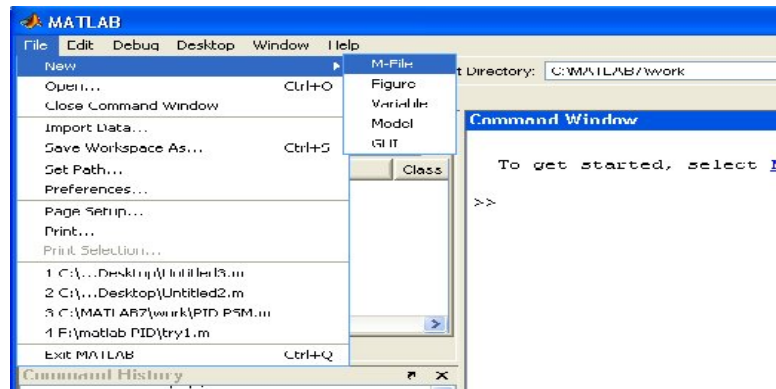


Figure 3.7 Designed using m-file

Then the following commands are typing into m-file. (Refer Figure 3.8)

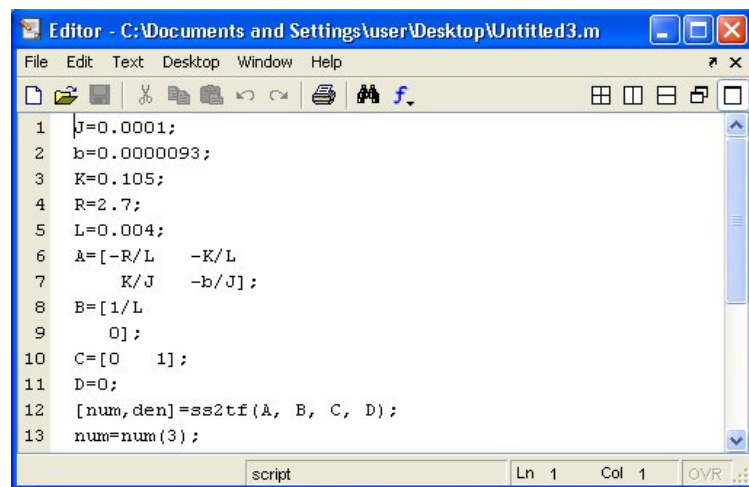
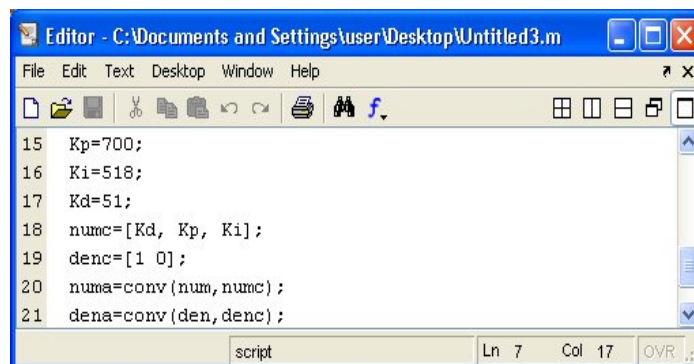


Figure 3.8 Typing program

In Figure 3.8, '[num,den] = ss2tf(A,B,C,D)' command creates the numerator and denominator of the transfer function of DC servo motor. This numerical

inconsistency can be eliminated by adding the following 'num=num(3)' command after the ss2tf command to get rid of the numbers that are not supposed to be there.

The transfer function of PID controller is recalled using following commands. The value of the proportional gain, Kp, integral gain Ki and derivative gain, Kd can be adjust by changing the value. (Refer Figure 3.9):



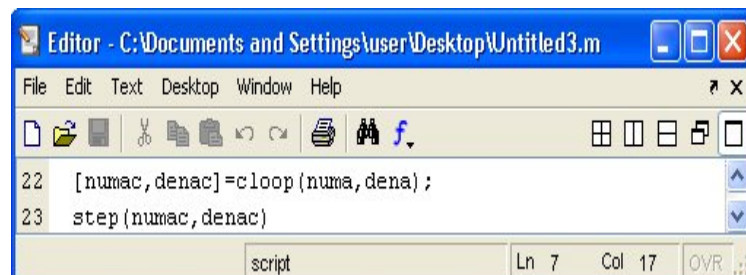
```

15 Kp=700;
16 Ki=518;
17 Kd=51;
18 numc=[Kd, Kp, Ki];
19 denc=[1 0];
20 numa=conv(num,numc);
21 dena=conv(den,denc);
  
```

The image shows a MATLAB Editor window titled 'Editor - C:\Documents and Settings\user\Desktop\Untitled3.m'. The window contains a script with the following code: Line 15: Kp=700; Line 16: Ki=518; Line 17: Kd=51; Line 18: numc=[Kd, Kp, Ki]; Line 19: denc=[1 0]; Line 20: numa=conv(num,numc); Line 21: dena=conv(den,denc);. The status bar at the bottom indicates 'script', 'Ln 7', 'Col 17', and 'OVR'.

Figure 3.9 Changing the value

The closed loop of the system is determined by 'cloop' command and the command 'step (numac,denac)' is to see how the step response looks as in Figure 3.10.



```

22 [numac,denac]=cloop(numa,dena);
23 step(numac,denac)
  
```

The image shows a MATLAB Editor window titled 'Editor - C:\Documents and Settings\user\Desktop\Untitled3.m'. The window contains a script with the following code: Line 22: [numac,denac]=cloop(numa,dena); Line 23: step(numac,denac);. The status bar at the bottom indicates 'script', 'Ln 7', 'Col 17', and 'OVR'.

Figure 3.10 Closed loop system

Then, save and run it such in Figure 3.11.

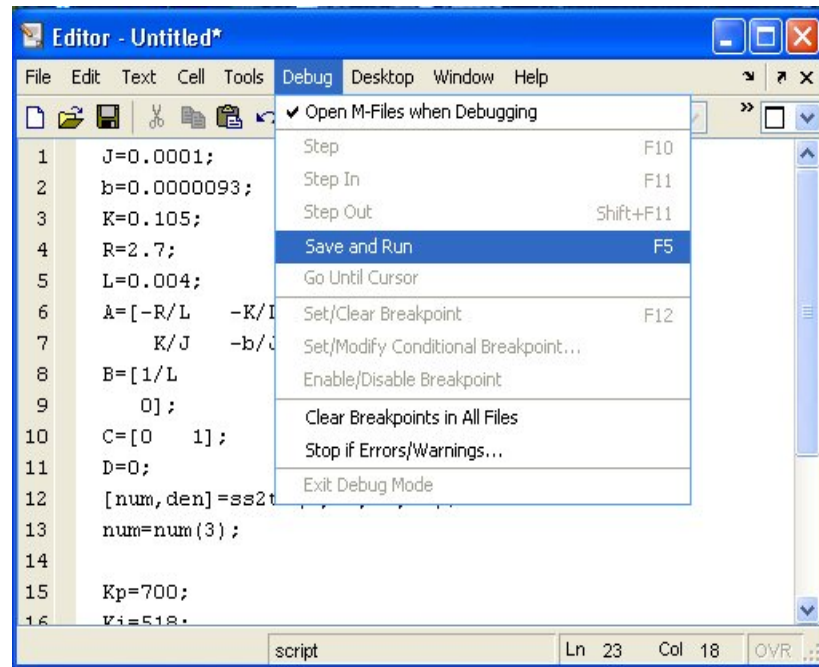


Figure 3.11 Save and run

The result of this system is obtained by changing the value of the proportional gain, K_p , integral gain K_i and derivative gain, K_d . The best five result for the proportional (P) controller, proportional-Integral (PI) controller, proportional-derivative (PD) controller and proportional-integral-derivative (PID) controller that is apply in this system is obtained.

(iii) Open the Melabs EPIC™ Programmer

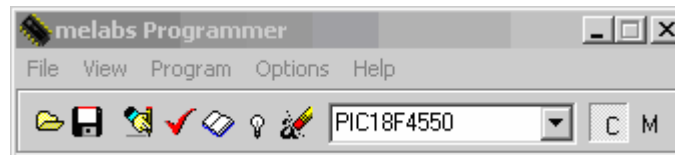


Figure 3.13 Select for PIC16F84A

Open Melabs EPIC™ Programmer then select for PIC16F84A. The melabs EPIC™ Programmer connects to a PC compatible parallel printer port. The melabs Serial Programmer connects to a PC compatible serial port. The melabs USB Programmer and melabs U2 Programmer connect to a PC USB port or powered USB hub. Each programmer may be controlled by the melabs Programmer software.

(iv) Open melabs configuration

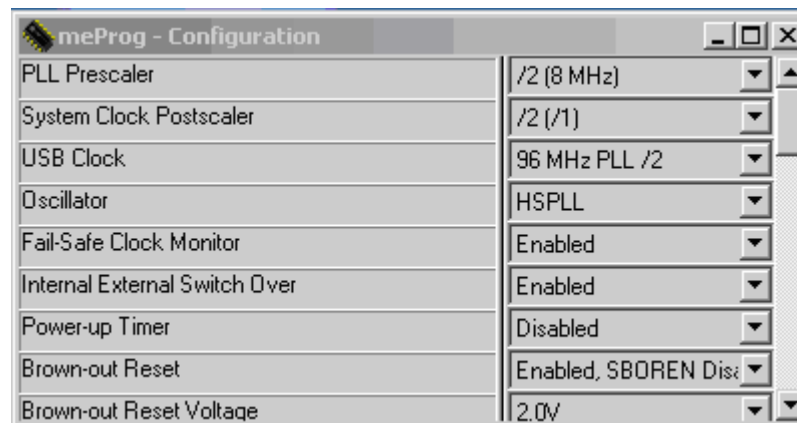


Figure 3.14 Setup for PLL

After the selection of the PIC type Figure 3.14 show the step to setup the melabs configuration for PLL application. Seen the 4 MHz crystal is used and the HS mode oscillator for frequencies is up to 48 MHz, the configuration must be fill correctly.

(v) Open the compile programming

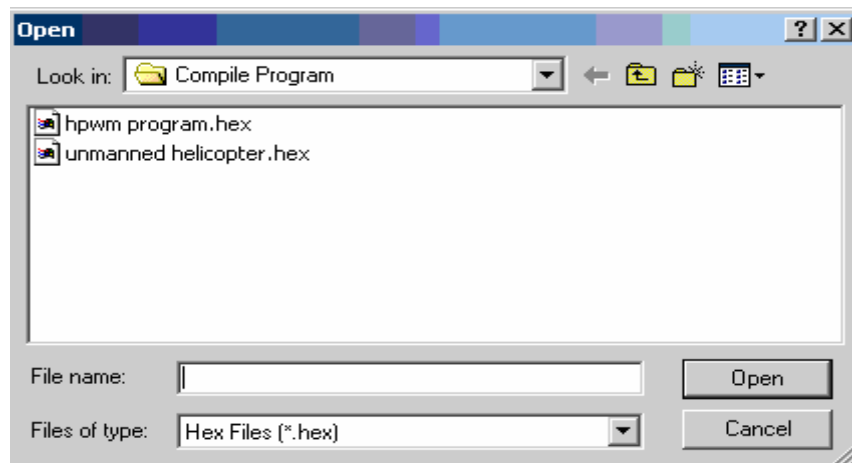


Figure 3.15 Programming selected

Figure 3.15 show the step for selecting the programming that had been saving in .HEX type documentation. This were done after the deleting the entire previous program in the PIC.

(vi) Verify the Program

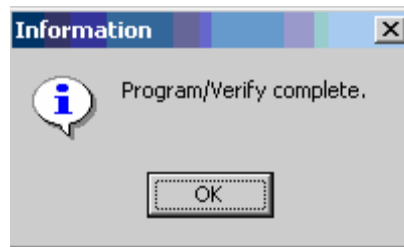


Figure 3.16 Verifying the program

Figure 3.16 shows the complete of verifying program and the PIC 16F84A is ready to be used.

3.5 Circuit Diagram

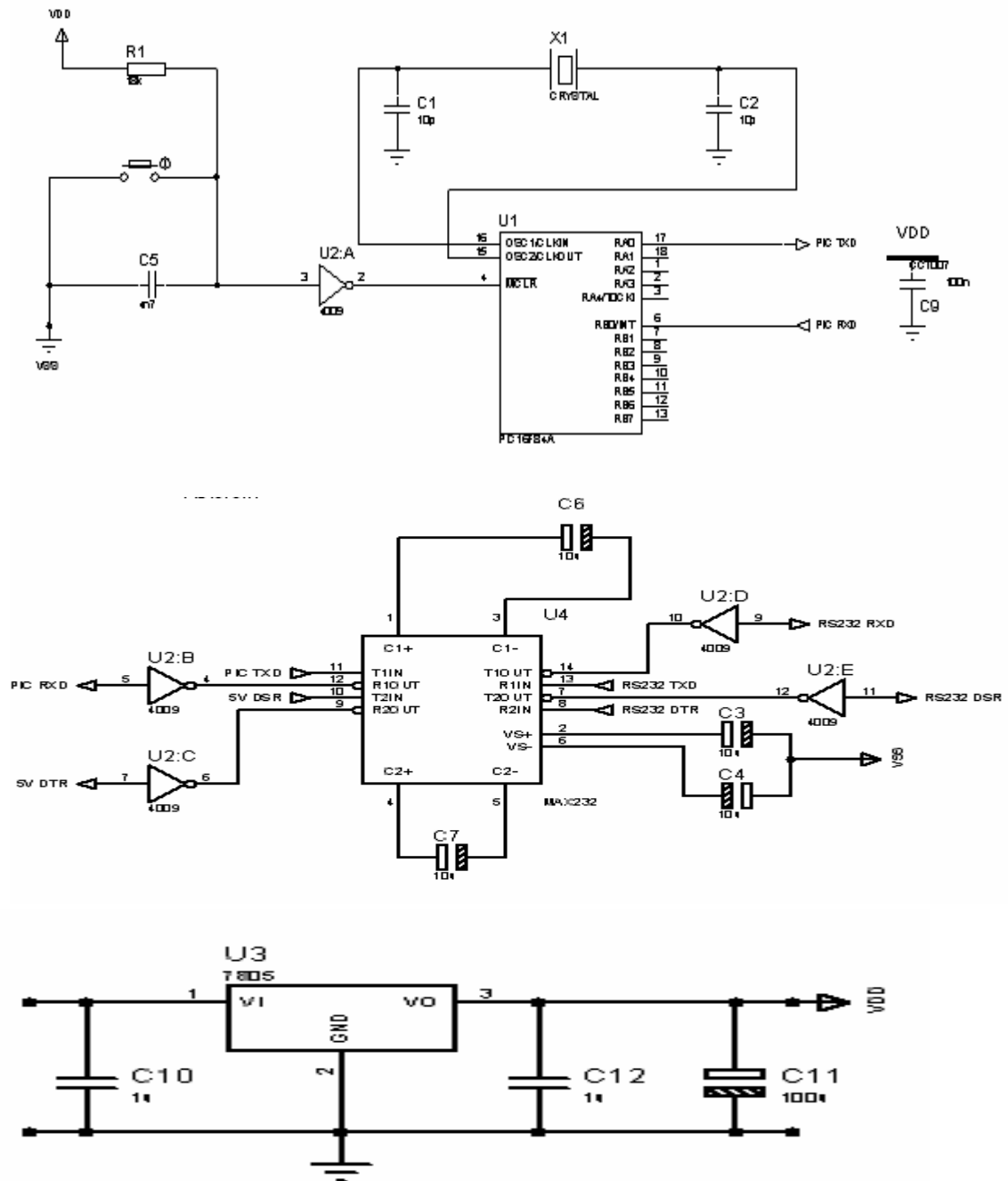


Figure 3.17 Circuit diagram for the whole system

3.6 Final Prototype

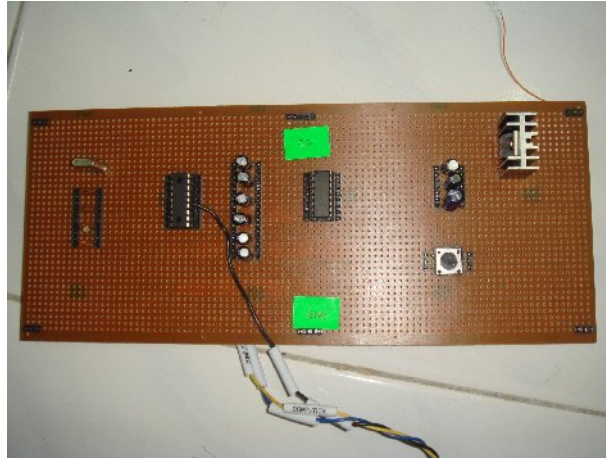


Figure 3.18 Main Circuit



Figure 3.19 Motor and encoder

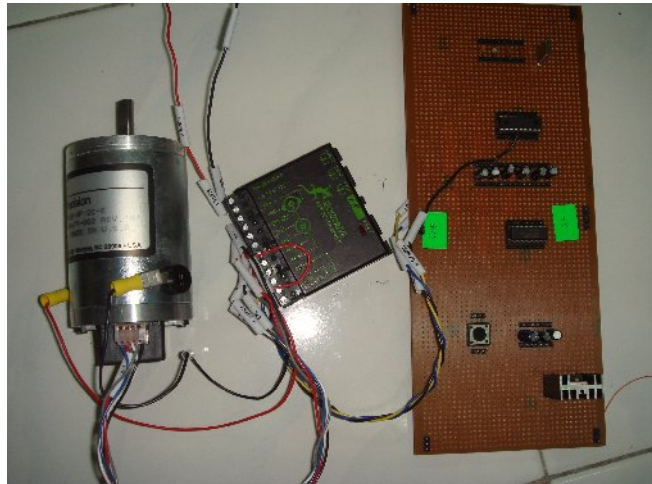


Figure 3.20 The whole system

CHAPTER 4

RESULT AND ANALYSIS

This chapter discusses the result of simulation using no controller, Proportional, Proportional-Integral, Proportional-Derivative, and Proportional-Integral-Derivative Controller by MATLAB software.

4.1 No Controller

The result simulates using no controller is shown below:

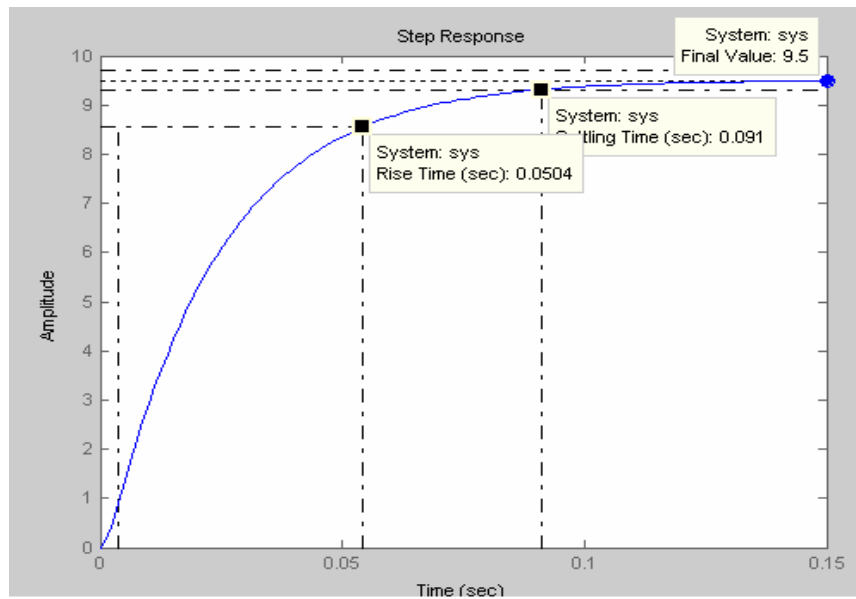


Figure 4.1 No controller

Table 4.1 No controller

COMPARISON	TIME RISE (s)	SETTLING TIME (s)	OVERSHOOT (%)	STEADY STATE
NO PID	0.0504	0.091	0	9.5

Table 4.1 show the system using no controller has a large of value time rise, settling time and steady state but it has no overshoot. The steady state of the system is calculated using the input substitution. Refer equation 4.1 to 4.3.

$$A = \begin{bmatrix} -675 & -26.25 \\ 1050 & -0.093 \end{bmatrix} \quad B = \begin{bmatrix} -250 \\ 0 \end{bmatrix} \quad C = [0 \ 1]$$

$$\begin{aligned} E(\infty) &= 1 + CA^{-1}B \\ &= 1 + [26.25 \ 0.093] \begin{bmatrix} 250 \\ 0 \end{bmatrix} \\ &= 1 + (6562.5 + 0) \\ &= 6563.5 \end{aligned} \tag{4.1}$$

$$E(\infty) = \left\| \lim (1 + CA^{-1}B) + (1 + C(A^{-1})^2 B) \right\| \tag{4.2}$$

$$\begin{aligned} 1 + C(A^{-1})^2 B &= [0 \ 1] \begin{bmatrix} 428062.5 & -708847.65 \\ 17721.19 & -27562.49 \end{bmatrix} \begin{bmatrix} 250 \\ 0 \end{bmatrix} \\ &= [17721.19 \ -27562.49] \begin{bmatrix} 250 \\ 0 \end{bmatrix} \\ &= 4430297.5 \end{aligned}$$

$$\begin{aligned} E(\infty) &= \left[\lim_{t \rightarrow \infty} (1 + CA^{-1}B)t + (1 + C(A^{-1})^2 B) \right] \\ &= [\lim(6563.5)t + (4430297.5)] \\ &= \infty \end{aligned} \tag{4.3}$$

4.2 Proportional Controller

The performance of Proportional Controller result simulate by tuning the value of proportional gain, K_p are shown in Figure 4.2 to Figure 4.6 using the value of 70, 228, 386, 594 and 700.

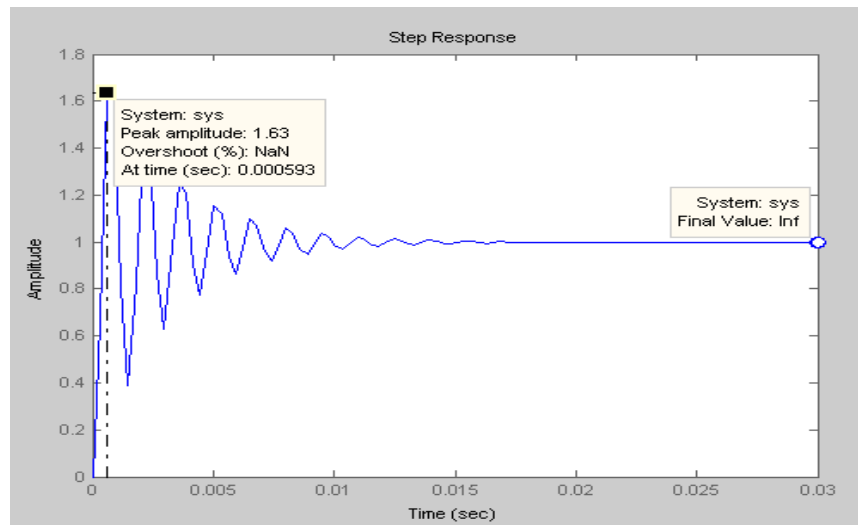


Figure 4.2 Proportional controller $K_p=70$

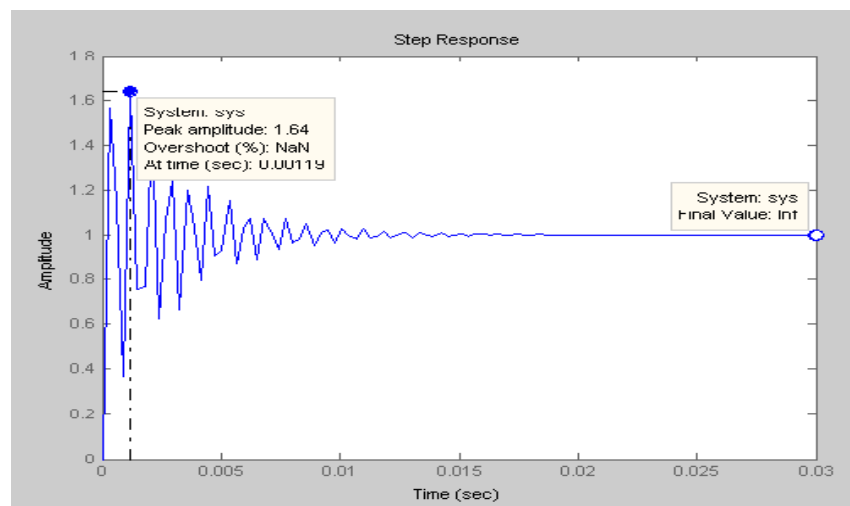


Figure 4.3 Proportional controller $K_p=228$

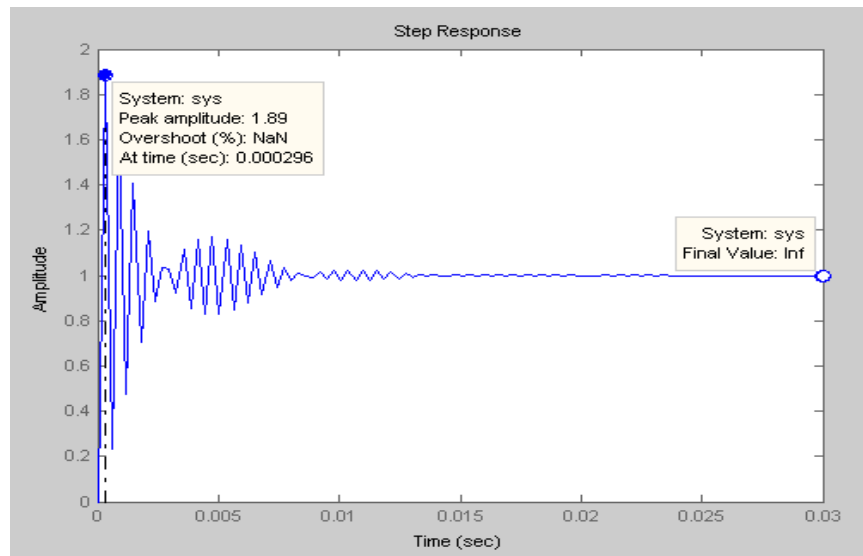


Figure 4.4 Proportional controller $K_p=386$

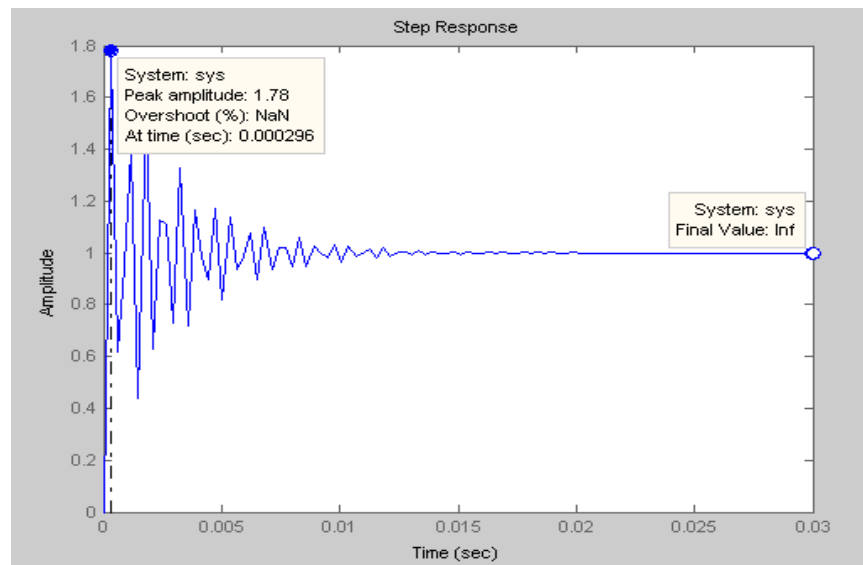


Figure 4.5 Proportional controller $K_p=594$

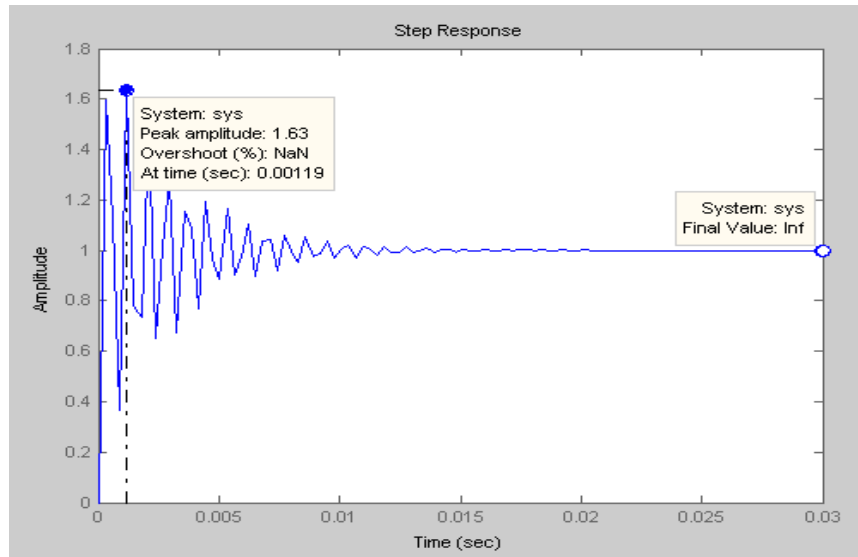


Figure 4.6 Proportional controller $K_p=700$

Table 4.2 Comparison of Proportional controller

COMPARISON	PEAK	TIME(s)	OVERSHOOT	STEADY STATE
K_p	AMPLITUDE		(%)	
70	1.63	0.000593	NaN	Inf
228	1.64	0.00119	NaN	Inf
386	1.89	0.000296	NaN	Inf
594	1.78	0.000296	NaN	Inf
700	1.63	0.00119	NaN	Inf

Table 4.2 show increasing the value of Proportional gain, K_p reducing the time to achieve the peak amplitude but in high value of peak amplitude. The overshoot is NaN (not a number) and the steady state is infinity.

4.3 Proportional-Integral Controller

The performance of Proportional-Integral Controller result simulate by tuning the value of proportional gain, K_p and Integral gain, K_i are shown in Figure 4.7 to Figure 4.10 using the value of Proportional gain, K_p 70 and 700 and the value of Integral gain, K_i 0.518 and 51.8.

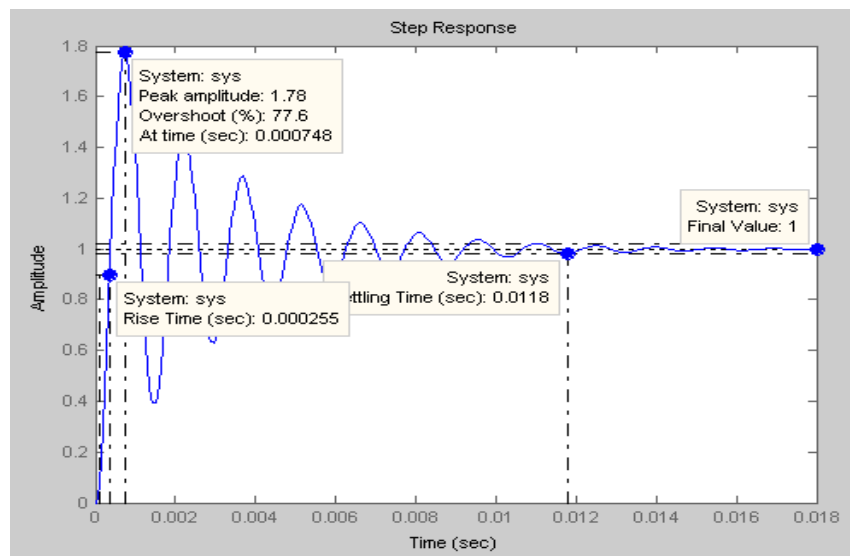


Figure 4.7 Proportional-integral controller $K_p=70$ $K_i= 0.518$

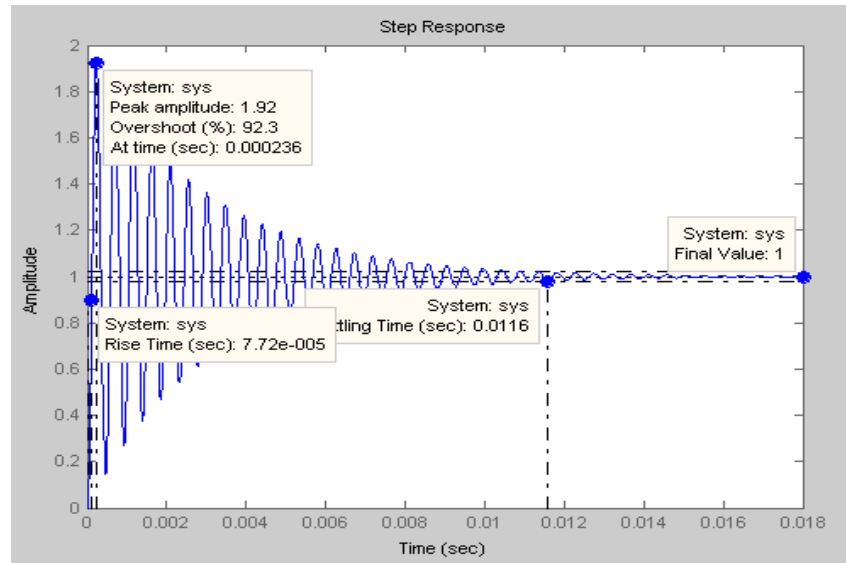


Figure 4.8 Proportional-integral controller $K_p=700$ $K_i=0.518$

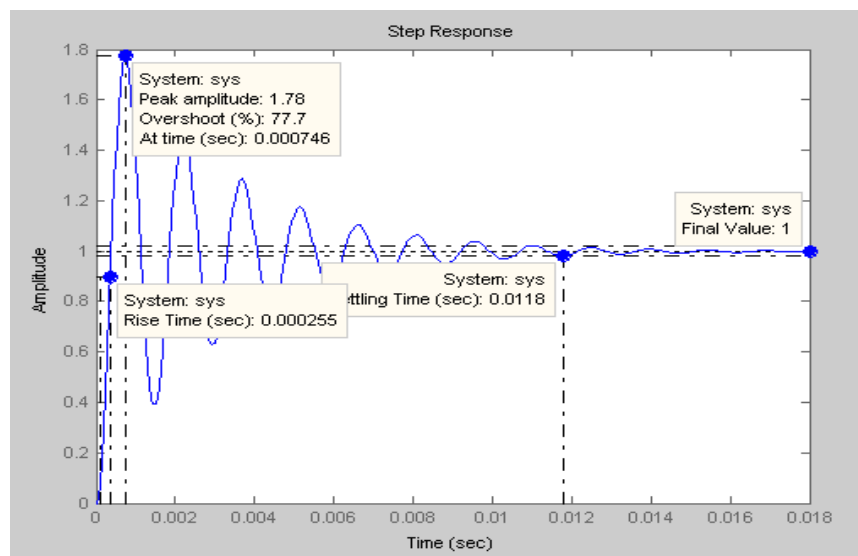


Figure 4.9 Proportional-integral controller $K_p=70$ $K_i=51.8$

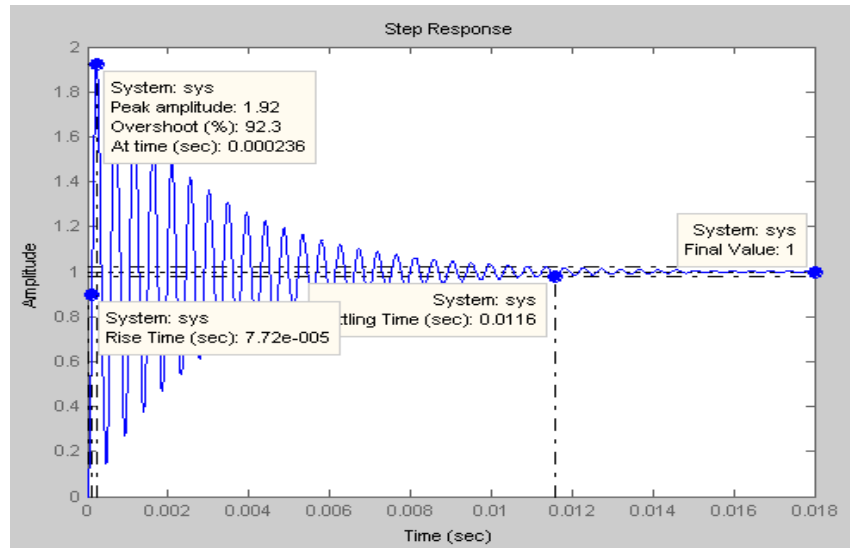


Figure 4.10 Proportional-integral controller $K_p=700$ $K_i= 51.8$

Table 4.3 Comparison of Proportional-integral controller

COMPARISON		PEAK	TIME	OVER	TIME	SETTLING	STEADY
K_p	K_i	AMPLITUDE	(s)	SHOOT (%)	RISE(s)	TIME (s)	STATE
70	0.518	1.78	0.000748	77.6	0.000255	0.0118	1
700	0.518	1.92	0.000236	92.3	0.052	0.0116	1
70	51.8	1.78	0.000748	77.7	0.000255	0.0118	1
700	51.8	1.92	0.000236	92.3	0.052	0.0116	1

Table 4.3 show reducing the value of Proportional gain, K_p and Integral gain, K_i will reduce the value of time rise, settling time, steady state and also reduce the time to achieve the peak amplitude.

4.4 Proportional-Derivative Controller

The performance of Proportional-Derivative Controller result simulate by tuning the value of proportional gain, K_p and Derivative gain, K_d are shown in Figure 4.11 to Figure 4.14 by using the value of Proportional gain, K_p 140 and 700, the value of Derivative gain, K_d 2.59 and 51.8.

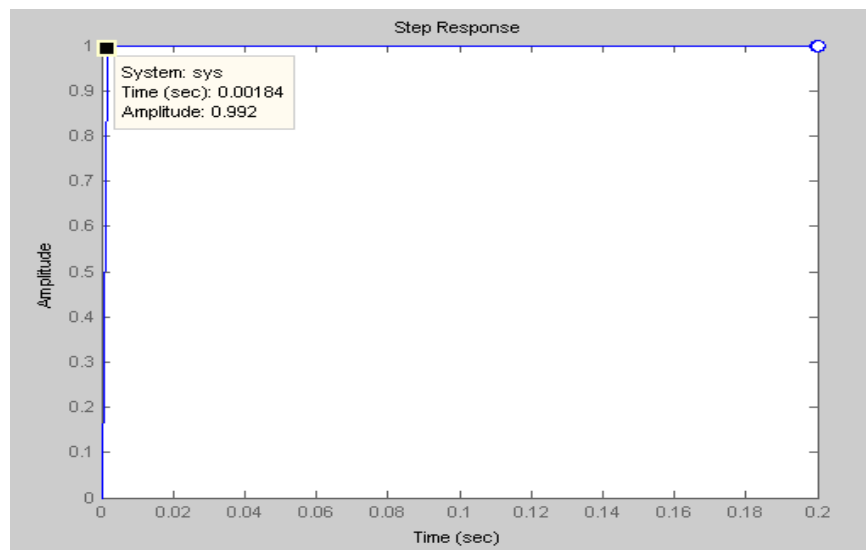


Figure 4.11 Proportional-derivative controller $K_p=140$ $K_d= 2.59$

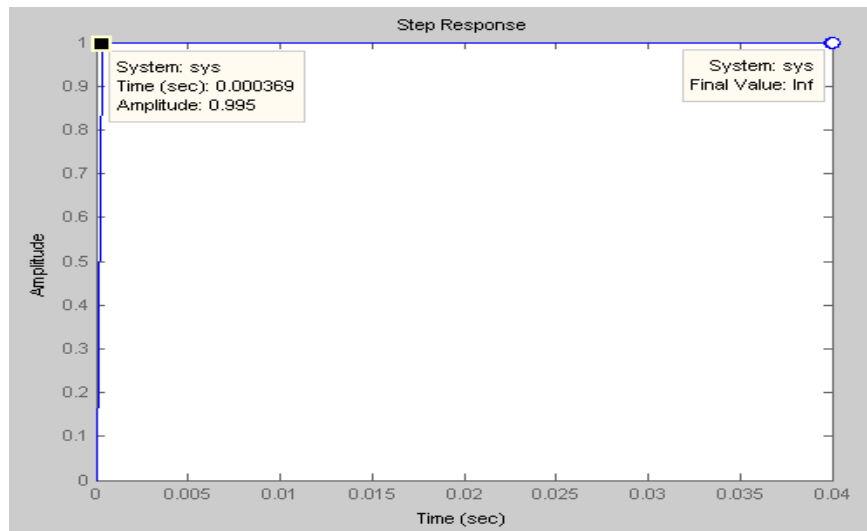


Figure 4.12 Proportional-derivative controller $K_p=700$ $K_d=2.59$

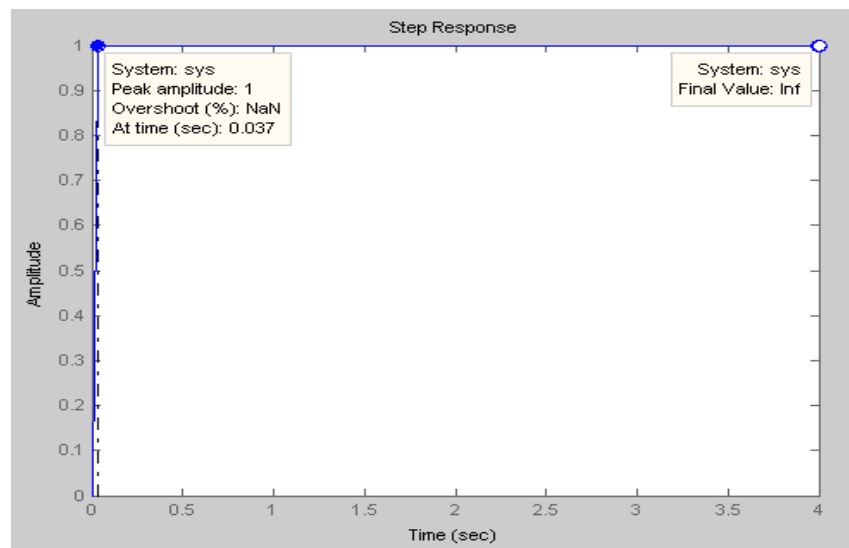


Figure 4.13 Proportional-derivative controller $K_p=140$ $K_d=51.8$

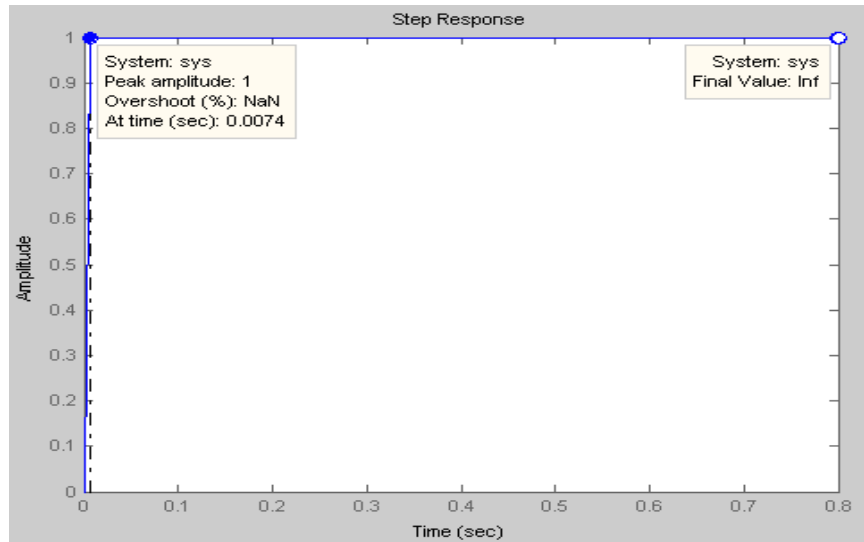


Figure 4.14 Proportional-derivative controller $K_p=700$ $K_d= 51.8$

Table 4.4 Comparison of Proportional-derivative controller

COMPARISON		PEAK	TIME(s)	OVER	STEADY
K_p	K_d	AMPLITUDE		SHOOT (%)	STATE
140	2.59	0.992	0.00184	NaN	Inf
700	2.59	0.995	0.000369	NaN	Inf
140	51.8	1	0.037	NaN	Inf
700	51.8	1	0.0074	NaN	Inf

Table 4.4 show increasing the value of Proportional gain, K_p and reducing the value of Derivative gain, K_d will reduce time to achieve the peak amplitude. The percentage overshoot is NaN (not a number) and the steady state is infinity.

4.5 Proportional-Integral-Derivative Controller

The performance of Proportional-Integral-Derivative (PID) Controller result simulate by tuning the value of proportional gain, K_p , Integral gain, K_i and Derivative gain, K_d are shown in Figure 4.15 to Figure 4.22 using the value of Proportional gain, K_p 70 and 700 and the value of Integral gain, K_i 5.18 and 518, and the value of Derivative gain, K_d 2.59 and 51.8.

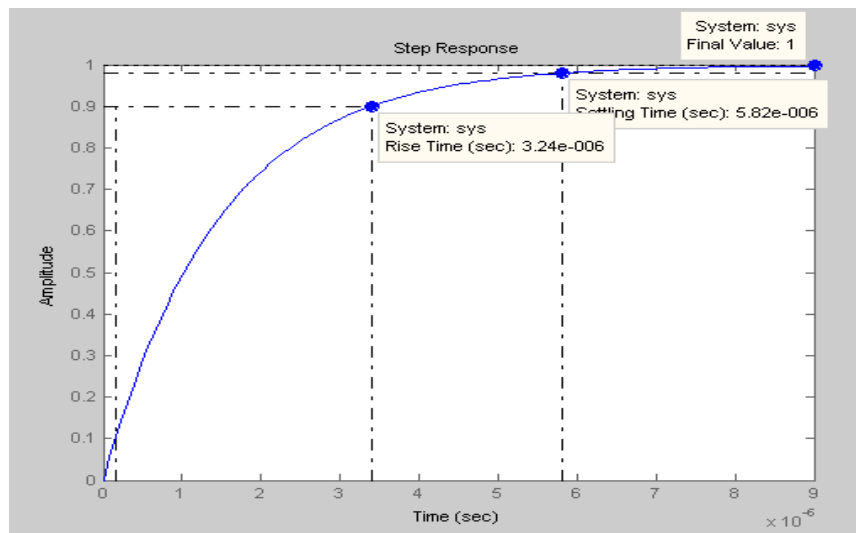


Figure 4.15 PID controller $K_p=140$ $K_i=5.18$ $K_d=2.59$

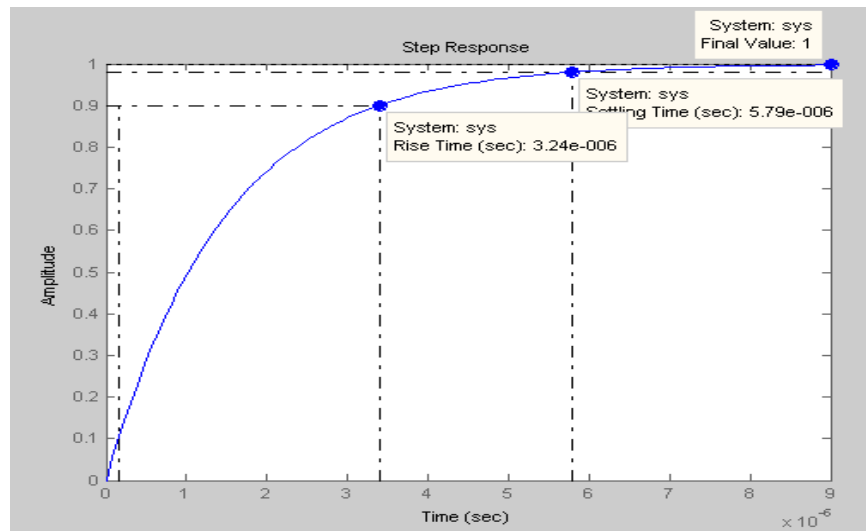


Figure 4.16 PID controller $K_p=700$ $K_i=5.18$ $K_d=2.59$

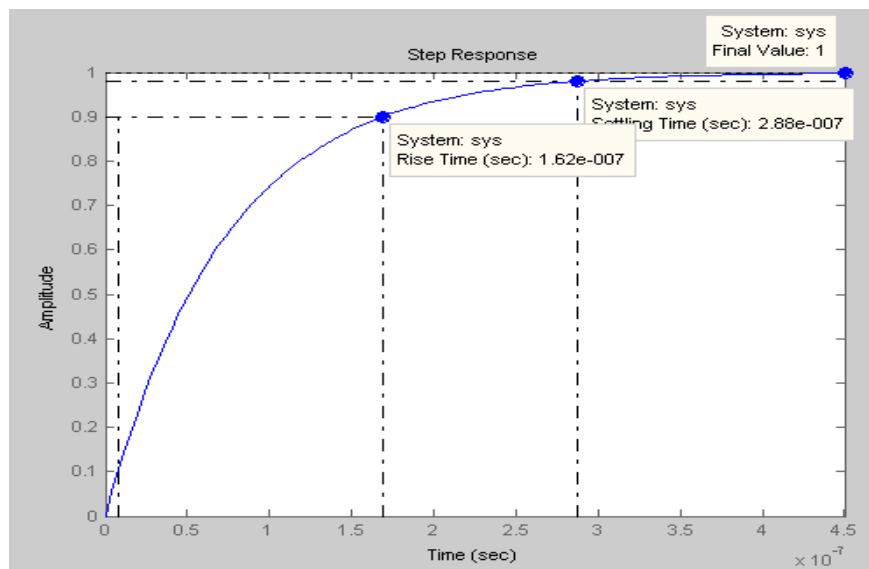


Figure 4.17 PID controller $K_p=140$ $K_i=518$ $K_d=51.8$

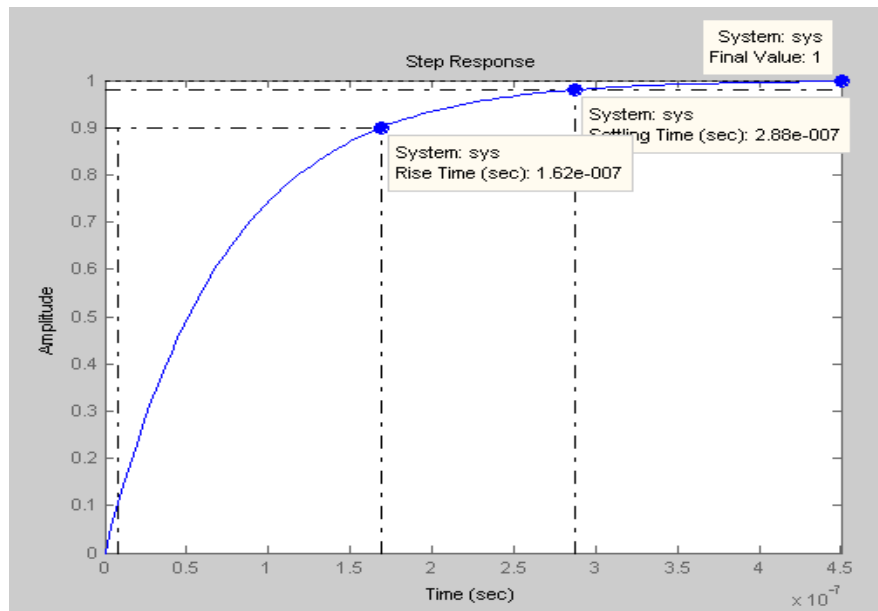


Figure 4.18 PID controller $K_p=700$ $K_i=518$ $K_d=51.8$

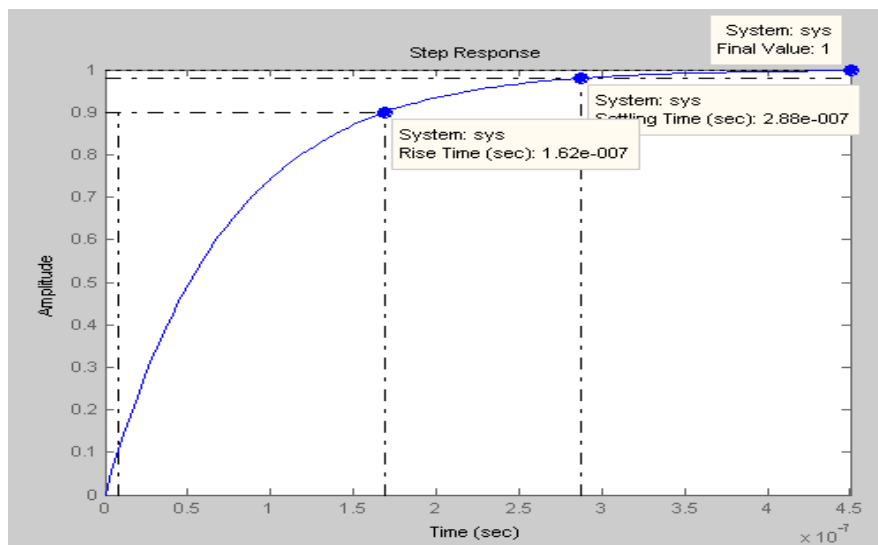


Figure 4.19 PID controller $K_p=140$ $K_i=5.18$ $K_d=51.8$

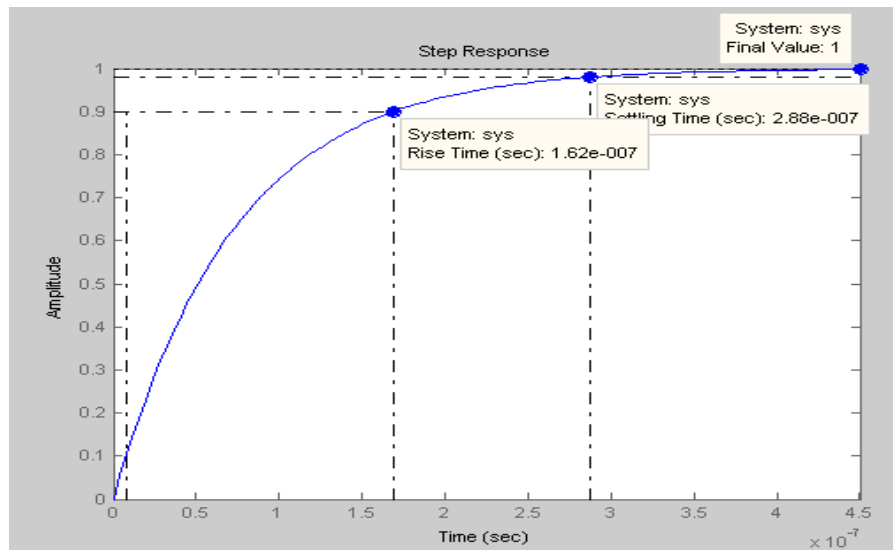


Figure 4.20 PID controller $K_p=700$ $K_i=5.18$ $K_d=51.8$

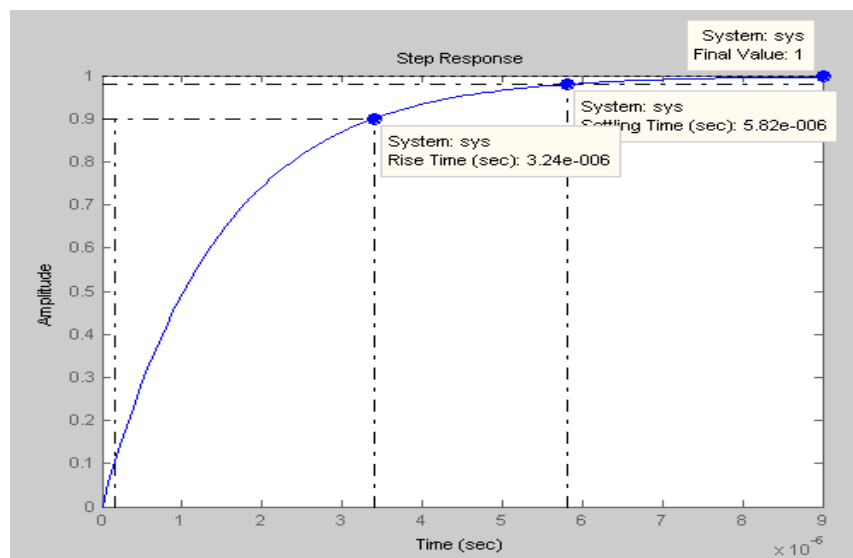


Figure 4.21 PID controller $K_p=140$ $K_i=518$ $K_d=2.59$

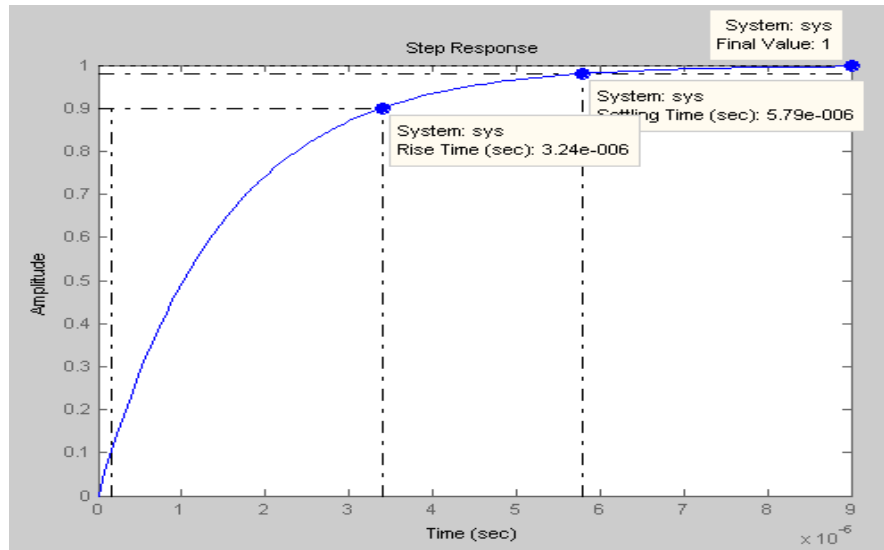


Figure 4.22 PID controller $K_p=700$ $K_i=518$ $K_d=2.59$

Table 4.5 Comparison of PID controller

COMPARISON			TIME RISE(s)	SETTLING TIME (s)	STEADY STATE
K_p	K_i	K_d			
140	5.18	2.59	0.00803	0.0144	1
700	5.18	2.59	0.00803	0.0144	1
140	518	51.8	0.00148	0.00262	1
700	518	51.8	0.00148	0.00262	1
140	5.18	51.8	0.00148	0.00262	1
700	5.18	51.8	0.00148	0.00262	1
140	518	2.59	0.00803	0.0144	1
700	518	2.59	0.00803	0.0144	1

Table 4.5 show increasing the value of Integral gain, K_i and Derivative gain, K_d will reduce the value of time rise, settling time, and steady state.

According to the graphs and tables in this chapter, it can prove that using Proportional-Integral-Derivative (PID) Controller is best controller compared to using no controller, Proportional Controller, Proportional-Integral (PI) Controller and Proportional-Derivative (PD) Controller

CHAPTER 5

CONCLUSION AND RECOMMENDATION

5.1 Conclusion

The controlling speed of DC servo motor using Proportional-Integral-Derivative (PID) algorithm had been presented using m-file simulation in MATLAB and PIC microcontroller is an implementation to prove it. This project's objective is not successfully achieved. The simulation of controlling the speed of DC servo motor using PID algorithm is successfully simulate in MATLAB but had some problem to implement PID algorithm in PIC microcontroller. It is because of difficulty to develop the program of the PIC microcontroller. Besides that, it also had problem to interface DC servo motor to the circuit hardware.

5.2 Recommendation

There are some suggestions for the future work

- i. Make simulation of controlling the speed of DC motor using other software such as LABVIEW.
- ii. Combining PID controller with the other controller such as fuzzy logic in order to get a high performance system.

5.3 Commercialization

Making analysis for the project is the first step before make real project. The analysis is doing based on the simulation of the project. The successfully of the project is depend on the analysis of the simulation that be done before.

In industry, the engineers always make analysis of the performance on the machine to improve quality and production and also to overcome their problems. By making the simulation on that machine using various tools is the important step for some mostly project.

This project presented the simulation by using MATLAB/Simulink as a tool in making analysis. So, this project can be used as reference for the engineering student or any engineer that work with simulation and doing analysis.

5.4 List and Cost of the Component

Table 5.1 List and cost of components

Components	Specifications	Quantity	Cost
PIC16F84		1	25.00
IC base	18 pin	1	0.20
	16 pin	1	0.18
Capacitor	10pF	2	0.16
	4.7nF	1	0.08
	100nF	2	0.20
	10uF	4	0.48
	1uF	2	0.24
	100uF	1	0.20
Resistor	16K Ω	1	0.04
Header		10	8.00
Heat Sink		1	0.70
Regulator 7805		1	2.00
Reset switch		1	0.60
Crystal 4MHz		1	1.90
Wire Wrap		1	40.00
Wrap Tool		1	-
RS232		1	0.60
MAX232		1	4.00
Ribbon Wire		1	3.30
Strip Board		1	5.00
TOTAL			92.88

The total estimation for this project is approximately RM92.88. This cost of this project does not involve the cost of DC servo motor. The amount is affordable.

REFERENCES

- [1] 1 August 2008, Citing Internet Sources URL
<http://www.hansen-motor.com/servo-motors.htm>

- [2] 15 February 2008, Citing Internet Sources URL
<http://www.geocities.com/nozomsite/pic6.htm>

- [3] 14 February 2008, Citing Internet Sources URL
http://en.wikipedia.org/wiki/PID_controller

- [4] 20 July 2008, Citing Internet Sources URL
http://en.wikipedia.org/wiki/Ziegler%E2%80%93Nichols_method

- [5] 14 February 2008, Citing Internet Sources URL
<http://www.jashaw.com/pid/tutorial/pid6.html>

- [6] Sergey E. Lyshevski. Electromechanical Systems, Electric Machines, and Applied Mechatronics. Boca Raton, Florida. CRC Press LLC. 2000

- [7] D.W. Smith. PIC in Practice, A Project-Based Approach. 2nd Ed. Jordan Hill. Elsevier

[8] 25 Mac 2008, Citing Internet Sources URL

<http://www.seattlerobotics.org/encoder/200001/simplemotor.htm>

[9] 1 April 2008, Citing Internet Sources URL

<http://www.mstarlabs.com/control/znrule.html>

[10] 29 June 2008, Citing Internet Sources URL

<http://www.sciencedirect.com/science>

[11] Dogan Ibrahim. PIC BASIC Projects, 30 Projects Using PIC BASIC and PIC BASIC PRO. Jordan Hill. Elsevier

APPENDIX A
Project Programming

PIC 16F84 Project Programming

```
motor var portb.0
```

```
trisb = 0
```

```
trisa = 1
```

```
backward:  PULSOUT 0,130  
           PULSOUT 1,170  
           PAUSE 20  
           goto backward
```

```
motor var portb.0
```

```
J var word
```

```
trisb = 0
```

```
loop:  PULSOUT 0,130  
       PAUSE 20  
       goto loop
```

```
loop2: PULSOUT 1,170  
       PAUSE 20  
       goto loop2
```

```
end
```

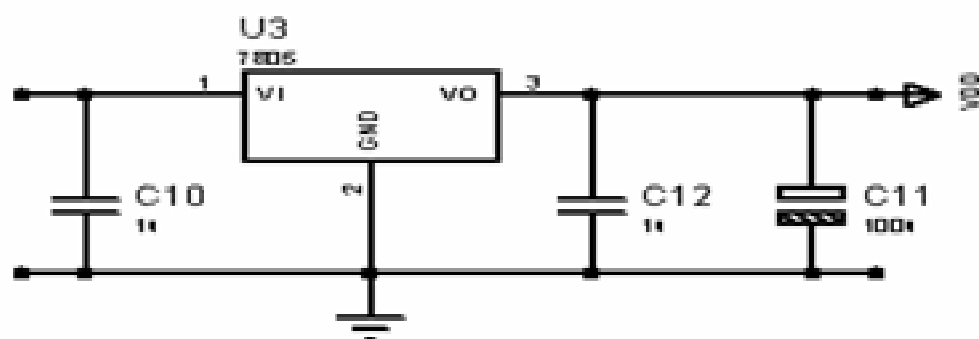
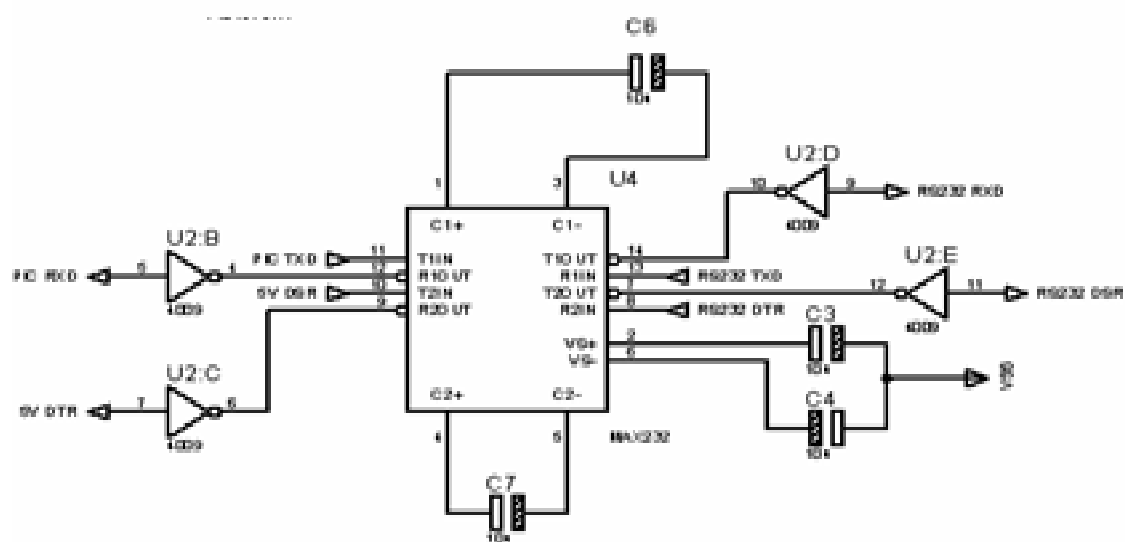
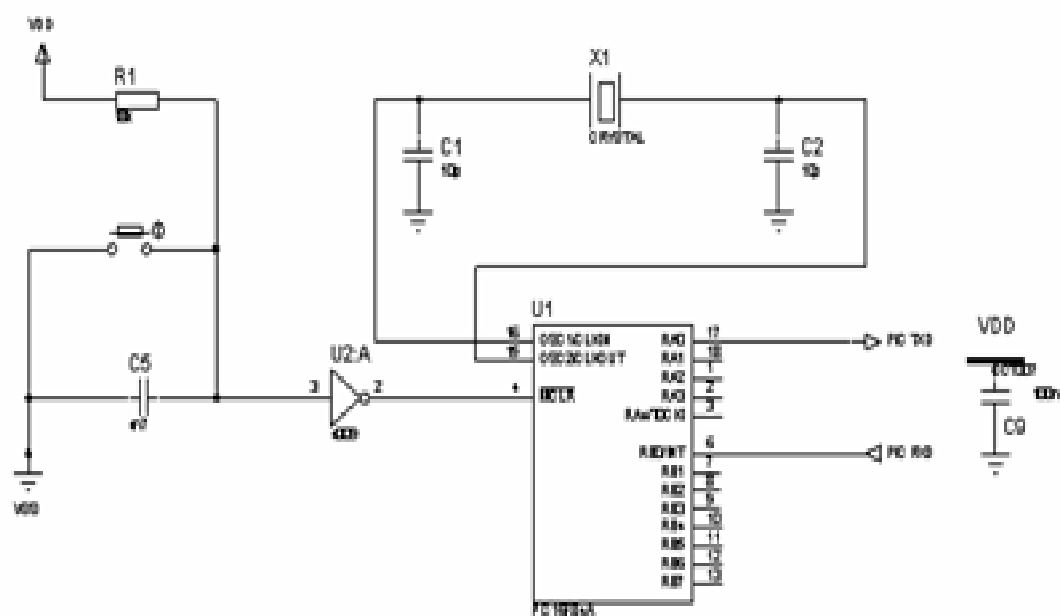
MATLAB Project Programming

```
J=0.01;
b=0.1;
K=0.01;
R=1;
L=0.5;

A=[-b/J  K/J
    -K/L  -R/L];
B=[0
    1/L];
C=[1  0];
D=0;
[num,den]=ss2tf(A, B, C, D);
num=num(3);

Kp=100;
Ki=1;
Kd=1;
numc=[Kd, Kp, Ki];
denc=[1 0];
numa=conv(num,numc);
dena=conv(den,denc);
[numac,denac]=cloop(numa,dena);
step(numac,denac)
```

APPENDIX B
Whole Circuit Diagram



APPENDIX C
PIC 16F84A Datasheet



PIC16F84A

18-pin Enhanced Flash/EEPROM 8-Bit Microcontroller

Devices Included in this Data Sheet:

- PIC16F84A
- Extended voltage range device available (PIC16LF84A)

High Performance RISC CPU Features:

- Only 35 single word instructions to learn
- All instructions single cycle except for program branches which are two-cycle
- Operating speed: DC - 20 MHz clock input
DC - 200 ns instruction cycle
- 1024 words of program memory
- 68 bytes of data RAM
- 64 bytes of data EEPROM
- 14-bit wide instruction words
- 8-bit wide data bytes
- 15 special function hardware registers
- Eight-level deep hardware stack
- Direct, indirect and relative addressing modes
- Four interrupt sources:
 - External RBO/INT pin
 - TMR0 timer overflow
 - PORTB<7:4> interrupt on change
 - Data EEPROM write complete

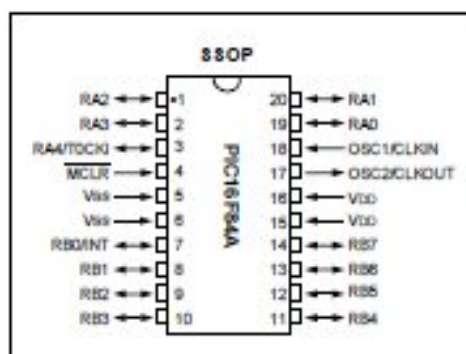
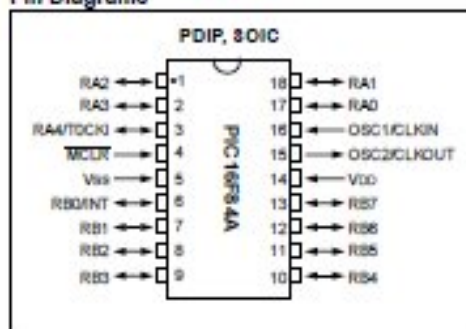
Peripheral Features:

- 13 I/O pins with individual direction control
- High current sink/source for direct LED drive
 - 25 mA sink max. per pin
 - 25 mA source max. per pin
- TMR0: 8-bit timer/counter with 8-bit programmable prescaler

Special Microcontroller Features:

- 1000 erase/write cycles Enhanced Flash program memory
- 1,000,000 typical erase/write cycles EEPROM data memory
- EEPROM Data Retention > 40 years
- In-Circuit Serial Programming (ICSP™) - via two pins
- Power-on Reset (POR), Power-up Timer (PWRT), Oscillator Start-up Timer (OST)
- Watchdog Timer (WDT) with its own on-chip RC oscillator for reliable operation
- Code-protection
- Power saving SLEEP mode
- Selectable oscillator options

Pin Diagrams



CMOS Enhanced Flash/EEPROM Technology:

- Low-power, high-speed technology
- Fully static design
- Wide operating voltage range:
 - Commercial: 2.0V to 5.5V
 - Industrial: 2.0V to 5.5V
- Low power consumption:
 - < 2 mA typical @ 5V, 4 MHz
 - 15 µA typical @ 2V, 32 kHz
 - < 0.5 µA typical standby current @ 2V

PIC16F84A

1.0 DEVICE OVERVIEW

This document contains device-specific information for the operation of the PIC16F84A device. Additional information may be found in the PICmicro™ Mid-Range Reference Manual, (DS33023), which may be downloaded from the Microchip website. The Reference Manual should be considered a complementary document to this data sheet, and is highly recommended reading for a better understanding of the device architecture and operation of the peripheral modules.

The PIC16F84A belongs to the mid-range family of the PICmicro™ microcontroller devices. A block diagram of the device is shown in Figure 1-1.

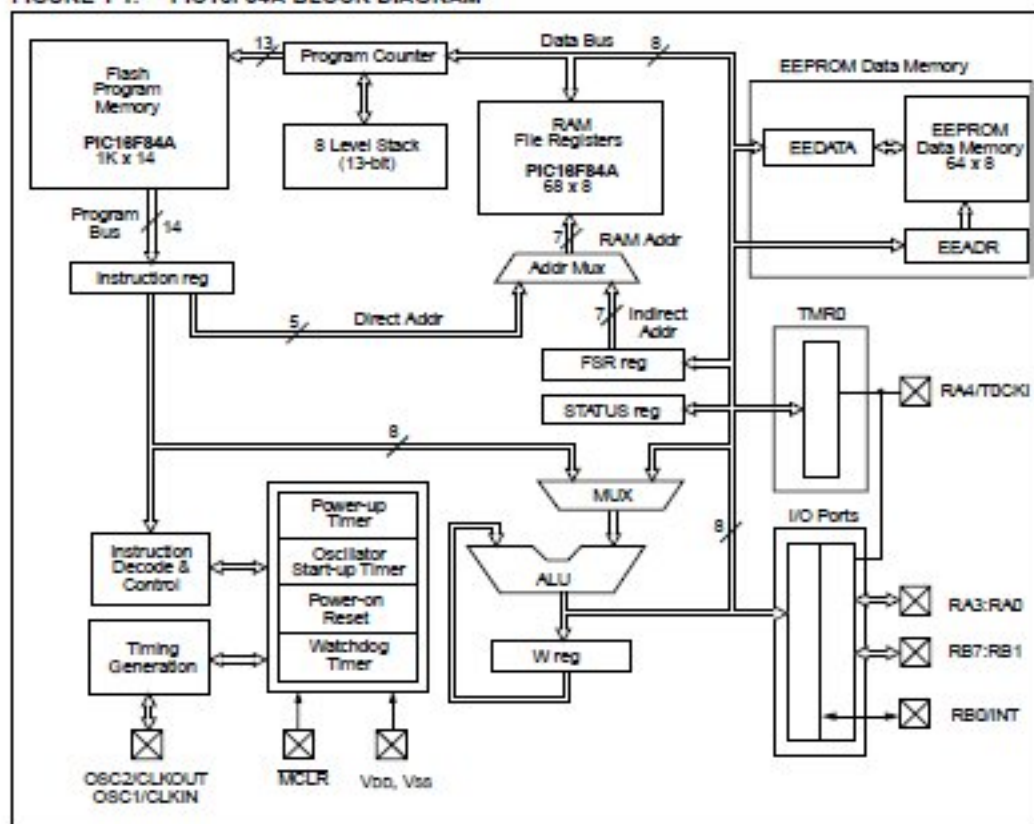
The program memory contains 1K words, which translates to 1024 instructions, since each 14-bit program memory word is the same width as each device instruction. The data memory (RAM) contains 68 bytes. Data EEPROM is 64 bytes.

There are also 13 I/O pins that are user-configured on a pin-to-pin basis. Some pins are multiplexed with other device functions. These functions include:

- External interrupt
- Change on PORTB interrupt
- Timer0 clock input

Table 1-1 details the pinout of the device with descriptions and details for each pin.

FIGURE 1-1: PIC16F84A BLOCK DIAGRAM



PIC16F84A

TABLE 1-1 PIC16F84A PINOUT DESCRIPTION

Pin Name	DIP No.	SOIC No.	SSOP No.	I/O/P Type	Buffer Type	Description
OSC1/CLKIN	16	16	18	I	ST/CMOS ⁽²⁾	Oscillator crystal input/external clock source input.
OSC2/CLKOUT	15	15	19	O	—	Oscillator crystal output. Connects to crystal or resonator in crystal oscillator mode. In RC mode, OSC2 pin outputs CLKOUT which has 1/4 the frequency of OSC1, and denotes the instruction cycle rate.
MCLR	4	4	4	I/P	ST	Master clear (reset) input/programming voltage input. This pin is an active low reset to the device.
RA0	17	17	19	I/O	TTL	PORTA is a bi-directional I/O port. Can also be selected to be the clock input to the TMR0 timer/counter. Output is open drain type.
RA1	18	18	20	I/O	TTL	
RA2	1	1	1	I/O	TTL	
RA3	2	2	2	I/O	TTL	
RA4/T0CKI	3	3	3	I/O	ST	
RB0/INT	6	6	7	I/O	TTL/ST ⁽¹⁾	PORTB is a bi-directional I/O port. PORTB can be software programmed for internal weak pull-up on all inputs. RB0/INT can also be selected as an external interrupt pin. Interrupt on change pin. Interrupt on change pin. Interrupt on change pin. Serial programming clock. Interrupt on change pin. Serial programming data.
RB1	7	7	8	I/O	TTL	
RB2	8	8	9	I/O	TTL	
RB3	9	9	10	I/O	TTL	
RB4	10	10	11	I/O	TTL	
RB5	11	11	12	I/O	TTL	
RB6	12	12	13	I/O	TTL/ST ⁽²⁾	
RB7	13	13	14	I/O	TTL/ST ⁽²⁾	
Vss	5	5	5,6	P	—	Ground reference for logic and I/O pins.
Vdd	14	14	15,16	P	—	Positive supply for logic and I/O pins.

Legend: I = input O = output I/O = Input/Output P = power
 — = Not used TTL = TTL input ST = Schmitt Trigger input

- Note 1: This buffer is a Schmitt Trigger input when configured as the external interrupt.
 2: This buffer is a Schmitt Trigger input when used in serial programming mode.
 3: This buffer is a Schmitt Trigger input when configured in RC oscillator mode and a CMOS input otherwise.

PIC16F84A

2.0 MEMORY ORGANIZATION

There are two memory blocks in the PIC16F84A. These are the program memory and the data memory. Each block has its own bus, so that access to each block can occur during the same oscillator cycle.

The data memory can further be broken down into the general purpose RAM and the Special Function Registers (SFRs). The operation of the SFRs that control the "core" are described here. The SFRs used to control the peripheral modules are described in the section discussing each individual peripheral module.

The data memory area also contains the data EEPROM memory. This memory is not directly mapped into the data memory, but is indirectly mapped. That is, an indirect address pointer specifies the address of the data EEPROM memory to read/write. The 64 bytes of data EEPROM memory have the address range 0h-3Fh. More details on the EEPROM memory can be found in Section 5.0.

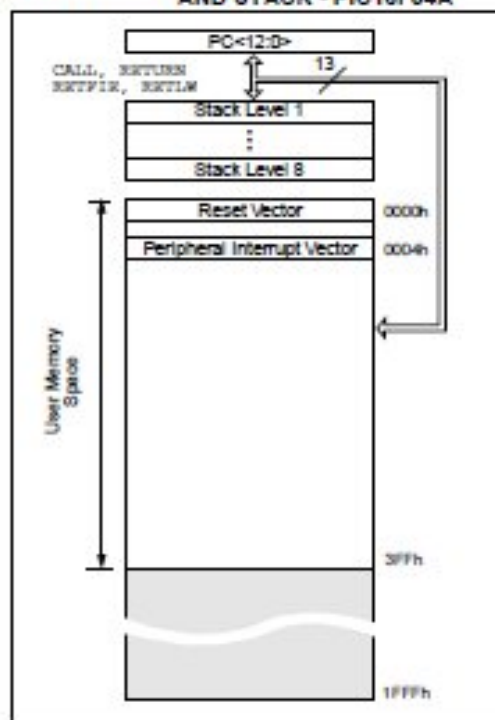
Additional information on device memory may be found in the PICmicro™ Mid-Range Reference Manual, (DS33023).

2.1 Program Memory Organization

The PIC16FXX has a 13-bit program counter capable of addressing an 8K x 14 program memory space. For the PIC16F84A, the first 1K x 14 (0000h-03FFh) are physically implemented (Figure 2-1). Accessing a location above the physically implemented address will cause a wraparound. For example, for locations 20h, 420h, 820h, C20h, 1020h, 1420h, 1820h, and 1C20h will be the same instruction.

The reset vector is at 0000h and the interrupt vector is at 0004h.

FIGURE 2-1: PROGRAM MEMORY MAP AND STACK - PIC16F84A



PIC16F84A

2.2.2 SPECIAL FUNCTION REGISTERS

The Special Function Registers (Figure 2-1 and Table 2-1) are used by the CPU and Peripheral functions to control the device operation. These registers are static RAM.

The special function registers can be classified into two sets, core and peripheral. Those associated with the core functions are described in this section. Those related to the operation of the peripheral features are described in the section for that specific feature.

TABLE 2-1 REGISTER FILE SUMMARY

Addr	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on Power-on Reset	Value on all other resets (Note3)	
Bank 0												
00h	INDF	Uses contents of FSR to address data memory (not a physical register)								----	----	
01h	TMR0	8-bit real-time clock/counter								XXXX XXXX	XXXX XXXX	
02h	PCL	Low order 8 bits of the Program Counter (PC)								0000 0000	0000 0000	
03h	STATUS ⁽²⁾	IRP	RP1	RP0	TO	PD	Z	DC	C	0001 1xxx	000q quuu	
04h	FSR	Indirect data memory address pointer 0								XXXX XXXX	XXXX XXXX	
05h	PORTA ⁽⁴⁾	—	—	—	RA4/T0CK1	RA3	RA2	RA1	RA0	---x xxxx	---u uuuu	
06h	PORTB ⁽⁵⁾	RB7	RB6	RB5	RB4	RB3	RB2	RB1	RB0/INT	XXXX XXXX	XXXX XXXX	
07h		Unimplemented location, read as '0'								-----	-----	
08h	EEDATA	EEPROM data register								XXXX XXXX	XXXX XXXX	
09h	EEADR	EEPROM address register								XXXX XXXX	XXXX XXXX	
0Ah	PCLATH	—	—	—	Write buffer for upper 5 bits of the PC ⁽¹⁾				---	0 0000	---	0 0000
0Bh	INTCON	GIE	EEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF	0000 000x	0000 000u	
Bank 1												
80h	INDF	Uses contents of FSR to address data memory (not a physical register)								----	----	
81h	OPTION_REG	RBP0	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0	1111 1111	1111 1111	
82h	PCL	Low order 8 bits of Program Counter (PC)								0000 0000	0000 0000	
83h	STATUS ⁽²⁾	IRP	RP1	RP0	TO	PD	Z	DC	C	0001 1xxx	000q quuu	
84h	FSR	Indirect data memory address pointer 0								XXXX XXXX	XXXX XXXX	
85h	TRISA	—	—	—	PORTA data direction register				---	1 1111	---	1 1111
86h	TRISB	PORTB data direction register								1111 1111	1111 1111	
87h		Unimplemented location, read as '0'								-----	-----	
88h	EECON1	—	—	—	EEIF	WRERR	WREN	WR	RD	---0 x000	---0 q000	
89h	EECON2	EEPROM control register 2 (not a physical register)								-----	-----	
0Ah	PCLATH	—	—	—	Write buffer for upper 5 bits of the PC ⁽¹⁾				---	0 0000	---	0 0000
0Bh	INTCON	GIE	EEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF	0000 000x	0000 000u	

Legend: x = unknown, u = unchanged, — = unimplemented read as '0', q = value depends on condition.

Note 1: The upper byte of the program counter is not directly accessible. PCLATH is a slave register for PC<12:8>. The contents of PCLATH can be transferred to the upper byte of the program counter, but the contents of PC<12:8> is never transferred to PCLATH.

2: The TO and PD status bits in the STATUS register are not affected by a MCLR reset.

3: Other (non power-up) resets include: external reset through MCLR and the Watchdog Timer Reset.

4: On any device reset, these pins are configured as inputs.

5: This is the value that will be in the port output latch.

PIC16F84A

2.2.2.1 STATUS REGISTER

The STATUS register contains the arithmetic status of the ALU, the RESET status and the bank select bit for data memory.

As with any register, the STATUS register can be the destination for any instruction. If the STATUS register is the destination for an instruction that affects the Z, DC or C bits, then the write to these three bits is disabled. These bits are set or cleared according to device logic. Furthermore, the \overline{TO} and \overline{PD} bits are not writable. Therefore, the result of an instruction with the STATUS register as destination may be different than intended.

For example, `CLRF STATUS` will clear the upper-three bits and set the Z bit. This leaves the STATUS register as 000u u1uu (where u = unchanged).

Only the BCF, BSF, SWAPF and MOVWF instructions should be used to alter the STATUS register (Table 7-2) because these instructions do not affect any status bit.

Note 1: The IRP and RP1 bits (STATUS<7:6>) are not used by the PIC16F84A and should be programmed as cleared. Use of these bits as general purpose R/W bits is NOT recommended, since this may affect upward compatibility with future products.

Note 2: The C and DC bits operate as a BORROW and digit borrow out bit, respectively, in subtraction. See the SUBWF and SUBWF instructions for examples.

Note 3: When the STATUS register is the destination for an instruction that affects the Z, DC or C bits, then the write to these three bits is disabled. The specified bit(s) will be updated according to device logic.

FIGURE 2-1: STATUS REGISTER (ADDRESS 03h, 83h)

R/W-0	R/W-0	R/W-0	R-1	R-1	R/W-x	R/W-x	R/W-x
IRP	RP1	RP0	\overline{TO}	\overline{PD}	Z	DC	C
bit7							bit0

R = Readable bit
W = Writable bit
U = Unimplemented bit, read as '0'
-x = Value at POR/reset

bit 7: **IRP:** Register Bank Select bit (used for indirect addressing)
The IRP bit is not used by the PIC16F84A. IRP should be maintained clear.

bit 6-5: **RP1:RP0:** Register Bank Select bits (used for direct addressing)
00 = Bank 0 (00h - 7Fh)
01 = Bank 1 (80h - FFh)
Each bank is 128 bytes. Only bit RP0 is used by the PIC16F84A. RP1 should be maintained clear.

bit 4: **\overline{TO} :** Time-out bit
1 = After power-up, `CLRWDT` instruction, or `SLEEP` instruction
0 = A WDT time-out occurred

bit 3: **\overline{PD} :** Power-down bit
1 = After power-up or by the `CLRWDT` instruction
0 = By execution of the `SLEEP` instruction

bit 2: **Z:** Zero bit
1 = The result of an arithmetic or logic operation is zero
0 = The result of an arithmetic or logic operation is not zero

bit 1: **DC:** Digit carry/BORROW bit (for `ADDWF` and `ADDLW` instructions) (For BORROW the polarity is reversed)
1 = A carry-out from the 4th low order bit of the result occurred
0 = No carry-out from the 4th low order bit of the result

bit 0: **C:** Carry/borrow bit (for `ADDWF` and `ADDLW` instructions)
1 = A carry-out from the most significant bit of the result occurred
0 = No carry-out from the most significant bit of the result occurred
Note: For BORROW the polarity is reversed. A subtraction is executed by adding the two's complement of the second operand. For rotate (`RRF`, `RLF`) instructions, this bit is loaded with either the high or low order bit of the source register.

PIC16F84A

2.2.2.2 OPTION_REG REGISTER

The OPTION_REG register is a readable and writable register which contains various control bits to configure the TMR0/WDT prescaler, the external INT interrupt, TMR0, and the weak pull-ups on PORTB.

Note: When the prescaler is assigned to the WDT (PSA = '1'), TMR0 has a 1:1 prescaler assignment.

FIGURE 2-1: OPTION_REG REGISTER (ADDRESS 81h)

R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
RBP0	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0
bit7				bit0			

R = Readable bit
 W = Writable bit
 U = Unimplemented bit, read as '0'
 - n = Value at POR reset

bit 7: **RBP0:** PORTB Pull-up Enable bit
 1 = PORTB pull-ups are disabled
 0 = PORTB pull-ups are enabled (by individual port latch values)

bit 6: **INTEDG:** Interrupt Edge Select bit
 1 = Interrupt on rising edge of RB0/INT pin
 0 = Interrupt on falling edge of RB0/INT pin

bit 5: **T0CS:** TMR0 Clock Source Select bit
 1 = Transition on RA4/T0CKI pin
 0 = Internal instruction cycle clock (CLKOUT)

bit 4: **T0SE:** TMR0 Source Edge Select bit
 1 = Increment on high-to-low transition on RA4/T0CKI pin
 0 = Increment on low-to-high transition on RA4/T0CKI pin

bit 3: **PSA:** Prescaler Assignment bit
 1 = Prescaler assigned to the WDT
 0 = Prescaler assigned to TMR0

bit 2-0: **PS2:PS0:** Prescaler Rate Select bits

Bit Value	TMR0 Rate	WDT Rate
000	1:2	1:1
001	1:4	1:2
010	1:8	1:4
011	1:16	1:8
100	1:32	1:16
101	1:64	1:32
110	1:128	1:64
111	1:256	1:128

PIC16F84A

TABLE 3-3 PORTB FUNCTIONS

Name	Bit	Buffer Type	I/O Consistency Function
RB0/INT	bit0	TTL/ST ⁽¹⁾	Input/output pin or external interrupt input. Internal software programmable weak pull-up.
RB1	bit1	TTL	Input/output pin. Internal software programmable weak pull-up.
RB2	bit2	TTL	Input/output pin. Internal software programmable weak pull-up.
RB3	bit3	TTL	Input/output pin. Internal software programmable weak pull-up.
RB4	bit4	TTL	Input/output pin (with interrupt on change). Internal software programmable weak pull-up.
RB5	bit5	TTL	Input/output pin (with interrupt on change). Internal software programmable weak pull-up.
RB6	bit6	TTL/ST ⁽²⁾	Input/output pin (with interrupt on change). Internal software programmable weak pull-up. Serial programming clock.
RB7	bit7	TTL/ST ⁽²⁾	Input/output pin (with interrupt on change). Internal software programmable weak pull-up. Serial programming data.

Legend: TTL = TTL input, ST = Schmitt Trigger.

Note 1: This buffer is a Schmitt Trigger input when configured as the external interrupt.

2: This buffer is a Schmitt Trigger input when used in serial programming mode.

TABLE 3-4 SUMMARY OF REGISTERS ASSOCIATED WITH PORTB

Addr	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on Power-on Reset	Value on all other resets
08h	PORTB	RB7	RB6	RB5	RB4	RB3	RB2	RB1	RB0/INT	xxxx xxxx	xxxx xxxx
09h	TRISB	TRISB7	TRISB6	TRISB5	TRISB4	TRISB3	TRISB2	TRISB1	TRISB0	1111 1111	1111 1111
B1h	OPTION_REG	REPU	INTEDG	TOCS	TOSE	PSA	PS2	PS1	PS0	1111 1111	1111 1111

Legend: x = unknown, u = unchanged. Shaded cells are not used by PORTB.

APPENDIX D
Regulator 7805 Datasheet

7805 • THREE-TERMINAL POSITIVE VOLTAGE REGULATOR IC

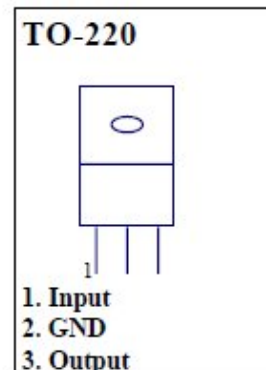
FEATURES :

- OUTPUT CURRENT IN EXCESS OF 1A;
- NO EXTERNAL COMPONENTS REQUIRED;
- INTERNAL SHORT CIRCUIT CURRENT LIMITING;
- INTERNAL THERMAL OVERLOAD PROTECTION;
- OUTPUT TRANSISTOR SAFE-AREA COMPENSATION;
- OUTPUT VOLTAGE OFFERED IN 4% TOLERANCE.

ABSOLUTE MAXIMUM RATINGS ($T_a = 25^\circ\text{C}$)

Characteristic	Symbol	Norm	Unit
Input Voltage	V_{in}	V	35
Maximum Dissipated Power(with heat sink)	$P_{tot(max)}$	W	15
Maximum Dissipated Power(without heat sink)	$P_{tot(max)}$	W	1.5
Thermal Resistance Junction to Case	θ_{jC}	$^\circ\text{C/W}$	5.0
Thermal Resistance, Junction to Air	θ_{jA}	$^\circ\text{C/W}$	65
Junction Temperature	T_j	150	$^\circ\text{C}$

$T_c = -45 + +70^\circ\text{C}$



ELECTRICAL CHARACTERISTICS

($V_{in}=10\text{V}$, $I_o=0.5\text{A}$, $C_i=0.33\text{mkF}$, $C_o=0.1\text{mkF}$, $T_j=0+125^\circ\text{C}$, unless otherwise noted.)

Characteristic	Symbol	Norm			Unit
		Min	TYP	Max	
Output Voltage($T_j=25^\circ\text{C}$)	V_o	4.8		5.2	V
Output Voltage ($5.0\text{mA} \leq I_o \leq 1.0\text{A}$, $P_o \leq 15\text{W}$) $7.0\text{V} \leq V_{in} \leq 20\text{V}$	V_o	4.75		5.25	V
Line Regulation($T_j=+25^\circ\text{C}$) $7.0\text{V} \leq V_{in} \leq 25\text{V}$ $8.0\text{V} \leq V_{in} \leq 12\text{V}$	ΔV_v			100 50	mV
Load Regulation($T_j=+25^\circ\text{C}$) $5.0\text{mA} \leq I_o \leq 1.5\text{A}$ $0.25\text{A} \leq I_o \leq 0.75\text{A}$	ΔV_i			100 50	mV
Quiescent Current($T_j=+25^\circ\text{C}$)	I_b			8.0	mA
Quiescent Current Change $7.0\text{V} \leq V_{in} \leq 25\text{V}$ $5.0\text{mA} \leq I_o \leq 1.0\text{A}$	ΔI_b			1.3 0.5	mA
Dropout Voltage($I_o=1.0\text{A}$, $T_j=+25^\circ\text{C}$)	$V_i - V_o$		2.0		V
Short Circuit Current Limit($T_a=+25^\circ\text{C}$), $V_{in}=35\text{V}$	I_{sc}		0.4		A
Peak Output Current($T_j=+25^\circ\text{C}$)	I_{max}		2.2		A
Average Temperature Coefficient of Output Voltage	TCV_o		0.3		$\text{mV}/^\circ\text{C}$

APPENDIX E

Servo Motor

Clifton Precision JDH 2250-HF-2C-E

G340 INSTALLATION NOTES

(April 3, 2008)

Thank you for purchasing the G340 drive. The G340 DC servo drive is warranted to be free of manufacturing defects for 1 year from the date of purchase. Also anyone who is dissatisfied with it or is unable to make it work will be cheerfully refunded the purchase price if the G340 is returned within 15 days of the purchase date.

PLEASE READ FIRST BEFORE USING THE G340:

If you are not familiar with DC servo drives please do the following setup instructions with the motor on the bench before mounting it on the mechanism it will eventually run. This will allow you to get a baseline motor behavior of what to expect.

Before you start, you must have a suitable encoder mounted and properly aligned on the motor. Follow the manufacturer's instructions on mounting and aligning the encoder if the motor doesn't already come with one.

Next you must have a DC power supply suitable for the motor. Do not use a power supply voltage more than 5 volts in excess of the motor's rated voltage. The power supply current rating must equal the maximum current you expect to run the motor at.

Finally have a STEP and DIRECTION pulse source available.

Before going on, turn the current LIMIT trimpot a quarter to half of full scale. Turn the GAIN trimpot fully off and turn the DAMP trimpot to a quarter of full scale. The trimpots are single-turn, do not over-torque them with a screwdriver.

REMOVING AND REPLACING THE COVER:

The STEP PULSE MULTIPLIER and the INPUT OPTION settings are jumpered internally. It is necessary to remove the cover of the drive to change these settings from their default values. Please follow the steps below on how to remove and replace the cover to avoid damaging the drive.

REMOVING THE COVER:

- 1) Remove the two 2-58 phillips-head screws on the bottom of the drive.
- 2) Gently lift the back of the cover upward until the LED clears the rectangular hole.
- 3) Slide the cover backwards while still lifting until it clears the drive.

REPLACING THE COVER:

- 1) Make sure the LED is vertical relative to the drive printed circuit board.
- 2) Slide the cover forward over the drive while lifting the back of the cover.
- 3) When the cover is fully forward, look to see the LED is aligned with the hole.
- 4) Gently press the cover down, making sure the LED moves into the hole.
- 5) Replace the screws on the bottom of the drive.

It is recommended to use small needle-nose pliers or tweezers to move the jumpers on the internal headers.

G340 MULTIPLIER AND INPUT OPTION HEADERS:

MULTIPLIER OPTION HEADER

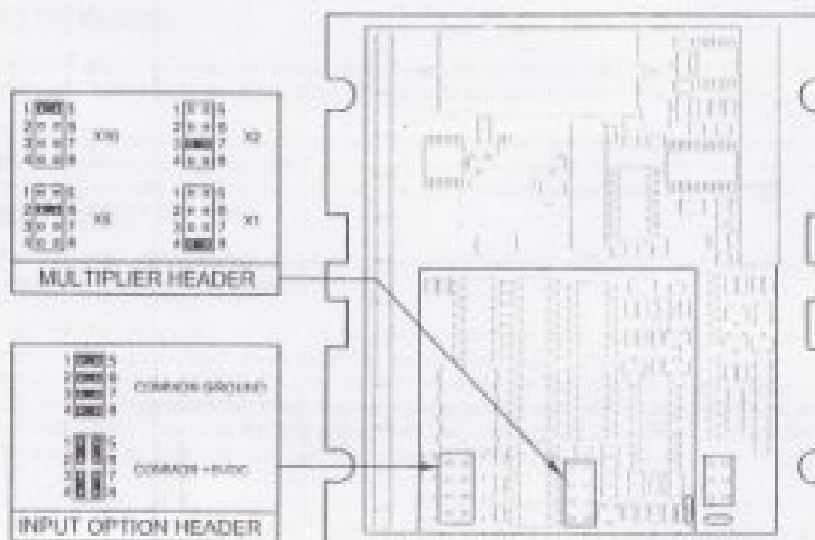
The G340 has a built in STEP PULSE MULTIPLIER. This circuit puts out 1, 2, 5, or 10 pulses for every input STEP pulse. All motor moves will be in these increments. For example, if X10 is selected, then the motor will move 10 encoder counts for every STEP pulse sent to the G340. Move the jumper on the MULTIPLIER HEADER to select the desired multiplier. Do not operate the drive without a jumper.

INPUT OPTION HEADER

The INPUT OPTION HEADER allows the STEP and DIRECTION opto-isolators to be configured as either common ground or common +5VDC.

If the G340 inputs are driven by a source that has only ground available, such as a PC parallel port, move the 4 jumpers on the header so that it looks like the COMMON GROUND setting. Connect the input driver ground to TERM 12 on the main connector.

If the G340 inputs are driven by open collector transistors or standard TTL gates, move the 4 jumpers on the header so that it looks like the COMMON +5VDC setting.



G340 TERMINAL WIRING:

IMPORTANT: When first testing the G320, connect ERR/RES (term. 5) to ENC+ (term. 7). Please follow the next steps in the sequence they are given.

STEP 1: ENCODER HOOK-UP

The encoder must be at minimum a 25 line-count digital quadrature encoder and must operate on a single +5VDC power supply. If the encoder supply current is more than 50 mA, use an external +5VDC supply. It may have an INDEX output, which will not be used. If it has differential outputs, use only the "I" phase outputs. **IMPORTANT:** Connect a 470-ohm resistor from TERM. 6 to TERM. 7 if an external power supply is used for the encoder.

(TERM. 6) ENC- Connect the encoder power supply ground to this terminal.

(TERM. 7) ENC+ Connect the encoder +5VDC to this terminal

(TERM. 8) PHASE A Connect the encoder phase "A" to this terminal

(TERM. 9) PHASE B Connect the encoder phase "B" to this terminal

To determine the optimal encoder line count, please follow the instructions below.

- 1.) Determine motor's no load RPM
- 2.) Calculate rated RPM as 80% of no load RPM
- 3.) Divide (#2) by 60 to get revolutions per second
- 4.) Determine the CNC program's maximum step pulse frequency (in Hz)
- 5.) Divide (#4) by (#3), which will give you the maximum counts per revolution
- 6.) Divide (#5) by 4, which will give you the max line count
- 7.) Pick the first standard line count below (#6)

An example of using that formula with a 45kHz step pulse frequency and a maximum motor RPM of 3000:

$$(45\text{kHz} / 60) / 4 = 281.25$$

STEP 2: POWER SUPPLY HOOK-UP

CAUTION: Never put a switch on the DC side of the power supply! This will damage, if not destroy, your drive!

Keep the power supply leads short and use the largest wire gauge that will fit in the terminals. If the lead length is more than 18" use a 1000 uF capacitor across the G340 power supply terminals. Make sure your power supply can provide the peak current the motor may draw. The power supply voltage must be between 18 VDC and 80 VDC. The actual voltage should not be more than 5 volts higher than the motor's rated voltage.

(TERM. 1) POWER GROUND Connect the power supply ground to this terminal.

(TERM. 2) +18 TO 80 VDC Connect the power supply "+" to this terminal

STEP 3: TESTING THE ENCODER

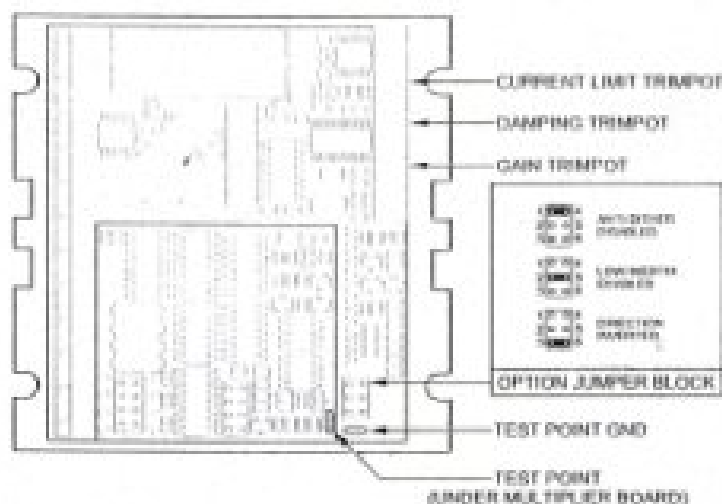
At this point the encoder should be tested for functionality. If you wish to monitor the POSITION ERROR test point with a voltmeter or oscilloscope, then remove the cover of the drive now. If you have a choice, pick the oscilloscope.

The POSITION ERROR test point shows the difference between the command position and the actual motor position. When both are the same, the voltage will be +5VDC. For every count the motor is clockwise of the command position, the voltage will decrease by 0.04 volts. When it drops to 0.4 volts, the protection circuit takes over and resets the drive for 3 seconds. While reset, the FAULT light is on.

For counter-clockwise position errors the voltage will increase by 0.04 volts for every count until it reaches 9.6 volts when again the protection circuit takes over as before.

VOLTMETER MONITORING: Place the red lead on the test point and the black lead on the large blue capacitor lead (GND) furthest from the main connector. Turn on the power supply. The FAULT light should be on for 3 seconds and then turn off. The voltmeter should read +5VDC. Turn the motor shaft clockwise VERY slowly. The voltmeter reading should decrease 0.04 volts for every encoder count. When the reading reaches 0.4 volts, the red light will turn on and the voltage will jump back to +5VDC. After 3 seconds the light will turn off. You may turn the motor shaft counter-clockwise as well. The voltage will increase then by 0.04 volts per count until it reaches 9.6 volts and trips the protection circuit.

OSCILLOSCOPE MONITORING: Set the scope to 2 volts / cm vertical and about 1 millisecond per cm horizontal. Zero the trace to the bottom line on the screen. DC couple the input. Place the probe on the test point and the ground clip to the blue capacitor ground lead. Follow the steps in VOLTMETER TESTING above.



STEP 4: CONTROL INPUT HOOK-UP

The control input group is the standard step motor drive STEP, DIRECTION and +5VDC lines. The STEP and DIRECTION signal drivers must be TTL compatible and have edge transition times of 100 ns or faster. The +5VDC is the opto-isolator common anode line and must be returned to the pulse source +5VDC supply.

(TERM. 10) DIR Connect the DIRECTION line to this terminal.

(TERM. 11) STEP Connect the STEP line to this terminal.

(TERM. 12) +5VDC Connect this terminal to the controller +5VDC power supply

STEP 5: TESTING THE CONTROL INPUTS

You may wish to test the functionality of these inputs. If you used an oscilloscope in the previous section, leave it connected to the test point. If you used a voltmeter, then remove it from the drive.

Set the STEP pulse generator to about 40 pulses per second and set the DIRECTION output to clockwise (logical '1'). Turn on the power supply. After the power-on reset period of 5 seconds the FAULT light will turn off.

If you are using an oscilloscope, then the test point voltage will begin to increase until 3 seconds later it trips the protection circuit at 9.6 volts. The FAULT light will turn on for 5 seconds and voltage will snap back to +5VDC. After the FAULT turns off, the sequence will repeat again.

If you are not using an oscilloscope, just see if the FAULT light turns on and off every three seconds.

STEP 6: MOTOR HOOK-UP

Make sure the power is off and the STEP pulse source is set to zero pulses per second. Check to see if the trimpot settings are set according to the instructions on page 2. You may wish to secure the motor so it can't jump off the bench.

(TERM. 3) **ARM-** Connect the BLACK motor lead to this terminal.

(TERM. 4) **ARM+** Connect the RED motor lead to this terminal.

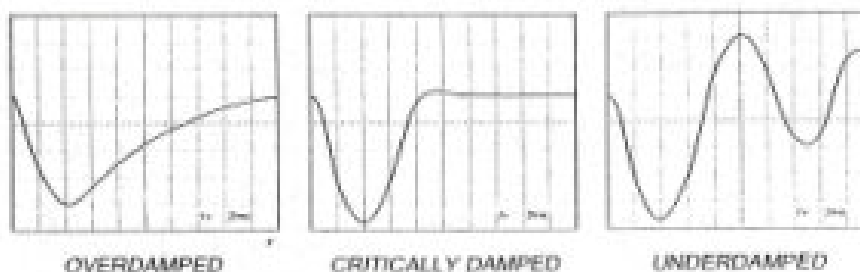
STEP 7: TUNING THE SERVO

Turn on the power supply. The FAULT light should turn off after 3 seconds. If everything is correct you should hear the motor "singing". This is normal. The motor is dithering or bouncing between adjacent encoder counts. The integral term in a PID loop has infinite DC gain over time and will amplify even the smallest position error. Because encoder feedback can only occur on count edges, the loop is "blind" until it encounters an encoder count edge. It then reverses the motor direction until another edge is found, then the process repeats.

If the motor jumps slightly and the FAULT light immediately turns back on, then either the motor is wired backwards or the trimpots are misadjusted. Check the trimpot settings. If they seem right then switch the motor leads and try again. If it still doesn't work and you followed all the previous steps, call me at the number at the end of this document.

Now turn on your STEP pulse source and ramp the speed up to see if the motor turns. It should turn clockwise with a logical "1" on the DIRECTION input.

The optimum way to tune the servo is to induce an impulse load on the motor while watching an oscilloscope to see how the motor behaves in response, then adjusting the PID co-efficient for optimal behavior.



In all cases the motor must return to the command position, what matters is how it does it. The manner in which the motor returns to its command position is called damping. At one extreme, called over damped response, the motor returns to position after a long, drawn out delay. At the other extreme, called under damped response, the motor returns to its position too rapidly, overshoots, returns and undershoots and so on until it finally settles at its command position. This is also called ringing; when extreme, the over/undershoot builds in amplitude until the motor enters violent oscillation. Between the two extremes is the optimal response called critical damping. Here the motor rapidly returns to its position with little or no overshoot in the minimal amount of time.

GAIN AND DAMPING

GAIN and DAMPING settings generally track each other. If you increase GAIN (greater stiffness), then increased DAMPING is needed as well to restore critical damping. Be careful, increasing GAIN without increasing DAMPING may cause the motor to break out into violent oscillation.

The higher GAIN is set, the noisier the motor will be when stopped. This is because higher gain causes more vigorous dithering between encoder counts at rest. There is a trade-off between high gain (high stiffness) one hand and excessive dithering (noise and motor heating) on the other. Use judgment here.

To see how your servo is compensated it is first necessary to induce a disturbance. The easiest way is to switch the DIRECTION input while commanding a constant speed via the STEP input. The abrupt direction change puts just the momentary load needed on the motor while you watch how it responds.

If you are using an oscilloscope, use channel 1 on the test point and channel 2 on the DIRECTION input. Set the trigger to "normal", trigger source to channel 2 and trigger edge to "+". You should see a single sweep for every clockwise change in direction.

Slowly increase STEP speed until you get a picture similar to one of the three above, and then do the following:

- 1) OVERDAMPED: Decrease DAMPING or increase GAIN
- 2) CRITICALLY DAMPED: Do nothing, you're there
- 3) UNDERDAMPED: Decrease GAIN or increase DAMPING

(POSITION ERROR TEST POINT NOTE)

Don't confuse the POSITION ERROR with the motor or machine position. The signal is actually the differential position error between the command speed and the motor speed. As noted above, sending clockwise STEP pulses moves the POSITION ERROR voltage more positive while turning the motor clockwise moves the POSITION ERROR voltage more negative.

When the motor encoder counts match the number of STEP pulses being sent one for one, the POSITION ERROR voltage stays at +5VDC. If the motor gets ahead of the STEP pulses such as during very rapid deceleration, the voltage will decrease by 0.04 volts for every encoder count the motor is ahead of the STEP pulses sent. The PD algorithm will force the motor to match the STEP input over time and restore the POSITION ERROR voltage back to +5 VDC.

CURRENT LIMIT

The current LIMIT tripot sets maximum current the motor is permitted to have. It is adjustable from 0 amps to 20 amps. Normally the LIMIT tripot is set to maximum (20 amps) unless you want to limit motor torque to a lower value.

Motor speed and position is unaffected by the current LIMIT setting unless the torque demand due to load exceeds this setting, then the motor position will fall behind the command position because of insufficient torque.

FAULT INDICATOR

The FAULT indicator is on while the drive is in power-on reset, the DISABLE input is held "low" or if the protection circuit is tripped due to a fault condition. All power MOSFETs are turned off and all internal counters are reset. The FAULT condition lasts for 3 seconds, and then self-resets to try again. If the protection circuit tripped it and the cause is not cleared, then it will immediately re-enter the FAULT state again and repeat the cycle.

There are two conditions that will trip the protection circuit. One condition is if a short-circuit occurs and current exceeds 20 amps. The other condition is if the POSITION ERROR exceeds +/- 120 counts causing a break of the servo-lock. This condition can have several causes:

- 1) The loop settings are severely under-damped and the motor breaks out into oscillation.
- 2) Excessive motor load due to acceleration or workload.
- 3) The speed command in excess of what the motor can deliver.
- 4) The current LIMIT is set too low.
- 5) The power supply current is insufficient for the demand.
- 6) The power supply voltage is below 18 VDC.
- 7) The motor is wired backwards, is broken or disconnected.
- 8) Encoder failure.

REVERSING DEFAULT MOTOR DIRECTION

The G340 will turn the motor in the CW direction when the DIRECTION input is "high" (logical "1", or +5VDC). If instead CCW is preferred, then:

- 1) Reverse the motor "+" and "-" leads (term. 3 with term. 4)
- 2) Reverse the encoder "channel A" and "channel B" leads (term. 8 with term. 9)

USING THE G340 WITH VERY SMALL MOTORS

Very small motors have low inductance relative to their operating current. Consequently ripple current due to pulse-width modulation can quickly overheat and destroy these motors. If the G340 will be used with these motors, then an external low pass filter must be used to attenuate ripple current to tolerable levels.

A suggested filter consists of two 150 micro Henry inductors in series with each motor lead and a 2 microfarad, low inductance film capacitor across the motor leads. The inductors must be rated for the anticipated peak motor current.

This filter is also need if ironless-armature or "pancake" motors are used. These motors have very low inductance as well and will overheat if driven directly by the G340.

(TERM. 5) ERR / RES

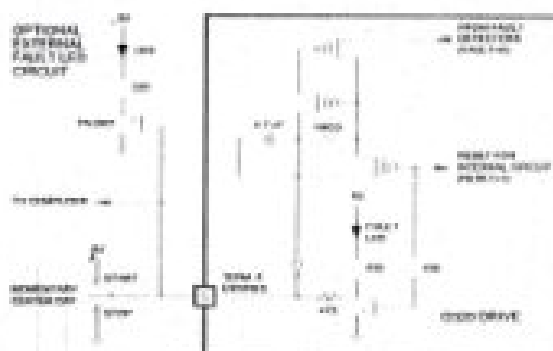
This terminal functions as an ERROR output and as a RESET input. Because this terminal functions as both an input and an output, some detailed description is necessary.

When first testing the G340, ERR/RES (term. 5) was connected to ENC+ (term. 7). It can be left that way if it is not necessary to read the state of the ERROR output. Otherwise, the following details are important.

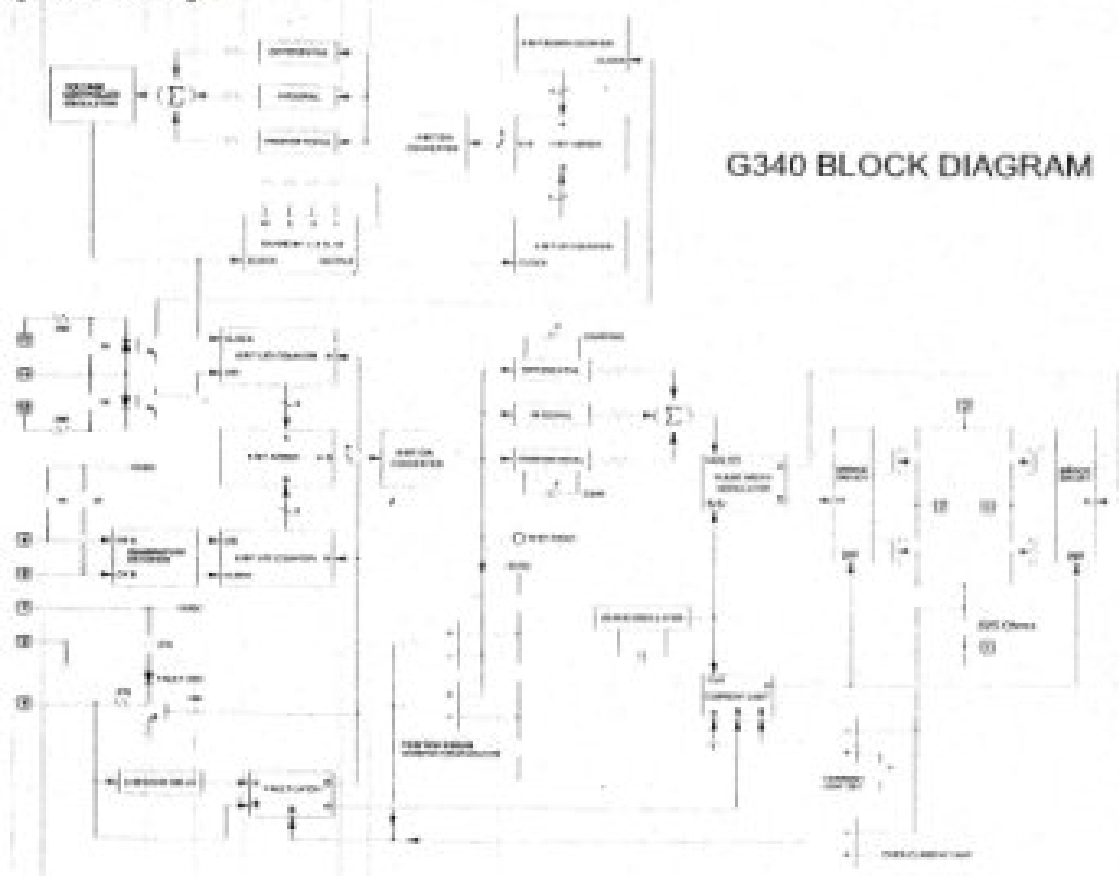
The ERROR output is latched in the "ERROR" state (term. 5 = "0") by the power-on reset circuitry in the G340. It will stay in this state indefinitely until it is cleared by applying +5V to this terminal for at least 1 second.

The voltage on this terminal is +5VDC when the G340 is functioning normally. The voltage on this terminal goes to 0VDC whenever the FAULT indicator is lit. This output can be used to signal your controller that an error has occurred.

Normally when the G340 is first powered up, it will be necessary to push the momentary switch to START for 5 seconds. This will clear the power-on reset condition and extinguish the FAULT LED. The motor will then be enabled and the drive will begin to operate. If at anytime after that a condition occurs that causes the G340 to "fault out", such as not being able to complete a step command, the ERR/RES terminal will go to "0", signaling the computer an error has occurred. This will require the operator to correct the problem that caused the fault and then push the switch to "START" for 5 seconds to re-enable the G340.

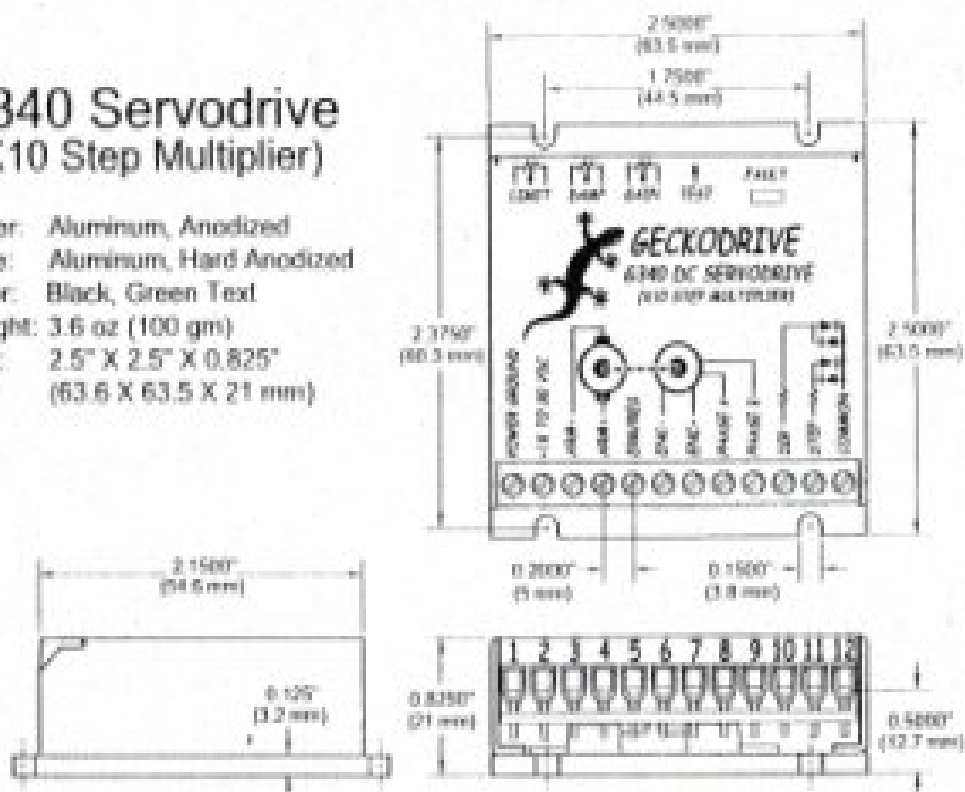


At anytime the operator can push the switch to the "STOP" position to immediately halt the G340 drive. Anytime the G340 is in the "FAULT" state (FAULT LED lit), all switching action stops and the motor freewheels and is unpowered. This will light the "FAULT" light.



G340 Servodrive (X10 Step Multiplier)

Cover: Aluminum, Anodized
 Plate: Aluminum, Hard Anodized
 Color: Black, Green Text
 Weight: 3.6 oz (100 gm)
 Size: 2.5" X 2.5" X 0.825"
 (63.5 X 63.5 X 21 mm)



G340 SPECIFICATIONS:

Power Supply	18 to 80 VDC
Motor Current	0 to 20 Amps
Lock Range	±128 count following error
Feedback	Quadrature TTL Encoder
Feedback Resolution	X4 Encoder Line Count
Switching Frequency	25 kHz
Current Limit	0 to 20 Amp, Trimptol Adjustable
Analog PID	Damping and Gain Trimptols
Step Pulse Frequency	0 to 250 kHz
Step Pulse "0" Time	0.5 Microseconds Min.
Step Pulse "1" Time	3.5 Microseconds Min.
Multiplier Settings	X1, X2, X4, X5 and X10
Size	2.5" X 2.5" X 0.825"
Weight	3.6 oz weight
Encoder Supply	+5VDC, 50mA max

Geckodrive Inc.
 14962 Franklin Ave
 Suite E
 Tustin, CA 92780

Phone: 1-714-832-8874
 Fax: 1-714-832-8882
 Web Site: www.geckodrive.com

A Full Service Motion Control Distributor and Systems Integrator.

JDH-2250-HF-2C-E
 DM-683
 CLIFTON PRECISION
REF. ONLY..

Supply Voltage, V_{cc} -.05 to 7V
 Output Voltage, V_o -.05 to V_{cc}
 Output Current per Channel..... -1.0 ma to 5 ma


PIN OUT

- [1] GND
- [2]
- [3] Channel A
- [4] + V_{cc}
- [5] Channel B



• Pin One



Designed by FABIAN JALDI	Checked by HELME JALDI	Approved by - date	File name SPP	Date 22/03/2008	Scale 1
 EASY TECHNOLOGY INDUSTRIES			SERVO DRIVER		
					Sheet

APPENDIX F
CD Content