Fall 1996

Vibration control of robotic modules using input shaping algorithm

Murat Eren
New Jersey Institute of Technology

Copyright Warning & Restrictions

The copyright law of the United States (Title 17, United States Code) governs the making of photocopies or other reproductions of copyrighted material.

Under certain conditions specified in the law, libraries and archives are authorized to furnish a photocopy or other reproduction. One of these specified conditions is that the photocopy or reproduction is not to be "used for any purpose other than private study, scholarship, or research." If a, user makes a request for, or later uses, a photocopy or reproduction for purposes in excess of "fair use" that user may be liable for copyright infringement,

This institution reserves the right to refuse to accept a copying order if, in its judgment, fulfillment of the order would involve violation of copyright law.

Please Note: The author retains the copyright while the New Jersey Institute of Technology reserves the right to distribute this thesis or dissertation

Printing note: If you do not wish to print this page, then select "Pages from: first page # to: last page #" on the print dialog screen

ABSTRACT

VIBRATION CONTROL OF ROBOTIC MODULES USING INPUT SHAPING ALGORITHM

by
Murat Eren

In this thesis, vibration control using input shaping algorithm is studied. Vibration Control of flexible structures is an important problem and has been an active research area. Different approaches have been developed for vibration control which can be divided roughly into feedback and feedforward methods. Feedback methods need measurements and on-line calculation of the controller outputs. Although feedback methods are generally more robust and have a number of well known performance advantages, proper use of feedforward control can also significantly improve the speed of response of the system.

Input shaping is one of these feedforward methods. It has been successfully applied to many control problems even in the presence of modeling uncertainties and structural nonlinearities. In many industrial problems, the objective is to position a load in minimum time without exciting the vibratory modes. In input shaping, the aim is to give zero energy to these modes by performing "input prefiltering" or equivalently pole-zero cancellation in the command feedforward path. To carry out this prefiltering function, the natural frequency (ω_n) and the damping ratio (ζ) of the plant are required for the shaper design [1].

This work is organized as follows; in the hardware part, basic information about a cartesian robotic module, an EXC controller, a VME controller, and a Dalanco Spry digital signal processing board is given. In Chapter 3 the input shaping technique is introduced. In Chapter 4 control system design and implementation of Zero Vibration (ZV), Zero Vibration & Derivative (ZVD), and Extra Insensitive (EI) shapers are given. In Chapter 5 results of ZV, ZVD, and EI shapers will be given.

Comparison and suggestions for improvement are also given in this chapter. Finally, concluding remarks are given in Chapter 6.

VIBRATION CONTROL OF ROBOTIC MODULES USING INPUT SHAPING ALGORITHM

by
Murat Eren

A Thesis
Submitted to the Faculty of
New Jersey Institute of Technology
in Partial Fulfillment of the Requirements for the Degree of
Master of Science in Electrical Engineering

Department of Electrical and Computer Engineering

January 1997

APPROVAL PAGE

VIBRATION CONTROL OF ROBOTIC MODULES USING INPUT SHAPING ALGORITHM

Murat Eren

Dr. Timothy N. Chang, Thesis Advisor — Date
Associate Professor of Electrical and Computer Engineering, NJIT

Dr. Andrew Meyer, Committee Member — Date
Professor of Electrical and Computer Engineering, NJIT

Dr. Edwin Hou, Committee Member — Date
Associate Professor of Electrical and Computer Engineering, NJIT

BIOGRAPHICAL SKETCH

Author: Murat Eren

Degree: Master of Science in Electrical Engineering

Date: January 1997

Undergraduate and Graduate Education:

- Master of Science in Electrical Engineering,
  New Jersey Institute of Technology, Newark, NJ, 1997
- Bachelor of Science in Electronics and Telecommunication Engineering,
  Istanbul Technical University, Istanbul, Turkey, 1992

Major: Electrical Engineering

To my family

ACKNOWLEDGMENT

First of all, I would like to thank the committee members. I would especially like to thank Dr. Timothy N. Chang, for financially supporting me. During my thesis he helped me in every possible way. He is a great advisor. I got a lot of theoretical and practical knowledge from him,

I would like to thank to my family, for their support and understanding. I do not think I would have been able to accomplish much without their support. Especially my sister Nalan Eren, she always gave positive comments when I had problems.

I also would like to thank Kedar A. Godboley for helping me in software and debugging programs and algorithms.

TABLE OF CONTENTS

TABLE OF CONTENTS
(Continued)

LIST OF TABLES

LIST OF FIGURES

LIST OF FIGURES
(Continued)

Figure Page

CHAPTER 1

INTRODUCTION

1.1 Objective

Control of flexible structures is an important problem and an active area of research. There are a number of control algorithms currently available which can be roughly divided in to two groups: feedback and feedforward approaches. Input shaping is one these feedforward algorithms. It has been successfully applied for vibration control of flexible structures.

The objective of this thesis is to implement several types of input shapers to reduce residual vibration of a 2 axis cartesian robot. A significant effort has been spent in hardware construction of this work, therefore hardware will first be described in Chapter 2. The limitations of each hardware will be given in this chapter. Input shaping is a versatile and effective method to for command vibration control. The derivation and implementation of the method will be given in Chapter 3 and Chapter 4. Results and suggestions for improvements will be in Chapter 5 and finally conclusions will be given in Chapter 6.

1.2 General Description of System

The system hardware consists of a two axis cartesian robot, an Adept VME controller, a DSP installed IBM compatible PC, and an interface circuitry between the VME controller-robot and the VME controller-PC. Figure (1.1) shows system overview and following figures, Figure (1.2), (1.3), and (1.4) shows actual system view. From Figure (1.1) VME controller sends position data to PC via its serial port. DSP in PC gets position data, calculates control action, and sends control action to linear module via amplifier. Before PC sends control action to linear module, proportional and integral gains of VME must be zeroed. The purpose of

this step is to disable VME to control the linear module. Figure (1.2) shows the two axis cartesian robot mounted on vibration free table. H module is the x axis and M module is the y axis. Figure (1.3) shows VME control part of the system. On the left part there is Manual Control Pendant (MCP) and on the right side of figure partly VMI Interface Panels (VMP) can be seen. And finally in Figure (1.4), PC part of the system can be seen. In the middle of figure there is charge amplifier and signal generator. Charge amplifier is used for accelerometer and signal generator is used for debugging software and DSP. To interface PC and VME a summer circuit is used. This circuit basically makes DAC voltage matching between VME and PC. It simply multiplies DSP output by 2 since VME output range is ± 10 V and DSP's output is ± 5 V. For DSP Dalanco Spry's Model 310 data acquisition and signal processing board is used. This board is a TMS320C31 based board and it has 4 ADC (14 bits) inputs and 2 DAC (12 bits) outputs. In early stages of this thesis before VME Controller was shipped to us, an EXC controller was applied to perform the initialization tests. Some descriptions of the EXC Controller will also be given in this chapter.

For control algorithm input shaping will be used. Input shaping is an open loop compensator of which vibrational motion is eliminated after the input ends. In input shaping a sequence of impulses is convolved with reference input. Closed loop eigenvalues determine the amplitudes, and application times of these impulse sequence. So frequencies and damping of the modes of vibration has to be known. There are several methods for shaper design. In this work Zero Vibration (ZV), Zero Vibration & Derivative (ZVD), and Extra Insensitive (EI) methods will be explained and applied. In ZV shaper method residual vibration will be zero after impulse sequence ends. To increase robustness of the shaper design to modeling errors in the natural frequency ω_0, and damping ratio ζ, the partial derivatives of the ZV constraints with respect to ω_0 and ζ are also constrained to zero. Basically

in ZVD an extra zero is put to the exact location of the closed loop poles. In EI a finite residual vibration at the modeled frequency is allowed. But frequencies slightly above and below the modeled frequency will be zero. This method will increase the frequency insensitivity.

Input shaping has become an active area of research after the publication of [2]. For more information readers should refer to [3, 4, 5, 6, 7, 8].

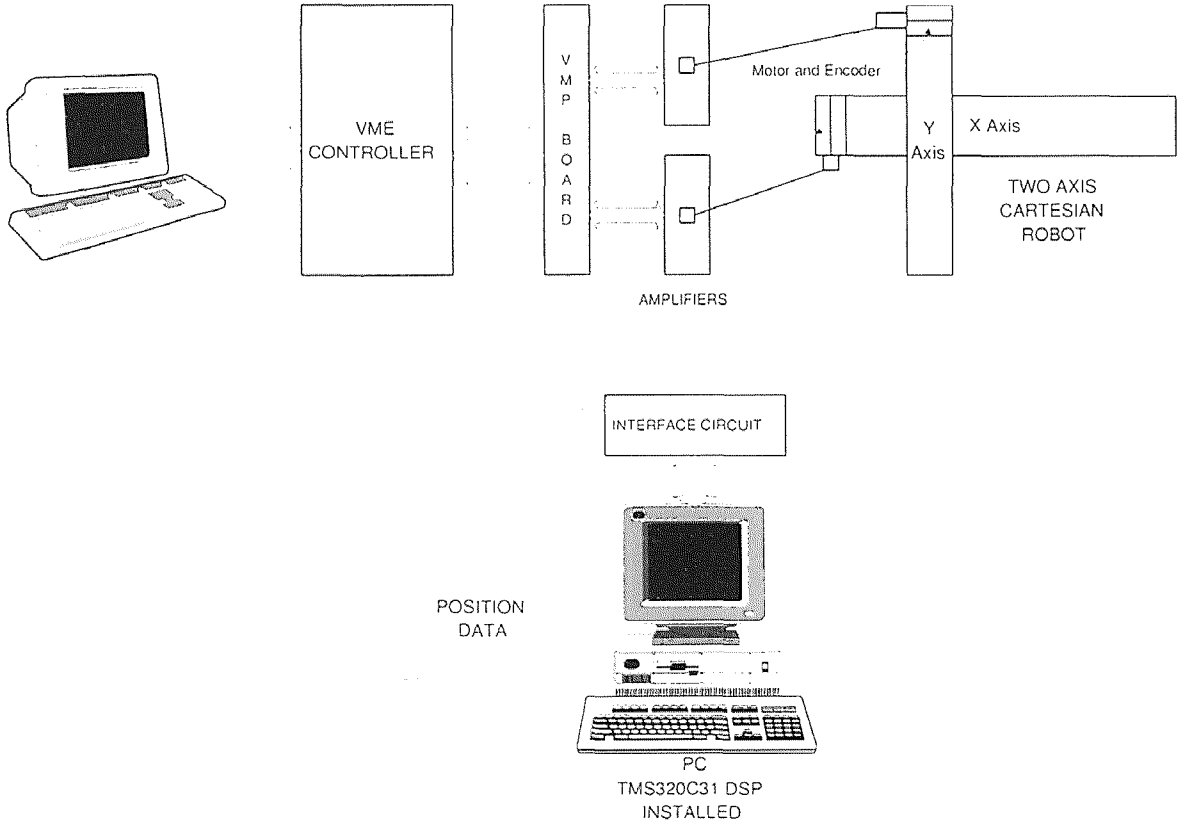

Figure 1.1 System Overview

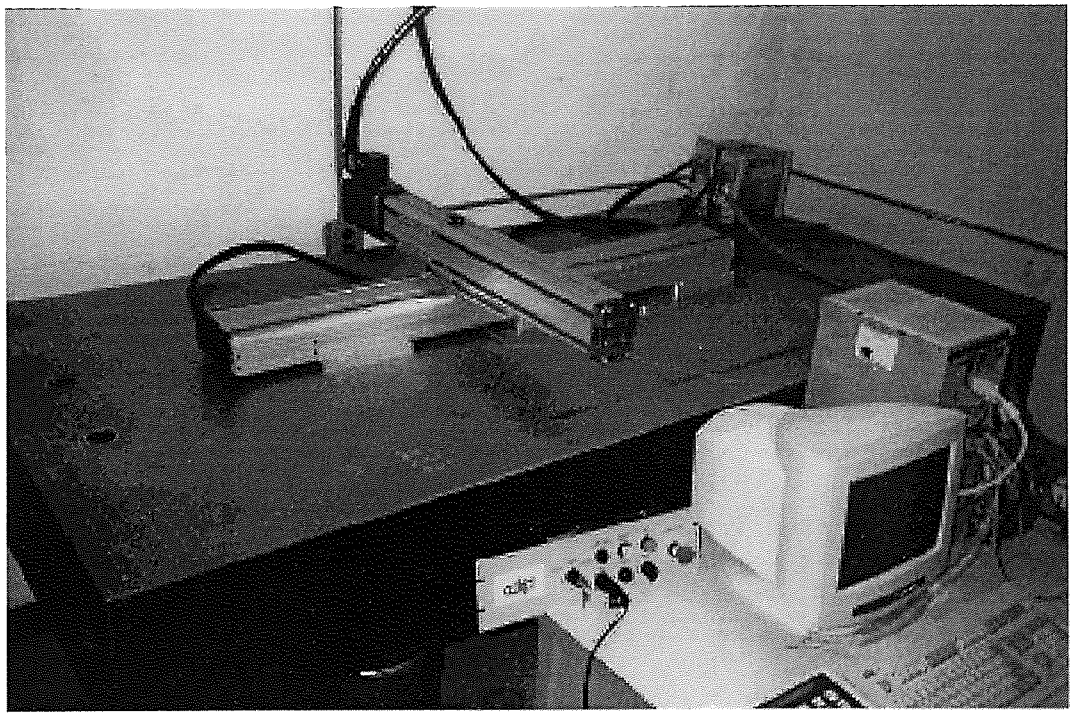

Figure 1.2 View of H and M Modules

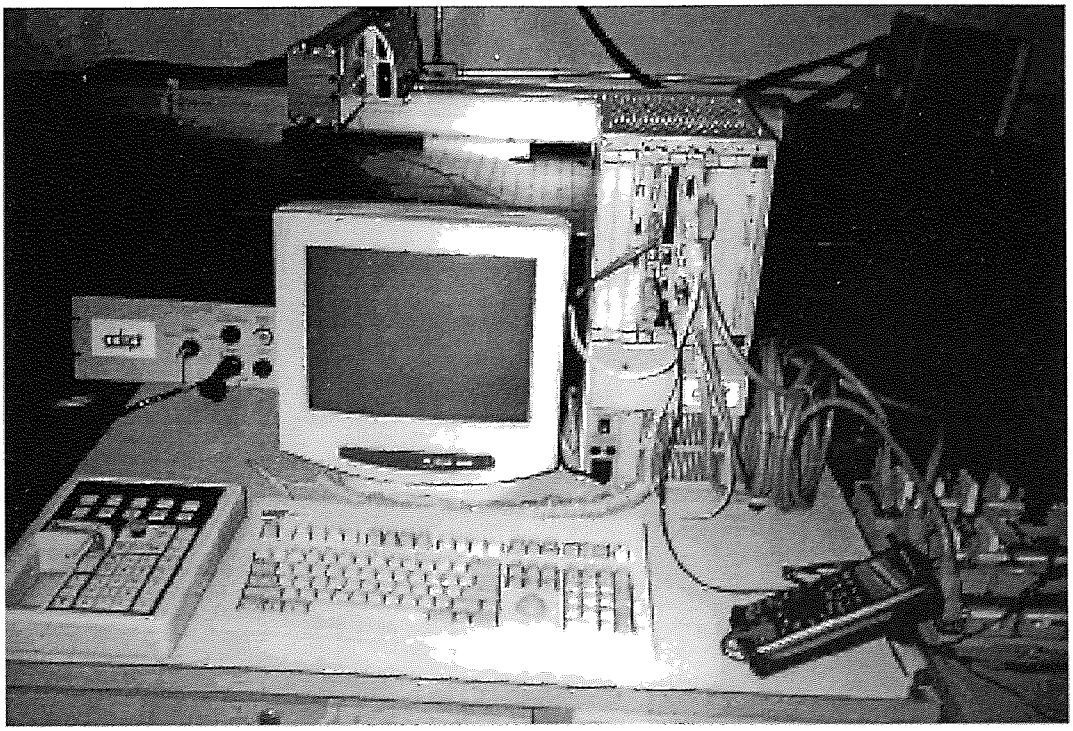

Figure 1.3 View of VME Controller

Figure 1.4 View of PC Part

CHAPTER 2

HARDWARE DESCRIPTION

In this chapter, the experimental hardware is described in the following five sections:

1. H and M linear robot modules [9];
2. EXC controller [10];
3. VME controller and robot interface;
4. Digital Signal Processor (DSP) [11];
5. VME and DSP interface.

2.1 Linear Robot Module

The cartesian robot system consists of one H Module, one M Module, and two amplifiers. Internal construction of the modules and their respective reference directions are shown in Figure (2.1). The robot modules are essentially ball-screw drives. Depending on the mechanical characteristics such as travel and payload ratings, the modules are classified as H, M, and S where the H and M types are used in this work.

2.1.1 H Module

The H module is the largest module with the highest payload and moment capacity. H module consists of a 20 mm pitch ball screw, two 25 mm linear guides and a 300 watt motor without a holding break. The dimensions are 180 mm (7") wide, 90 mm (3.5") in height and 1000 mm (40") length. Refer to Table (2.1) for the performance specifications.

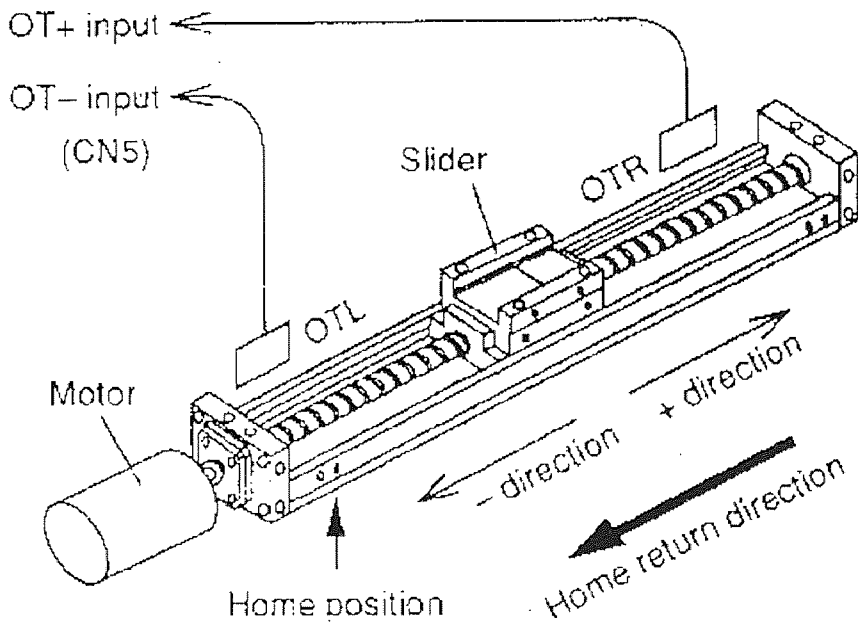

Figure 2.1 Inside view of H and M Modules

2.1.2 M Module

The M module is a mid size module. It consists of a 20 mm pitch ball screw, two 25 mm linear guides and a 300 watt motor without a holding break. The dimensions are 116 mm (4.6") wide, 85 mm (3.4") in height and 550 mm (21.34") length. The standard M modules are supplied with direct-mount motors. Performance specifications of the M modules are given in Table (2.1) whereas the motor and encoder specifications are given in Table (2.2). A more detailed table about motor and encoder is given in Appendix B.

Table 2.1 Specifications of H and M Modules

Module Type	Stroke (mm)	Max. Speed (mm/sec)	Repeatability (± mm/ ± in)	Ball screw Pitch (mm)	Max. Payload (kg/lb)	Motor Count	Rated Thrust Force (N/lb)
H-module	1000	1200	0.01/0.0004	20	60/132	Direct	300/67
M-module	550	1200	0.01/0.0004	20	60/132	Direct	300/67

Table 2.2 Motor and Encoder Specifications

Motor	300W AC servo motor
Position feedback	2000 lines/revolution Maximum frequency: 150 KHz
Maximum Motor Speed	3600 r.p.m.
Power	Single phase 180-240 VAC Required Power: 1170 VA max. (at max. torque)

2.2 EXC Controller [10]

The EXC Controller is a multi-function, multi-axis controller with a built-in dual-axis or triple-axis servo driver. It is capable of executing following set of motion commands:

- Sequential commands: Timer, conditional jump, repetition, etc.
- Move commands: Triple-axis linear interpolation (linear motion between two points in three-dimensional space), dual-axis arc interpolation (start point, transit point and end point), etc.
- Palletizing commands.
- Hand and other subordinate unit control through 32 general purpose inputs and outputs.
- Capacity of 100 programs and 5000 steps maximum.

With the above-shown features, the EXC controller is capable of feedforward compensation, digital filtration and some other control features for general performance motion control. Standard setup and block diagram of the EXC Controller is given in Figure (2.2) and Figure (2.3). In Figure (2.2), user has the option to give manual robot position commands or program controller from "Teaching Box". Figure (2.3) shows EXC controller block diagram. By tuning controller gains and filter parameters required motion characteristics can be obtained.

The EXC controller is available as a compact rack mount type and a stand-alone type. The stand alone type was used in this work.

For each modular axis, a filtered PI controller with feedforward control has been pre-programmed in the EXC for servo control of the modules. The user can further customize a number of parameters such as controller gains and filter frequencies.

2.2.1 Servo Parameter Setting

- **Position loop proportional gain:** Decrease this gain if overshoot, noise and/or vibrations are too great.
- **Velocity loop proportional gain:** Increase this gain if overshoot is too great. Decrease it if noises or vibrations are too great.
- **Velocity loop integration frequency:** Decrease this value if overshoot is too great.

Normally, change the settings of the velocity loop proportional gain, velocity loop integration frequency and position loop proportional gain in this order.

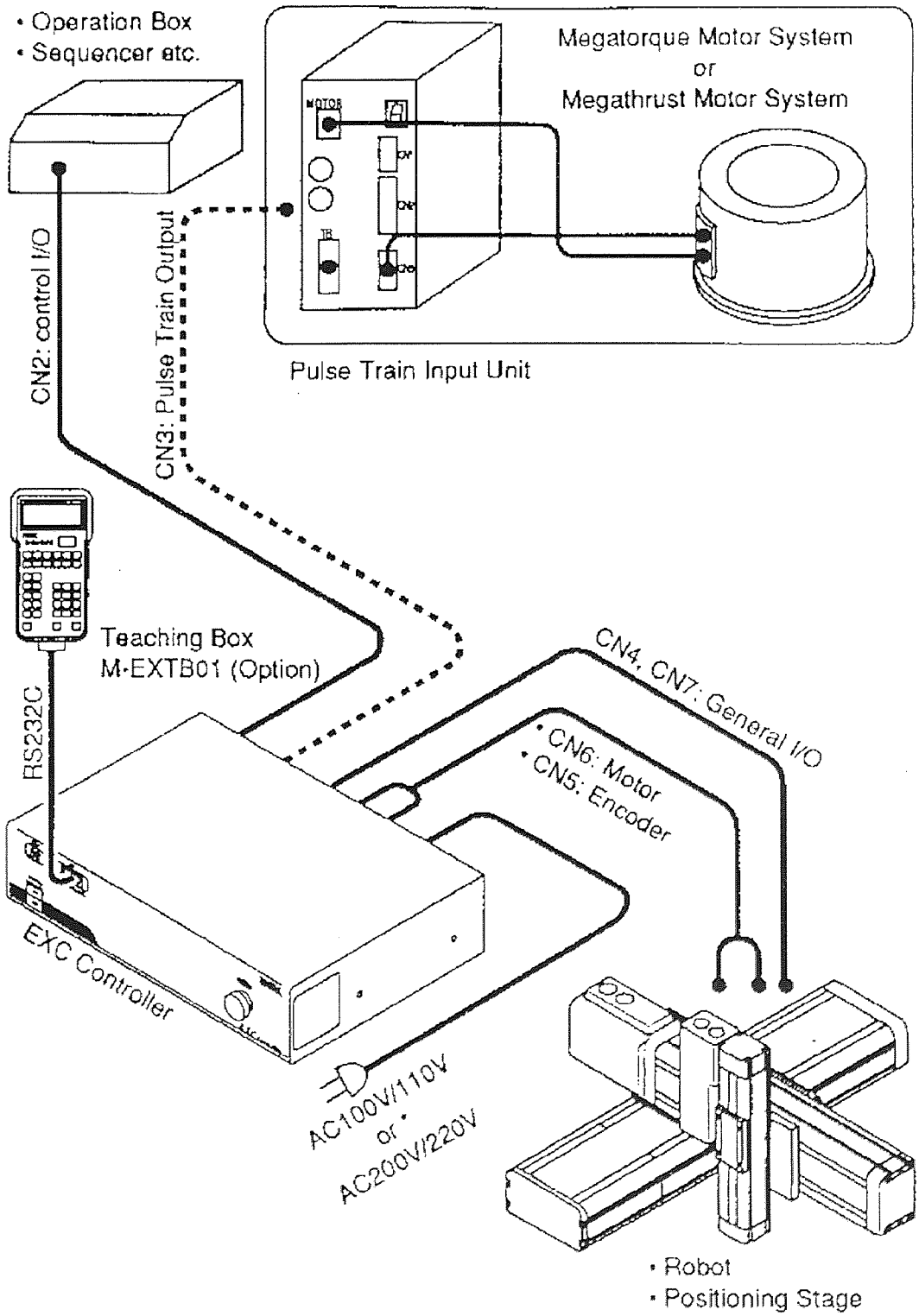

Figure 2.2 General Setup of EXC Controller

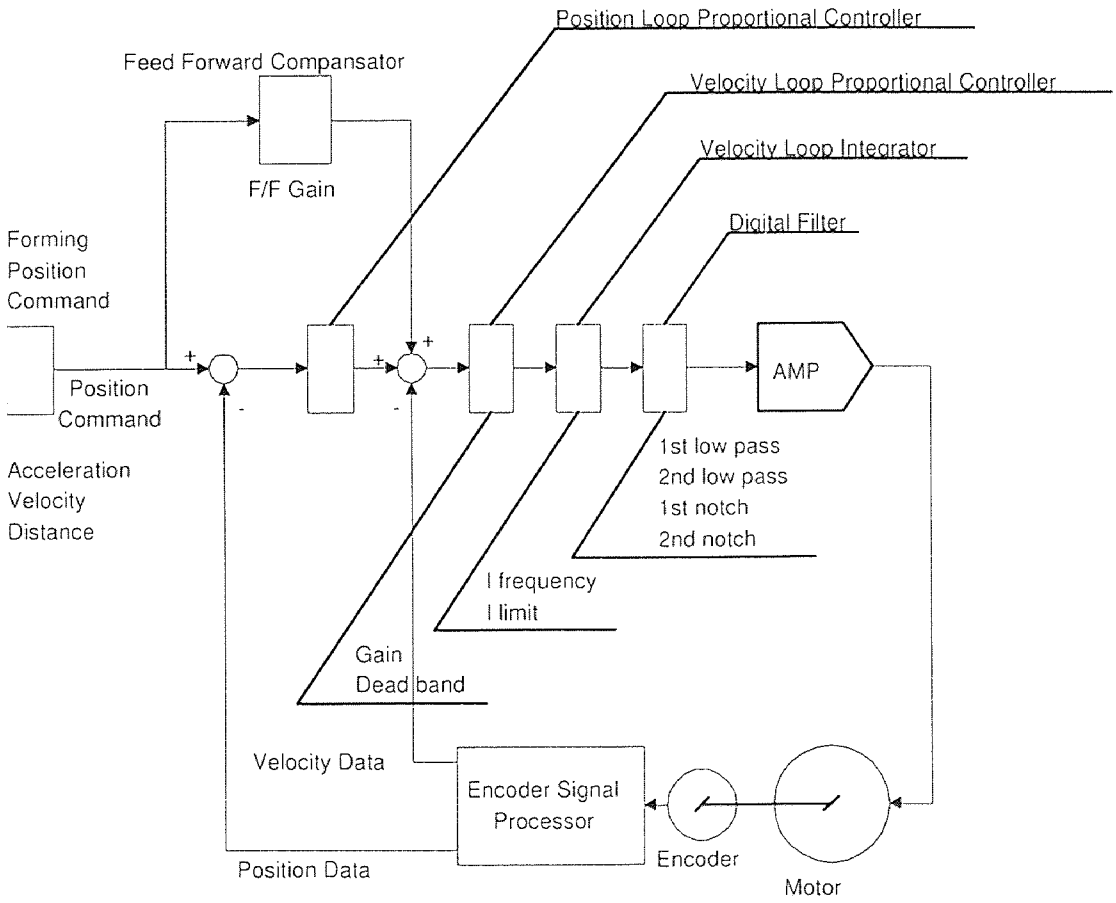

Figure 2.3 EXC Servo Block Diagram

Position Loop

Table (2.3) lists the position loop adjusting parameters and their setting procedures.

Table 2.3 Position Loop Parameters

Item	Description	Units	Setting Range
Gain	Position loop proportional gain. The positioning time is shorter with a greater gain. However, a greater gain causes greater overshoot or vibrations.	-	0.01~31.00
Dead band	The speed command is zeroed with a position error less than a set value in order to reduce minute vibrations due to small errors.	counts	0~4095
F/F Gain	Position loop feedforward compensation gain. Traceability to a command is better with a greater F/F gain. However, a greater gain causes greater overshoot or vibrations.	-	0.00 ~ 1.00

Velocity Loop

Table (2.4) lists the velocity loop adjusting parameters and their setting procedures.

Filter

Table (2.5) lists the filter adjusting parameters and their setting procedures.

The EXC Controller is useful in rapid system configuration and integration. However its lack of programmability severely limits its use in control system development. Therefore it was subsequently replaced by a VME controller which has a more open architecture.

Table 2.4 Velocity Loop Parameter Settings

Item	Description	Units	Setting Range
Gain	Velocity loop proportional gain. The velocity trace is more accurate with a greater gain. However, a greater gain causes more vibrations.	-	0.1~255.0
I frequency	Velocity loop integration frequency. The positioning time is shorter with a greater I frequency. However, a greater I frequency causes more overshoot or hunting.	Hz	0.1~127.0
I limit	The I limit gives an upper limit to pulses collected by integration to improve overshoot.		0.0~100.0
Dead Band	The torque command is zeroed with a position error less than a set value in order to reduce minute vibrations due to small errors.	counts	0~4095

Table 2.5 Filter Parameter Settings

Item	Description	Units	Setting Range
1st low pass	Cut-off frequency of the first low-pass filter	Hz	10~400
2nd low pass	Cut-off frequency of the second low-pass filter	Hz	10~400
1st notch	Center frequency of the first notch filter	Hz	10~400
2nd notch	Center frequency of the second notch filter	Hz	10~400

2.3 VME Controller

2.3.1 Introduction

In this work Adept VME controller is used. It is more flexible and more powerful than EXC controller. Figure (2.4) shows basic command flow of VME controller. The robot position commands in user program are interpreted by V^+ operating system and then motor position commands are formed.

The Adept MV series controllers are based on the VMEbus specification, and the slide-in modules are designed to the 6U VME size. Our system is an Adept MV-8 A-Series controller with the following features:

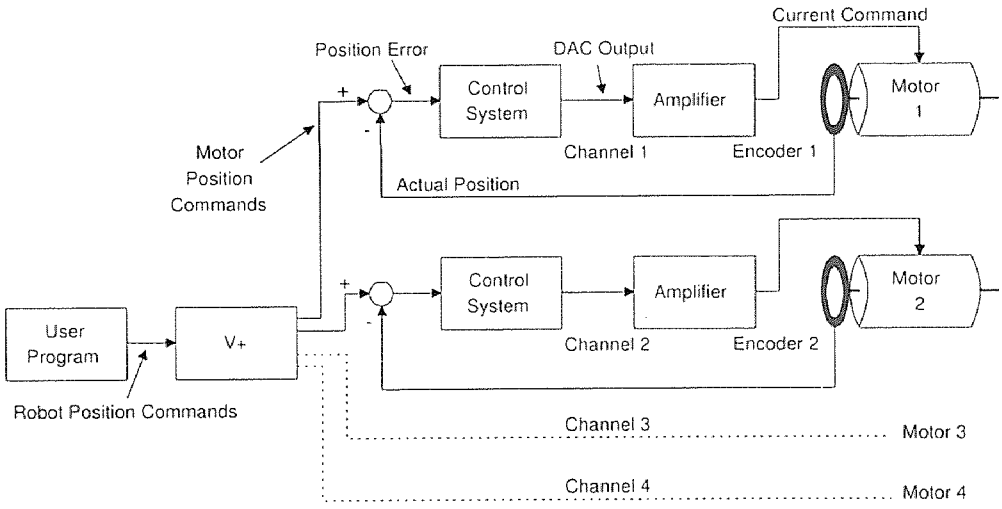

Figure 2.4 VME Controller Command Flow

2.3.2 Adept MV-8 A Series Controller

The Adept MV-8 controller is an 8-slot chassis. It requires two modules:

- The System Processor module (030 or 040);
- The System Input/Output (SIO) module;

The System Processor module takes one slot and System Input/Output module takes two slots. The remaining five slots can be populated with various combinations of optional Adept modules. Our system has one VGB graphics board, one VMI motion interface module and one AIO analog input-output module in additional to the SIO and the 030 System Processor. These electronic modules are now described:

2.3.3 System Processor (030)

The 030 is a single-slot 6U VME module that can serve as the main system processor for an Adept MV controller. The CPU for this module is a Motorola 68EC030 microprocessor running at 40 MHz. It is possible to populate 4 modules. The

module can be configured with 2, 4, or 8 MB of DRAM, and this memory can be upgraded in the field. This module also includes a Motorola 68882 math coprocessor. The 030 module has two serial I/O ports on the front of the module: one is an RS-232 port, and the other is an RS-422/485 port. The 030 can be used either as the main system processor, or as an auxiliary processor in Adept MV controller system. Each Processor Module must have a unique address setting. If additional 030 or 040 modules are installed one of them must be set to main, the others must be set to auxiliary processors. This setting is done by jumpers on the processors modules.

2.3.4 System Input/Output Module (SIO)

The System Input/Output module (SIO) is also required for all Adept MV controller. The SIO is a two-slot 6U VME slave module that handles the basic I/O functions for an Adept MV controller. Communication between the system processor(s) and this board occurs over the VMEbus. The Adept system information is stored in non-volatile RAM (NVRAM) on the SIO module. The SIO module features include:

- 3.5" high-density 1.44 MB floppy drive.
- Internal hard drive (256 MB).
- Digital I/O connector for 20 channels (12 input, 8 output). All channels are opto-isolated. Additional digital I/O can be made available by installing one or more DIO channels.
- Three general-purpose RS-232 serial I/O ports.
- Connector for an optional External Front Panel.

The SIO module controls the system Emergency Stop (E-Stop) circuitry. The system real-time clock/calendar functions are also handled by SIO module.

2.3.5 Adept Graphics Modules (VGB)

The VGB is required for A-Series Adept MV controllers. The VGB is a single-slot 6U VME module that serves as the graphics processor and controls the video output to the color monitor. The VGB has connectors for the monitor, keyboard, and pointing device (mouse, trackball, etc.). The VGB also has a direct Video Bus connection to the VIS module in AdeptVision VME system.

2.3.6 AdeptMotion VME Interface Module (VMI)

Adept Motion VME product consists of a hardware and software package that provides high performance coordinated motion control for industrial automation devices. The hardware consists of an Adept MV controller with one or several Adept-Motion VME modules (VMI) installed.

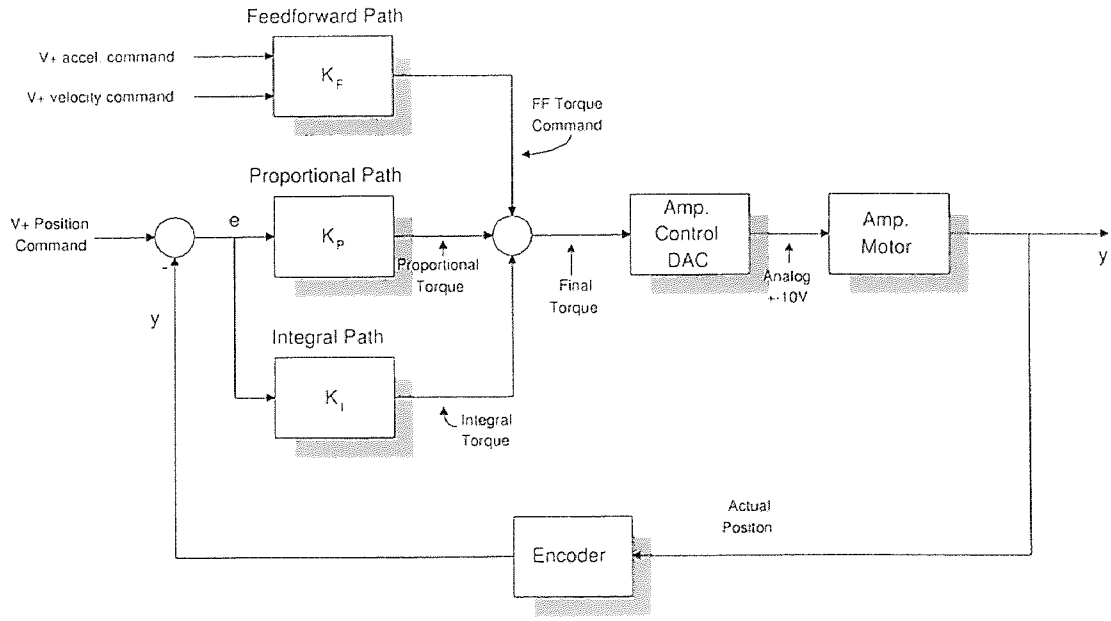

Figure 2.5 VME Controller Block Diagram

The VMI is required to run the AdeptMotion VME product. The VMI module is a single-slot 6U VME module designed to control four axes of motion for use with the Adept Motion VME product. Each VMI module has four servo drive outputs, four incremental encoder inputs, and digital I/O for each machine and amplifier

control. All external device inputs and outputs are opto-isolated. The VMI has four ±10 V analog command outputs that are used to command the motor drive amplifiers. The analog commands are generated by four DAC. The AdeptMotion VME control software runs on the V+ operating system. This product can be integrated into an Adept automation system with additional features such as: vision guidance and inspection, and conveyor tracking. AdeptMotion VME motion control system is compatible with most industry standard motor drives that accept a ±10 V analog input signal for current (torque) or velocity commands. AdeptMotion VME includes a motion interface module that plugs into the backplane of the Adept MV controller, thus permitting the use of Adept's powerful V+ programming language and operating system. The VMI module's connections to user equipment are divided into three groups:

- The encoder;
- The machine;
- The servo;

With a set of cables and Motion Interface mounting panels (VMP), connections between VMI and the application hardware can be made. VMP panels have sockets for user supplied opto isolation modules (12 VDC).

AdeptMotion VMI Interface Panels (VMP)

The three motion interface panels serve as the interface between the VMI Module and the application hardware. VMPM is interface panel for machine cable, VMPS interface panel for servo cable, and VMPE interface panel for encoder cable. The VMP panels also provide mounting sockets for all I/O modules used in conjunction with the dedicated discrete input/output signals. The VMP's also provide detectable barrier type screw terminal strips for all field wiring terminations.

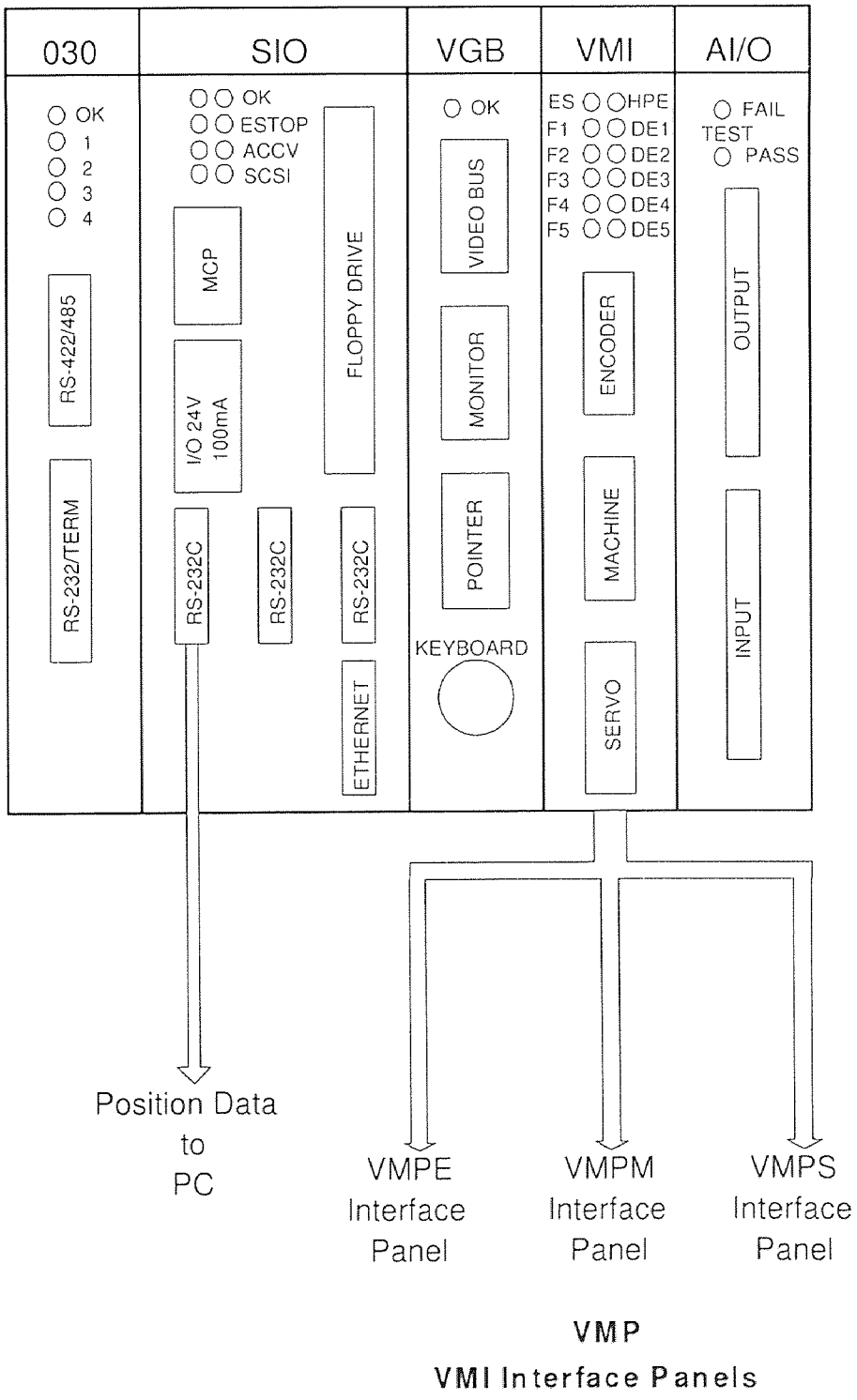

Figure 2.6 VME Controller and AdeptMotion VMI Interface Panels (VMP)

The *VMPS* is used to interface to the Servo Drive amplifiers. It provides one digital input (DF) and one digital output (DE) for each of four channels. It also provides one analog output (CD) for each of four channels. The function of VMPS is to optically isolate the signals, and to perform voltage level shifting, for example, to interface 24V user circuits to the Adept MV controller.

- DE (output) : The DE signals are outputs to the drives which command the drives to enable motor power.
- DF (input) : The DF input is used to indicate a drive fault, such as over-temperature, over-current, etc., and causes all drives to power down via the DE signals.
- CD (output) : The CD outputs provide a command signal to each of the drives. Maximum output is ± 10 V into a 10 KΩ input resistance. These analog outputs are rated at 100 mA (max) per channel.

The *VMPE* is used to interface to the encoders. It supports up to 4 encoder channels, with differential input (A, B, and Index) for each encoder. Each channel is designed to interface directly to the encoders which use industry AB quadrature outputs and an optional zero index channel. The control system measures all position variables in units of encoder counts. For a given encoder there will be 4 encoder counts per slot in encoder, because of the quadrature decoding of the encoder's A and B phases.

All torque values are in units of "DAC output counts". The AdeptMotion VME hardware contains one DAC per motor that accepts a digital input value (in DAC output counts) and produces an analog output voltage proportional to the input. A DAC command may range from -32767 to +32767 (depending of configuration in SPEC utility), corresponding to a -10V to 10V output, respectively. Then a current (or voltage, depending upon the amplifier design) is applied to the motor that is

proportional to the DAC command. The exact amount of current depends on the gain of amplifier. The resulting shaft torque applied to the load will depend upon the design of the motor. Electric motors generate a torque that is roughly proportional to the current by the amount called the "torque constant" of the motor.

2.3.7 Analog Input/Output Module (AIO)

The Analog Input/Output module (AIO) is a VMEbus compatible 6U VME slave I/O module, capable of performing both analog-to-digital and digital-to-analog conversions, with 12-bit resolution.. The AIO module can support 16 different input channels (or 32 single-ended inputs) and 4 output channels. In single ended mode each input is assigned a single input pin, referenced to a common analog ground. In differential mode, each input is assigned a separate + and - terminal. The two modes are mutually exclusive so that the module will accept either all single-ended inputs or all differential inputs but not a combination of both. Selection of differential vs. single ended inputs is totally independent of the voltage range. Choice will depend solely on the electrical characteristics of your analog voltage source/device. Selection is done via jumpers. Three jumper selectable voltage inputs ranges are supported. Within each range, there are four software selectable gain values (total 12 gain settings). Four outputs can be configured as either voltage (5 mA max) or current loop (4-20 mA). Power of analog outputs is provided by on-board DC/DC converter.

2.3.8 Manual Control Pendant (MCP)

The MCP is hand-held control unit that can be added to any Adept MV controller system that includes a VFP(VME Front Panel). The MCP connects to the front of the VFP. The MCP is available in two versions: the operator's pendant and the programmer's pendant. The operator's pendant has a palm-activated "hold-to-run" switch connected to the remote emergency stop circuit; the programmer's pendant

does not have this switch. MCP is often used to manually control a robot or a motion device during system development. The MCP includes function keys and a 2-line by 40-character LCD display that are fully programmable. An Emergency Stop push button switch on the MCP shuts off high power at the Adept MV controller.

2.4 Dalanco Spry Board [11]

During developing software in V^+ environment, we have seen that cycle time resolution was 1 ms and also V^+ operating system does not have a compiler, it has an interpreter, all the programs you write are interpreted line by line. In implementation of input shaping time resolution is important. For our application we needed ~0.1 ms time resolution So for these reasons we decided to use TMS320C31 DSP board with sampling rate of 10 KHz.

The controller implemented in Dalanco Spry's Model 310 data acquisition and signal processing board. Figure (2.7) shows PC to DSP data flow.

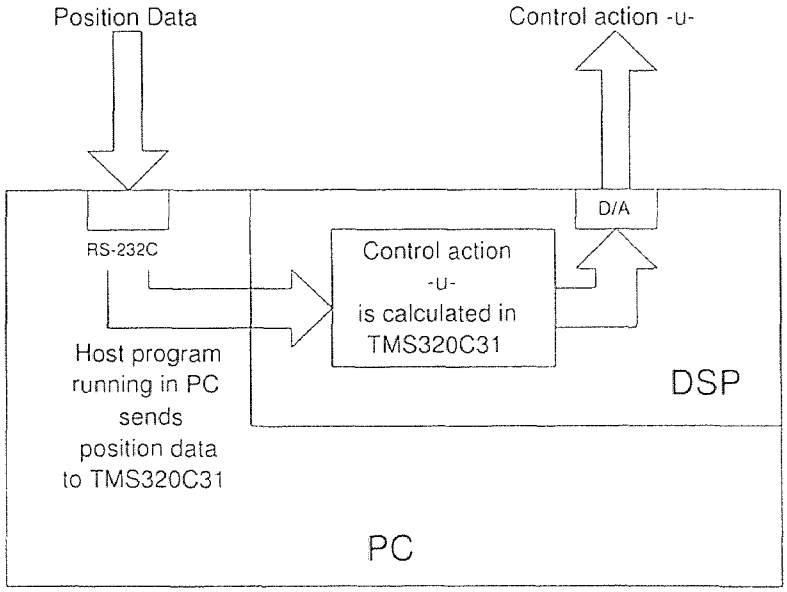

Figure 2.7 PC to DSP Data Flow

The heart of board is TMS320C31, a low cost 32 bit floating point digital signal processor with 60 ns single-cycle instruction execution time and 2Kx32 word

of internal RAM. It can perform single cycle multiplications on 24-bit integer and 32-bit floating point values. Parallel instructions can be used to gain even higher throughput. The arithmetic logic unit (ALU) performs single cycle operations on 32-bit integer, 32-bit logical, and 40-bit floating point data. The TMS320C31 has 28 internal registers. All of these registers can be operated upon by the multiplier and ALU, and can be used as general purpose registers. The total memory space of the TMS320C31 is 16M (million) 32-bit words. Program, data, and I/O space are contained within this 16M word address space. It has 2 general purpose 32-bit timer/event counter and 2 bi-directional serial ports. Each serial port can be configured to transfer 8, 16, 24, or 32 bits of data per word. The on-chip Direct Memory Access (DMA) controller can read from or write to any location in the memory map without interfering with the operation of the CPU [12].

The base IO address of the Model 310 in the host PC is selectable on 8 byte boundaries. Our board's base IO address is 300H. The Model 310 board is also equipped with 512K words of static memory. Model 310 holds a single bank of 32 bit wide memory. User can select one of the PC interrupts INT10, 11, 12, 15. And also the host PC can send an INT 0 interrupt to C31. It has two internal clocks, TCLK0 and TCLK1. TCLK1 is not used by the Model 310A, while TCLK0 is used to trigger conversion on the A/D board. The Model 310 has 4 ADC and 2 DAC channels. The TMS320C31 controls the following Model 310 parameters by writing to a latch mapped at memory location 0FFFFFFh:

- The next A/D input channel to be sampled. (Chan 0-3)
- The gain setting of the Programmable Gain Amplifier. (1-8)

In terms of software tools, there are two board-specific programs: the D300 debugger and the A3xx assembler assembler. Furthermore, a Texas Instrument C30/C40 C-compiler/assembler/link is also available [13, 14].

2.4.1 A/D Converter

ADC starts conversion by bringing the convert pin low. This strobe is generated by the output of the onchip clock TCLK0. And then the data is read "if ready" at the Serial Port Data Receive Register. TCLK0 can be programmed in two ways. First it can be programmed as an IO pin; second as a clock signal. In the second way desired sampling rate can be given.

Type	Maxim 121 or equivalent (14bit)
Maximum Range	150 KHz (310A), 300 KHz (310B)
Input Voltage Range	5 Volts
Data Format	16 bit, two's complement binary code placed in 14 MSB

2.4.2 D/A Converter

The D/A Converter is connected to the three transmit lines of the serial port. The DSP polls the Serial Port Global Control Register to determine if the serial port transmit buffer is ready for a new data. Upon notification of readiness, the new data word may be written to the Serial Port Data Transmit Register. The D/A output will be updated with the value in the lower 16 bits of the 32 bit data word after reception of the data word.

Type	1 AD7243 (310A), 2 AD7243 (310B)
Range	250 KHz Single / 140 KHz dual channel
Input Voltage Range	5 Volts
Data Format	16 bit, two's complement binary code placed in 12 LSB.

2.5 Interface Circuitry

In interface circuitry we need a summer to match the ± 5 V output of DSP board to the ± 10 V output of VME. It will also do buffering. Figure (2.8) is the interface circuit between C31 and VME controller.

$$V_O \frac{R_5}{R_4 + R_5} = V_X$$

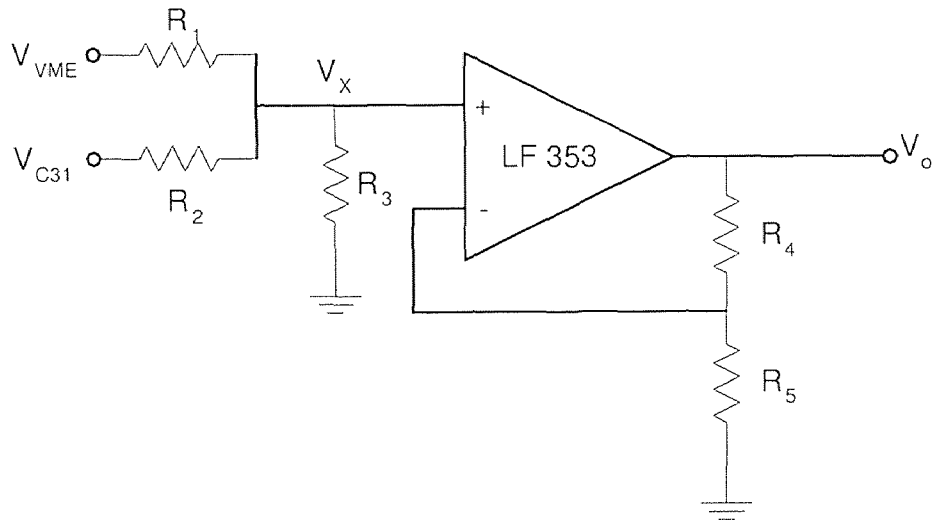

Figure 2.8 Interface Circuit

Let

$$V_{X1} = V_X \quad \text{when} \quad (V_{C31} = 0)$$

$$V_{X2} = V_X \quad \text{when} \quad (V_{VME} = 0)$$

then

$$V_{X1} = V_{VME}\left(\frac{R_2//R_3}{R_1 + R_2//R_3}\right)$$

$$V_{X2} = V_{C31}\left(\frac{R_1//R_3}{R_2 + R_1//R_3}\right)$$

If we let $R_1 = 2R$ and $R_2 = R_3 = R$ then

$$V_{X1} = \frac{1}{5}V_{VME}$$

$$V_{X2} = \frac{2}{5}V_{C31}$$

$$V_X = V_{X1} + V_{X2}$$

$$V_X = \left(\frac{1}{5}V_{VME} + \frac{2}{5}V_{C31}\right)V_O$$

if we choose $R_4 = R$ and $R_5 = 4R$ then we get

$$V_O = V_{VME} + 2V_{C31}$$

CHAPTER 3

INPUT SHAPING ALGORITHM

Input shaping is an open loop compensator which shapes the actuator input such that vibrational motion is eliminated after the input is ended. The method is based on the notion that superimposed impulse responses can exactly cancel one after the last impulse. Implementing input shaping control involves convolving a sequence of impulses with the reference inputs. The critical parameters of the scheme are the amplitudes and application times of the impulses. Impulse amplitudes are a function of modal damping while application times are a function of both modal damping and frequency. An important assumption of input shaping control is that the frequencies and dampings of the modes of vibration are known a priori. Figure (3.1) illustrates a typical input shaping scheme where the closed loop eigenvalues affect the shaper design. Note that numerator dynamics of the system (zeros) are not considered in designing an impulse sequence [5, 10].

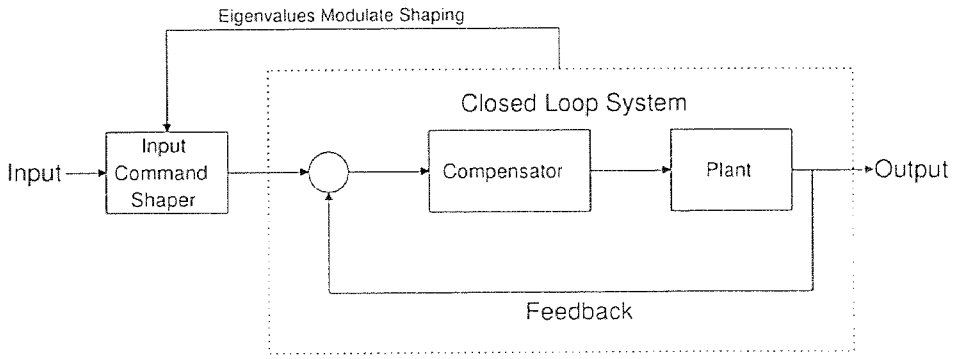

Figure 3.1 General System Configuration for Input Shaping System

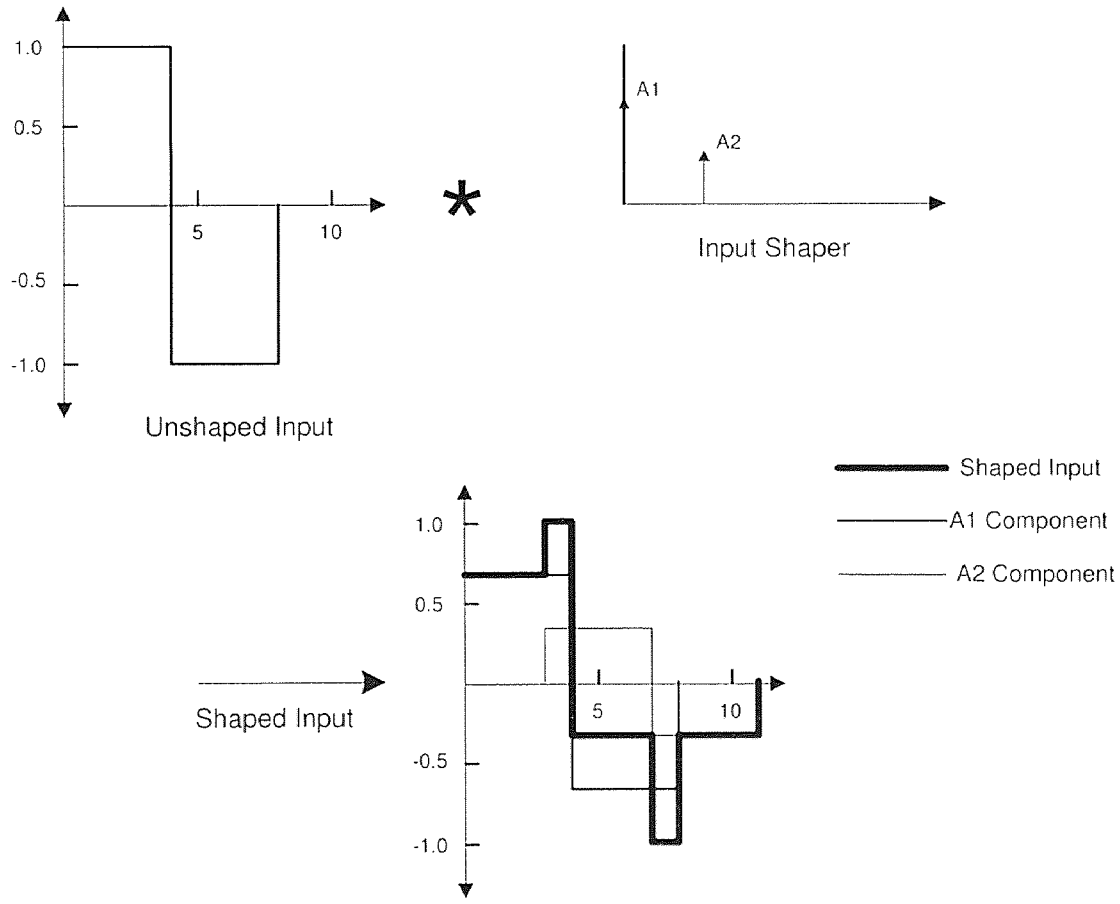

Figure 3.2 Illustration of Input Shaping Algorithm

Figure (3.2) is an illustration of input shaping algorithm. The amplitudes A_1, A_2 , and the application times of these impulses must be determined. After these parameters are found, the unshaped input is convolved with these impulses and result of this convolution gives the shaped input. This input is the control input to the closed loop system.

3.1 Derivation and Details of Input Shaping Method [4, 5]

An uncoupled, linear, vibratory system of any order can be specified as a cascaded set of second order poles with the decaying sinusoidal response:

$$y(t) = \left[A \frac{\omega_0}{\sqrt{1.0 - \zeta^2}} e^{-\zeta \omega_0 (t - t_0)} \right] \sin\left(\omega_0 \sqrt{1.0 - \zeta^2} (t - t_0) \right) \tag{3.1}$$

where A is the amplitude of the impulse, ω_0 is the undamped natural frequency of the plant, ζ is the damping ratio of the plant, t is time, t_0 is the time of the impulse input. The impulse is usually a torque or velocity command to an actuator. Equation (3.1) specifies the acceleration or velocity response, $y(t)$, at some point of interest in the system. First the two impulse case will be considered. The end or the duration of the input is the time of the second (last) impulse.

$$B_1 \sin(\alpha t + \phi_1) + B_2 \sin(\alpha t + \phi_2) = A_{amp} \sin(\alpha t + \psi) \tag{3.2}$$

where

$$A_{amp} = \sqrt{(B_1 \cos\phi_1 + B_2 \cos\phi_2)^2 + (B_1 \sin\phi_1 + B_2 \sin\phi_2)^2}$$

$$\psi = \arctan\left(\frac{B_1 \cos\phi_1 + B_2 \cos\phi_2}{B_1 \sin\phi_1 + B_2 \sin\phi_2} \right)$$

The amplitude of the vibration for a multi-impulse input is given by:

$$A_{amp} = \sqrt{\left(\sum_{j=1}^{N} B_j \cos\phi_j \right)^2 + \left(\sum_{j=1}^{N} B_j \sin\phi_j \right)^2} \tag{3.3}$$

$$\phi_j = \omega \sqrt{(1 - \zeta^2)} t_j$$

The B_j are the coefficients of the sine term in (3.1) for each of the N impulse inputs, the t_j are the times at which the impulses occur, and ω is the natural frequency. Elimination of vibration after the input has ended requires that the impression for

A_{amp} equal to zero at the time at which the input ends, t_N. This is true if both squared terms in (3.3) are independently zero, yielding:

$$B_1 \cos\phi_1 + B_2 \cos\phi_2 + \cdots + B_N \cos\phi_N = 0 \tag{3.4}$$

$$B_1 \sin\phi_1 + B_2 \sin\phi_2 + \cdots + B_N \sin\phi_N = 0 \tag{3.5}$$

with

$$B_j = \frac{A_j \omega}{\sqrt{(1-\zeta^2)}} e^{-\zeta\omega(t_N - t_j)}$$

where A_j is the amplitude of the jth impulse, t_j is the time of the jth impulse, and t_N is the time at which the sequence ends. Equations (3.4) and (3.5) can be simplified further, yielding:

$$\sum_{j=1}^{N} A_j e^{-\zeta\omega(t_N - t_j)} \sin(t_j \omega\sqrt{1-\zeta^2}) = 0 \tag{3.6}$$

$$\sum_{j=1}^{N} A_j e^{-\zeta\omega(t_N - t_j)} \cos(t_j \omega\sqrt{1-\zeta^2}) = 0 \tag{3.7}$$

In order to increase robustness of impulse shaping sequence to modeling errors in the natural frequency ω_0 and the damping ratio ζ, the partial derivatives of ZV constraints with respect ω_0 and ζ are also constrained to zero. This yields:

$$\sum_{j=1}^{N} A_j t_j e^{-\zeta\omega(t_N - t_j)} \sin(t_j \omega\sqrt{1-\zeta^2}) = 0 \tag{3.8}$$

$$\sum_{j=1}^{N} A_j t_j e^{-\zeta\omega(t_N - t_j)} \cos(t_j \omega\sqrt{1-\zeta^2}) = 0 \tag{3.9}$$

Therefore two more unknowns must be added (A_3 and t_3) by increasing the input from two to three inputs. The general formula concerning this process is:

$$\sum_{j=1}^{N} A_j (t_j)^q e^{-\zeta\omega(t_N - t_j)} \sin(t_j \omega\sqrt{1-\zeta^2}) = 0 \tag{3.10}$$

$$\sum_{j=1}^{N} A_j (t_j)^q e^{-\zeta\omega(t_N - t_j)} \cos(t_j \omega\sqrt{1-\zeta^2}) = 0 \tag{3.11}$$

After this process input is robust for system frequency variations of $\approx \pm 20$ percent. For the four impulse case input is robust for system frequency variations of $\approx$ -30 percent +40 percent.

The aim of input shaping is essentially, to add zeros to the system function at the exact locations of the system modes (poles). Let's assume that the system consists of a pair of complex poles at $s = -\sigma \pm j\omega$. If an impulse sequence $u(t) = \alpha_0\delta(t) + \alpha_1\delta(t - t_1)$ is input to the system then laplace transform of the output will be

$$\begin{aligned} Y(s) &= \alpha_0 H(s) + \alpha_1 e^{-st_1} H(s) \\ &= H(s)(\alpha_0 + \alpha_1 e^{-st_1}), \end{aligned} \tag{3.12}$$

If we let $\frac{\alpha_0}{\alpha_1} + e^{-st_1} = 0$, and set the location of complex zeros contributed by shaped input, equal to complex system poles. This results,

$$\frac{\alpha_0}{\alpha_1} = -e^{\sigma t_1} e^{\pm j\omega t_1} \tag{3.13}$$

where t_1 is the application time of the second impulse and is got from $e^{\pm j\omega t_1} = -1$, or $t_1 = \frac{\pi}{\omega}$. Thus

$$t_1 = \frac{\pi}{\omega_n \sqrt{1 - \zeta^2}}$$

now we got

$$\frac{\alpha_0}{\alpha_1} = e^{\sigma t_1}$$

if we put $t_1 = \frac{\pi}{\omega_n \sqrt{1-\zeta^2}}$ and $\sigma = \omega_n \zeta$ in to this equation we get:

$$\frac{\alpha_0}{\alpha_1} = e^{\frac{\zeta\pi}{\sqrt{1-\zeta^2}}} \tag{3.14}$$

if we let $\alpha_0 + \alpha_1 = 1$ to ensure that the shaped input voltage has the same amplitude as the reference amplitude, we get

$$\alpha_0 = \frac{1}{1 + e^{\frac{-\zeta\pi}{\sqrt{1-\zeta^2}}}} \tag{3.15}$$

$$\alpha_1 = \frac{e^{\frac{-\zeta\pi}{\sqrt{1-\zeta^2}}}}{1+e^{\frac{\zeta\pi}{\sqrt{1-\zeta^2}}}} \tag{3.16}$$

For ZVD case two pairs of complex zeros, rather than one, are added to the system function at the exact location of the pair of the complex poles. Robustness to the system uncertainty is improved since there are two zeros in the vicinity of each complex pole rather than one. Full detail can be found in [6]. Figures (3.3), (3.4), and (3.5) show the parameters for Zero Vibration (ZV), Zero Vibration & Derivative (ZVD), and Extra Insensitive (EI) shapers.

$$K = e^{-\frac{\zeta\pi}{\sqrt{1-\zeta^2}}} \tag{3.17}$$

$$\Delta T = \frac{\pi}{\omega_0\sqrt{1-\zeta^2}} \tag{3.18}$$

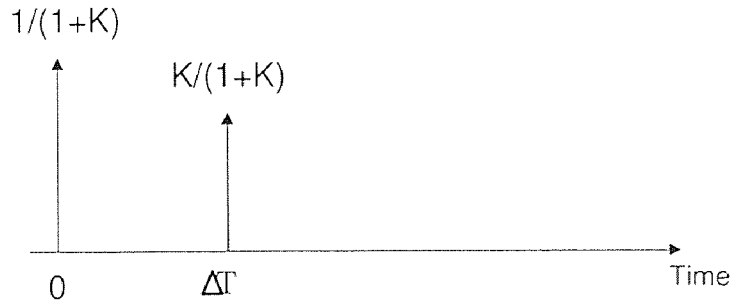

Figure 3.3 ZV Shaper

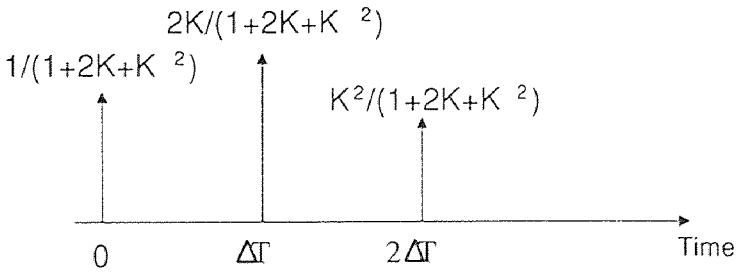

Figure 3.4 ZVD Shaper

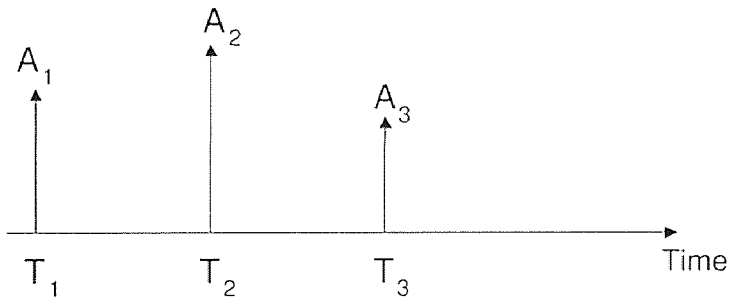

Figure 3.5 EI Shaper

Curve fit formulas for EI shapers are:

$$
\begin{aligned}
A_1 &= 0.2497 + 0.2496V + 0.8001\zeta + 1.233V\zeta + \\
&\quad 0.4960\zeta^2 + 3.173V\zeta^2 \\
A_2 &= 1 - (A_1 + A_2) \\
A_3 &= 0.2515 + 0.2147V - 0.8325\zeta + 1.415V\zeta + \\
&\quad 0.8518\zeta^2 - 4.901V\zeta^2 \\
T_1 &= 0 \\
T_2 &= (0.5000 + 0.4616V\zeta + 4.262V\zeta^2 + 1.756V\zeta^3 + \\
&\quad 8.578V^2\zeta - 108.6V^2\zeta^2 + 337.0V^2\zeta^3)Td \\
T_3 &= Td
\end{aligned}
$$

(good for $0 \leq \zeta \leq 0.3$ and $0 \leq V \leq 0.15$)

CHAPTER 4

CONTROL SYSTEM DESIGN

In this chapter, a design model for the modules will be derived from experimental data. This is followed by a discussion of various input shaping designs: Zero Vibration (ZV), Zero Vibration & Derivative (ZVD), and Extra Insensitive (EI).

4.1 Closed Loop System

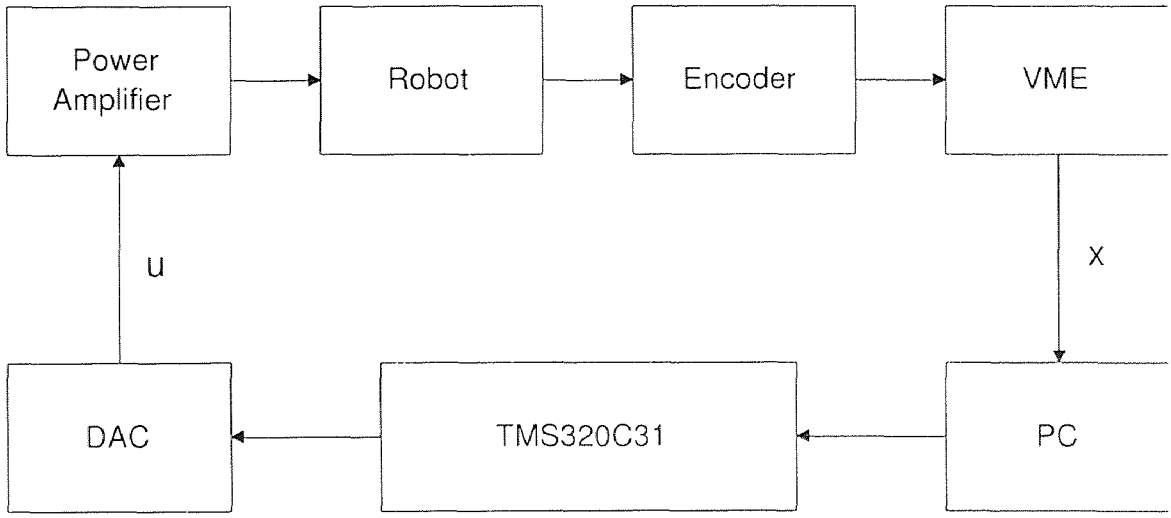

Figure 4.1 Closed Loop Diagram of System

A schematics of the closed loop system is shown in Figure (4.1) where the module's position value is generated by encoder. A V$^+$ program residing in the VME system sends this value to the PC via its serial port. The PC then writes this position data to a dual ported memory location (1000H) which is "simultaneously" accessible by the TMS320C31 DSP to calculate the proper control action. Finally, the control signal is sent to robot via the power amplifier.

4.2 Plant Transfer Function

In this section the process for determining plant transfer function will be explained. In our case, the plant is the linear module whose transfer function is in the form of:

$$G(s) = \frac{K_0}{s(s\tau + 1)}$$

In order to find the transfer function, proportional feedback is applied to the

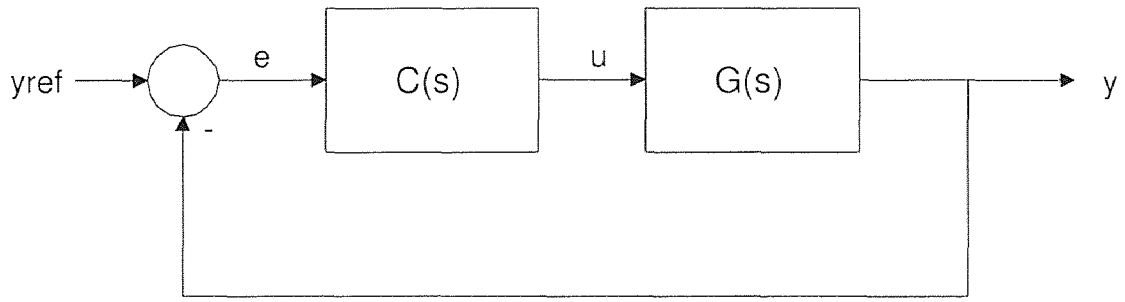

Figure 4.2 Plant with Proportional Controller

plant as shown in Figure (4.2). Since $C(s) = K_p$ closed loop transfer function after rearranging the coefficients becomes:

$$G_c(s) = \frac{\frac{K_0 K_p}{\tau}}{s^2 + \frac{s}{\tau} + \frac{K_0 K_p}{\tau}} \tag{4.1}$$

For $K_p = 6$, a set of closed loop responses are plotted in Figure (4.3). Furthermore, data collected from an accelerometer with sampling rate of 10 KHz is first filtered and then plotted in Figure (4.4).

From these responses, T_d (period of damped vibration), and M_p (overshoot) can be readily determined according to the following formulas:

$$\begin{aligned} \omega_d &= 2\pi / T_d \\ \omega_0 &= \omega_d \sqrt{1 - \zeta^2} \\ M_n &= e^{\frac{-\pi\zeta}{\sqrt{1-\zeta^2}}} \end{aligned}$$

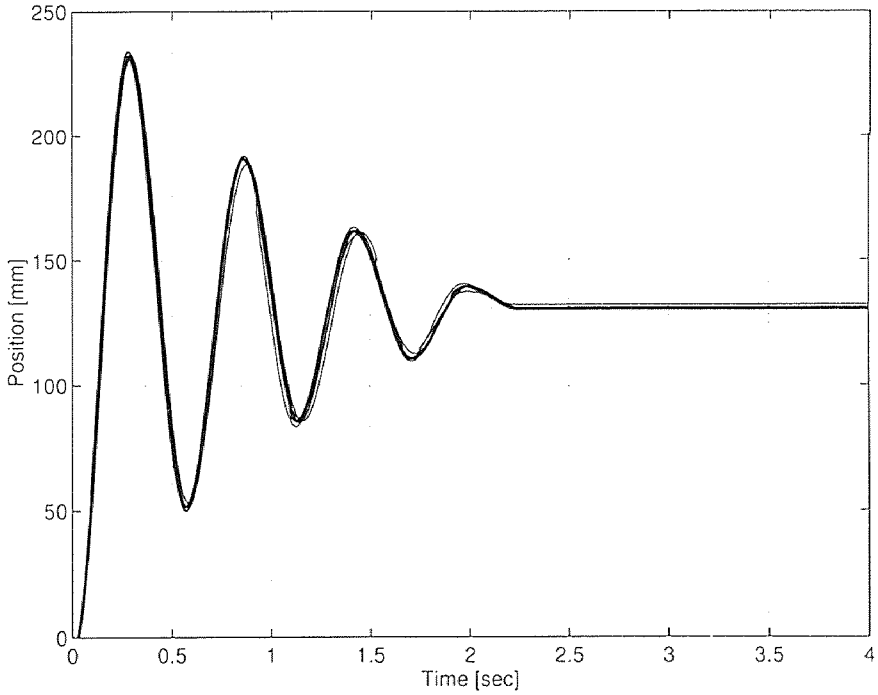

Figure 4.3 10 Sequential Step Response of System for K_p=6

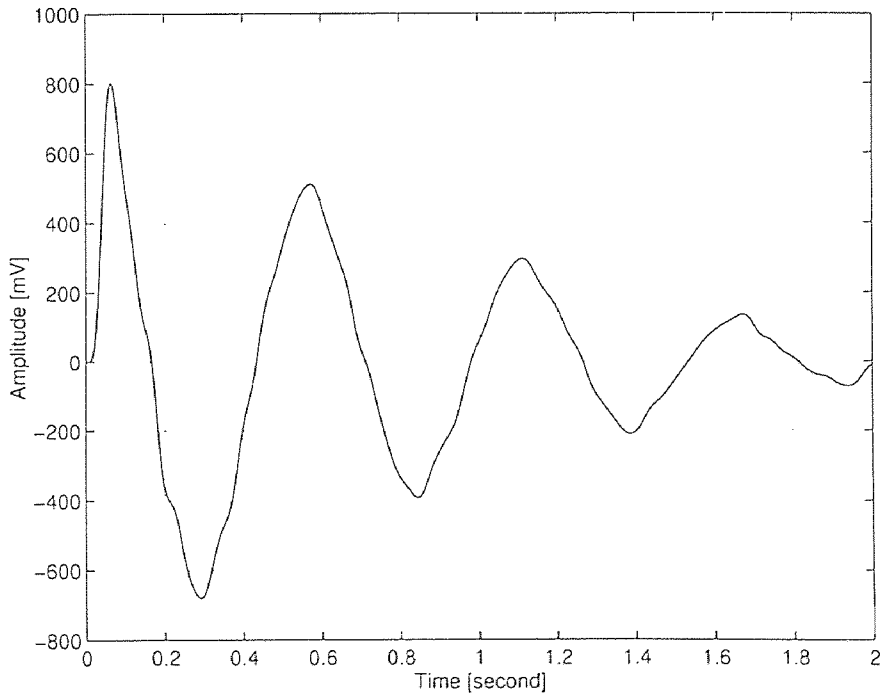

Figure 4.4 Filtered Acceleration Data (f_c=15 Hz)

where ω_0 and ζ are respectively, the natural frequency and damping factor. Moreover, since:

$$\begin{aligned} 2\zeta\omega_0 &= \frac{1}{\tau} \\ \omega_0^2 &= \frac{K_0 K_p}{\tau} \end{aligned}$$

from these formulas, it is determined that $\tau = 0.5010$ and $K_0 = 10.9811$. Therefore, the open loop transfer function of the plant is given by:

$$G(s) = \frac{21.92}{s(s + 19.96)} \tag{4.2}$$

4.3 Implementing Input Shaping

Since the switch time of the shaper depends directly on the plant's (open loop or closed loop) natural frequency, it is advantageous to apply pre-conditioning feedback to increase the natural frequency.

Before implementing input shaping, a PD control is first applied to the plant to increase the natural frequency. For all of the tests, movement will be from $x = 154000$ encoder counts to $x = 144000$ encoder counts or equivalently from $x = 385$ mm to $x = 360$ mm.

The PD control was tuned online to optimize the closed loop response so that a suitable balance between natural frequency and damping factor is obtained. It was observed that with $K_p = 40$ and $K_d = 100$, the plant characteristics are most suited for input shaping. Step response of this pre-conditioned closed loop system plotted in Figure (4.5) from which it is determined that:

$$\begin{aligned} \zeta &= 0.1315 \\ T_d &= 0.2050 sec \end{aligned}$$

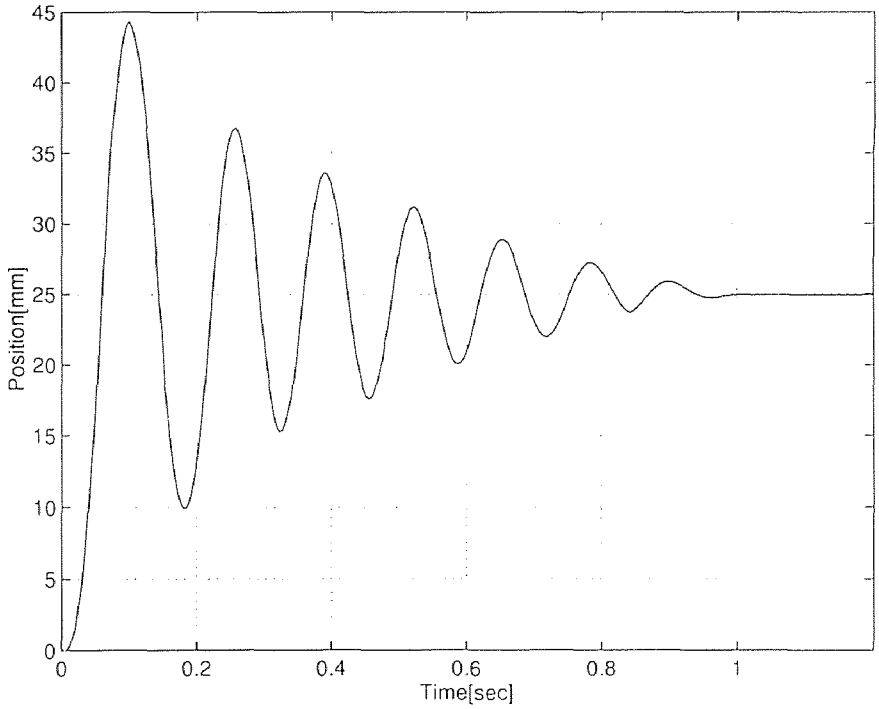

Figure 4.5 Step Response of Closed Loop System With PD Control

4.3.1 Implementing ZV Shaper

First, a ZV shaper will be applied to position the linear robot. The parameters of a ZV shaper are given by equations (3.17) and (3.18) which are repeated below:

$$K = e^{-\frac{\pi\zeta}{\sqrt{1-\zeta^2}}} \qquad \text{and} \qquad \Delta T = \frac{T_d}{2}$$

$$\begin{aligned} A_1 &= \frac{1}{1+K} \\ &= 0.6027 \\ A_2 &= \frac{K}{1+K} \\ &= 1 - A_1 \\ &= 0.3973 \\ \Delta T &= \frac{T_d}{2} \\ &= 0.1025 sec \end{aligned}$$

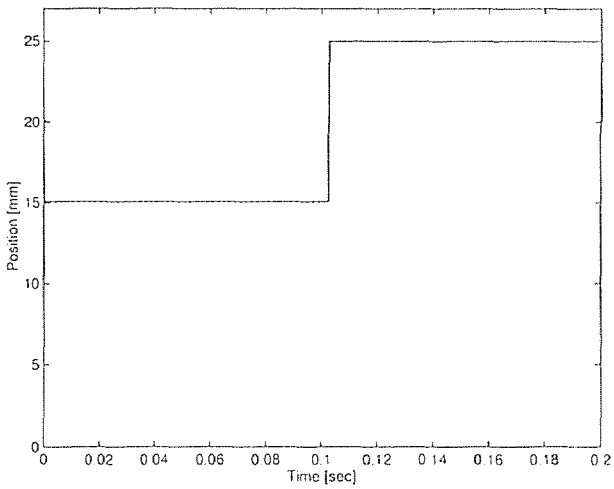

Figure 4.6 ZV Shaper After Convolution

Figure (4.6) shows ZV shaper output. This will be the input to the plant instead of one step amplitude of 25 mm. From this figure the switch time is 0.1025 sec. After switch time output of shaper will jump from A_1 * 25 mm to 25 mm.

4.3.2 Implementing ZVD Shaper

For ZVD shaper coefficients are:

$$\begin{aligned} A_1 &= \frac{1}{(1+K)^2} \\ &= 0.3633 \\ A_2 &= \frac{2K}{(1+K)^2} \\ &= 0.4789 \\ A_3 &= \frac{K^2}{(1+K)^2} \\ &= 1-(A_1+A_2) \\ &= 0.1578 \\ \Delta T &= \frac{T_d}{2} \\ &= 0.1025sec \end{aligned}$$

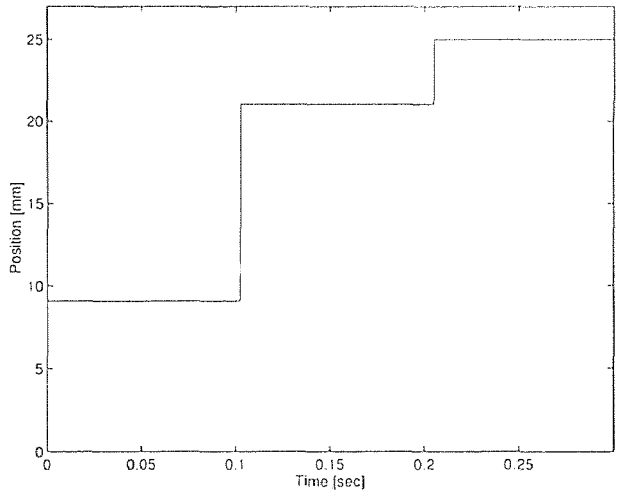

Figure 4.7 ZVD Shaper After Convolution

Figure (4.7) shows ZVD shaper output. This will be the input to the plant instead of one step amplitude of 25 mm. ZVD shaper has 2 switch times. First is at 0.1025 sec and second at 0.2050 sec.

4.3.3 Implementing EI Shaper

For $V = 0.05 \quad \zeta = 0.1315 \quad$ and $\quad T_d = 0.2050$ sec ,

$$A_1 = 0.3868$$

$$A_2 = 0.4406$$

$$A_3 = 0.1726$$

$$T_1 = 0 \text{ sec}$$

$$T_2 = 0.1039 \text{ sec}$$

$$T_3 = 0.2050 \text{ sec}$$

Figure (4.8) shows EI shaper output. This will be the input to the plant instead of one step amplitude of 25 mm. Like ZVD shaper, EI shaper has 2 switch times. First is at 0.1039 sec and the second at 0.2050 sec.

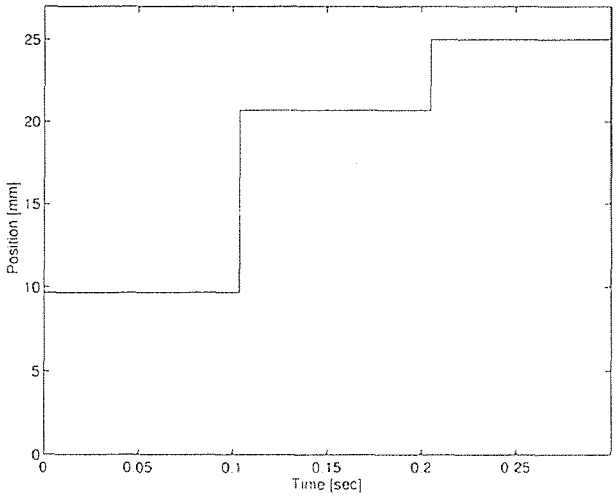

Figure 4.8 EI Shaper After Convolution

CHAPTER 5

RESULTS

In this chapter, the results of various input shaping designs will be given. This is followed by suggestions for improvement of system performance.

5.1 Results of The ZV Shaper

Figures (5.1), (5.2), (5.3), (5.4) and (5.5) show the results of ZV shaper under 5 consecutive trials. From these results, it is observed that the vibration of the linear module has been significantly attenuated. In fact, the amount of overshoot beyond the 25 mm reference level has been reduced from 80 % to about 2 % . The settling times, for all cases, are within 0.25 sec. Although all the trials are satisfactory, the lack of robustness of the ZV design is evident in the significant difference of transient responses.

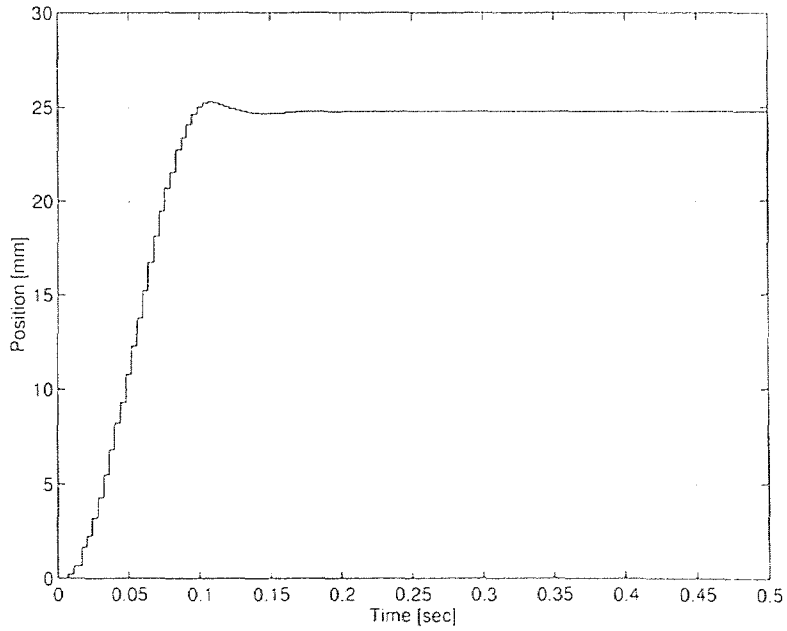

Figure 5.1 Result of ZV Shaper: Experiment #1

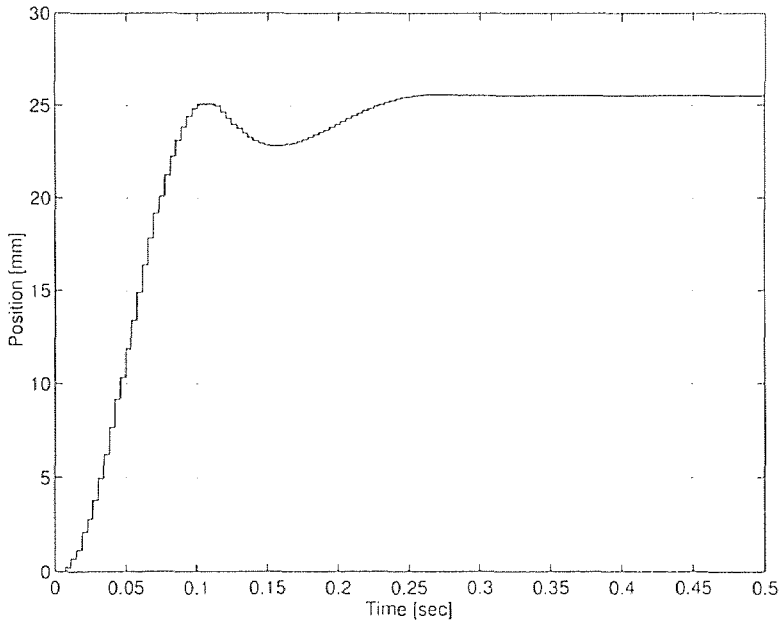

Figure 5.2 Result of ZV Shaper: Experiment #2

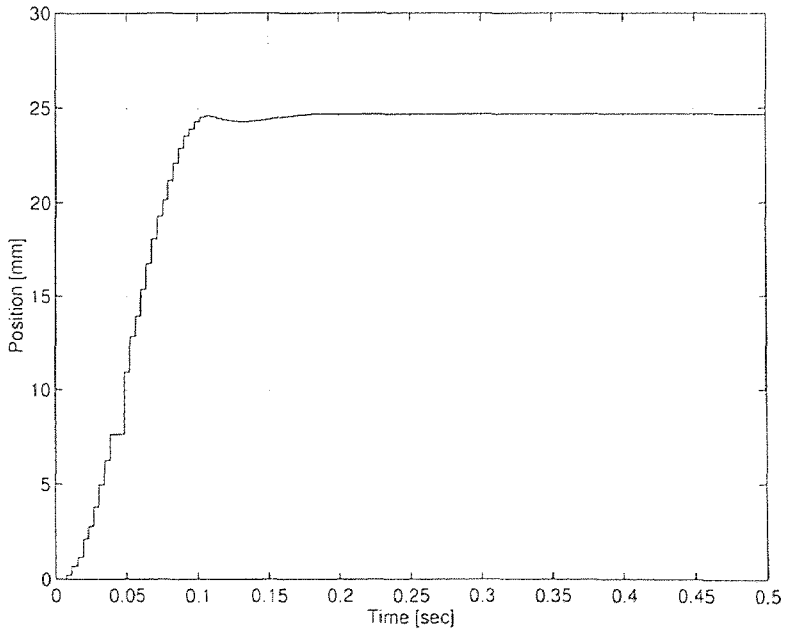

Figure 5.3 Result of ZV Shaper: Experiment #3

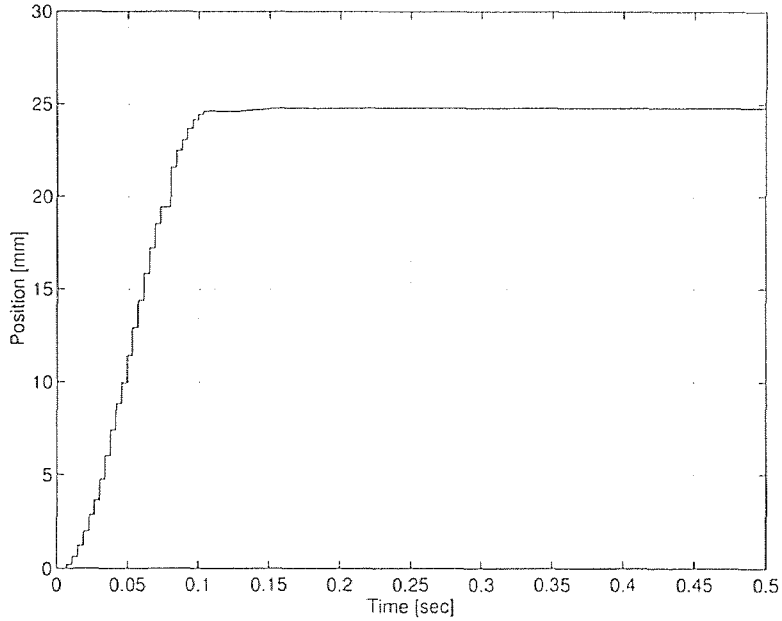

Figure 5.4 Result of ZV Shaper: Experiment #4

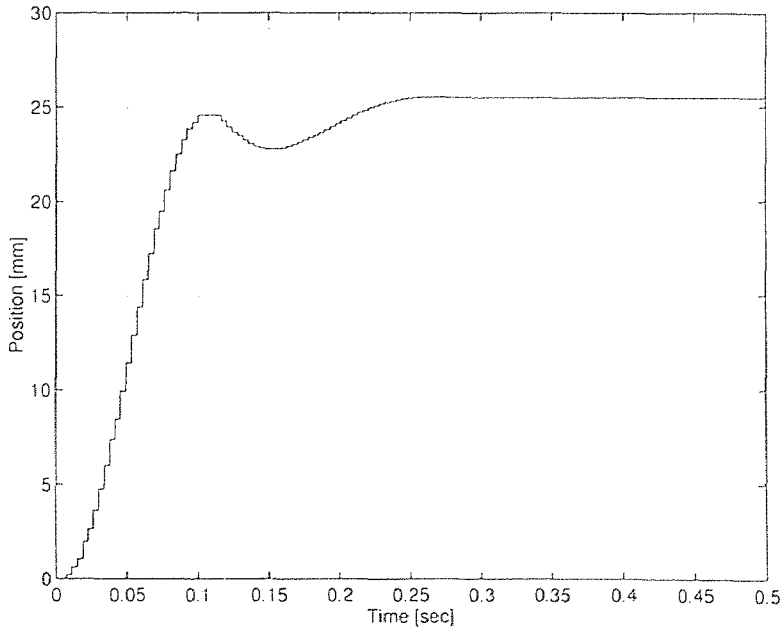

Figure 5.5 Result of ZV Shaper: Experiment #5

5.2 Results of ZVD Shaper

Figures (5.6), (5.7), (5.8), (5.9) and (5.10) show the results of ZVD shaper under 5 consecutive trials. Same as the ZV shaper, it is observed that the vibration of the linear module has been significantly attenuated. The amount of overshoot beyond the 25 mm reference level has been reduced from 80 % to about 2 % . The settling time for ZVD shaper is longer than ZV shaper since there are 2 switching times. But most importantly ZVD is more robust than ZV shaper. Transient responses are more consistent than ZV case.

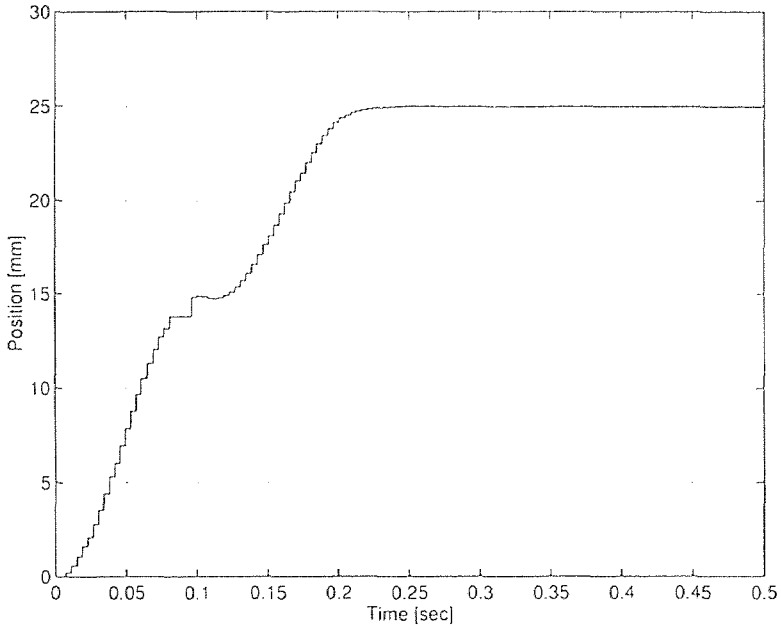

Figure 5.6 Result of ZVD Shaper: Experiment #1

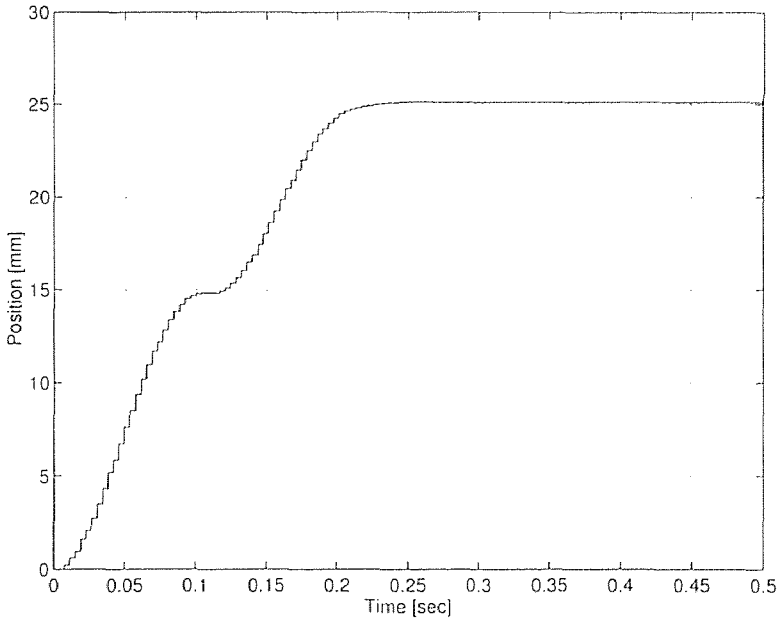

Figure 5.7 Result of ZVD Shaper: Experiment #2

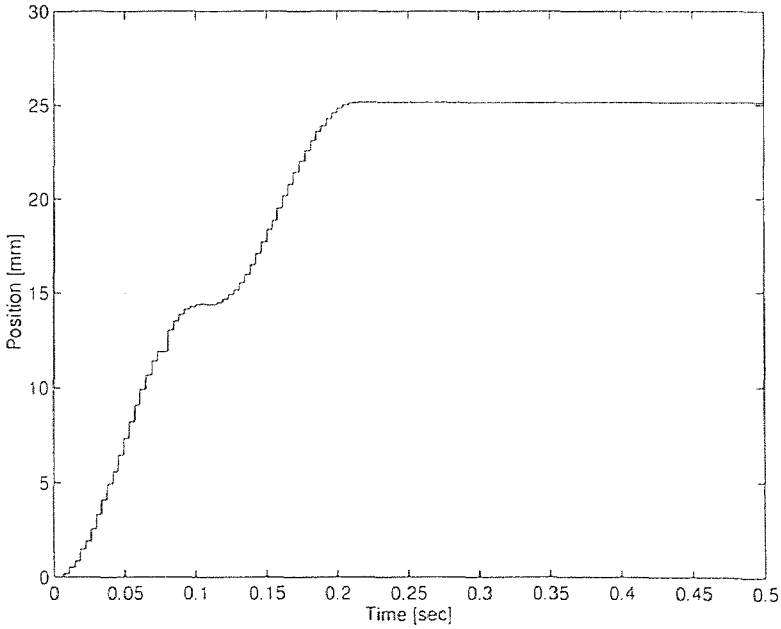

Figure 5.8 Result of ZVD Shaper: Experiment #3

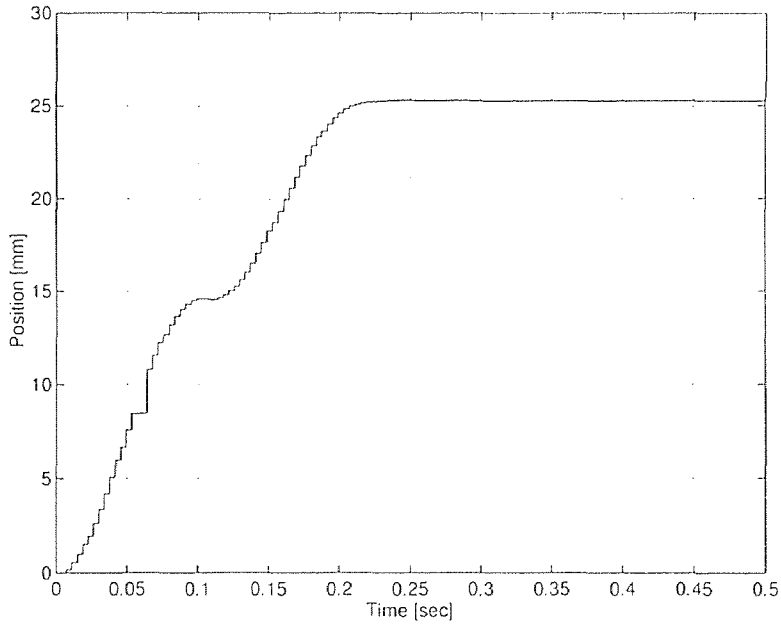

Figure 5.9 Result of ZVD Shaper: Experiment #4

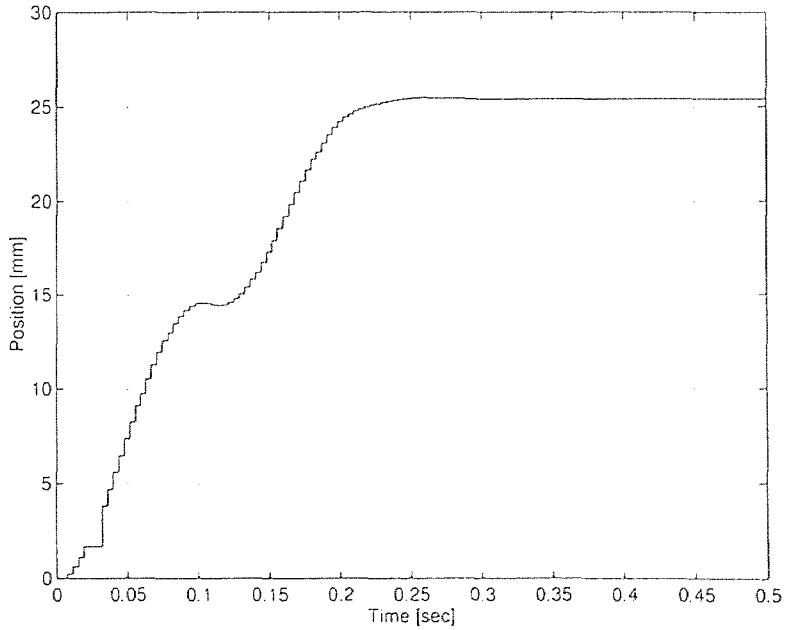

Figure 5.10 Result of ZVD Shaper: Experiment #5

5.3 Results of EI Shaper for V=0.05

Figures (5.11), (5.12), (5.13), (5.14) and (5.15) show the results of EI shaper under 5 consecutive trials. EI shaper also significantly reduces the system vibration after impulse sequence ends. The amount of overshoot beyond the 25 mm reference level has been reduced from 80 % to about 2 % . The settling time for EI, as in ZVD shaper is longer than ZV shaper since there are 2 switching times. The transient response is more consistent than ZV shaper. V is a design parameter and must be tuned for the best system response.

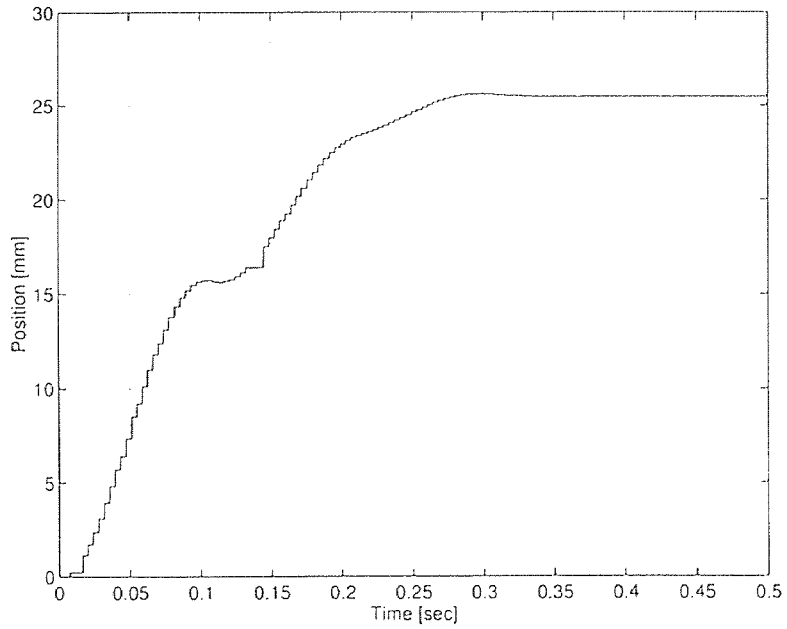

Figure 5.11 Result of EI Shaper For V=0.05: Experiment #1

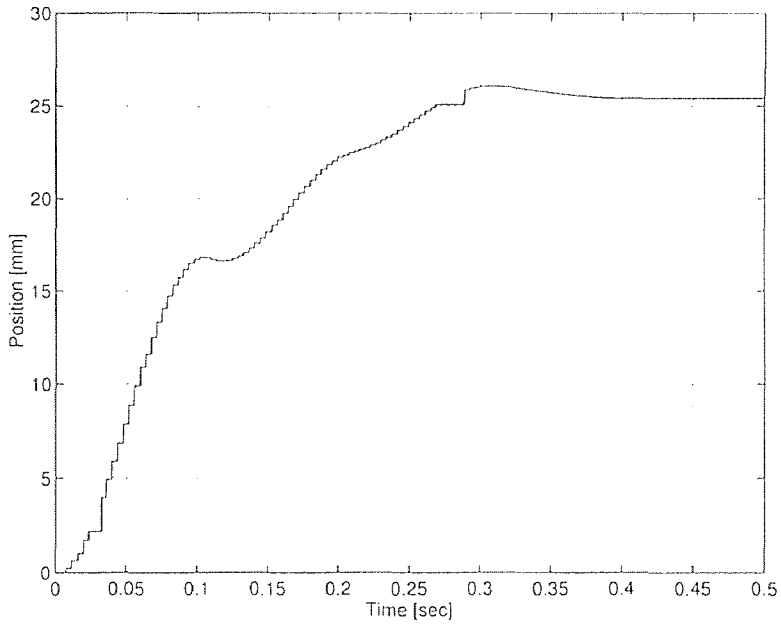

Figure 5.12 Result of EI Shaper For V=0.05: Experiment #2

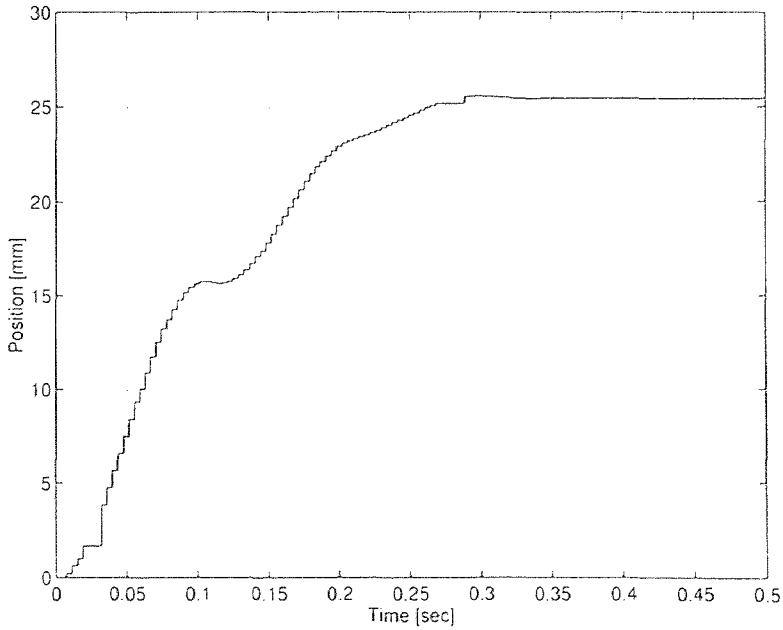

Figure 5.13 Result of EI Shaper For V=0.05: Experiment #3

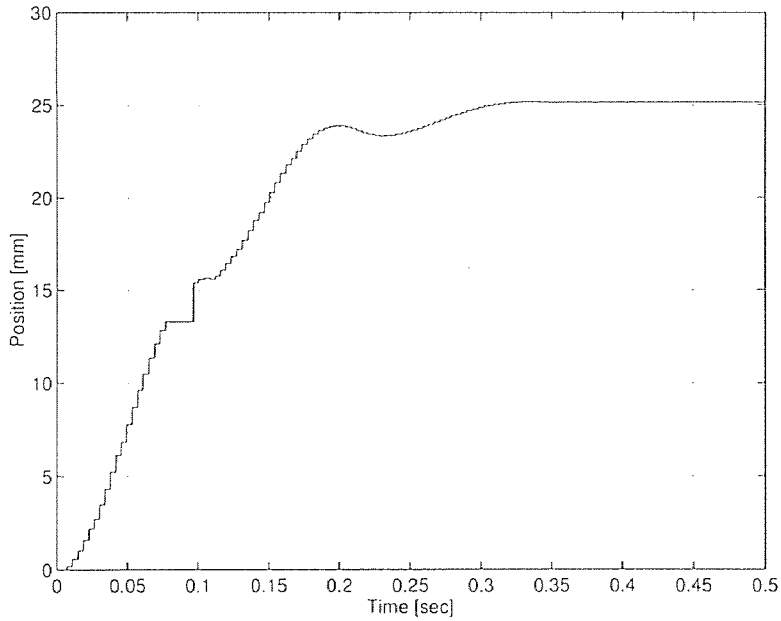

Figure 5.14 Result of EI Shaper For V=0.05: Experiment #4

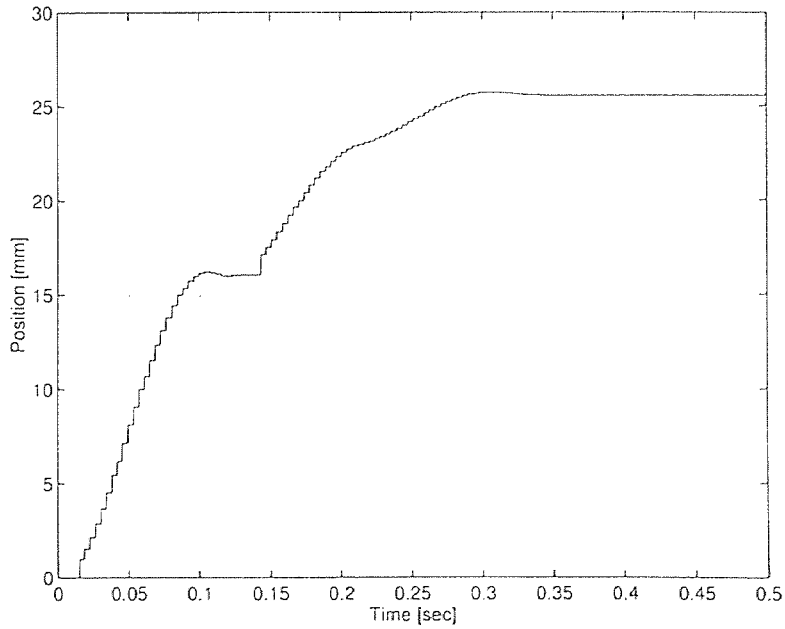

Figure 5.15 Result of EI Shaper For V=0.05: Experiment #5

5.4 Comparison Between ZV, ZVD, and EI

In this section comparison between different algorithms will be given ($\Delta T = 0.1025sec$).

Table 5.1 Position error [mm] in different times for ZV

ZV	$2\Delta T$	$3\Delta T$	$4\Delta T$
#1	+0.2600	+0.2425	+0.2425
#2	+0.7350	-0.5400	-0.5375
#3	+0.3375	+0.3300	-0.3300
#4	+0.2250	+0.2200	-0.2200
#5	+0.5875	-0.4850	-0.4825
RMS	0.2114	0.1724	0.1718
% Vibration	0.8457	0.6895	0.6872

Table 5.2 Position error [mm] in different times for ZVD

ZVD	$2\Delta T$	$3\Delta T$	$4\Delta T$
#1	+0.4675	+0.0050	+0.0050
#2	+0.5150	-0.1200	-0.1200
#3	+0.0000	-0.1400	-0.1375
#4	+0.1775	-0.2625	-0.2625
#5	+0.5525	-0.3875	-0.3875
RMS	0.1812	0.1006	0.1005
% Vibration	0.7247	0.4025	0.4019

From these plots, the following conclusions can be drawn:

- ZVD & EI (V=0.05) are more "robust" than ZV, as evident from the consistency of their transient responses.
- As expected % Vibration for EI is less than % 5.
- ZV performs faster because of shorter switching and significant system damping.
- In terms of the level of residual vibration, it is noted that ZVD has significantly lower level of steady state error, in agreement with the theoretical predictions.

Table 5.3 Position error [mm] in different times for EI (V=0.05)

EI	$2\Delta T$	$3\Delta T$	$4\Delta T$
#1	+1.9725	-1.5625	-0.4725
#2	+2.6675	-1.0750	-0.4350
#3	+1.9300	-0.5125	-0.4425
#4	+1.1900	+0.0300	-0.1450
#5	+2.2675	-0.7175	-0.5525
RMS	0.9228	0.4183	0.1933
% Vibration	3.6912	1.6734	0.7734

5.5 Suggestions for Improvement

The installation of the linear robots is an evolutionary sequence of effort. The EXC controller is the easiest to install yet the least flexible. For the VME controller, the lack of compiler is an important disadvantage for system development. During this work, it is observed that every additional line added to V+ program severely decreased the execution time of the application software. As a result V+ may be sufficient for end-user applications but for advanced control, the VME controller must be augmented by a compiler for faster program execution times. This was the main reason to add TMS320C31 to system. To further improve the system response, a number of changes can be incorporated.

Software Improvements

- deadbeat control for fast, feedback control,
- accelerator feedback loop to improve transient,
- an integrator loop to overcome the friction in the system,
- for faster program execution VME system has to have a compiler.

Hardware Improvements

- a decoder to read the modules' position directly, thereby increasing the bandwidth of the system,
- better shielding to reduce noise level.

CHAPTER 6
CONCLUSIONS

In this work vibration control of robotic modules using input shaping algorithm is studied. Input shaping is an actively used algorithm for vibration control of flexible structures. The effectiveness of input shaping has been proved on may different systems. There are various input shaper design methods of which ZV, ZVD, and EI techniques are applied in this work. Robustness to the modeling errors lead different types of input shapers. To increase robustness to these modeling errors additional constraints are added to ZV constraints. Otherwise if natural frequencies and damping ratios of the closed loop system were known ZV shaper would be the optimum case. Input shaping is a feedforward method. Addition of the feedforward methods can dramatically reduce the complexity of feedback methods.

In this thesis we have installed the application hardware and implemented several input shaper methods and made comparison between them. We have seen that input shaping is an extremely easy and effective method. We have reduced overshoot from 80 % to 2 %. For all of the methods the settling time reduced from 1 sec to 0.25 sec. ZV shaper gave the best performance because of shorter switching time and significant system damping. ZVD and EI shaper were more robust since they had more consistent transient responses.

In this work significant effort spent on hardware application. First we have started with EXC controller. Due to its lack of flexibility and programming capabilities we have switched to VME controller. VME controller was much more flexible and powerful than EXC controller but program execution times were not fast enough. Its lack of compiler was the major disadvantage. For example in a simple program which basically gets data from ADC multiplies data by 2 and sends it to output, the execution time was 0.174 ms. Every "aio.in" command which gets data from ADC adds 0.1 ms to program execution time. For these reasons we have

decided to use Dalanco Spry's DSP board. First VME controller's proportional and integral gains zeroed to disable VME to control the linear module. A program residing in VME sent position via its serial port to PC with 250 Hz sampling rate. To increase the time resolution in DSP we have used 10 KHz sampling rate.

As a result we have shown that input shaping improves settling time and positioning accuracy by reducing residual vibrations in computer controlled machines.

APPENDIX A

SOFTWARE

In this chapter software part of thesis will be explained. In this work following programs have been developed.

- Program in VME part.
- Program in C31 part.
    1. PID
    2. ZV
    3. ZVD
    4. EI
- Program in host part.

A.1 VME Program

The program in VME is written in V^+ which is a specific language used for Adept VME modules. This program basically gets the encoder position by using Advanced Servo Library [15] function "sl.read". As it'll be seen in program in stead of directly sending position value, conversion is done in order to reduce the packet size and to increase overall sampling rate. By doing so packet size became from 9 bytes (if position is 6 digits) to 4 bytes with suppressing CR and LF.

In early stages of thesis we've seen that Adept uses following packet for serial line communication. Let's suppose we want to send 123456 encoder counts position value to PC: what we see at PC is:

32 49 50 51 52 53 54 13 10

by using "S" option in write command we suppressed CR and LF (13 10). Since position is in [0-400000] we needed actually 3 bytes to represent encoder value. For this reason before we send position value some conversions made. And resulting 3 bytes value are sent over serial line. With this program sampling rate is 250 samples/sec in 19200bps.

```
;Program in VME that send position value to PC
.PROGRAM serial()
AUTO $password[15]
AUTO slun
AUTO encpos
$password[1] = "adept"
CALL sl.init($password[], status[])
ATTACH (slun, 4) "SERIAL:1"
IF IOSTAT(slun) < 0 GOTO 100

TYPE "Serial Communication Program is in infinite loop"
WHILE TRUE DO

10        c = GETC(slun,1) ; program waits 55 (arbitrarily chosen)
  IF c <> 55 GOTO 10 ; from PC for synchronization

  CALL sl.read(1, sl.motpos, encpos, error) ; get position
  $a = $LNGB(encpos) ; convert position value
  a1 = ASC($MID($a,4,1)) ; to 3 bytes
  a2 = ASC($MID($a,3,1))
  a3 = ASC($MID($a,2,1))
  WRITE (slun) $CHR(a1), $CHR(a2), $CHR(a3), /S ; send position
  IF IOSTAT(slun) < 0 GOTO 100  ; if there is an error go to 100

END

CALL sl.asl.cleanup()
   100 IF (IOSTAT(slun) < 0) THEN
    TYPE IOSTAT(slun), " ", $ERROR(IOSTAT(slun))
END
DETACH (slun)

.END
```

A.2 C31

A.2.1 PID

Following program is used to apply PID to plant.

```
/* TIMPER0        SAMPLING FREQUENCY
    2500           2.5khz
     625          10  khz
    1250           5  khz
   25000         250  hz
  sampling freq=6.25Mhz / Timper0
*/

extern long y,ui,count,newpos,error;

void main(){
long deltat,yref,en,enm1,shape,sum_e,u;
float kp,ki,kd,kpp,kii,kdd,a1,prop,integ,deriv;
int first;
asm("   .bss       _y,1");
asm("   .global    _y");
asm("   .bss       _ui,1");
asm("   .global    _ui");
asm("   .bss       _count,1");
asm("   .global    _count");
asm("   .bss       _newpos,1");
asm("   .global    _newpos");
asm("   .bss       _error,1");
asm("   .global    _error");

asm(" .word      start");
asm(" .space     3fh");
asm("start:");
asm(" LDP     0h");
asm(" LDI     1800H,ST");
asm(" LDI     0985H,SP");
asm(" LDI     @IOF_SET_XF1,R0");
asm(" LDI     R0,IOF");
asm(" LDI     @CTRL,AR7");
asm(" LDI     1000h,AR1");
asm(" LDI     @POINTER,AR2");
asm(" LDI     @LATCH_AREA,AR3");
asm(" LDI     @LATCH_VAL,R0");
asm(" STI     R0,*+AR3(0)");
```

```
asm(" LDI     @SERGLOBA,R0");
asm(" STI     R0,*+AR7(64)");
asm(" LDI     @SERTIMOVAL,R0");
asm(" STI     R0,*+AR7(70)");
asm(" LDI     @SERTIMO,R0");
asm(" STI     R0,*+AR7(68)");
asm(" LDI     @SERPRTXO,R0");
asm(" STI     R0,*+AR7(66)");
asm(" LDI     @SERPRTRO,R0");
asm(" STI     R0,*+AR7(67)");
asm(" LDI     @SERGLOBO,R0");
asm(" STI     R0,*+AR7(64)");
asm(" LDI     18H,R0");
asm(" STI     R0,*+AR7(64H)");
asm(" LDI     0,R0");
asm(" STI     R0,*+AR7(20H)");
asm(" LDI     @TIMPERO,R0");
asm(" STI     R0,*+AR7(28H)");
asm(" LDI     @TIMGBOSTART,R0");
asm(" STI     R0,*+AR7(20H)");
count=0;
newpos=10;
infinite_loop:
asm("next_sample:");
asm(" LDI     *+AR7(20H),R0");
asm(" AND     800H,R0");
asm(" BZ      next_sample");
asm("n1:");
asm(" LDI     *+AR7(20H),R0");
asm(" AND     800H,R0");
asm(" BNZ     n1");
asm(" NOP");
asm(" NOP");
asm(" LDI     @IOF_SET_XF1,R0");
asm(" LDI     R0,IOF");
asm("wait_sample:");
asm(" LDI     *+AR7(64),R2");
asm(" AND     01H,R2");
asm(" BZ      wait_sample");
asm(" LDI     @IOF_RESET_XF1,R0");
asm(" LDI     R0,IOF");
asm(" LDI     *+AR7(76),R3");
asm(" ASH     -18,R3");
 asm(" LDI     6e20h,R2");
```

```
asm(" LDI     AR2,R1");
asm(" CMPI    R2,R1");
asm(" BZ      skip_mem_write");
asm(" STI     R3,*AR2++(1)");
asm("skip_mem_write:");

/*********************************************************************/
/***** CONTOL ALGORITHM STARTS FROM HERE *****************************/
asm(" LDI     *+AR1(0),R3"); /*get position from mem. location 1000h */
asm(" STI     R3,@_y");

yref=144000;          /* move from 154000 to 144000 */
kp=40.0;
ki=0.0;
kd=100.0;
enm1=en;
en=y-yref;
if(first==0) {
enm1=en;
first=1;
}
asm(" LDI     *+AR1(1),R3");
asm(" STI     R3,@_newpos");
if(newpos==10) {
    sum_e=sum_e+en;
    prop=kp*en;
    integ=ki*sum_e;
    deriv=kd*(en-enm1);
    asm(" LDI     0,R3");
    asm(" STI     R3,*+AR1(1)");
}
  u=prop+integ+deriv;
  u=u*0.0051175;   /* 1/400000*2047=0.0051175 */
  ui=(long)u;
  if(ui>2045)  ui=2045;
  if(ui<-2046) ui=-2046;
asm("try_send:");
asm(" LDI     *+AR7(64),R2");
asm(" AND     02H,R2");
asm(" BZ      try_send");
asm(" LDI     @_ui,R3");
asm(" AND     0FFFH,R3");
asm(" STI     R3,*+AR7(48H)");
```

```
goto infinite_loop;

asm("IOOF_AMASK       .word   0EH");
asm("IOF_SET_XF1      .word   62H");
asm("IOF_RESET_XF1    .word   22H");
asm("CTRL       .word   808000H");
asm("POINTER          .word   2000H");
asm("TIMGB0CONHI      .word   6H");
asm("TIMGB0CONLO      .word   2H");
asm("TIMGB0START      .word   3C1H");
asm("TIMGB0STOP       .word   381H");
asm("TIMGB0RESTART    .word   341H");
asm("TIMPER0          .word   625");
asm("SERGLOBA         .word   150144H");
asm("SERGLOB0         .word   0C150144H");
asm("SERPRTX0         .word   111H");
asm("SERPRTR0         .word   111H");
asm("SERTIM0          .word   3CFH");
asm("SERTIM0VAL       .word   01h");
asm("XVALUES          .word   809800H");
asm("LATCH_VAL        .word   0H");
asm("LATCH_AREA       .word   0FFFFFFH");

}
```

A.2.2 ZV

In order to apply ZV just small part of the PID.C program is changed, therefore only difference will be given:

```
/*********************************************************************/
/***** CONTOL ALGORITHM STARTS FROM HERE *****************************/
asm(" LDI     *+AR1(0),R3"); /*get position from mem. location 1000h */
asm(" STI     R3,@_y");

yref=144000;          /* move from 154000 to 144000 */
shape=-10000;
kp=40;
ki=0;
kd=100;
deltat=1025;
a1=0.6027;
kpp=kp;
kii=ki;
kdd=kd;
if(count<deltat) shape=shape*a1;

if(count<20000) {
    kpp=40;
    kii=0;
    kdd=100;
    sum_e=0;
    count++;
}
yref=154000+shape;
enm1=en;
en=y-yref;

probe=y;
asm(" LDI     @_probe,R3");
asm(" LDI     6e20h,R2");
asm(" LDI     AR2,R1");
asm(" CMPI    R2,R1");
asm(" BZ      skip_mem_write");
asm(" STI     R3,*AR2++(1)");
asm("skip_mem_write:");

if(first==0) {
enm1=en;
```

```
first=1;
}
asm(" LDI     *+AR1(1),R3");
asm(" STI     R3,@_newpos");
if(newpos==10) {
    sum_e=sum_e+en;
    prop=kpp*en;
    integ=kii*sum_e;
    deriv=kdd*(en-enm1);
    asm(" LDI     0,R3");
    asm(" STI     R3,*+AR1(1)");
}
  u=prop+integ+deriv;
  u=u*0.0051175;   /* 1/400000*2047=0.0051175 */
  ui=(long)u;
  if(ui>2045)  ui=2045;
  if(ui<-2046) ui=-2046;


asm("try_send:");
asm(" LDI     *+AR7(64),R2");
asm(" AND     02H,R2");
asm(" BZ      try_send");
asm(" LDI     @_ui,R3");
asm(" AND     0FFFH,R3");
asm(" STI     R3,*+AR7(48H)");

goto infinite_loop;
```

A.2.3 ZVD

```
/*********************************************************************/
/***** CONTOL ALGORITHM STARTS FROM HERE ****************************/
asm(" LDI     *+AR1(0),R3"); /*get position from mem. location 1000h */
asm(" STI     R3,@_y");

yref=144000;          /* move from 154000 to 144000 */
shape=-10000;
kp=40;
ki=0;
kd=100;
deltat=1025;
a1=0.3633;
a2=0.4789;
kpp=kp;
kii=ki;
kdd=kd;
if(count<deltat) shape=shape*a1;
if((count>=deltat) && (count<2*deltat))shape=shape*(a1+a2);

if(count<200000) {
    kpp=40;
    kii=0;
    kdd=100;
    sum_e=0;
    count++;
}
yref=154000+shape;
enm1=en;
en=y-yref;

error=y;
asm(" LDI     @_error,R3");
asm(" LDI     6e20h,R2");
asm(" LDI     AR2,R1");
asm(" CMPI    R2,R1");
asm(" BZ      skip_mem_write");
asm(" STI     R3,*AR2++(1)");
asm("skip_mem_write:");

if(first==0) {
enm1=en;
first=1;
```

```
}
asm(" LDI     *+AR1(1),R3");
asm(" STI     R3,@_newpos");
if(newpos==10) {
    sum_e=sum_e+en;
    prop=kpp*en;
    integ=kii*sum_e;
    deriv=kdd*(en-enm1);
    asm(" LDI     0,R3");
    asm(" STI     R3,*+AR1(1)");
}
  u=prop+integ+deriv;
  u=u*0.0051175;   /* 1/400000*2047=0.0051175 */
  ui=(long)u;
  if(ui>2045)  ui=2045;
  if(ui<-2046) ui=-2046;


asm("try_send:");
asm(" LDI     *+AR7(64),R2");
asm(" AND     02H,R2");
asm(" BZ      try_send");
asm(" LDI     @_ui,R3");
asm(" AND     0FFFH,R3");
asm(" STI     R3,*+AR7(48H)");

goto infinite_loop;
```

A.2.4 EI

```
/*********************************************************************/
/***** CONTOL ALGORITHM STARTS FROM HERE ****************************/
asm(" LDI     *+AR1(0),R3"); /*get position from mem. location 1000h */
asm(" STI     R3,@_y");

yref=144000;          /* move from 154000 to 144000 */
shape=-10000;
kp=40;
ki=0;
kd=100;

/*for V=0.05*/
T2=1039;
T3=2050;
A1=0.3868;
A2=0.4406;

kpp=kp;
kii=ki;
kdd=kd;
if(count<T2) shape=shape*A1;
if((count>=T2) && (count<T3))shape=shape*(A1+A2);

if(count<200000) {
    kpp=40;
    kii=0;
    kdd=100;
    sum_e=0;
    count++;
}
yref=154000+shape;
enm1=en;
en=y-yref;

error=y;
asm(" LDI     @_error,R3");
asm(" LDI     6e20h,R2");
asm(" LDI     AR2,R1");
asm(" CMPI    R2,R1");
asm(" BZ      skip_mem_write");
asm(" STI     R3,*AR2++(1)");
asm("skip_mem_write:");
```

```
if(first==0) {
enm1=en;
first=1;
}
asm(" LDI     *+AR1(1),R3");
asm(" STI     R3,@_newpos");
if(newpos==10) {
    sum_e=sum_e+en;
    prop=kpp*en;
    integ=kii*sum_e;
    deriv=kdd*(en-enm1);
    asm(" LDI     0,R3");
    asm(" STI     R3,*+AR1(1)");
}
  u=prop+integ+deriv;
  u=u*0.0051175;   /* 1/400000*2047=0.0051175 */
  ui=(long)u;
  if(ui>2045)  ui=2045;
  if(ui<-2046) ui=-2046;


asm("try_send:");
asm(" LDI     *+AR7(64),R2");
asm(" AND     02H,R2");
asm(" BZ      try_send");
asm(" LDI     @_ui,R3");
asm(" AND     0FFFH,R3");
asm(" STI     R3,*+AR7(48H)");

goto infinite_loop;
```

A.3 Host Program

This is the program that runs in PC. It gets position and writes it to certain memory position (1000H) in C31.

```
#include<stdio.h>
#include<stdlib.h>
#include<bios.h>
#include<dos.h>
#define COM2        1
#define DATA_READY 0x100
#define SETTINGS ( 0xE0 | 0x00 | 0x00 | 0x03)
extern co80320();
void main(void){

   char a,c[10];
   int run=0,i=0,status,*newdata;
   long perr,s[2];
   union  x{
   char c[4];
   long position;
   }y;
   clrscr();

   a=inportb(0x02f8);
   a=inportb(0x02f8);
   a=inportb(0x02f8);
   a=inportb(0x02f8);
   a=inportb(0x02f8);
   a=inportb(0x02f8);

xx:  outportb(0x2f8,55);

   status = bioscom(3, 0, COM2);
   if (status & DATA_READY){
 a=inportb(0x02f8);
 y.c[i]=a;
 i++;
   }
   status = bioscom(3, 0, COM2);
   if (status & DATA_READY){
 a=inportb(0x02f8);
 y.c[i]=a;
 i++;
```

```
    }
    status = bioscom(3, 0, COM2);
    if (status & DATA_READY){
 a=inportb(0x02f8);
 y.c[i]=a;
 i++;
    }

      if(i==3){
        y.c[3]=0;
        i=0;
        if((y.c[2] & 0x80 )!=0){
 y.c[3]=0xff;
        }
        if((y.position<0) || (y.position>400000)) goto vv;
        s[0]=y.position;
        s[1]=10;
        sendio(s,2,0x10001,0x300);
        perr=y.position-144000;
        printf("\n%ld",perr);
        flushall();
        if(run==0){
 run=1;
 go320(0x300);
        }
      }
vv:

      if(kbhit()==0) goto xx;
      hlt320(0x300);
}
```

APPENDIX B

HARDWARE SPECIFICATIONS

Table B.1 Motor and Encoder Specifications

Control Input	Torque Command Rated torque at 3.3v Maximum torque=3x rated torque
Motor	300w AC servo motor
Position feedback	2000 lines/revolution Maximum frequency:150 Khz
Maximum Motor Speed	3600 r.p.m.
I/O	**Input** Drive Enable Alarm Clear Brake Release Output: Drive Fault Positive/Negative Overtravel
Alarms	Main Voltage Error(MVE) Overhead(OH) Control Power Under Voltage(CUV) Motor Over Current(OC) Encoder Cable Break(LOS)
Power	Single phase 180-240VAC Required Power: 1170VA max. (at max. torque)

CN1 is the connector between VMP interface panel and amplifier. Table (B.2) gives CN1 connector pinouts. Table (B.3) gives signal decriptions of CN1 connector.

Table B.2 Connector CN1

1	24V GND	In	24VDC Ground
2	BR-	In	Brake Release (-)
3	BR+	In	Brake Release (+)
4	Not connected		
5	CD-	In	Torque Command Ground
6	CD+	In	Torque Command
7	CD-	In	Torque Command Ground
8	Alarm Clear	In	Alarm Clear
9	DE-	In	Servo On (-)
10	DE+	In	Servo On (+)
11	Not connected		
12	+24VDC	In	24VDC
13	Not connected		
14	Not connected		
15	Not connected		
16	Not connected		
17	NO-	Out	Negative Over Travel (-)
18	NO+	Out	Negative Over Travel (+)
19	PO-	Out	Positive Over Travel (-)
20	PO+	Out	Positive Over Travel (+)
21	DF-	Out	Driver Fault (-)
22	DF+	Out	Driver Fault (+)
23	S GND		Signal Ground (Line driver ground)
24	S GND		Signal Ground (Line driver ground)
25	Not connected		
26	I-	Out	Encoder Phase I (-)
27	I+	Out	Encoder Phase I (+)
28	B-	Out	Encoder Phase B (-)
29	B+	Out	Encoder Phase B (+)
30	A-	Out	Encoder Phase A (-)
31	A+	Out	Encoder Phase A (+)
32	Not connected		
33	Not connected		
34	Not connected		

Table B.3 Signal Description

DE+ DE-	Drive Enable/Servo On input. Enabling DE after DF output closed makes motor servo turn on.
Alarm Clear	Alarm Clear Input. All alarms except LOS can be reset by this input. Alarm clear input should be one-shot input. If the Alarm clear input is always on, the alarm function cannot work. Before clearing alarm output, the cause of alarm should be removed. (to prevent a fatal error.
BR+ BR-	Brake release input. Enabling BR releases the motor break. 24 VDC has to be supplied to pin 12 (+24 VDC) referenced to pin 1 (24 V GND) for releasing the brake.
DF+ DF-	Drive Fault/Alarm Output. In case of alarm condition, DF contact will be open state and motor servo will be turned off.
NO+ NO- PO+ PO-	Over travel sensor output. The slider moves into over travel position and the NO (PO) contact will be in open state. If the modules are not connected to the amplifier or sensor cables are broken, NO (PO) contacts will be open state.
CD+ CD-	Torque command input. Analog voltage input is proportional to the motor output torque. Command input voltage: ± 10 VDC. Input impedance: 20 KΩ
A+ A- B+ B-	Encoder outputs. 2000 counts/rev. I+, I- 1 count/rev. Maximum frequency is 150 KHz for Phase A&B.

Table B.4 Analog Outputs

Number of channels	4
Accuracy	
Resolution	12 bits
Overall error	± 1/2 LSB
Differential Linearity	± LSB
Voltage output characteristics	
Ranges	0-5 V, 0-10 V, ±5 V, ±10 V
Output Current, maximum	5 mA @ ±10 V
Settling Time	7 μs
Offset Temperature Coefficient	75 ppm/°C
Gain Temperature Coefficient	100 ppm/°C
Current Loop Output Characteristics	
Range	4-20 mA, Non-isolated
Compliance voltage	10 V @ 20 mA
Loop supply voltage	+15 V to +30 V
Settling time	50 μs
Load resistancerange	50-500W
Offset temperature coefficient	75 ppm/°C
Gain temperature coefficient	100 ppm/°C

Table B.5 Analog Inputs

Number of Channels	
In single-ended mode	32
In differential mode	16
A/D input full scale voltage ranges (gain=1)	
Unipolar	0-5 V, 0-10 V
Bipolar	±2.5 V, ±5 V, ±10V
Programmable Gain	
Range 1	1, 2, 5, or 10
Range 2	1, 2, 5, or 10 or 40
Range 3	10, 20, 50, or 100
Maximum input voltage	
Power on	44 V
Power off	30 V
Input impedance	
With 22 MW resistor	17 MW min.
Without 22 MW resistor	100 MW min.
Bias Current	±100 nA
Input Capacitance	225 pF, max.
Operating Common Mode Voltage	14V
Accuracy	
Resolution	12 bits
Overall errror	± 1/2 LSB
Differential linearity	± 1/2 LSB
Common Mode Rejection Ratio	60 dB, min.
Speed	
Conversion time	
Single mode	25 μs
All other modes	50 μs
Throughput	
Single mode	40 KHz
All other modes	20 KHz
External trigger to sample	25 μs

REFERENCES

1. Timothy N. Chang Lucy Y. Pao and Edwin Hou. Input shaper designs for minimizing the expected level of residual vibration in flexible structures. *Submitted in September 1996 for the American Control Conference.*

2. NSK Ltd. *EXC Controller User's Manual*, July 1994.

3. Dalanco Spry. *Model 310 Data Acquisition and Signal Processing Board for The IBM PC and ISA Bus Compatibles*, 1993.

4. N.C.Singer and W.P.Seering. Preshaping command inputs to reduce system vibration. *Journal of Dynamic Systems, Measurement, and Control*, 112:76–81, March 1990.

5. K.L.Hillsley and S.Yurkovich. Vibration control of two-link flexible robot arm. *Proceedings of the 1991 IEEE International Conference on Robotics and Automation*, pages 2121–2125, April 1991.

6. Neil C.Singer. *Residual Vibration Reduction in Computer Controlled Machines.* PhD thesis, Department of Mechanical Engineering, MIT, Fall 1988.

7. James M.Hyde and W.P.Seering. Using input command pre-shaping to suppress multiple mode vibration. *Proceedings of the 1991 IEEE International Conference on Robotics and Automation, Sacramento, CA*, pages 2604–2609, April 1991.

8. Neil C. Singer B. Whitney Rappole, Jr. and W.P.Seering. Input shaping with negative sequences for reducing vibration in flexible structures. *Proceedings of the American Control Conference*, pages 2695–2699, June 1993.

9. Timothy D. Tuttle and W.P.Seering. A zero placement technique for designing shaped inputs to suppress multiple-mode vibration. *Proceedings of the American Control Conference, Baltimore, Maryland*, pages 2533–2537, June 1994.

10. William Singhose and Neil Singer. Initial investigations into the effects of input shaping on trajectory following. *Proceedings of the American Control Conference, Baltimore, Maryland*, pages 2526–2532, June 1994.

11. Adept Technology Inc. *Adept MV Controller Developer's Guide*, version 11.2 edition, December 1995.

12. Texas Instruments. *TMS320C3x User's Guide*, October 1994.

13. Texas Instruments. *TMS320 Floating-Point Optimizing C Compiler User's Guide*, February 1995.

14. Texas Instruments. *TMS320 Floating-Point DSP Assembly Language Tools User's Guide*, February 1995.

15. Adept Technology Inc. *Advanced Servo Library Reference Guide*, version 11.2 edition, December 1995.