

DEVELOPMENT OF A PID CONTROLLER FOR DC MOTOR USING  
MICROSOFT VISUAL BASIC

MOHD AIZUDDIN BIN ABU BAKAR

A report submitted in partial fulfillment of the  
requirements for the award of the degree of  
Bachelor of Electrical (Electronics) Engineering

Faculty of Electrical and Electronics Engineering  
Universiti Malaysia Pahang

NOVEMBER 2008



## UNIVERSITI MALAYSIA PAHANG

**BORANG PENGESAHAN STATUS TESIS♦**

**JUDUL:** DEVELOPMENT OF A PID CONTROLLER FOR DC MOTOR USING MICROSOFT VISUAL BASIC

**SESI PENGAJIAN:** 2008/2009

Saya MOHD AIZUDDIN BIN ABU BAKAR (840531-06-5615)  
(HURUF BESAR)

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

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

☐

**SULIT**

(Mengandungi maklumat yang berdarjah keselamatan atau kepentingan Malaysia seperti yang termaktub di dalam AKTA RAHSIA RASMI 1972)

☐

**TERHAD**

(Mengandungi maklumat TERHAD yang telah ditentukan oleh organisasi/badan di mana penyelidikan dijalankan)

☒

**TIDAK TERHAD**

Disahkan oleh:

\_\_\_\_\_  
(TANDATANGAN PENULIS)

\_\_\_\_\_  
(TANDATANGAN PENYELIA)

Alamat Tetap:

**NO.71 LORONG BUKIT SETONGKOL 12  
PERKAMPUNGAN CENDERAWASIH  
25200 KUANTAN  
PAHANG**

**HASZURAIDAH BINTI ISHAK**

Tarikh: **17 NOV 2008**

Tarikh: : **17 NOV 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

DEVELOPMENT OF A PID CONTROLLER FOR DC MOTOR USING  
MICROSOFT VISUAL BASIC

MOHD AIZUDDIN BIN ABU BAKAR

A report submitted in partial fulfillment of the  
requirements for the award of the degree of  
Bachelor of Electrical (Electronics) Engineering

Faculty of Electrical and Electronics Engineering  
Universiti Malaysia Pahang

NOVEMBER 2008

I declare this thesis entitled Develop a PID controller for DC motor using Microsoft Visual Basic is the result of my own research except as cited in the references. This thesis has not been accepted for any degree and is not concurrently submitted in candidature of any other degree

Signature: \_\_\_\_\_

Name: MOHD AIZUDDIN BIN ABU BAKAR

Date: 17 NOVEMBER 2008

## **ACKNOWLEDGEMENT**

I am greatly indebted to my supervisor, Puan Haszuraidah binti Ishak for her advice and guidance throughout my project. Thank you.

I would like to thank my family member for giving me their loves and supports throughout my study in Universiti Malaysia Pahang.

Special thanks to FKEE staffs for helping me to complete my project. Suggestions and criticisms from my friends have always been helpful in finding solutions to my problems. Thank you all.

Finally, I would like to express my thanks to those who involves directly or indirectly in completion of my project.

## **ABSTRACT**

This main of this project is to develop a PID (Proportional, Integral, Derivatives) controller and interface with a device. The controller is PID and the software is Microsoft Visual Basic 6.0. The MATLAB software is used for simulation of this system. The methodology is divided into two parts which is software and hardware. The first part is simulation for this system by using Matlab software to determine the value of  $K_p$ ,  $K_i$  and  $K_d$ . The range value for PID is determined by using Ziegler Nichols method. For second part is to interface the controller with hardware. The controller is using Microsoft visual basic 6.0 software. Then, the controller need to interface with DAQ card first. After interfacing success, the system can be implementing to servo motor. The feedback value can be received from servo motor encoder. After finished the first and second part, this system can be tuned up by using the PID value from simulation.

## **ABSTRAK**

*Tujuan utama projek ini adalah untuk membangunkan sebuah pengawal PID (Proportional, Integral, Derivatives) yang boleh berantaramuka dengan peralatan. Pengawal yang digunakan adalah PID dan perisian yang digunakan adalah Microsoft*



*Visual Basic 6.0. Perisian Matlab digunakan untuk membuat simulasi pada sistem ini. Metodologi dibahagikan kepada dua bahagian iaitu perkakasan dan perisian. Bahagian pertama adalah simulasi kepada sistem ini dengan menggunakan perisian Matlab untuk menentukan nilai  $K_p$ ,  $K_i$  dan  $K_d$ . Julat nilai PID ditentukan dengan menggunakan kaedah Ziegler Nicholes. Bahagian kedua adalah untuk antaramuka antara pengawal dan perkakasan. Pengawal menggunakan perisian Microsoft Visual Basic 6.0. Kemudian, pengawal perlu berantaramuka dengan kad DAQ (Data Acquisition). Selepas berjaya berantaramuka, sistem ini akan diimplementasikan pada motor servo. Nilai suapbalik akan diterima dari pengekod motor servo. Setelah selesai bahagian pertama dan kedua, sistem ini akan dilaraskan dengan menggunakan nilai PID dari proses simulasi.*

## TABLE OF CONTENTS

CHAPTER	TITLE	PAGE
---------	-------	------

<b>1</b>	<b>INTRODUCTION</b>	<b>1</b>
	1.1 Overview	1
	1.2 Project Objectives	2
	1.3 Scope of Project	2
	1.4 Problem Statement	3
	1.5 Thesis Organization	3
<b>2</b>	<b>LITERATURE REVIEW</b>	<b>4</b>
	2.1 PID controller	4
	2.2 Direct Current motor	7
	2.3 Microsoft Visual Basic 6.0	8
	2.4 Data acquisition card	9
<b>3</b>	<b>METHODOLOGY</b>	<b>11</b>
	3.1 Introduction	11
	3.2 Hardware development	14
	3.2.1. Servo motor	16
	3.2.2. Modelling DC servo motor	17
	3.2.3. DAQ card	20
	3.3 Software Development	21
	3.3.1. Matlab	21
	3.3.2. Microsoft Visual Basic 6.0	21
	3.3.3. PID Method	23

		11
<b>4</b>	<b>RESULT &amp; DISCUSSION</b>	<b>30</b>
	4.1 No controller	31
	4.2 Proportional controller	34
	4.3 Proportional Integral controller	38
	4.4 Proportional Derivative controller	41
	4.5 Proportional Integral Derivative controller	45
	4.6 Developing PID controller using Microsoft Visual Basic	52
	6.0	
<b>5</b>	<b>CONCLUSION AND RECOMMENDATION</b>	<b>53</b>
	5.1 Conclusion	53
	5.2 Future Recommendation	54
	5.3 Commercialization	54
	5.4 List and Cost of the Component	55
	<b>REFERENCES</b>	<b>56-88</b>
	<b>APPENDICES A-C</b>	

## LIST OF FIGURES

FIGURE NO.	TITLE	PAGE
2.1	PID controller equations	6
2.2	VB project selection panel	8
2.3	VB development environments	9
3.1	Flow chart of the project	12
3.2	Block diagram of PID controller	13
3.3	Ziegler Nicholes table	14
3.4	Ziegler Nicholes range value	15
3.5	Servo motor	16
3.6	USB DAQ card	20
3.7	System before using PID controller	23
3.8	System with PID controller	24
3.9	Designed using m-file	26
3.10	Typing program	26
3.11	Changing the value	27
3.12	Closed loop system	28
3.13	Save and run	28
4.1	No controller Step input for the system	31
4.2	Proportional controller $K_p=70$ Output graph for the system	34
4.3	Proportional controller $K_p=235$	35
4.4	Proportional controller $K_p=390$	35
4.5	Proportional controller $K_p=586$	36
4.6	Proportional controller $K_p=700$	36
4.7	Proportional-integral controller $K_p=70$ $K_i= 0.518$	38

4.8	Proportional-integral controller $K_p=700$ $K_i=0.518$	39
4.9	Proportional-integral controller $K_p=70$ $K_i=51.8$	39
4.10	Proportional-integral controller $K_p=700$ $K_i=51.8$	40
4.11	Proportional-derivative controller $K_p=140$ $K_d=2.59$	42
4.12	Proportional-derivative controller $K_p=700$ $K_d=2.59$	42
4.13	Proportional-derivative controller $K_p=140$ $K_d=51.8$	43
4.14	Proportional-derivative controller $K_p=700$ $K_d=51.8$	43
4.15	PID controller $K_p=70$ $K_i=5.18$ $K_d=2.59$	45
4.16	PID controller $K_p=700$ $K_i=5.18$ $K_d=2.59$	46
4.17	PID controller $K_p=70$ $K_i=51.8$ $K_d=51.8$	46
4.18	PID controller $K_p=700$ $K_i=51.8$ $K_d=51.8$	47
4.19	PID controller $K_p=70$ $K_i=5.18$ $K_d=51.8$	48
4.20	PID controller $K_p=700$ $K_i=5.18$ $K_d=51.8$	48
4.21	PID controller $K_p=70$ $K_i=51.8$ $K_d=2.59$	49
4.22	PID controller $K_p=700$ $K_i=51.8$ $K_d=2.59$	50
4.23	PID controller using Microsoft Visual Basic 6.0	52

**LIST OF EQUATIONS**

<b>FIGURE NO.</b>	<b>PAGE</b>
3.1	17
3.2	17
3.3	17
3.4	18
3.5	18
3.6	18
3.7	19
4.1	31
4.2	32
4.3	32

**LIST OF TABLE**

<b>TABLE</b>	<b>TITLE</b>	<b>PAGE</b>
4.1	No controller	32
4.2	Comparison of Proportional controller	37
4.3	Comparison of Proportional-integral controller	40
4.4	Comparison of Proportional-derivative controller	44
4.5	Comparison of PID controller	50

**LIST OF APPENDICES**

<b>APPENDIX</b>	<b>TITLE</b>	<b>PAGE</b>
A	DAQ card manual	30
B	Servo motor manual installation	52
C	Microsoft Visual Basic programmed	61





## **CHAPTER 1**

### **INTRODUCTION**

#### **1.1 Overview**

In 18<sup>th</sup> century, James Watt invented the flyball speed governor to control the speed of steam engines. In this device, two spinning flyballs rise as rotational speed increases. A steam valve connected to the flyball mechanism closes with the ascending flyballs and opens with the descending flyballs, thus regulating the speed.

PID Control (proportional-integral-derivative) is by far the widest type of automatic control used in industry nowadays. Even though it has a relatively simple algorithm/structure, there are many subtle variations in how it is applied in industry. PID control action allows the process control to accurately maintain set point by adjusting the control outputs.

In order to eliminate this problem, PID controller is introduced to the system. There's few type of controller but in this project, PID controller is chosen to interface

with the DC motor. This is because PID controller helps get the output, where we want it in a short time, with minimal overshoot and little error.

## **1.2 Project Objectives**

At the end of this project:-

- i. To develop a PID controller design for DC motor speed using Microsoft visual basic.
- ii. To control the speed of DC motor with PID controller using Microsoft Visual Basic (Design the PID controller and tune it).

## **1.3 Scope of Project**

The scope of this project is:-

- i. To derive mathematical model of dc motor and develop PID controller for the motor.
- ii. To develop GUI in Vb as an environment to applied the PID controller for the motor.
- iii. Perform computer simulation of the PID controller by using Matlab simulink to investigate the effectiveness of PID controller.

## **1.4 Problem Statements**

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.

By using PID controller, it can overcome this problems because it is sensitive to disturbances and able to correct this error quickly.

## **1.5 Thesis Organization**

This thesis will consist of five chapters including this chapter. The contents of each chapter are outlined as follows;

Chapter 2 will present the detailed description of the PID system, Microsoft Visual Basic software, DC motor and DAQ card. Chapter 3 will describe the methodology used in the project, including how the project is organized and flow of the project. Chapter 4 will discuss about the results and all of this will be concluded in Chapter 5.

## **CHAPTER 2**

### **LITERATURE REVIEW**

#### **2.1 PID controller**

The PID controller (Proportional, Integral and Derivative) has been known for several decades in many fields of automatic control. It has had powerful applications and several modifications anywhere where automatic control has been applicable. In spite of its many modified structures and forms the basic idea has remained the same. The underlying working principle relies on feedback control and the PID controller is the most common embodiment of feedback control. It involves three different terms, each of which have a specific purpose. Proportional and integral terms were known in the 1930s but derivative control was not invented until the 1940s. In basic terms, the PID controller is a numerical recipe, an algorithm. The algorithm may be implemented, that is, programmed using any programming language supporting numerical computation. It can be written in Visual Basic, Fortran, Pascal, C or Java. Also, there are numerous platforms for the algorithm such as personal computers, distributed control systems (DCS), programmable logic controllers (PLC), and field devices or microchips

enabling embedded solutions. In spite of the PID controller syntax language and its platform or even application, there are some essential features that should be involved in the PID controller algorithm: The basic calculation covering arithmetic around three different terms is not enough. Other issues of automatic feedback control must also be taken care of. In addition to this, there are potential characteristics that may be added to increase functionality and to improve applicability of the PID controller. There are different types of PID controllers such as ratio, cascade or split range controllers -and there are different algorithm types such as position or incremental (velocity) algorithms. The position algorithms can be categorized into ISA, series and parallel algorithms. The most typical operation modes for PID controllers are automatic, manual, cascade and remote. Some manufacturers hide their algorithms, but they should be available so that users can properly tune the instrument. The PID controller is not a secret and it should not be treated as one, although it is very rare for the numerical robustness of the algorithm to be available at all.[1]

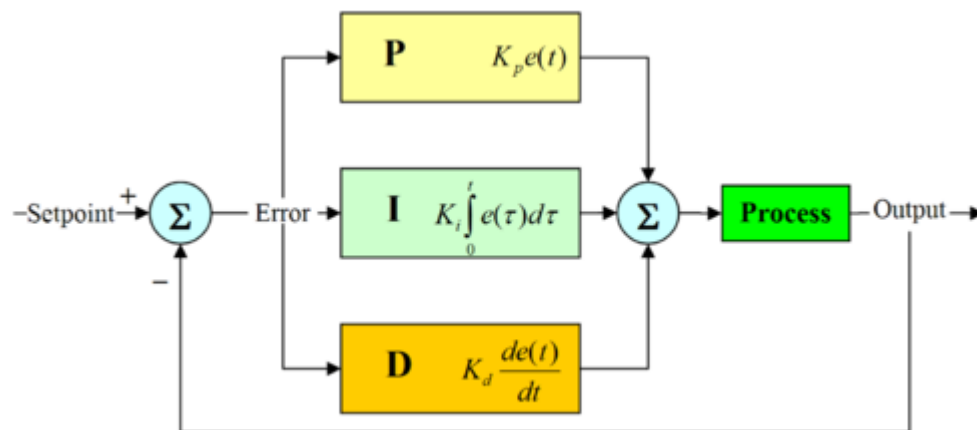
A proportional–integral–derivative controller (PID controller) is a generic control loop feedback mechanism widely used in industrial control systems. A PID controller attempts to correct the error between a measured process variable and a desired set point by calculating and then outputting a corrective action that can adjust the process accordingly.[2]

Figure 2.1 shows 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.

By tuning the three constants in the PID controller algorithm the PID 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 set point and the degree of system oscillation. Note that the use of the PID algorithm for control does not guarantee optimal control of the system.

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 prevents the system from reaching its target value due to the control action.



**Figure 2.1** PID controller equations

## 2.2 Direct current motor

Almost every mechanical movement is caused by a DC (direct current) electric motor. An electric motor is a device that transforms electrical energy into mechanical energy by using the motor effect.

Every DC motor has six basic parts which is axle, rotor (armature), stator, commutator, field magnet, and brushes. In most common DC motors, the external magnetic field is produced by high-strength permanent magnets. The stator is the stationary part of the motor which includes the motor casing, as well as two or more permanent magnet pole pieces. The rotor rotates with respect to the stator. The rotor consists of windings and the windings being electrically connected to the commutator. Industrial applications use dc motors because the speed-torque relationship can be varied to almost any useful form which is for both dc motor and regeneration applications in either direction of rotation. Dc motors are often applied where they momentarily deliver three or more times their rated torque. In emergency situations, dc motors can supply over five times rated torque without stalling. Dc motors feature a speed, which can be controlled smoothly down to zero, immediately followed by acceleration in the opposite direction. Dc motors respond quickly to changes in control signals due to the dc motor's high ratio of torque to inertia.[3]



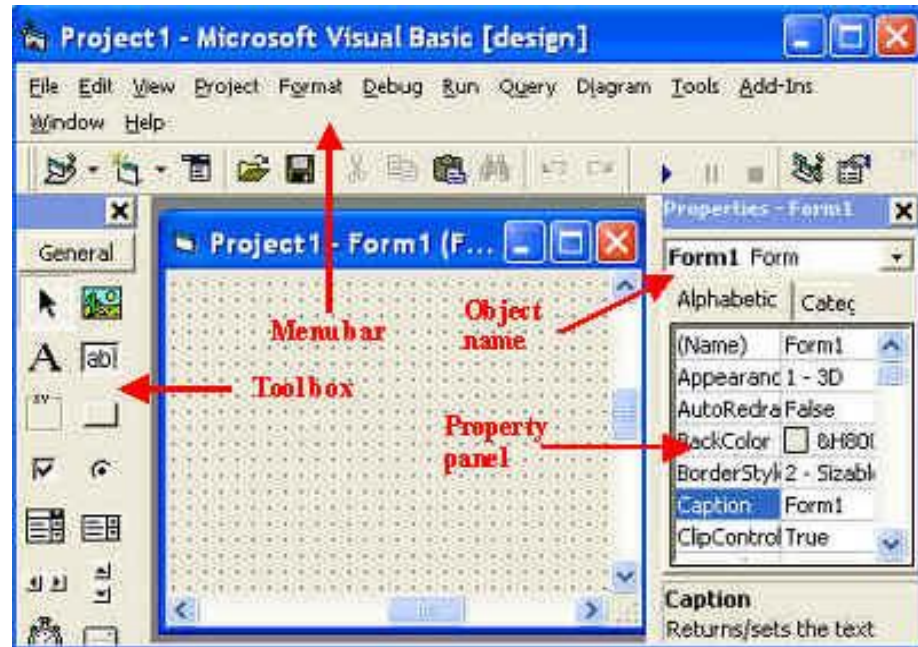
### 2.3 Microsoft Visual Basic (VB)

VB is a very easy yet very powerful application development tool under the Microsoft Windows family. It is possible to get your first program running in less than an hour. There are three editions of VB, they are the learning edition, the professional edition, and the enterprise edition. To develop software for control, a professional edition is necessary.[4]

By using this software as a tuner, it is easier to interface using RS232 port and USB port. It also can show the input and output data graphically.



**Figure 2.2** VB project selection panel



**Figure 2.3** VB development environments

## 2.4 Data acquisition card

DAQ is an abbreviation for data acquisition. Therefore a DAQ card is a basic A/D converter coupled with an interface that allows a personal computer to control the actions of the A/D, as well as to capture the digital output information from the converter. A DAQ card is designed to plug directly into a personal computer's bus. All the power required for the A/D converter and associated interface components is obtained directly from the PC bus.

A DAQ card today is more than a simple A/D function on a board. A data acquisition card can offer measurements of up to 64 channels at a resolution of 16 bits, (one part in 65,536) with data throughput rates up to 20 million samples per second. A data acquisition card can often include discrete digital bi-directional I/O lines, counter timers, and D/A converters for outputting analog signals for control applications.

A high-performance DAQ card will work in a very wide range of test and measurement, and control applications. Combined with powerful software, DAQ cards will turn a personal computer into powerful measurement system that may be used to automate experiments, construct product test stands, monitor and control production equipment or be embedded in products ranging from medical monitoring systems to automobile test simulators.

A DAQ card converts analog signals into a digital output form, which can be manipulated with software. Using software in conjunction with a personal computer, analog data can be displayed, logged, charted, graphed, or stored in memory as needed.

Stored data can later be used and compared with a set of established limits. Control decisions are made if the stored data is at the limit, above or below the limit. A DAQ card can make repetitive measurements, for continuous monitoring and controlling.[5]

## **CHAPTER 3**

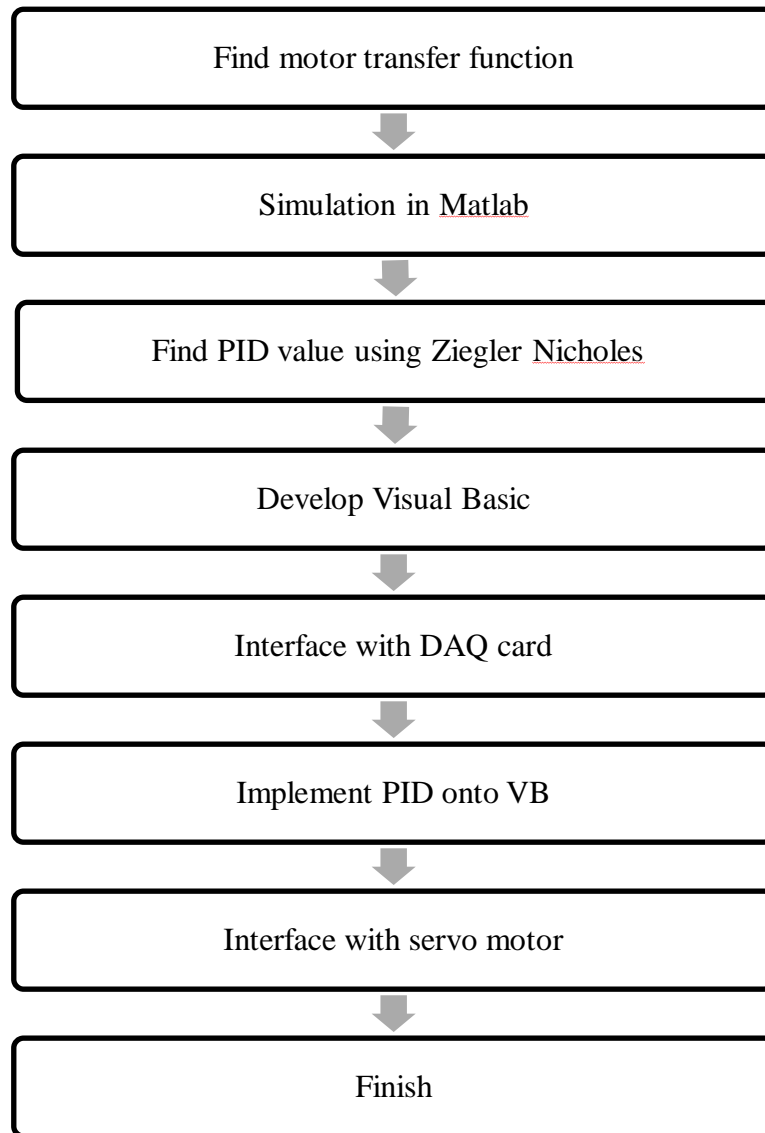
### **METHODOLOGY**

#### **3.1 Introduction**

This chapter will explain the methodology used in this project. The methodology is divided into two parts which is hardware and software. The first part is simulation for this system by using Matlab software to determine the value of  $K_p$ ,  $K_i$  and  $K_d$ . The range value for PID is determined by using Ziegler Nicholes method.

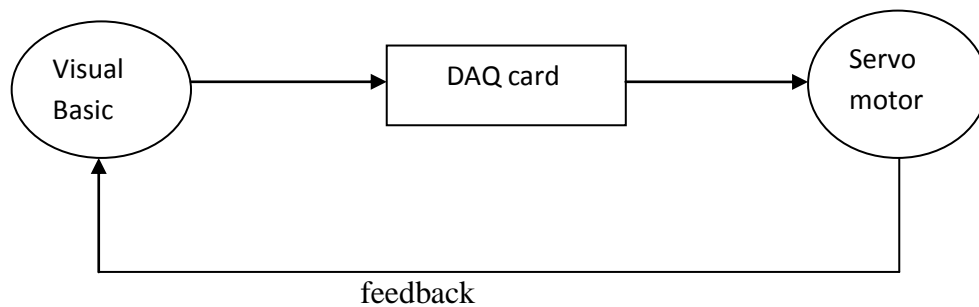
The second part is to interface the controller with hardware. The controller is using Microsoft visual basic 6.0 software. Then, the controller need to interface with DAQ card first. After interfacing success, the system can be implementing to servo motor. The feedback value can be received from servo motor encoder.

Figure 3.1 shows the flow chart of the project. After finished the first and second part, this system can be tuned up by using the PID value from simulation.



**Figure 3.1** Flow chart of the project

The DAQ card is used as an interface within Visual Basic software and servo motor. It has analog input, digital input, analog output and digital output port. For this system, only analog input and output port is used. Servo motor is used to show the output from controller. It also has a decoder which is used to give a feedback voltage to the controller. This servo motor input voltage is 5V to 80V.



**Figure 3.2** Block diagram of PID controller

Figure 3.2 shows the block diagram of PID controller. The controller software is Microsoft Visual Basic 6.0. The PID system is implemented in this software where user can tune the value of  $K_p$ ,  $K_i$  and  $K_d$  manually. The value of feedback voltage and error value shows in this software.

### 3.2 Hardware Development

Before starting develop the hardware, the value of P,PI,PD and PID need to determine first. It is determine using Ziegler Nicholes method. This value is very important because it will be used as a reference value to tune dc motor using visual basic software. Figure 3.3 shows Ziegler Nicholes table;

Controller	Kp	Ki	Kd
PID	K <sub>pt</sub> [0.1 0.5] K <sub>p</sub> max	K <sub>it</sub> [0.1 10] K <sub>p</sub> max T <sub>osc</sub>	K <sub>dt</sub> [0.05 1] K <sub>p</sub> max T <sub>osc</sub>
PD	K <sub>pt</sub> [0.1 0.5] K <sub>p</sub> max	0	K <sub>dt</sub> [0.05 1] K <sub>p</sub> max T <sub>osc</sub>
PI	K <sub>pt</sub> [0.1 0.5] K <sub>p</sub> max	K <sub>it</sub> [0.01 1] K <sub>p</sub> max T <sub>osc</sub>	0
P	K <sub>pt</sub> [0.05 0.5] K <sub>p</sub> max	0	0

T<sub>osc</sub> value: 0.037s

K<sub>p</sub> max value: 1400

**Figure 3.3** Ziegler Nicholes table

The  $T_{osc}$  and  $K_p$  max value is fixed for this dc motor model [6]. Figure 3.4 shows the result of Ziegler Nicholes range value;

Controller	$K_p$	$K_i$	$K_d$
PID	140 - 700	5.18 - 518	2.59 – 51.8
PD	140 - 700	0	2.59 – 51.8
PI	140 - 700	0.518 – 51.8	0
P	70 - 700	0	0

**Figure 3.4** Ziegler Nicholes range value

The hardware is used after the PID value determined. The hardware used is dc motor and daq card.



### 3.2.1 Servo motor

This dc motor is very suitable for my project. It has encoder to give feedback for actual speed of the motor. The specification of this dc motor is very suitable for this project.

Model: CLIFTON PRECISION SERVO MOTOR MODEL JDH-2250-HF-2C-E

Supplier: Servo Systems Company

Specification:

- Torque Constant: 15.76 oz-in. / A
- Back EMF: 11.65 VDC / KRPM
- Peak Torque: 125 oz-in.
- Cont. Torque: 16.5 oz-in.
- Encoder: 250 counts / rev.
- Channels A, B in quadrature, 5 VDC input (no index)
- Body Dimensions: 2.25" dia. x 4.35" L (includes encoder)
- Shaft Dimensions: 8 mm x 1.0" L w/flat



**Figure 3.5** Servo motor

### 3.2.2 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.

$$R= 2.7 \, \Omega$$

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

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

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

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

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

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

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

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

$$y = \begin{bmatrix} 0 & 1 \end{bmatrix} \begin{bmatrix} i_a \\ \omega_r \end{bmatrix} + \begin{bmatrix} 0 \end{bmatrix} V_a \quad (3.4)$$

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

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

$$C = \begin{bmatrix} 0 & 1 \end{bmatrix} \quad D = \begin{bmatrix} 0 \end{bmatrix}$$

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

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

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

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

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

(3.7)

$$\begin{aligned}
T(s) &= \frac{Y(s)}{U(s)} = C[sI - A]^{-1}B + D \\
&= [0 \quad 1] \frac{\begin{bmatrix} s - 0.093 & -26.25 \\ 1050 & s - 675 \end{bmatrix}}{s^2 + 675.093s + 27625.275} \begin{bmatrix} 250 \\ 0 \end{bmatrix} + [0] \\
&= \frac{262500}{s^2 + 675.093s + 27625.275}
\end{aligned}$$

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

### 3.2.3 DAQ card

This is the best DAQ card that can support Microsoft Visual Basic software. This DAQ card use USB port to interface within pc and dc motor. The specification is;

Model: USB 4716

Supplier: Advantech Co. Ltd.

Main Features:

- Supports USB 2.0
- Portable
- Bus-powered
- 16 analog input channels
- 16-bit resolution AI
- Sampling rate up to 200 kS/s
- 8DI/8DO, 2 AO and 1 32-bit counter
- Wiring terminal on modules
- Suitable for DIN-rail mounting
- Lockable USB cable for rigid connection



**Figure 3.6** USB DAQ card

### **3.3 Software Development**

#### **3.3.1 Matlab**

Before run the VB programming, a simulation of controller using Ziegler Nicholes value and Matlab software. With this simulation, we can determine the best value for  $K_p$ ,  $K_i$  and  $K_d$ .

#### **3.3.2 Microsoft Visual Basic 6.0**

There are many methods to implement into PID controller such as speed, angular and acceleration. This controller only measure speed (RPM) by using Microsoft Visual Basic 6.0 edition as a tuner.

Microsoft Visual Basic 6.0 is an object-oriented computer language that can be viewed as an evolution of Microsoft's Visual Basic (VB) implemented on the Microsoft .NET framework. Its introduction has been controversial, as significant changes were made that broke backward compatibility with older versions and caused a rift within the developer community.

VB Advantage is a powerful VB development productivity utility that enhances VB's design time environment. VB Advantage has many powerful, helpful, and easy-to-use features and tools that were conceived to support software engineering development activities that developers do as they create and test application code.

Like the BASIC programming language, Visual Basic was designed to be easy to learn and use. The language not only allows programmers to create simple GUI applications, but can also develop complex applications. Programming in VB is a combination of visually arranging components or controls on a form, specifying attributes and actions of those components, and writing additional lines of code for more functionality. Since default attributes and actions are defined for the components, a simple program can be

created without the programmer having to write many lines of code. Performance problems were experienced by earlier versions, but with faster computers and native code compilation this has become less of an issue.

Although programs can be compiled into native code executables from version 5 onwards, they still require the presence of runtime libraries of approximately 2 MB in size. This runtime is included by default in Windows 2000 and later, but for earlier versions of Windows or Windows Vista, it must be distributed together with the executable.

Forms are created using drag-and-drop techniques. A tool is used to place controls on the form. Controls have attributes and event handlers associated with them. Default values are provided when the control is created, but may be changed by the programmer. Many attribute values can be modified during run time based on user actions or changes in the environment, providing a dynamic application. For example, code can be inserted into the form resize event handler to reposition a control so that it remains centered on the form, expands to fill up the form. By inserting code into the event handler for a keypress in a text box, the program can automatically translate the case of the text being entered, or even prevent certain characters from being inserted.

Visual Basic can create executables (EXE files), ActiveX controls, DLL files, but is primarily used to develop Windows applications and to interface web database systems. Dialog boxes with less functionality can be used to provide pop-up capabilities. Controls provide the basic functionality of the application, while programmers can insert additional logic within the appropriate event handlers. For example, a drop-down combination box will automatically display its list and allow the user to select any element. An event handler is called when an item is selected, which can then execute additional code created by the programmer to perform some action based on which element was selected, such as populating a related list.

Alternatively, a Visual Basic component can have no user interface, and instead provide ActiveX objects to other programs using Component Object Model (COM). This allows for server-side processing or an add-in module.

The language is garbage collected using reference counting, has a large library of utility objects, and has basic object oriented support. Since the more common components are included in the default project template, the programmer seldom needs to specify additional libraries. Unlike many other programming languages, Visual Basic is generally not case sensitive, although it will transform keywords into a standard case configuration and force the case of variable names to conform to the case of the entry within the symbol table entry. String comparisons are case sensitive by default, but can be made case insensitive if so desired.

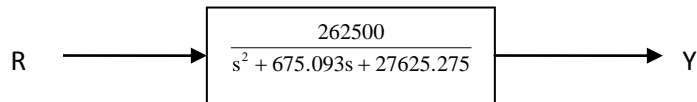
The Visual Basic compiler is shared with other Visual Studio languages (C, C++), but restrictions in the IDE do not allow the creation of some targets (Windows model DLL's) and threading models.

### 3.3.3 PID Method

From the modeling DC servo motor, the transfer function is

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

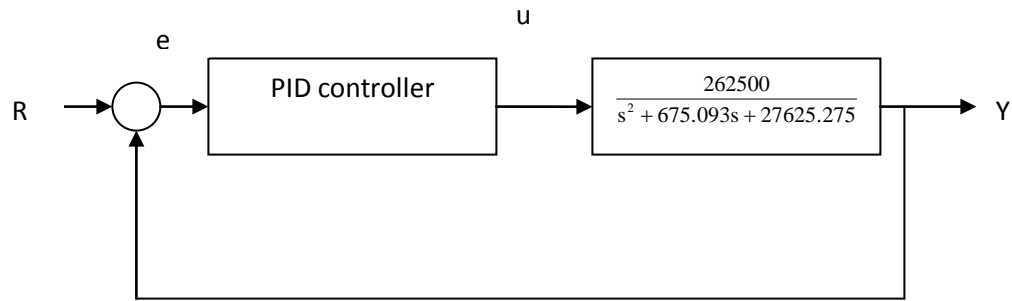
The system before using PID controller is looks like in Figure 3.7:



**Figure 3.7** System before using PID controller

Then, PID controller is added to the system. Now, the system looks like in Figure 3.8:





**Figure 3.8** System with PID controller

In Figure 3.8, 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:

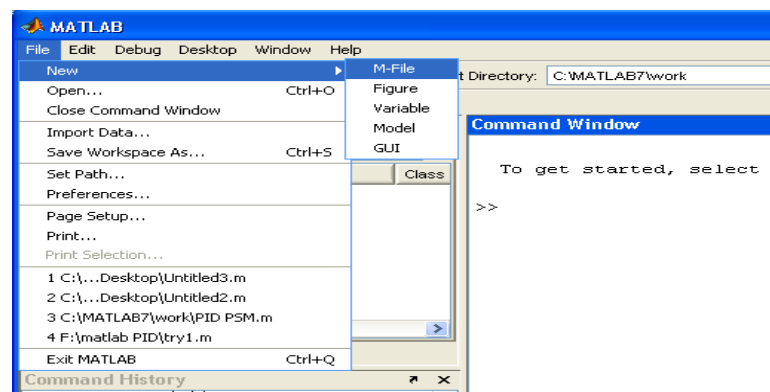
$$K_p + \frac{K_i}{s} + K_d s = + \frac{K_d s^2 + K_p s + K_i}{s} \quad (3.9)$$

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

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

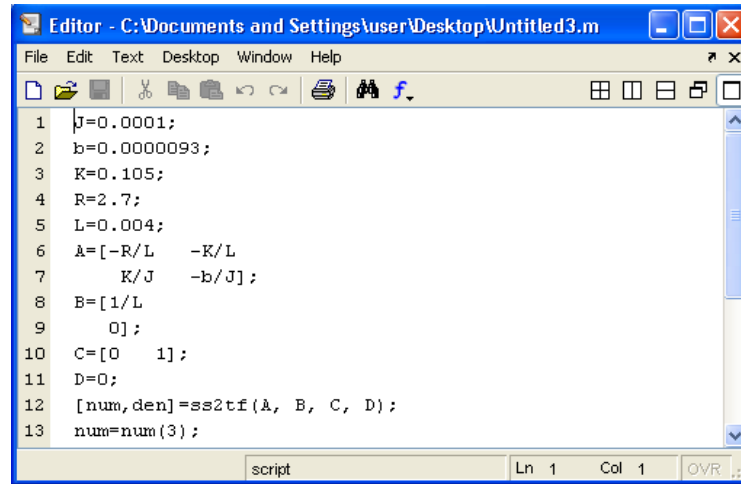
This signal (u) will be sent to the plant, and the new output (Y) will be obtained. This new output (Y) will be sent back to the sensor again to find the new error signal (e). The controller takes this new error signal and computes its derivative and its integral again. This process goes on and on.

In this project, the PID controller that was added into the system is designed using m-file in matlab software. (Refer Figure 3.9)



**Figure 3.9** Designed using m-file

Then the following commands are typing into m-file. (Refer Figure 3.10)



```

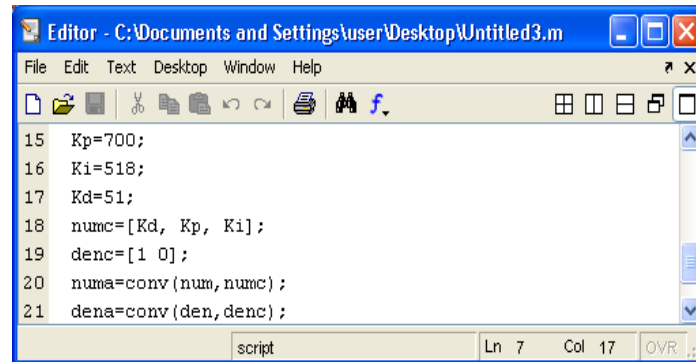
1 J=0.0001;
2 b=0.0000093;
3 K=0.105;
4 R=2.7;
5 L=0.004;
6 A=[-R/L    -K/L
7     K/J    -b/J];
8 B=[1/L
9     0];
10 C=[0    1];
11 D=0;
12 [num,den]=ss2tf(A, B, C, D);
13 num=num(3);

```

**Figure 3.10** Typing program

In Figure 3.10, '[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.

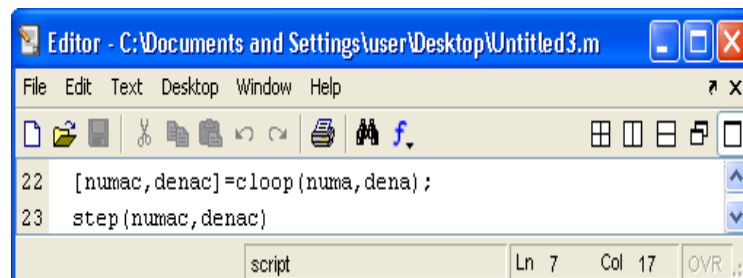
The transfer function of PID controller is recalled using following commands. The value of the proportional gain,  $K_p$ , integral gain  $K_i$  and derivative gain,  $K_d$  can be adjust by changing the value. (Refer Figure 3.11):



```
Editor - C:\Documents and Settings\user\Desktop\Untitled3.m
File Edit Text Desktop Window Help
[Icons]
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);
script Ln 7 Col 17 OVR
```

**Figure 3.11** Changing the value

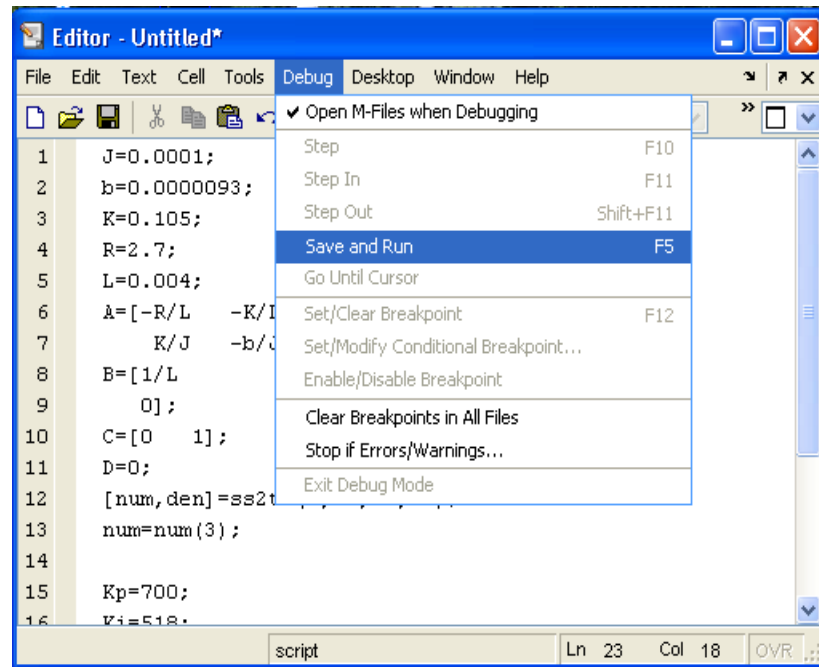
The closed loop of the system is determined by ‘cloop’ command and the command ‘step (numac,denac)’ is to see how the step response looks as in Figure 3.12.



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

**Figure 3.12** Closed loop system

Then, save and run it such in Figure 3.13.



**Figure 3.13** 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.

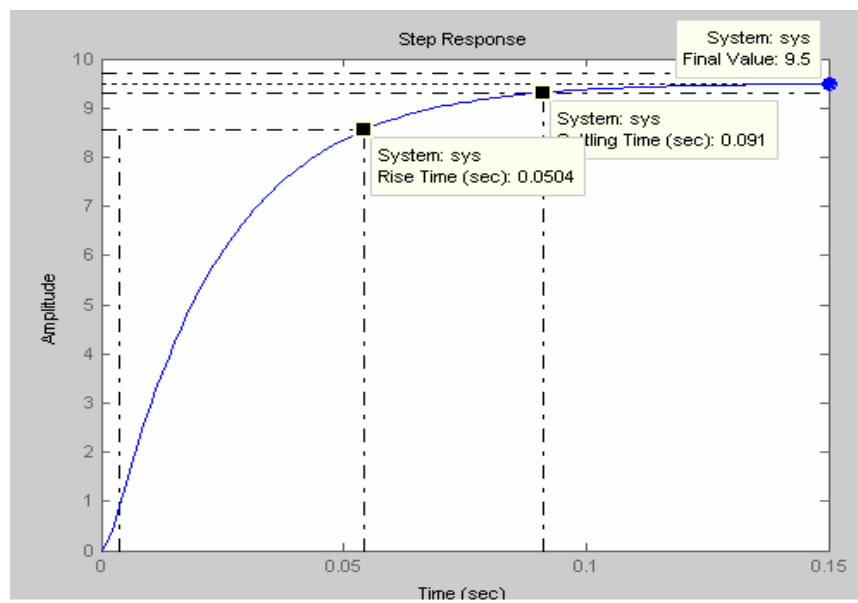
## **CHAPTER 4**

### **RESULTS DISCUSSIONS**

This chapter consists of two parts which is discussions on the results from Matlab simulation for the system and developing a PID controller using Microsoft Visual Basic 6.0 software. It 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

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

**Table 4.1** No controller

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.

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

$$E(\infty) = 1 + CA^{-1}B \quad (4.1)$$



$$\begin{aligned}
&= 1 + \begin{bmatrix} 26.25 & 0.093 \end{bmatrix} \begin{bmatrix} 250 \\ 0 \end{bmatrix} \\
&= 1 + (6562.5 + 0) \\
&= 6563.5
\end{aligned}$$

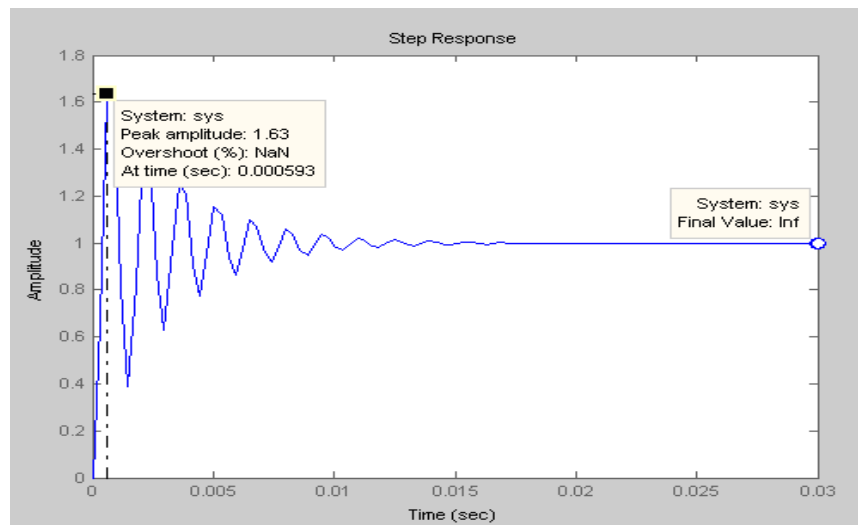
$$E(\infty) = \left\| \lim \left( 1 + CA^{-1}B \right) + \left( 1 + C(A^{-1})^2 B \right) \right\| \quad (4.2)$$

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

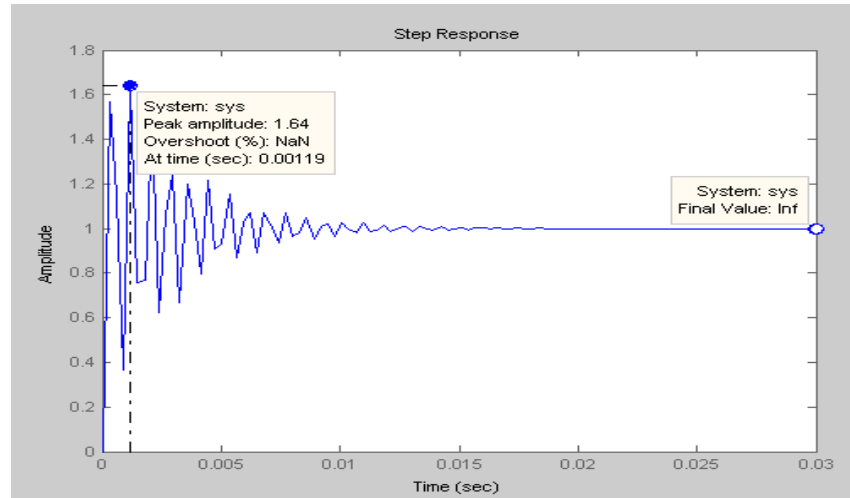
$$\begin{aligned}
E(\infty) &= \left[ \lim_{t \rightarrow \infty} \left( 1 + CA^{-1}B \right) t + \left( 1 + C(A^{-1})^2 B \right) \right] \quad (4.3) \\
&= \left[ \lim (6563.5)t + (4430297.5) \right] \\
&= \infty
\end{aligned}$$

## 4.2 Proportional Controller

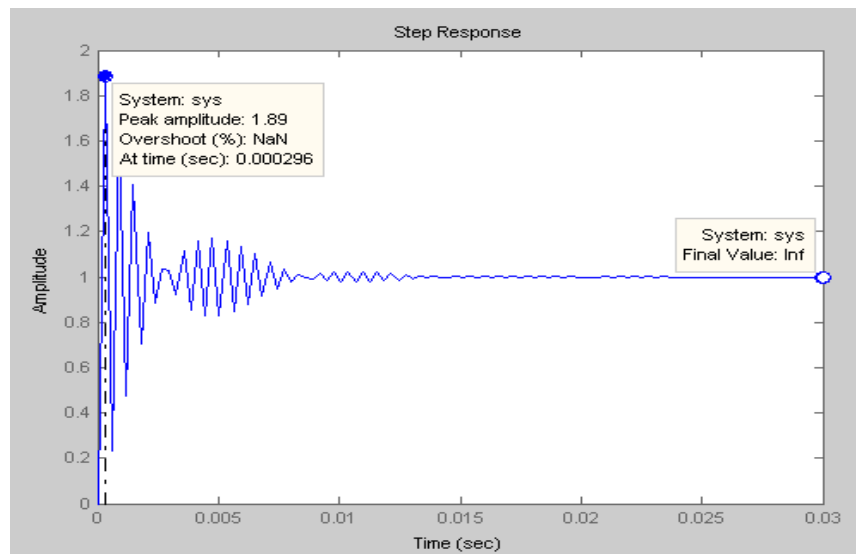
The performance of Proportional Controller result simulate by tuning the value of proportional gain,  $K_p$  are shown in Figure 4.2 to Figure 4.6 using the value of 70, 235, 390, 586 and 700.



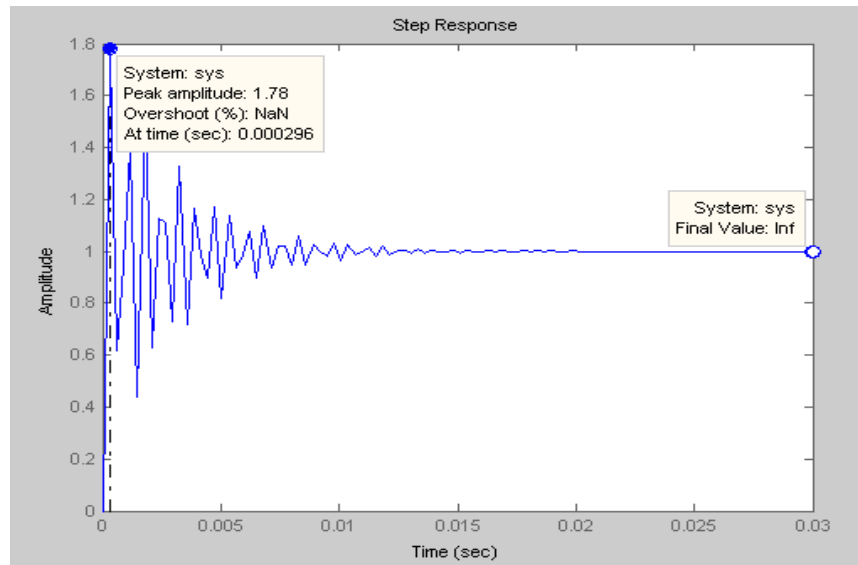
**Figure 4.2** Proportional controller  $K_p=70$



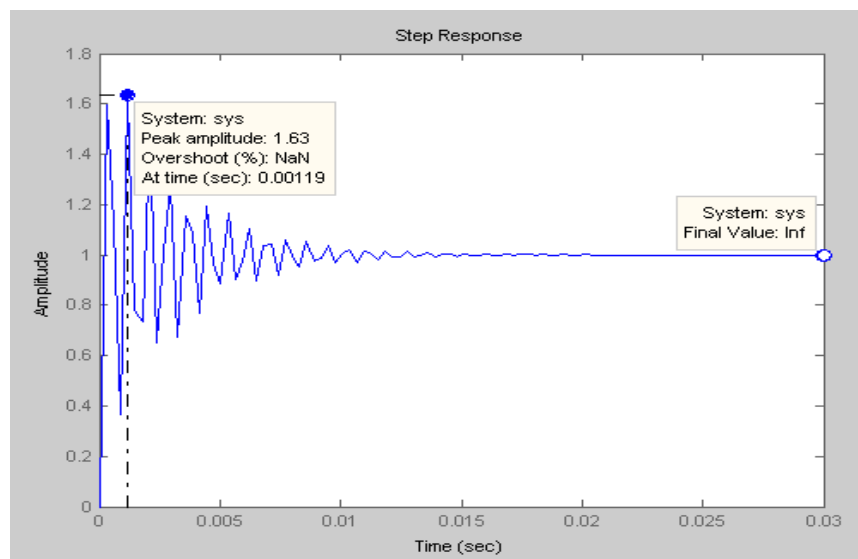
**Figure 4.3** Proportional controller  $K_p=235$



**Figure 4.4** Proportional controller  $K_p=390$



**Figure 4.5** Proportional controller  $K_p=586$



**Figure 4.6** Proportional controller  $K_p=700$

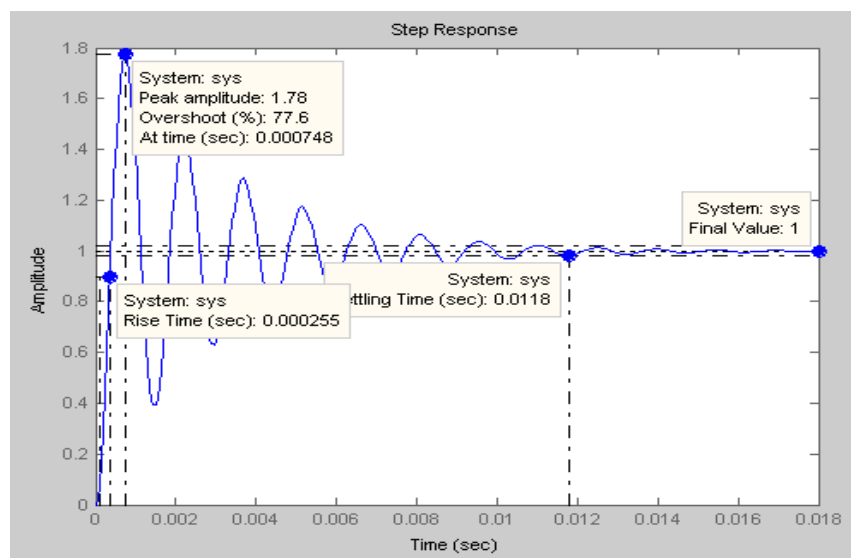
COMPARIS ON	PEAK AMPLITUDE	TIME(s)	OVERSHOO T (%)	STEADY STATE
Kp				
70	1.63	0.000593	NaN	Inf
235	1.64	0.00119	NaN	Inf
390	1.89	0.000296	NaN	Inf
586	1.78	0.000296	NaN	Inf
700	1.63	0.00119	NaN	Inf

**Table 4.2** Comparison of Proportional controller

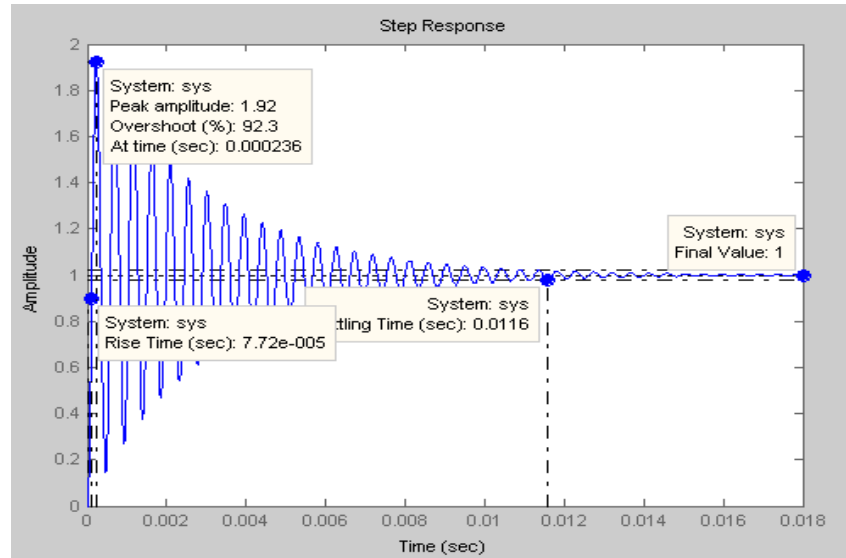
Table 4.2 show increasing the value of Proportional gain, Kp reducing the time to achieve the peak amplitude but in high value of peak amplitude. The overshoot is NaN (not a number) and the steady state is infinity.

### 4.3 Proportional-Integral Controller

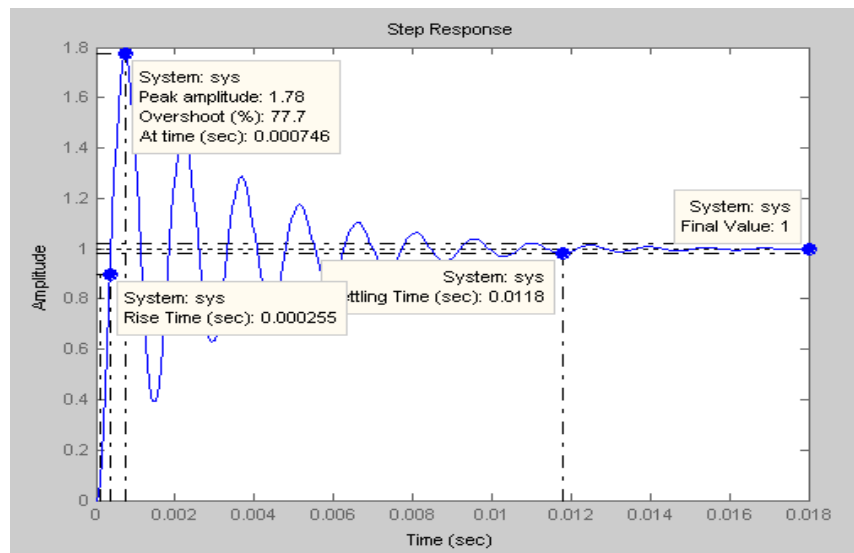
The performance of Proportional-Integral Controller result simulate by tuning the value of proportional gain,  $K_p$  and Integral gain,  $K_i$  are shown in Figure 4.7 to Figure 4.10 using the value of Proportional gain,  $K_p$  70 and 700 and the value of Integral gain,  $K_i$  0.518 and 51.8.



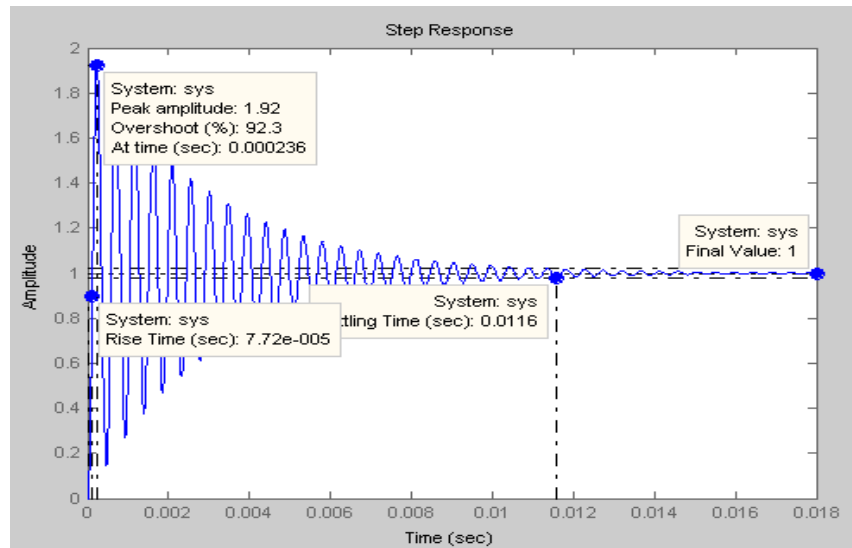
**Figure 4.7** Proportional-integral controller  $K_p=70$   $K_i= 0.518$



**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  $K_p=700$   $K_i= 51.8$

COMPARISON		PEAK AMPLITUDE	TIME (s)	OVER SHOOT (%)	TIME RISE(s)	SETTLING TIME (s)	STEADY STATE
$K_p$	$K_i$						
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

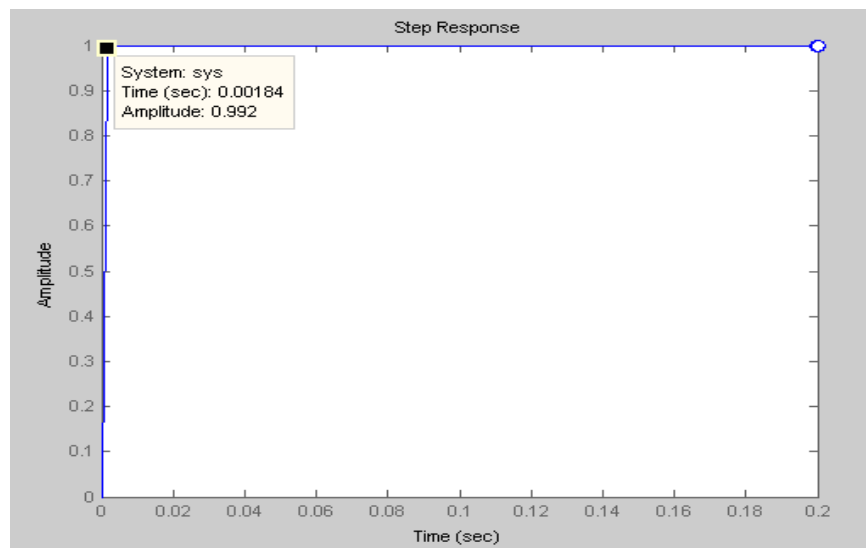
**Table 4.3** Comparison of Proportional-integral controller



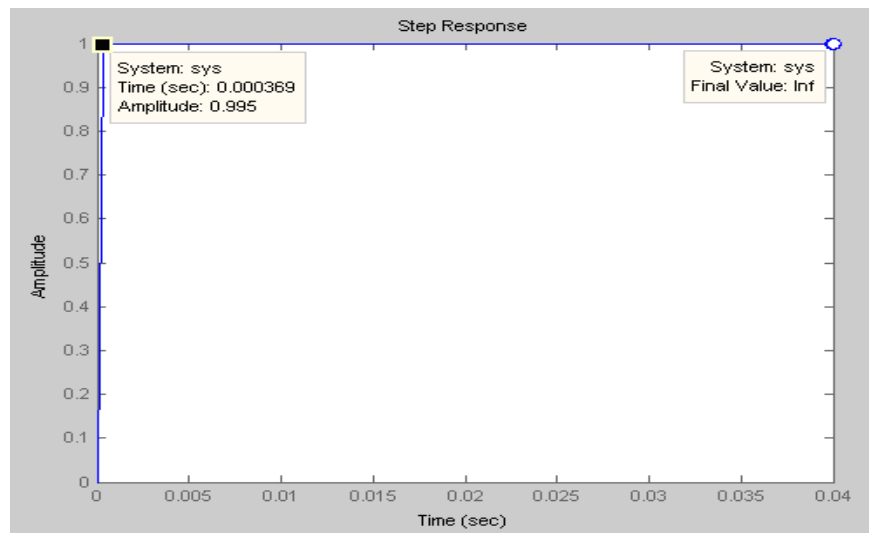
Table 4.3 shows reducing the value of Proportional gain,  $K_p$  and Integral gain,  $K_i$  will reduce the value of time rise, settling time, steady state and also reduce the time to achieve the peak amplitude.

#### 4.4 Proportional-Derivative Controller

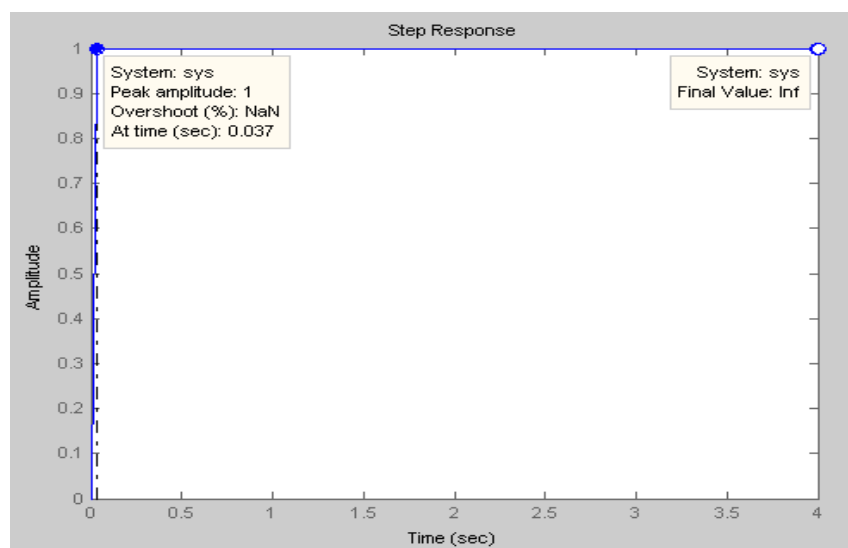
The performance of Proportional-Derivative Controller result simulate by tuning the value of proportional gain,  $K_p$  and Derivative gain,  $K_d$  are shown in Figure 4.11 to Figure 4.14 by using the value of Proportional gain,  $K_p$  140 and 700, the value of Derivative gain,  $K_d$  2.59 and 51.8.



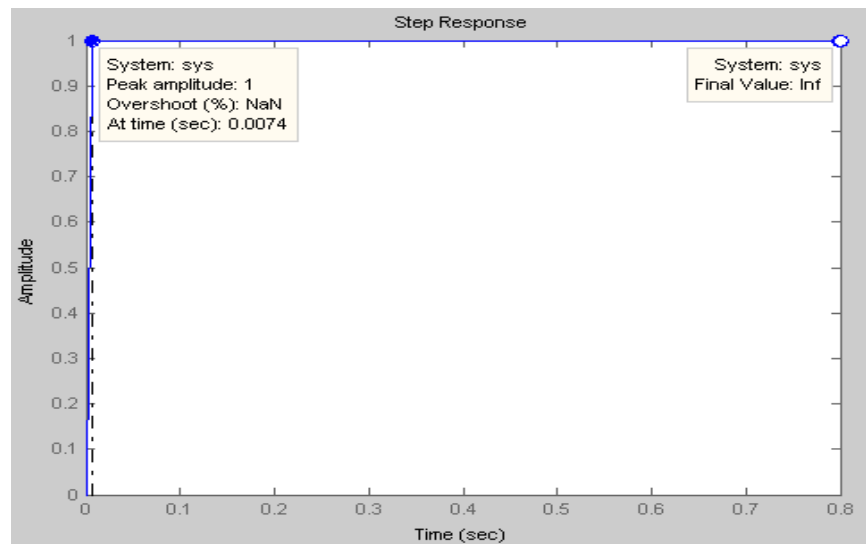
**Figure 4.11** Proportional-derivative controller  $K_p=140$   $K_d= 2.59$



**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  $K_p=700$   $K_d= 51.8$

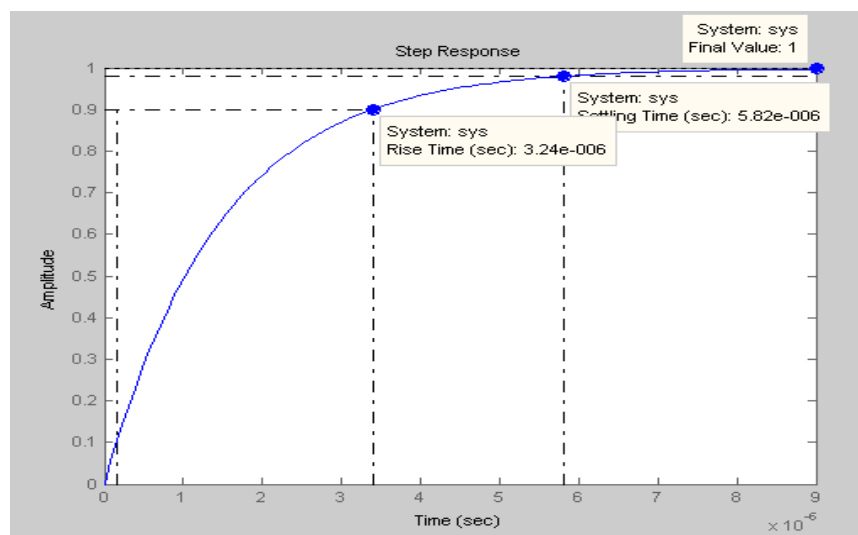
COMPARISON		PEAK	TIME(s)	OVER	STEADY
K <sub>p</sub>	K <sub>d</sub>	AMPLITUDE		SHOOT	STATE
		DE		(%)	
140	2.59	0.992	0.00184	NaN	Inf
700	2.59	0.995	0.000369	NaN	Inf
140	51.8	1	0.037	NaN	Inf
700	51.8	1	0.0074	NaN	Inf

**Table 4.4** Comparison of Proportional-derivative controller

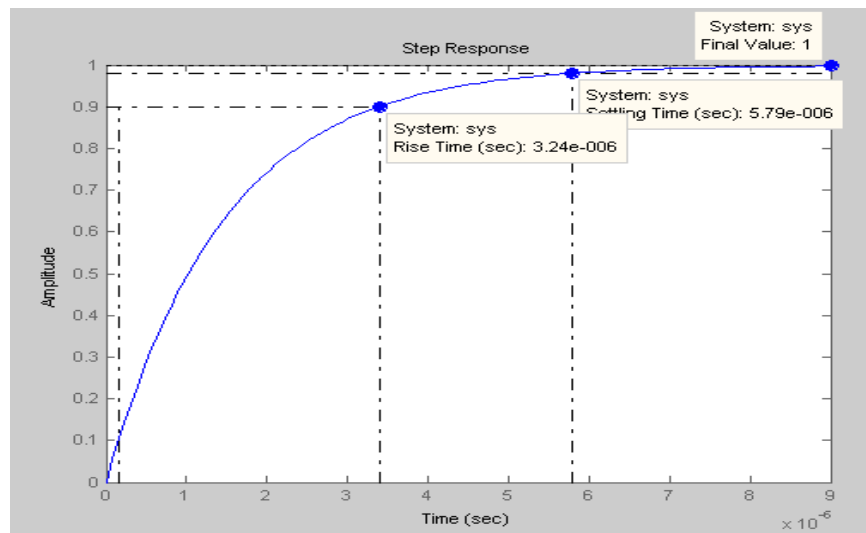
Table 4.4 show increasing the value of Proportional gain,  $K_p$  and reducing the value of Derivative gain,  $K_d$  will reduce time to achieve the peak amplitude. The percentage overshoot is NaN (not a number) and the steady state is infinity.

#### 4.5 Proportional-Integral-Derivative Controller

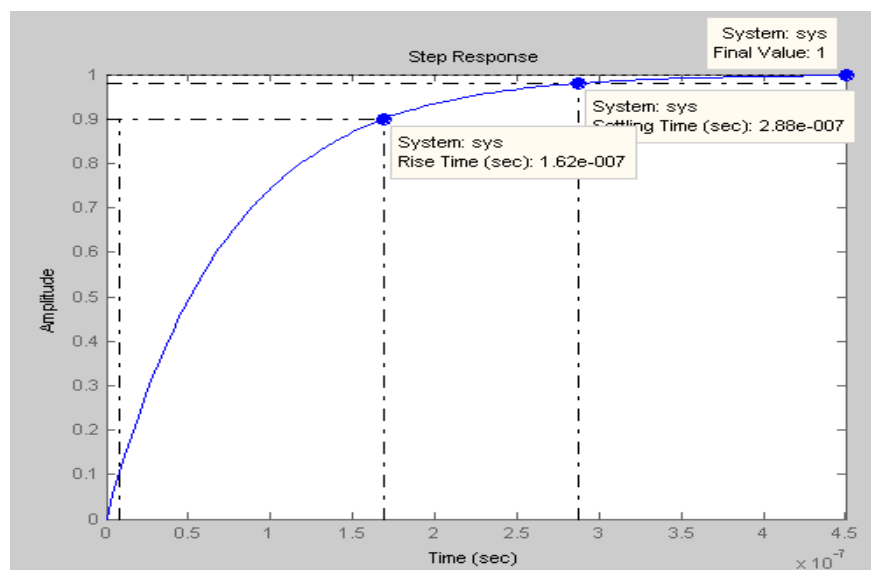
The performance of Proportional-Integral-Derivative (PID) Controller result simulate by tuning the value of proportional gain,  $K_p$ , Integral gain,  $K_i$  and Derivative gain,  $K_d$  are shown in Figure 4.15 to Figure 4.22 using the value of Proportional gain,  $K_p$  70 and 700 and the value of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



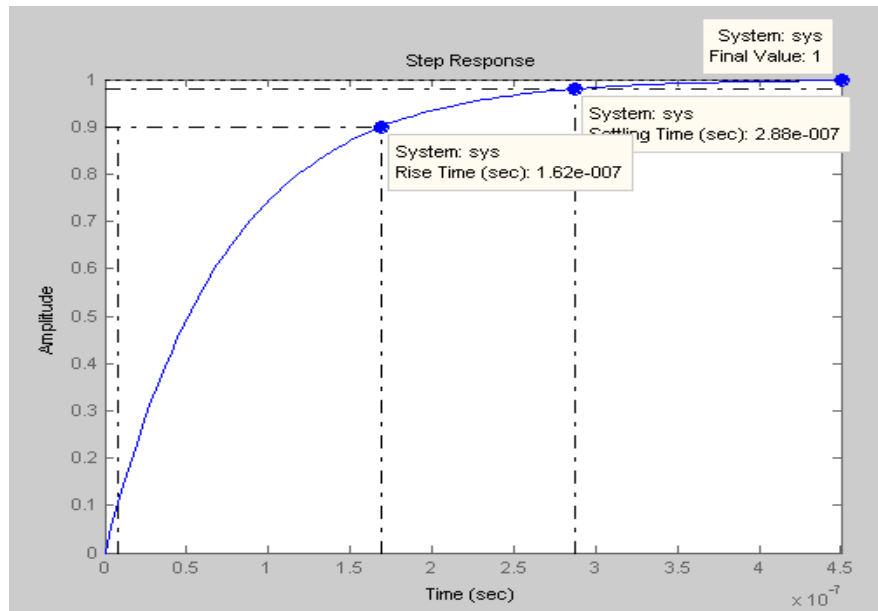
**Figure 4.15** PID controller  $K_p=70$   $K_i=5.18$   $K_d=2.59$



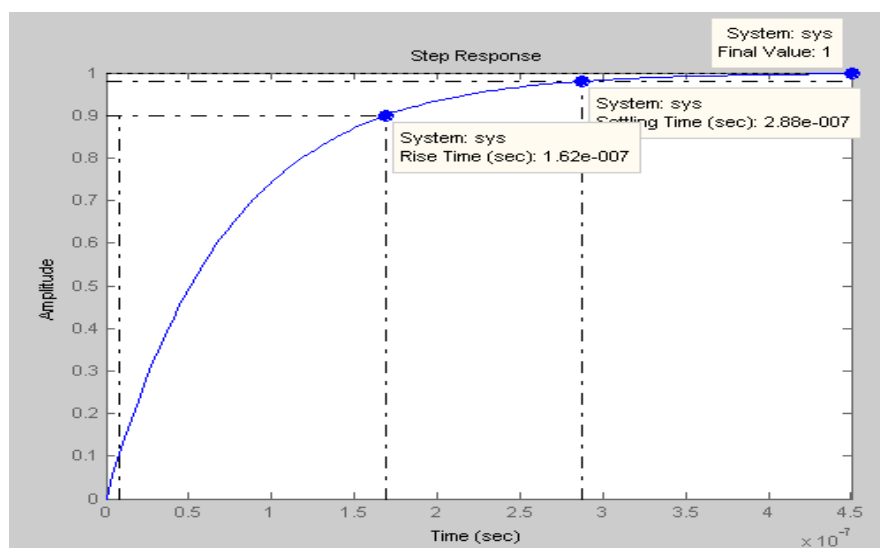
**Figure 4.16** PID controller  $K_p=700$   $K_i=5.18$   $K_d=2.59$



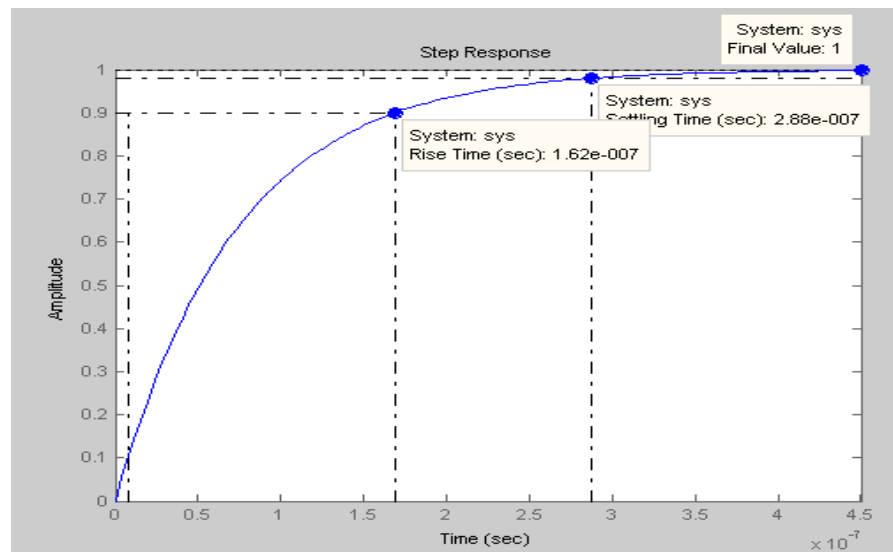
**Figure 4.17** PID controller  $K_p=70$   $K_i=518$   $K_d=51.8$



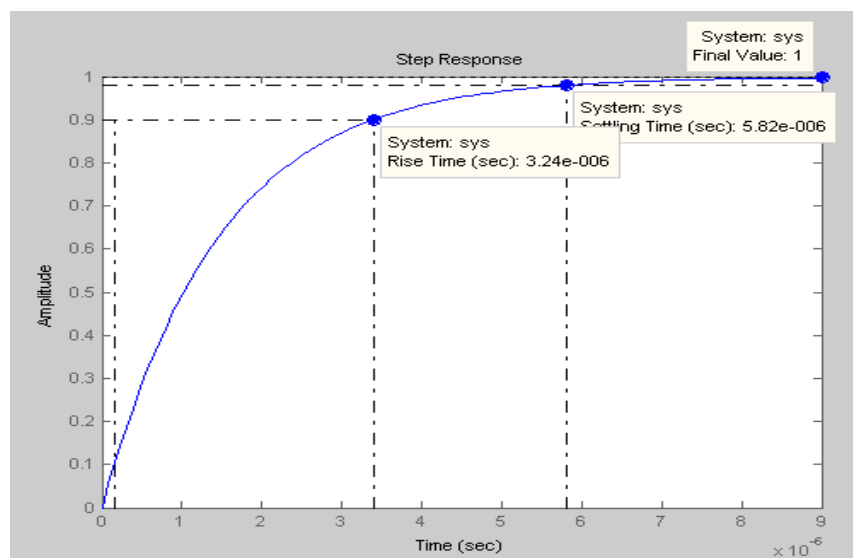
**Figure 4.18** PID controller  $K_p=700$   $K_i=518$   $K_d=51.8$

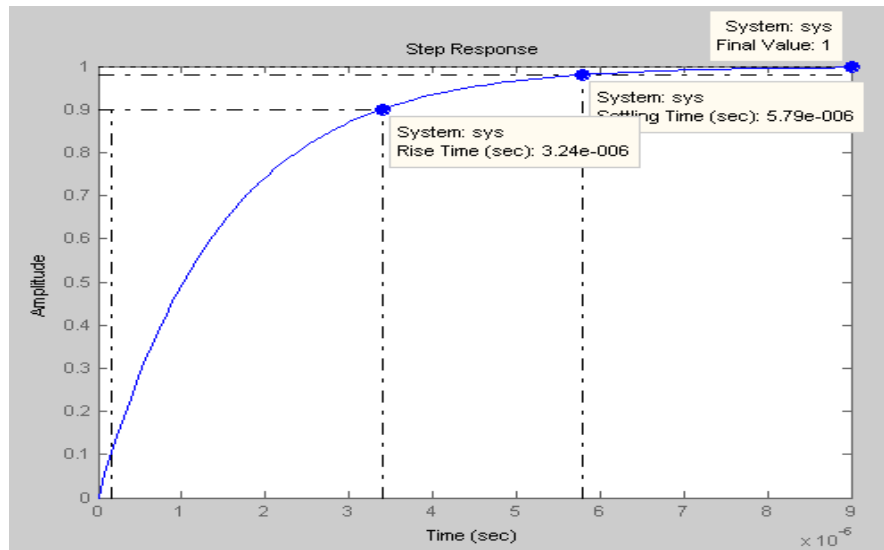


**Figure 4.19** PID controller  $K_p=70$   $K_i=5.18$   $K_d=51.8$



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



**Figure 4.21** PID controller  $K_p=70$   $K_i=518$   $K_d=2.59$ **Figure 4.22** PID controller  $K_p=700$   $K_i=518$   $K_d=2.59$ 

COMPARISON			TIME RISE(s)	SETTLING TIME (s)	STEADY STATE
$K_p$	$K_i$	$K_d$			
70	5.18	2.59	0.00803	0.0144	1
700	5.18	2.59	0.00803	0.0144	1
70	518	51.8	0.00148	0.00262	1
700	518	51.8	0.00148	0.00262	1
70	5.18	51.8	0.00148	0.00262	1
700	5.18	51.8	0.00148	0.00262	1
70	518	2.59	0.00803	0.0144	1
700	518	2.59	0.00803	0.0144	1

**Table 4.5** Comparison of PID controller

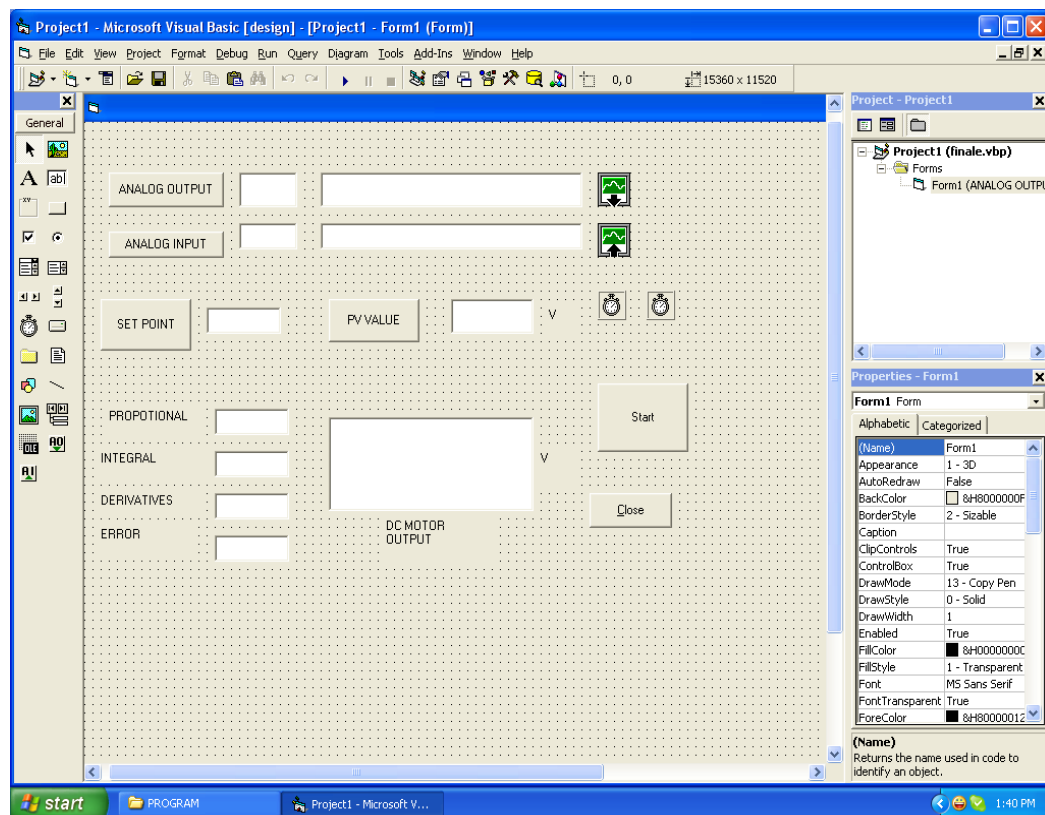


Table 4.5 show increasing the value of Integral gain,  $K_i$  and Derivative gain,  $K_d$  will reduce the value of time rise, settling time, and steady state.

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

#### 4.6 Developing PID controller using Microsoft Visual Basic 6.0

This controller is developing after the Matlab simulations succeed. There are 2 stage of developing this controller which is to programmed an interface with DAQ card and programmed a pid controller for this system. Figure 4.5 shows the picture of Microsoft Visual Basic 6.0 PID controller.



**Figure 4.23** PID controller using Microsoft Visual Basic 6.0

## **CHAPTER 5**

### **CONCLUSION AND RECOMMENDATION**

#### **5.1 Conclusion**

A development of a PID controller for DC motor using Microsoft Visual Basic has been presented in this project. The controller using Matlab show that using PID the system can stabilize effectively. Microsoft Visual Basic is very useful and easy to communicate with DAQ card. The language is simple and object oriented. It is easy to implement PID system onto this software.

The objectives to develop a PID controller using Microsoft Visual Basic achieved but have a problem with servo motor. The driver for servo motor cannot communicate with DAQ card.

## 5.2 Future Recommendation

Although most of the goals are achieved, there are still some enhancement should be introduced to improve its quality. Some suggestions for improvement are:

- Apply this controller with more output and application such as AC motor.
- Use another method such as LQR to determine  $K_p$ ,  $K_i$ ,  $K_d$  value.
- Apply the controller with digital input and output.

Other than that, different controller such as fuzzy logic and pole placement can be used to control the motor. Fuzzy logic is one of the controllers that keep developing nowadays. Using this controller, motor can be control accurately to what is desired and it is more efficient compared to other controller.

## 5.3 Commercialization

Designing a new project is one of implementing study skills. A few upgrades need to be done before this project want to commercialize. To run this project properly, it is wise to use a suitable motor driver that can endure high voltage since the DC motor that had been used works on high voltage. The motor driver also should be able to drives the motor using the PWM signal.

Otherwise, this project can be used as reference for the engineering student or any engineer that using PID controller.

#### **5.4 List and Cost of the Component**

The cost of this project is not necessary because there is much freeware software is develop outside. This project also does not involve the cost of DC servo motor and DAQ card.

## REFERENCES

- [1] 21<sup>st</sup> Januari 2008, The PID controller : Algorithm and implementation by Pasi Airikka, URL <http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=01267781>
- [2] 23<sup>rd</sup> January 2008, PID controller  
URL [http://en.wikipedia.org/wiki/PID\\_controller](http://en.wikipedia.org/wiki/PID_controller)
- [3] 18<sup>th</sup> April 2008, DC motor, Principles and Operation by Eric Seale  
URL [http://www.solarbotics.net/starting/200111\\_dcmotor/200111\\_dcmotor2.html](http://www.solarbotics.net/starting/200111_dcmotor/200111_dcmotor2.html)
- [4] 9<sup>th</sup> February 2008, Automation and control using visual basic tutorial by Dr. Wan  
URL <http://www.easyautomation.ca/tutorial.htm>
- [5] 6<sup>th</sup> June 2008, DAQ card solutions from measurement computing by Measurement Computing  
URL [http://www.measurementcomputing.com/daq\\_card.htm](http://www.measurementcomputing.com/daq_card.htm)
- [6] Page 116, Sergey E. Lysherski, Electromechanical Systems, electric machines and applied mechatronics.

## **APPENDIX A**

### **DAQ CARD MANUAL**

# **USB-4716**

## **200 kS/s, 16-bit, USB Multifunction Module User Manual**

### **Copyright**

The documentation and the software included with this product are copyrighted 2006 by Advantech Co., Ltd. All rights are reserved. Advantech Co., Ltd. reserves the right to make improvements in the products described in this manual at any time without notice. No part of this manual may be reproduced, copied, translated or transmitted in any form or by any means without the prior written permission of Advantech Co., Ltd. Information provided in this manual is intended to be accurate and reliable. However, Advantech Co., Ltd. assumes no responsibility for its use, nor for any infringements of the rights of third parties, which may result from its use.

### **Acknowledgements**

Intel and Pentium are trademarks of Intel Corporation.  
Microsoft Windows and MS-DOS are registered trademarks of Microsoft Corp.  
All other product names or trademarks are properties of their respective owners.

### **Product Warranty (2 years)**

Advantech warrants to you, the original purchaser, that each of its products will be free from defects in materials and workmanship for two years from the date of purchase.

This warranty does not apply to any products which have been repaired or altered by persons other than repair personnel authorized by Advantech, or which have been subject to misuse, abuse, accident or improper installation. Advantech assumes no liability under the terms of this warranty as a consequence of such events.

Because of Advantech's high quality-control standards and rigorous testing, most of our customers never need to use our repair service. If an Advantech product is defective, it will be repaired or replaced at no charge during the warranty period. For out-of-warranty repairs, you will be billed according to the cost of replacement materials, service time and freight. Please consult your dealer for more details.

If you think you have a defective product, follow these steps:

1. Collect all the information about the problem encountered. (For example, CPU speed, Advantech products used, other hardware and software used, etc.) Note anything abnormal and list any onscreen messages you get when the problem occurs.
2. Call your dealer and describe the problem. Please have your manual, product, and any helpful information readily available.
3. If your product is diagnosed as defective, obtain an RMA (return merchandise authorization) number from your dealer. This allows us to process your return more quickly.
4. Carefully pack the defective product, a fully-completed Repair and Replacement Order Card and a photocopy proof of purchase date (such as your sales receipt) in a shippable container. A product returned without proof of the purchase date is not eligible for warranty service.
5. Write the RMA number visibly on the outside of the package and ship it prepaid to your dealer.

#### **CE**

This product has passed the CE test for environmental specifications when shielded cables are used for external wiring. We recommend the use of shielded cables. This kind of cable is available from Advantech. Please contact your local supplier for ordering information.

#### **Technical Support and Assistance**

Step 1. Visit the Advantech web site at **[www.advantech.com/support](http://www.advantech.com/support)** where you can find the latest information about the product.

Step 2. Contact your distributor, sales representative, or Advantech's customer service center for technical support if you need additional assistance. Please have the following information ready before you call:

- Product name and serial number
- Description of your peripheral attachments
- Description of your software (operating system, version, application software, etc.)
- A complete description of the problem
- The exact wording of any error message

#### **Document Feedback**

To assist us in making improvements to this manual, we would welcome comments and constructive criticism. Please send all such - in writing to: [support@advantech.com](mailto:support@advantech.com).



### **Safety Precaution - Static Electricity**

Follow these simple precautions to protect yourself from harm and the products from damage.

1. To avoid electrical shock, always disconnect the power from your PC chassis before you work on it. Don't touch any components on the CPU card or other cards while the PC is on.
2. Disconnect power before making any configuration changes. The sudden rush of power as you connect a jumper or install a card may damage sensitive electronic components

## **Introduction**

This chapter will provide information on the features of the DAS module, a quick start guide for installation, and some brief information on software and accessories for the USB-4716 Module.

Sections include:

- Features
- Software Overview

The Advantech USB-4716 is a powerful data acquisition (DAS) module for the USB port. It features a unique circuit design and complete functions for data acquisition and control

### **1.1 Features**

- 2 USB-4716 has the most requested measurement & control functions:
- 3 • 16 single-ended/ 8 differential or combination analog input channels
- 4 • 16-bit resolution A/D converter, with up to 200 kS/s sampling rate
- 5 • 8 digital input & 8 digital output channels (TTL Level)
- 6 • 2 analog output channels
- 7 • 16-bit programmable counter/timer x 1
- 8 • Programmable gain for each analog input channel
- 9 • Automatic channel/gain scanning
- 10 • Onboard 1K samples FIFO buffer for AI channels
- 11 • Bus-powered

- 12 • Device status LED indicator
- 13 • Removable on-module wiring terminal
- 14 • Supports high-speed USB 2.0
- 15 • Auto calibration function
- 16 • Hot swappable

*Note: The USB chip on your system may have a limitation on the number of USB devices it will support. Normally, only five USB-4716 devices can be supported.*

*Note: The power output of an USB port is 500 mA, while the USB-4716 requires 360 mA (typical). This means that if an USB hub is used, it will need an external power supply to support more than one USB-4716 device.*

## 1.2 Software Overview

Advantech offers a rich set of DLL drivers, third-party driver support and application software on the companion CD-ROM to help fully exploit the functions of your device. Advantech's Device Drivers feature a complete I/O function library to help boost your application performance and work seamlessly with development tools such as Visual C++, Visual Basic, Inprise C++ Builder, and Inprise Delphi.

### 1.2.1 More on the CD

For instructions on how to begin programming in each development tool, Advantech offers some tutorial chapters in the Device Drivers Manual for your reference. Please refer to the corresponding sections in these chapters on the Device Drivers Manual to begin your programming efforts. You can also look at the example source code provided for each programming tool, since they can get you very well oriented. The Device Drivers Manual can be found on the companion CD-ROM. Alternatively, if you have already installed the Device Drivers on your system, The Device Drivers Manual can be readily accessed through the

Start button:

**Start/Programs/Advantech Automation/Advantech Device Manager /  
Device Driver's Manual**

The example source code can be found under the corresponding installation folder such as the default installation path:

**\Program Files\Advantech\ADSAPI\Examples**

## Installation

Sections include:

- Unpacking
- Driver Installation
- Hardware Installation
- Hardware Uninstallation

### 2.1 Unpacking

After receiving your USB-4716 package, please inspect its contents first. The package should contain the following items:

- USB-4716 Module
- Shielded USB 2.0 Cable (1.8 m)
- Companion CD-ROM (DLL driver included)
- User Manual

The USB-4716 Module harbors certain electronic components vulnerable to *electrostatic discharge* (ESD). ESD could easily damage the integrated circuits and certain components if preventive measures are not carefully paid attention to. ***Before removing the module from the antistatic plastic bag, you should take following precautions to ward off possible ESD damage:***

- Touch the metal part of your computer chassis with your hand to discharge static electricity accumulated on your body. One can also use a grounding strap.
- Make contact between the antistatic bag and ground before opening.

**After taking out the module, you should first:**

Inspect the module for any possible signs of external damage (loose or

damaged components, etc.). If the module is visibly damaged, please notify our service department or our local sales representative immediately. Avoid using a damaged module with your system.

- Avoid physical contact with materials that could hold static electricity such as plastic, vinyl and Styrofoam.

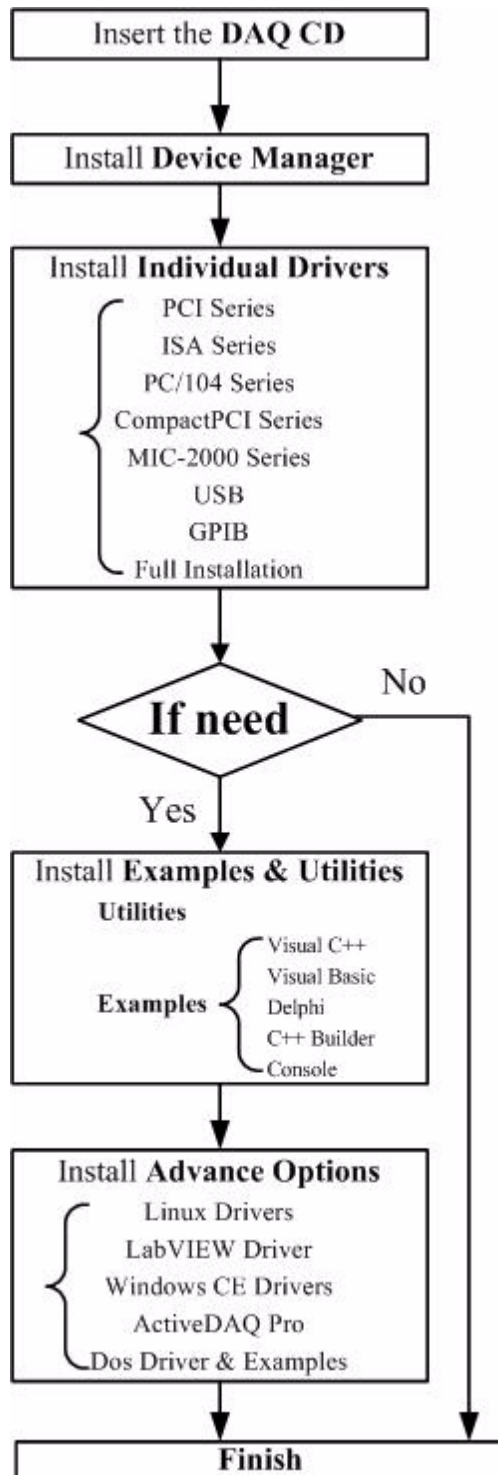
## 2.2 Driver Installation

We recommend you install the software driver before you install the USB-4716 module into your system, since this will guarantee a smooth installation process.

The 32-bit DLL driver Setup program for the USB-4716 module is included on the companion CD-ROM that is shipped with your module package. Please follow the steps on the following page to install the driver software:

For further information on driver-related issues, an online version of the Device Drivers Manual is available by accessing the following path:

**Start\Programs\Advantech Automation  
\Device Manager\Device Driver's Manual**



## 2.3 Hardware Installation

*Note: Make sure you have installed the software driver before you install the module (please refer to Section 2.2 Driver Installation)*

After the DLL driver installation is completed, you can now go on to install the USB-4716 module in any USB port that supports the USB 1.1/2.0 standard, on your computer. Please follow the steps below to install the module on your system.

**Step 1:** Touch the metal part on the surface of your computer to neutralize the static electricity that might be in your body.

**Step 2:** Plug your USB module into the selected USB port. Use of excessive force must be avoided; otherwise the module might get damaged.

*Note: In case you installed the module without installing the DLL driver, Win2000/XP will recognize your module as an “unknown device”. After reboot, it will prompt you to provide necessary driver. You should ignore the prompting messages and set up the driver according to the steps described in Sec.2.2.*

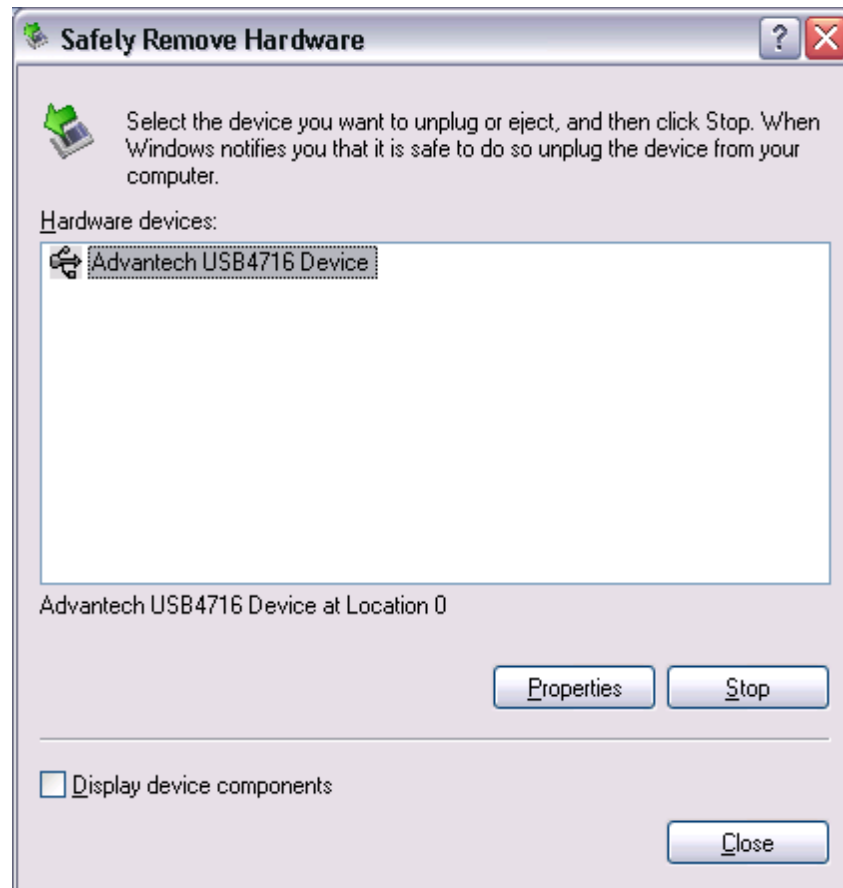
After your module is installed, you can configure it using the Advantech Device Manager. The Device Driver's Manual can be found at:  
*Start\Programs\Advantech Automation\Advantech Device Manager\Device Driver's Manual*

## 2.4 Hardware Uninstallation

Though the Advantech USB modules are hot swappable, we still recommend you to follow the hardware un-installation procedure to avoid any unpredictable damages to your device or your system.

**Step1:** Close the applications of the USB module.

**Step2:** Right click the “Unplug or Eject Hardware” icon on your task bar.



*Figure 2.1: Unplug or Eject Hardware Dialog*

**Step3:** Select “Advantech USB-4716 Device” and press “Stop” Button.



*Figure 2.2: Stop a Hardware device dialog box*

**Step4:** Unplug your USB device from the USB port.

*Note: Please make sure that you have closed the application before unplugging the USB device, otherwise unexpected system error or damage may occur.*



## Signal Connections

This chapter provides useful information on how to connect input and output signals to the USB-4716 via the I/O connectors.

Sections include:

- Overview
- I/O Connectors
- Analog Input Connections
- Analog Output Connections
- Trigger Source Connections
- Field Wiring Considerations

## Chapter 3 Signal Connections

### 3.1 Overview

Maintaining good signal connections is one of the most important factors in ensuring that your application system is sending and receiving data correctly. A good signal connection can avoid unnecessary and costly damage to your PC and other hardware devices.

### 3.2 I/O Connectors

USB-4716 is equipped with plug-in screw-terminal connectors that facilitate connection to the module without terminal boards or cables.

#### 3.2.1 Pin Assignment

Figure 3.1 on next page shows the pin assignments for the five 10-pin I/O connectors on USB-4716.

*Warning: The two ground references AGND and DGND should be used separately for their designated purpose. Do not connect them together.*

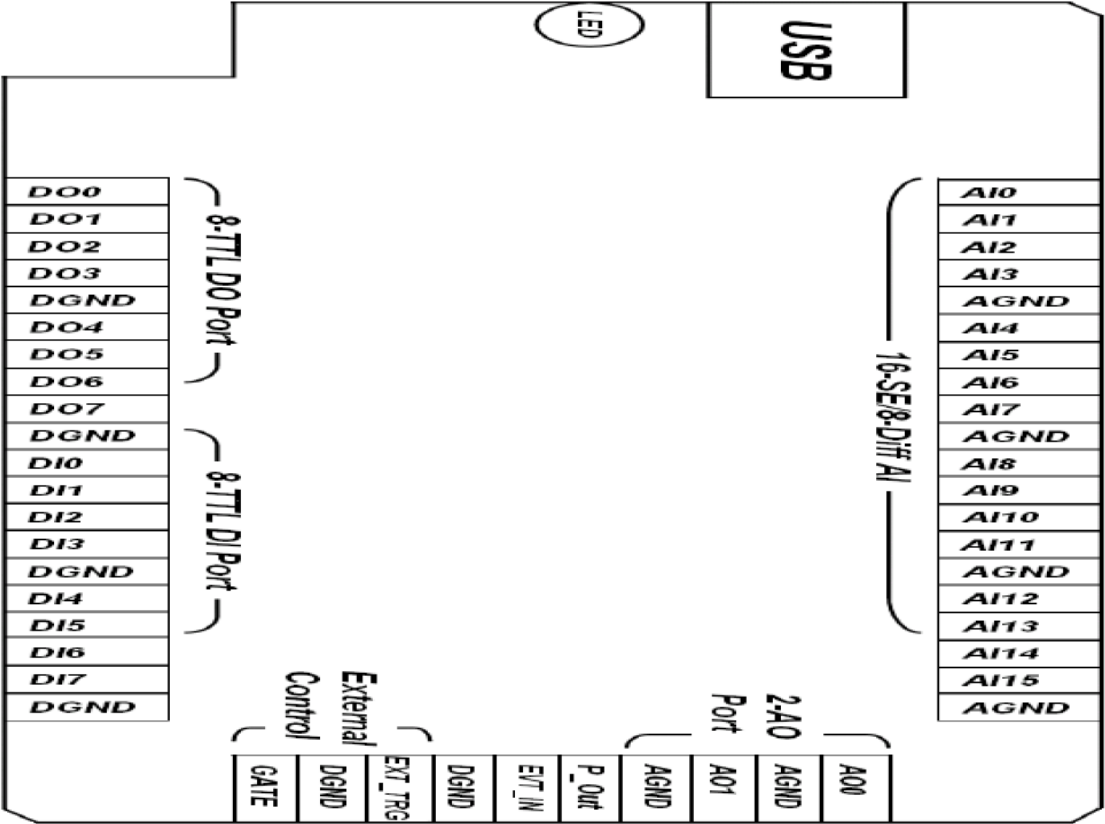


Figure 3.1: I/O Connector Pin Assignment

3.2.2 I/O Connector Signal Description

**Table 3.1: I/O Connector Signal Description**

Signal Name	Reference	Direction	Description
AI<0...15>	AGND	Input	Analog Input Channels 0 through 15.
AIGND	-	-	Analog Input Ground.
AO0 AO1	AGND	Output	Analog Output Channels 0/1.
AOGND	-	-	Analog Output Ground. The analog output voltages are referenced to these nodes.
DI<0..7>	DGND	Input	Digital Input channels.
DO<0..7>	DGND	Output	Digital Output channels.
DGND	-	-	Digital Ground. This pin supplies the reference for the digital channels at the I/O connector.
GATE	DGND	Input	A/D External Trigger Gate. When GATE is connected to +5 V, it will disable the external trigger signal to input.
EXT_TRG	DGND	Input	A/D External Trigger. This pin is external trigger signal input for the A/D conversion. A low-to-high edge triggers A/D conversion to start.
EVT_IN	DGND	Input	External events input channel.
P_OUT	DGND	Output	Pulse output channel

### 3.2.3 LED Indicator Status Description

The USB Module is equipped with a LED indicator to show the current status of the device. When you plug the USB device into the USB port, the LED indicator will blink five times and then stay lit to indicate that it is on. Please refer to the following table for detailed LED indicator status information.

<b>Table 3.2: LED Indicator Status Description</b>	
LED Status	Description
On	Device ready for work
Off	Device not ready to work
Slow Blinking (5 times)	Device initialization
Fast Blinking (Depends on data transfer speed).	Device working

## 3.3 Analog Input Connections

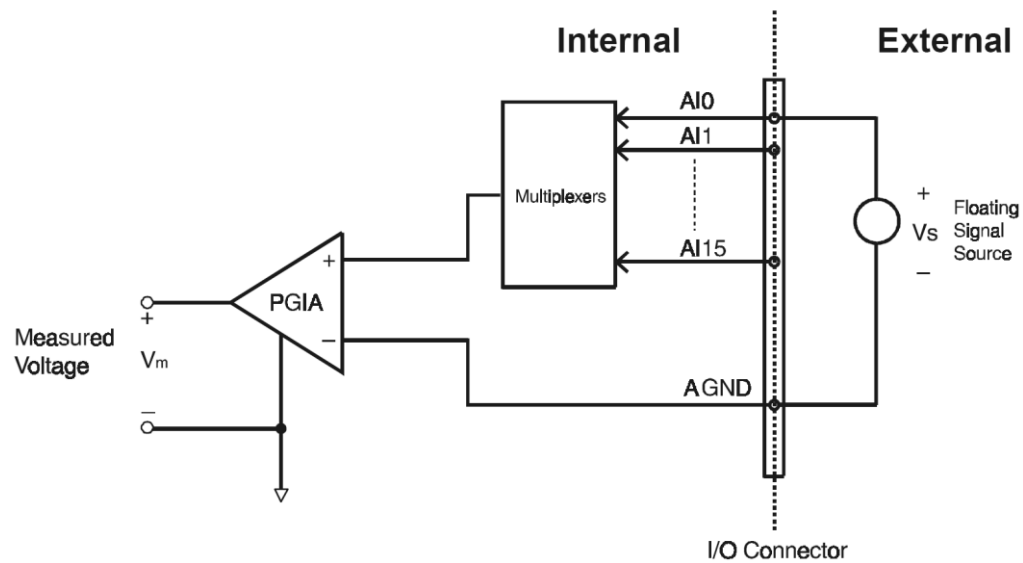
USB-4716 supports 16 single-ended/ 8 differential (or combination) analog inputs. Each individual input channel is software-selected.

### 3.3.1 Single-ended Channel Connections

channel, and the *measured voltage* ( $V_m$ ) is the voltage of the wire as referenced against the common ground.

A signal source without a local ground is also called a “floating source”. It is fairly simple to connect a single-ended channel to a floating signal source. In this mode, USB-4716 provides a reference ground for external floating signal sources.

Figure 3.2 shows a single-ended channel connection between a floating signal source and an input channel on USB-4716.



*Figure 3.2: Single-Ended Input Channel Connection*

### 3.3.4 Differential Input Connections

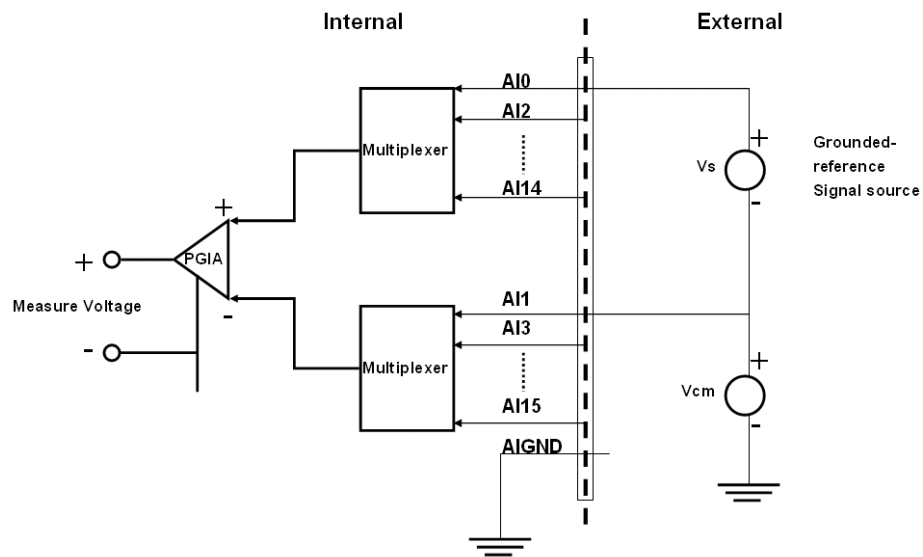
The differential input channels operate with two signal wires for each channel, and the voltage difference between both signal wires is measured. On USB-4716, when all channels are configured to differential input, up to 8 analog channels are available.

If one side of the signal source is connected to a local ground, the signal source is ground-referenced. Therefore, the ground of the signal source and the ground of the card will not be exactly of the same voltage. The difference between the ground voltages forms a commonmode voltage ( $V_{cm}$ ).

To avoid the ground loop noise effect caused by common-mode voltages, you can connect the signal ground to the *Low* input. Figure 3-3 shows a

differential channel connection between a grounded-reference signal source and an input channel on USB-4716. With this connection, the PGIA rejects a common-mode voltage  $V_{cm}$  between the signal source and USB-4716 ground, shown as  $V_{cm}$  in Figure 3-3.

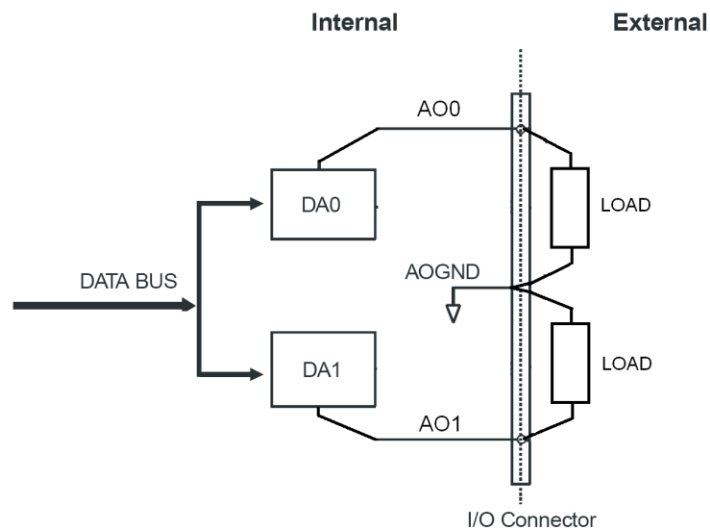
**Note:** In differential input mode, the input channel  $n$  should be used with channel  $n+1$ .  
( $n=0,2,4\ldots14$ )



*Figure 3.3: Differential Input Channel Connection*

### 3.4 Analog Output Connections (Voltage)

USB-4716 provides two analog output channels, AO0 and AO1. Figure 3-3 shows how to make analog output connections on USB-4716.



*Figure 3.4: Analog Output Channel Connections*

## 3.5 Trigger Source Connections

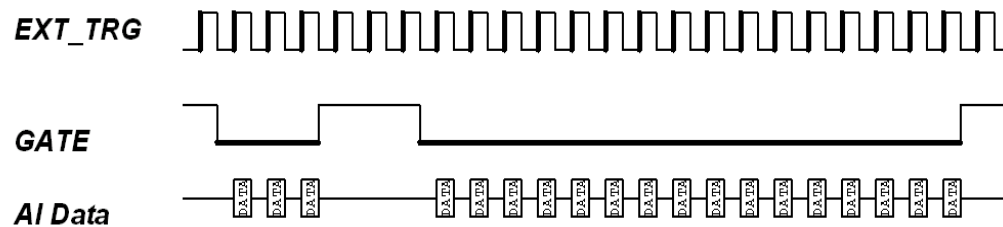
### 3.5.1 Internal Pacer Trigger Connection

USB-4716 provides two 16-bit counters connected to a 10 MHz clock. Counter 0 is a counter that counts events from an input channel. Counter 1 is a 16-bit timer for pacer triggering. A low-to-high edge from the Counter 1 output will trigger an A/D conversion on USB-4716.

### 3.5.2 External Trigger Source Connection

In addition to pacer triggering, USB-4716 also allows external triggering for A/D conversions. When GATE is connected to a +5V DC source, the external trigger function is thereby disabled. And the external trigger function will be enabled once the +5V DC source is removed.

### ***External Trigger Mode :***



## **3.6 Field Wiring Considerations**

When you use USB-4716 to acquire data from outside, noises in the environment might significantly affect the accuracy of your measurements if due cautions are not taken. The following measures will be helpful to reduce possible interference running signal wires between signal sources and the USB-4716.

- The signal cables must be kept away from strong electromagnetic sources such as power lines, large electric motors, circuit breakers or welding machines, since they may cause strong electromagnetic interference. Keep the analog signal cables away from any video monitor, since it can significantly affect a data acquisition system.
- If the cable travels through an area with significant electromagnetic interference, you should adopt individually shielded, twisted-pair wires as the analog input cable. This type of cable has its signal wires twisted together and shielded with a metal mesh. The metal mesh should only be connected to one point at the signal source ground.
- Avoid running the signal cables through any conduit that might have power lines in it.
- If you have to place your signal cable parallel to a power line that has a high voltage or high current running through it, try to keep a safe distance between them. Or place the signal cable in a right angle to the power line to minimize the undesirable effect.



## Specifications

### A.1 Analog Input

Channels	16-ch single-ended/ 8-ch differential						
Resolution	16 bits	FIFO Size	1024 Samples				
Sampling Rate		200 kS/s					
Input Range and Gain List	Gain	0.5	1	2	4	8	
	Gain Code	4	0	1	2	3	
	Bipolar (V)	±10	±5	±2.5	±1.25	±0.625	
	Unipolar (V)	N/A	0~10	0~5	0~2.5	0~1.25	
Drift	Gain	0.5	1	2	4	8	
	Zero (μV/° C)	±30					
	Gain (ppm/° C)	30	30	30	30	30	
Small Signal Bandwidth for PGA	Gain	0.5	1	2	4	8	
	Bandwidth (MHz)	1.1	1.1	1.1	1.1	1.1	
Input Protection		30 V max.					
Input Impedance		1GW					
Input Comm. Mode Voltage		11V					
Accuracy	DC	INLE	1LSB				
		DNLE	3LSB				
		Gain	0.5	1	2	4	8
		Gain Error (% of FSR )	0.015	0.03	0.03	0.05	0.1
	AC	SINAD	83 dB				
		THD	-88 dB				
		ENOB	13.5 bit				

## A.2 Analog Output

Channels	2		
Resolution	16 bits	FIFO Size	N/A
Throughput	2 kHz		
Operating Mode	Single output		
Output Range	0~5, 0~10, $\pm 5$ , $\pm 10$ V		
Accuracy	DC	INLE	$\pm 2$ LSB
		DNLE	$\pm 1$ LSB
Dynamic Performance	Slew Rate	0.125 V/ $\mu$ s	
	Settling Time	150 $\mu$ s (to $\pm 1/2$ LSB of FSB)	
Driving Capability	5 mA		
Output Impedance	0.1W max.		

## A.3 Non-Isolated Digital Input/Output

Input Channels	8 Non-Isolation TTL	
Input Voltage	Low	0.0 Vdc (Min) / 1.0Vdc (Max)
	High	2.0 Vdc (Min) / 5.0Vdc (Max)
Output Channels	8 Non-Isolation TTL	
Output Voltage	Low	0.4 Vdc / -8mA (Sink)
	High	2.4 Vdc / 8mA (Source)

## **APPENDIX B**

### **SERVO MOTOR MANUAL INSTALLATION**

## 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 2-56 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

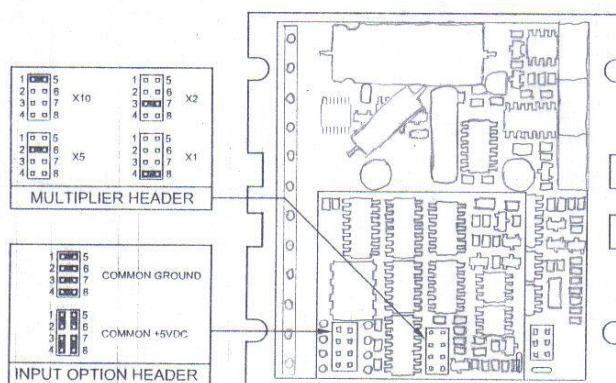
The G340 has a built in STEP PULSE MULTIPLIER. This circuit puts out 1, 2, 5, or 10 pulses for every input STEP pulse. All motor moves will be in these increments. For example, if X10 is selected, then the motor will move 10 encoder counts for every STEP pulse sent to the G340. Move the jumper on the 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 "+" 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:

$$(45\text{kHz} / 40) / 4 = 281.25$$

#### 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

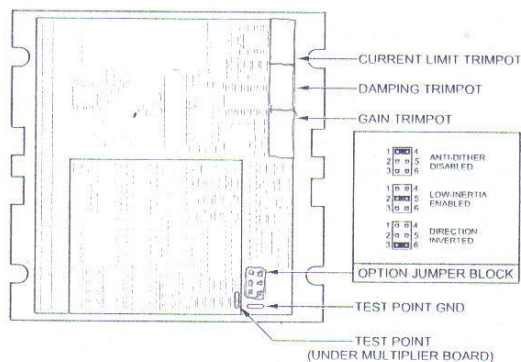
At this point the encoder should be tested for functionality. If you wish to monitor the POSITION ERROR test point with a voltmeter or oscilloscope, then remove the cover of the drive now. If you have a choice, pick the oscilloscope.

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.

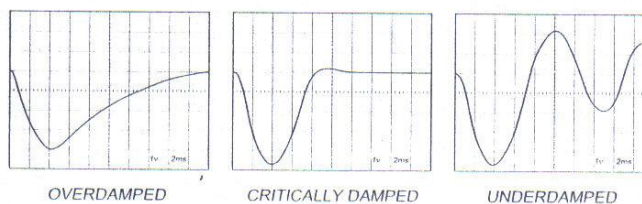
### STEP 7: TUNING THE SERVO

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 PID 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 trimpot sets maximum current the motor is permitted to have. It is adjustable from 0 amps to 20 amps. Normally the LIMIT trimpot 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.

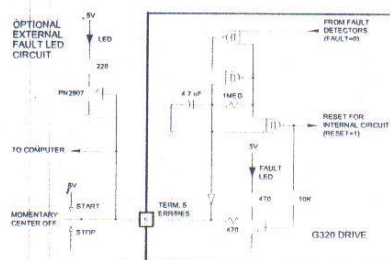


When first testing the G340, ERR/RES (term. 5) was connected to ENC+ (term. 7). It can be left that way if it is not necessary to read the state of the ERROR output. Otherwise, the following details are important.

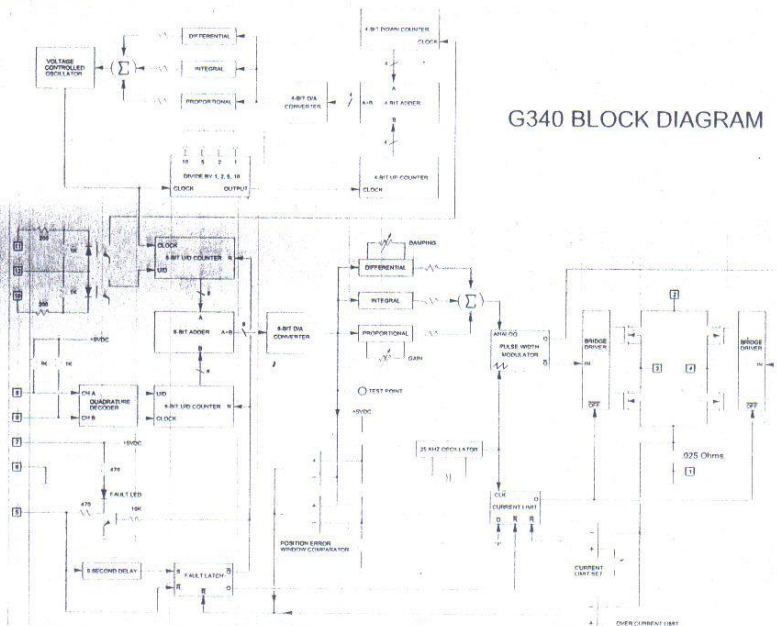
The ERROR output is latched in the "ERROR" state (term. 5 = "0") by the power-on reset circuitry in the G340. It will stay in this state indefinitely until it is cleared by applying +5V to this terminal for at least 1 second.

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.

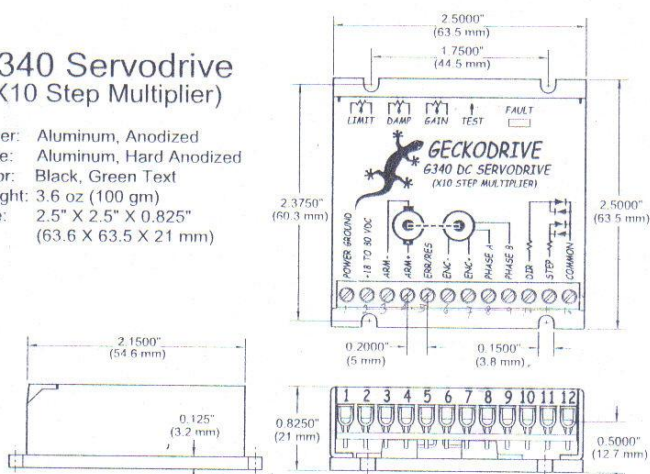


At anytime the operator can push the switch to the "STOP" position to immediately halt the G340 drive. Anytime the G340 is in the "FAULT" state (FAULT LED lit), all switching action stops and the motor freewheels and is unpowered. This will light the "FAULT" light.



## 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"  
(63.6 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, Trimpot Adjustable
Analog PID	Damping and Gain Trimpots
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.  
14662 Franklin Ave.  
Suite E  
Tustin, CA 92780

Phone: 1-714-832-8874  
Fax: 1-714-832-8082  
Web Site: [www.geckodrive.com](http://www.geckodrive.com)



A Full Service Motion Control Distributor and Systems Integrator.

JDH-2250-HF-2C-E

DM-683

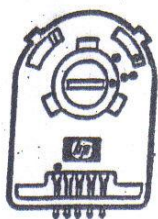
CLIFTON PRECISION

**REF. ONLY..**

Supply Voltage,  $V_{cc}$  ..... -0.5 to 7V  
 Output Voltage,  $V_o$  ..... -0.5 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

**APPENDIX C**  
**MICROSOFT VISUAL BASIC PROGRAMMED**

```
Private Sub cmdExit_Click()
```

```
Unload Me
```

```
End
```

```
End Sub
```

```
Private Sub Command1_Click()
```

```
DEMOAO.SelectDevice
```

```
Text1.Text = DEMOAO.DeviceNumber
```

```
Text2.Text = DEMOAO.DeviceName
```

```
End Sub
```

```
Private Sub Command2_Click()
```

```
tmrpid.Enabled = True
```

```
End Sub
```

```
Private Sub Command3_Click()
```

```
DAQAI1.SelectDevice
```

```
Text4.Text = DAQAI1.DeviceNumber
```

```
Text5.Text = DAQAI1.DeviceName
```

```
End Sub
```

```
Private Sub start_Click()
```

```
tmrpid.Interval = tmrin.Interval
```

```
End Sub
```

```
Private Sub tmrin_Timer()  
  
    DAQAI1.OpenDevice  
  
    pv.Text = DAQAI1.RealInput(0)  
  
End Sub
```

```
Private Sub tmrpid_Timer()  
  
    process = DAQAI1.RealInput(0)  
  
    setp = sp.Text  
  
    t = tmrpid.Interval
```

```
    er = setp - process  
  
    er_old = er  
  
    Kp = er * p.Text  
  
    Ki = i.Text * ((er_old - er) / t)  
  
    Kd = d.Text * ((er_old - er) * t)
```

```
    output.Text = Kp + Ki + Kd  
  
    output.Text = DAQAO.output  
  
End Sub
```