A DIGITAL SIGNAL PROCESSOR BASED OPTICAL POSITION SENSOR AND ITS APPLICATION TO FLEXIBLE BEAM CONTROL

by
Thomas J. Spirock

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

May 1996

APPROVAL PAGE

A DIGITAL SIGNAL PROCESSOR BASED OPTICAL POSITION SENSOR AND ITS APPLICATION TO FLEXIBLE BEAM CONTROL

Thomas J. Spirock

Dr. Timothy Chang, Thesis Advisor — Date
Assistant Professor of Electrical Engineering, NJIT

Dr. Walter Kosonocky, Committee Member — Date
Distinguished Professor of Electrical Engineering
and Holder of Foundation Chair in Optoelectronics and Solid-State Circuits, NJIT

Dr. Philip Goode, Committee Member — Date
Professor of Physics, NJIT

ABSTRACT

A DIGITAL SIGNAL PROCESSOR BASED OPTICAL POSITION SENSOR AND ITS APPLICATION TO FLEXIBLE BEAM CONTROL

by Thomas J. Spirock

A Digital Signal Processor (DSP) based optical position sensor was developed. The sensor system consists of the following components: 1) analog electronics, 2) the DSP based synchronous demodulation software, 3) PC based interface software which samples and saves the data, and 4) PC based control codes for a flexible beam experiment.

The ability of the system to determine the distance from the optical sensor to the power modulated light source was assessed by the following tests: 1) a stationary drift test to evaluate the system's noise, 2) a short-range test to determine the resolution of the optical sensor over a 25mm range and, 3) a long-range test to evaluate the ability of the system to predict the location of the optical sensor over a 600mm range. It was found that the resolution of the system is approximately 0.5mm for the short range test and 5mm for the long range test.

Finally, the sensor was deployed for the position feedback of a flexible beam experiment. Performance indices used to evaluate the response of the system were: 1) the sum of the squared position error, 2) the final steady state position error of the end of the flexible beam, and 3) the 5% settling time of the flexible beam. A number of control laws were evaluated and it was determined that a variable PID controller produced the best overall performance. The system can consistently position the end of the flexible beam from a +/- 20cm to within 5mm of the command position in approximately 8 seconds with a properly tuned controller.

Blank Page

BIOGRAPHICAL SKETCH

Author: Thomas J. Spirock

Degree: Master of Science in Electrical Engineering

Date: May, 1996

Undergraduate and Graduate Education:

- Master of Science in Electrical Engineering,
  New Jersey Institute of Technology, Newark, NJ, 1996
- Bachelor of Science in Electrical Engineering,
  New Jersey Institute of Technology, Newark, NJ, 1993

Major: Electrical Engineering

TABLE OF CONTENTS

TABLE OF CONTENTS
(Continued)

TABLE OF CONTENTS
(Continued)

TABLE OF CONTENTS
(Continued)

LIST OF FIGURES

LIST OF FIGURES
(Continued)

LIST OF FIGURES
(Continued)

LIST OF FIGURES
(Continued)

LIST OF FIGURES
(Continued)

LIST OF FIGURES
(Continued)

LIST OF TABLES

LIST OF TABLES
(Continued)

CHAPTER 1

INTRODUCTION

In this thesis work, an optical position sensor has been designed and implemented. A number of tests were performed to characterize the drift, short-range, and long-range properties. The system was then deployed in a flexible beam control experiment.

1.1 Overview

The purpose of the project is to design and build an optical position sensor and to apply this sensor in motion control systems. The sensor's operation is based on the inverse square law property of light propagation which is used to derive the distance between a power-modulated laser diode and a light intensity sensor.

A Digital Signal Processor (DSP) is used to 1) power-modulate the laser diode, 2) sample the light intensity signal from the optical sensor, 3) perform synchronous demodulation on the intensity signal, and 4) pass the data to the host PC for further calculations. To perform the stated functions of the system both the analog electronics; which power the laser diode and pre-process the signals from the light intensity sensors, and the software; which performs the synchronous demodulation and power-modulate the laser diode, have been developed.

1.2 Signal Flow and Description of Variables

In this section, a brief overview of the system configuration and pertinent signals is provided. All subsequent discussions will be referenced to the diagrams and table in this section. Block diagrams of the signal flow are shown in Figure 1.1 and Figure 1.2. A description of the signals appears in Table 1.1.

Figure 1.1 is a block diagram of sensor system and DSP signal flow. The signals from the analog electronics are sampled at 10KHz by the A/D (analog to digital converter). Synchronous demodulation is then performed on the sampled signals to extract their amplitudes which correspond to the non-normalized intensity signals from the sensor modules. The average values of the amplitudes are stored in the external memory where it can be accessed by the host PC for further analysis.

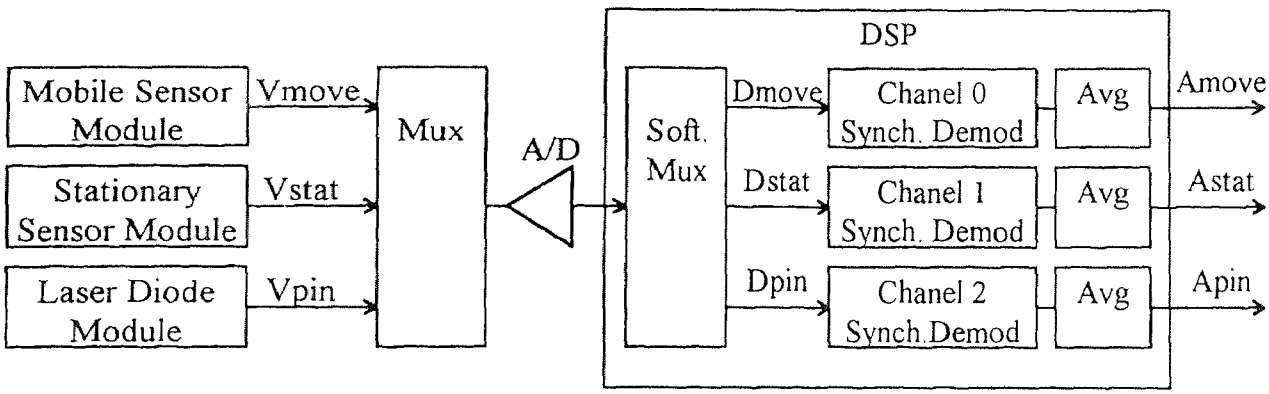


Figure 1.1 Block diagram of sensor system and DSP signal flow.

Figure 1.2 describes the signal flow in the host PC. The output data from the DSP are stored in dual ported external memory. Post processing of the sensor data, such as normalization and interpolation is performed by a C-program executed in the host PC.

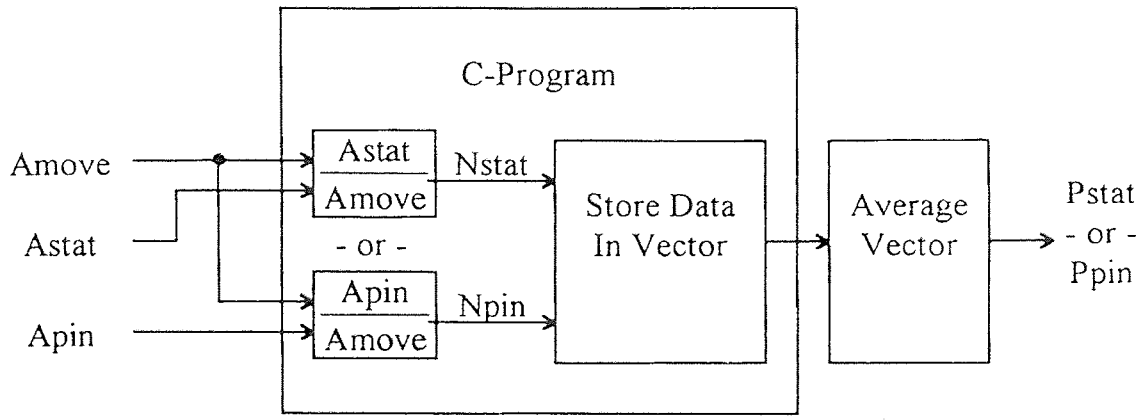


Figure 1.2 Block diagram of software signal flow in the host PC.

Table 1.1 Descriptions of Signals

Signal	Description
Vmove	The analog sine wave output from the mobile sensor module.
Vstat	The analog sine wave output from the stationary sensor module.
Vpin	The analog signal from the laser diode feedback circuit.
Dmove	The 10KHz sampled Vmove signal.
Dstat	The 10KHz sampled Vstat signal.
Dpin	The 10KHz sampled Vpin signal.
Amove	Average of the Non-normalized mobile sensor module's position.
Astat	Average of the Non-normalized stationary sensor module's position.
Apin	Average of the laser diode's output signal.
Nstat	Position signal normalized by the stationary sensor module.
Npin	Position signal normalized by the laser diode's feedback signal.
Pstat	Time average of the Nstat signal.
Ppin	Time average of the Npin signal.

1.3 Thesis Organization

The organization of this thesis is as follows: Chapter Two describes the hardware development such as the optical sensor modules, the laser-diode module, data sampling schemes, the one-dimensional testing setup and the application of the sensor module as the position sensor for a flexible beam experiment. Chapter Three describes the software development such as the DSP assembly language program which performs synchronous demodulation on the optical sensor output signals and the C-programs, which are executed on the host PC, which save the position data from the DSP and execute the control program for the flexible beam experiment. Chapter Four characterizes the performance of the sensor system. Finally, Chapter Five describes the application of the system to the flexible beam experiment.

CHAPTER 2

HARDWARE DEVELOPMENT

2.1 General Description

A general block diagram of the hardware is shown in Figure 2.1. A stationary, red laser diode, power-modulated by a 500Hz sine wave, illuminates a mobile light sensor module. The light sensor module consists of a TSL220 light sensor and analog electronics. The TMS320C25 Digital Signal Processor (DSP) performs the following tasks: 1) modulate the laser diode, 2) sample the signal from the sensor module, 3) demodulate the sensor signal, and 4) pass the results to the host PC for future signal processing.

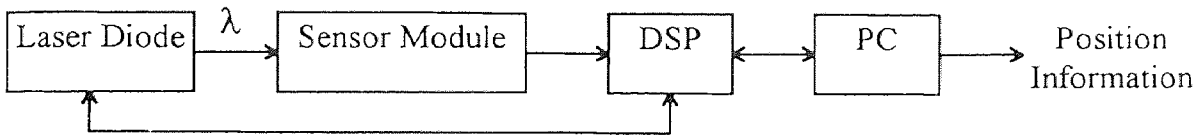


Figure 2.1 General hardware structure.

A detailed description of the hardware components are given in the following sections.

2.2 Light Sensor Module

The heart of the sensor module is the TSL220 light intensity to frequency converter chip from Texas Instruments. The TSL220 consists of a photo-diode and a current-to-frequency converter. The output voltage is a pulse train whose frequency is directly proportional to the incident light intensity on the photodiode. The output frequency range is determined by an external capacitor; so that the desired output frequency is adjustable

for a given intensity of light. The TSL220 circuit is shown in Figure 2.2. Figure 2.3 shows the output frequency of the TSL220 Vs it's external capacitor and Figure 2.4 shows the spectral response of the photodiode. Complete specifications of this chip are included in Appendix 1.

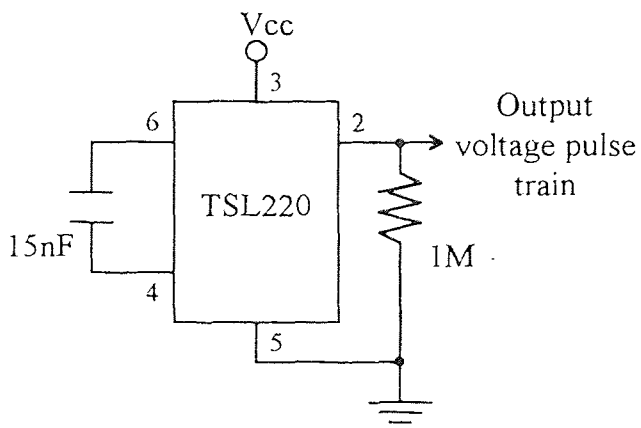


Figure 2.2 TSL220 circuit.

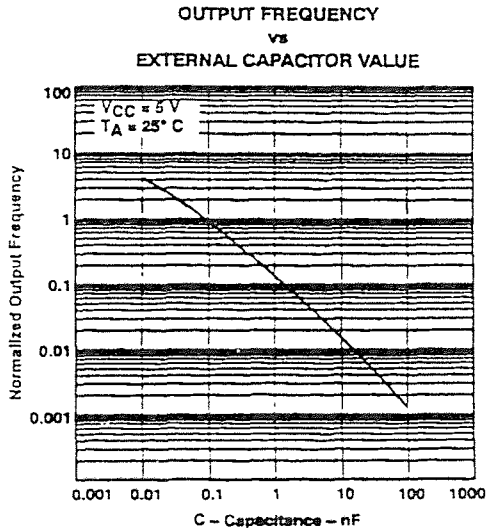


Figure 2.3 Output frequency of the TSL220 Vs the external capacitor.

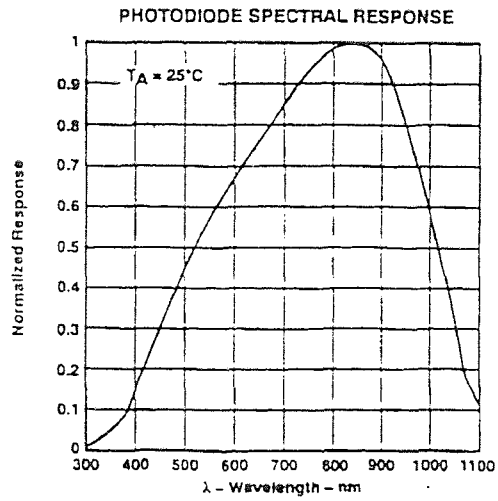


Figure 2.4 Photodiode spectral response.

2.3 Laser Diode Module

The light source which illuminates the light sensor modules is a Panasonic LN9705 laser diode. The LN9705 is a visible red GaAlAs laser diode with a nominal illumination wavelength of 670nm. Automatic power control is possible by utilizing a built-in pin photodiode to monitor light power.

The primary source of noise on the light sensor is the 120Hz overhead room lights. The frequency of oscillation of the laser diode was chosen to be 500Hz so that it is sufficiently far enough away from the frequency of the room lights for demodulating purposes.

The TMS320C25 Digital Signal Processor, which will be described in detail in the following sections, generates the signal to modulate the laser diode. The signal consists of

a 500Hz sine wave with an amplitude of 0.2 volts and a 2.6 volt DC bias. It should be noted that the drive signal should not exceed 3.0 volts, as indicated in the specification sheet, because the laser diode is very susceptible to over-voltage damage. Refer to Appendix 2 for the specifications of the laser diode.

The laser diode's power amplifier circuit consists of a set of unity gain inverters, implemented with LF353 op-amps, which buffer the DSP's D/A (Digital to Analog Converter). The current gain is provided by a unity gain amplifier implemented with an LM675 op-amp. A block diagram of the Laser Diode Module is shown in Figure 2.5. Figure 2.6 shows the schematic of the laser diode's power amplifier circuit. The calculation of the laser diode modulation voltage (V_laser) is described in Section 3.3.

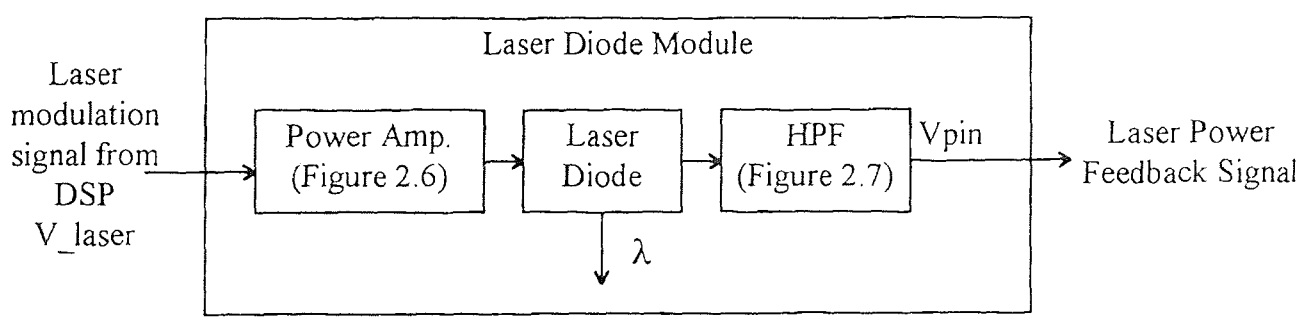


Figure 2.5 Block diagram of the laser diode power amplifier.

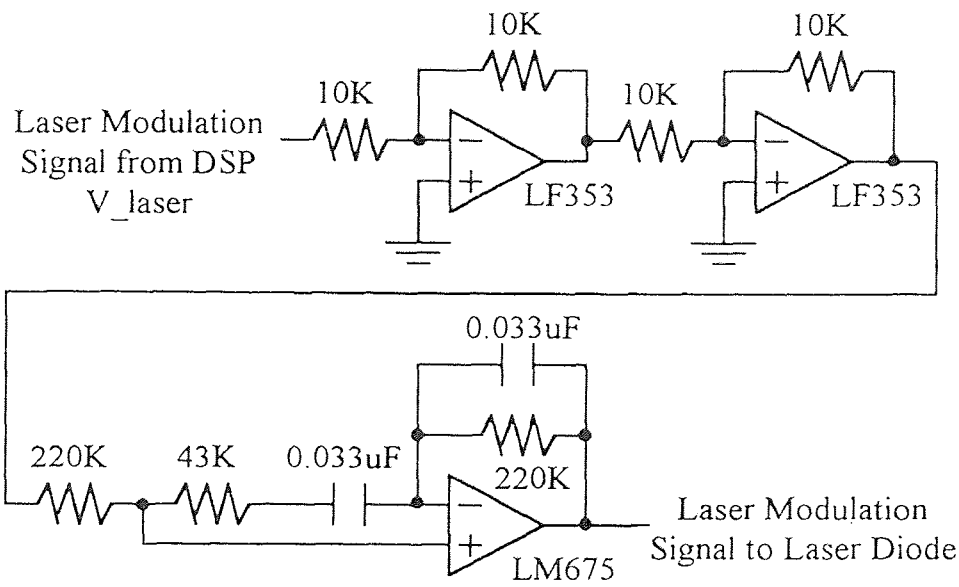


Figure 2.6 Schematic of the laser diode's power amplifier circuit.

The power monitoring feedback signal from the laser diode's built-in pin photodiode (Vpin) can be used to normalize the signal from the mobile sensor module, which will be discussed in detail in Section 3.4. The feedback signal from the laser diode is a sine wave with a DC bias whose voltage is directly proportional to the illumination power of the laser diode. A highpass filter is needed to remove the DC bias and a variable gain amplifier is synthesized to adjust the amplitude to match the +/- 5 volts range of the A/D. The result is a sine wave whose amplitude is directly proportional to the illumination power of the laser diode which is sampled at 10KHz by the DSP. Figure 2.7 shows the schematic of the first order highpass filter with a cutoff frequency of 50Hz and variable gain which filters and amplifies the laser diode's feedback signal so it can be sampled by the DSP's A/D (Analog to Digital converter).

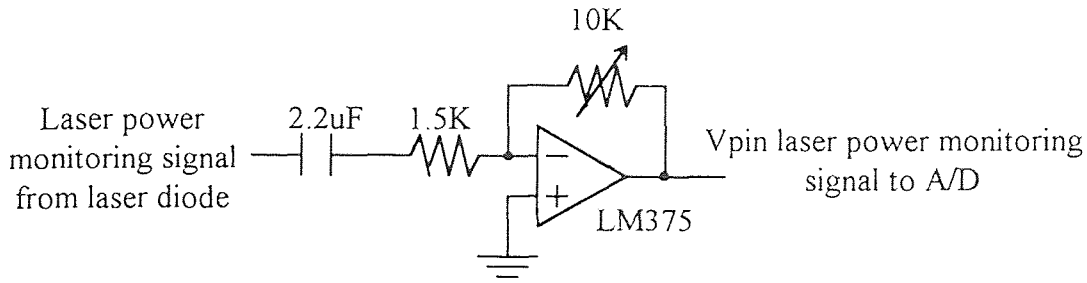


Figure 2.7 Schematic of the highpass filter with gain circuit.

2.4 TMS320C25 Digital Signal Processor

The TMS320C25 Digital Signal Processor (DSP) from Texas Instruments is used for sensor signal demodulation and laser modulation. It is located on a development plug-in card in the host PC. The DSP development card has the following peripherals: An internal timer with 0.1us resolution, eight analog input channels multiplexed to an A/D, two independent analog output channels, 128K words of dual-ported memory which is simultaneously accessible to both the DSP and the host PC. The DSP program is executed concurrently with the host PC. Therefore, a C-program can be executed in the host PC, in parallel with the DSP program, to manipulate the data in the DSP's memory. In the application stage, the DSP can be used in an embedded mode independent of a host PC to maximize efficiency.

2.5 Data Sampling Schemes

In order to acquire the sensor signal accurately and efficiently, two sampling approaches were investigated: Time Sampling and Analog Sampling. The Time Sampling approach uses the DSP's internal timer to clock the pulse stream generated by the TSL220. In the

Analog Sampling method the pulse stream from the TSL220 is first passed through a 500Hz analog bandpass filter (BPF) to extract the carrier's fundamental harmonic which is then sampled by the A/D. Figures 2.8 and 2.9 show block diagrams of the two sampling schemes investigated. These two approaches are now described.

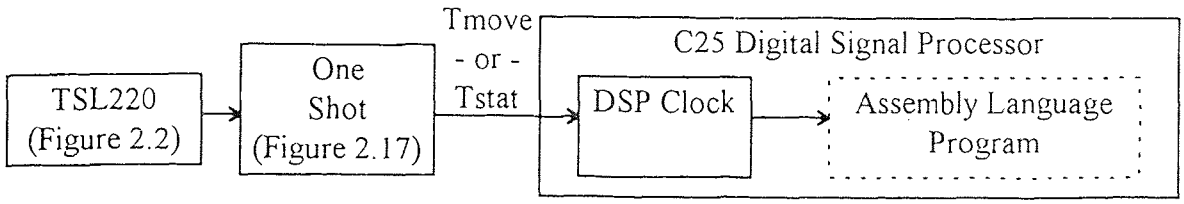


Figure 2.8 Block diagram of the Time Sampling scheme.

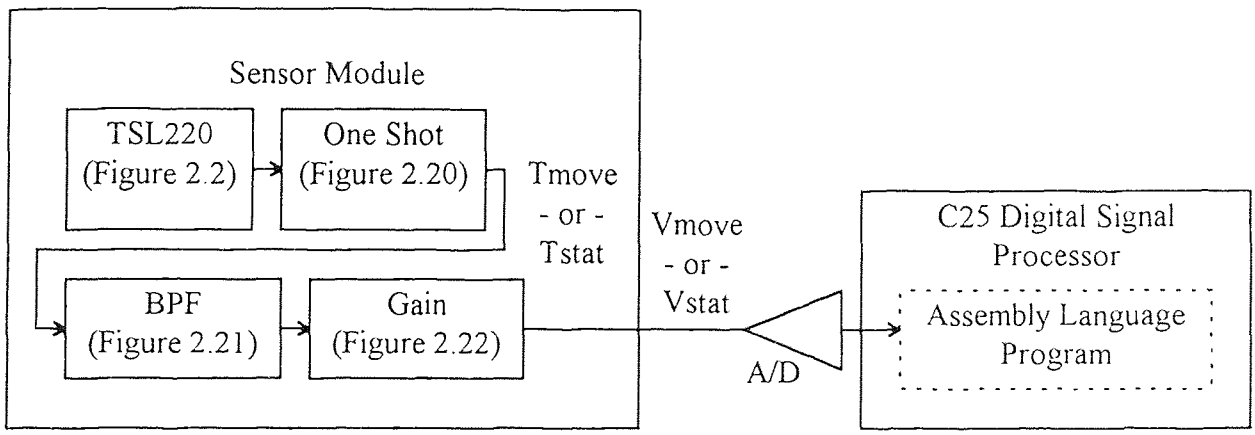


Figure 2.9 Block diagram of the Analog Sampling scheme.

2.5.1 Time Sampling Scheme

The first attempt to sample the TSL220 pulse stream was to use the internal timer of the DSP as a counter. A block diagram of the signal flow for the Time Sampling scheme is shown in Figure 2.10 and a description of the variables appears in Table 2.1

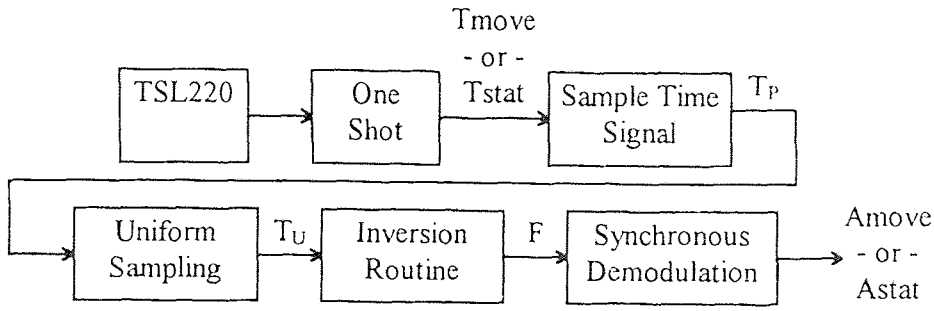


Figure 2.10 Signal flow in the Time Sampling scheme.

Table 2.1: Description of signals in the Time Sampling scheme.

Signal Name	Description
F	Output of Inversion Routine = the intensity of the incident light.
T_P	Non-uniform sampled time signal from TSL220
T_U	Uniformly sampled time signal from TSL220
Tmove	Output signal from the mobile TSL220
Tstat	Output signal from the stationary TSL220

The signal from the TSL220 sensor is fed into the DSP as an interrupt trigger. The interrupt routine determines the period of the incoming interrupts by performing the following operations: 1) Save the previous timer value, 2) Save the current timer value, 3) Reset the timer to its maximum value, and 4) Take the difference from the current and pervious timer values and save in memory as the time between the two previous pulses (T_P). The resultant value (T_P) would represent the number of clock ticks between two successive TSL220 pulses. Since the frequency of the DSP's internal timer is 10MHz, a timing resolution of 0.1uS can be achieved. A flow chart of this scheme is shown in Figure 2.11.

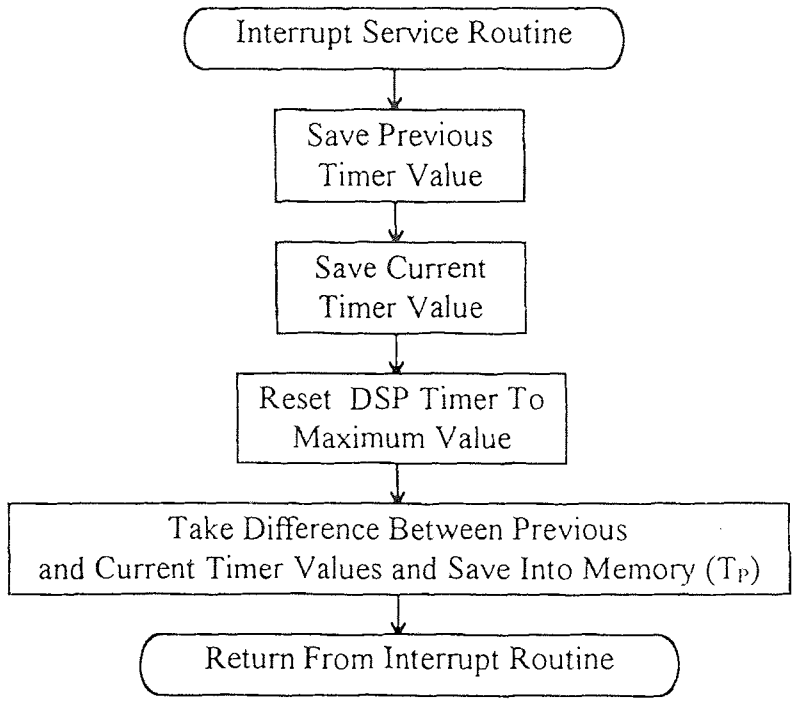


Figure 2.11 Flow chart of the Time Sampling scheme.

There are several problems associated with this intuitively appealing scheme. The first problem is non-uniform time sampling: the frequency of the TSL220 pulse train is continually changing due to the modulation of the laser diode and background fluctuations. This problem can be solved by uniformly sampling the timer data (T_P) at a constant sampling rate. To accomplish this, another lower priority interrupt service routine, operating at 10KHz, is used for the uniform sampling routine. Whenever the lower priority interrupt routine is triggered, the program copies the values of T_P saved in memory and saves it as the uniform sampled time value (T_U). A flow chart of the uniform sampling routine is shown in Figure 2.12.

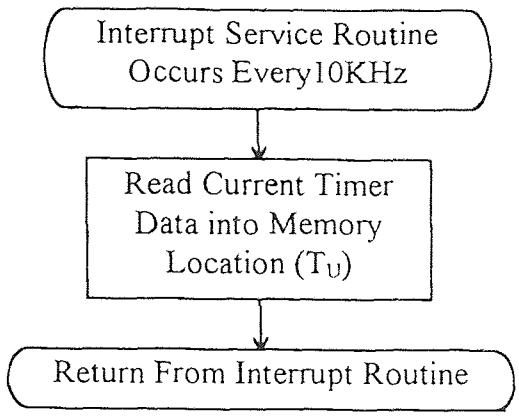


Figure 2.12 Flow chart of the Uniform Sampling scheme.

The second problem with using the TSL220 signal as the sampling trigger is that the DSP interrupts are both level and edge sensitive. The typical duration of a TSL220 pulse is approximately 3us whereas the cycle time of the interrupt service routine which saves the time value (T_P), is 2us. Now since the interrupt is level sensitive, it will immediately re-trigger upon returning from the interrupt service routine if the TSL220 signal is still low, resulting in the incorrect timer value being saved. Figure 2.13 shows the timing sequence of the TSL220 interrupt trigger signal and the resulting sampling.

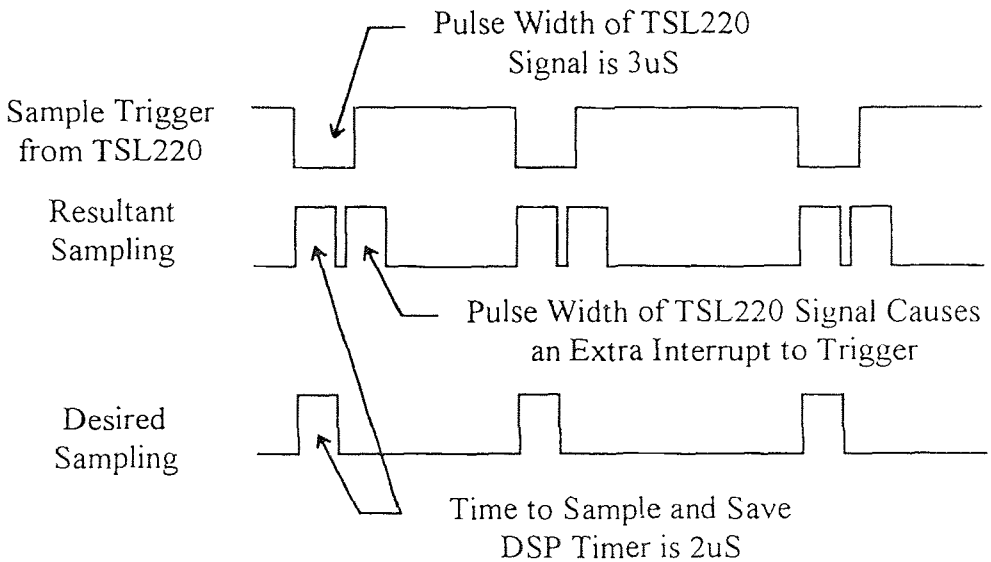


Figure 2.13 Timing sequence of the TSL220 interrupt trigger and the resultant sampling.

The output waveform specifications are shown in Figure 2.14 and the output pulse duration Vs the external capacitor is shown in Figure 2.15. Refer to Appendices 3 and 4 for the assembly codes for the Time Sampling and Uniform Sampling programs.

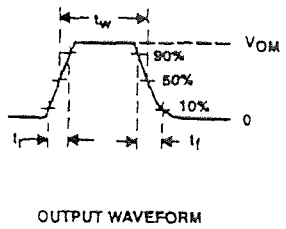


Figure 2.14 The output waveform specifications of the TSL220.

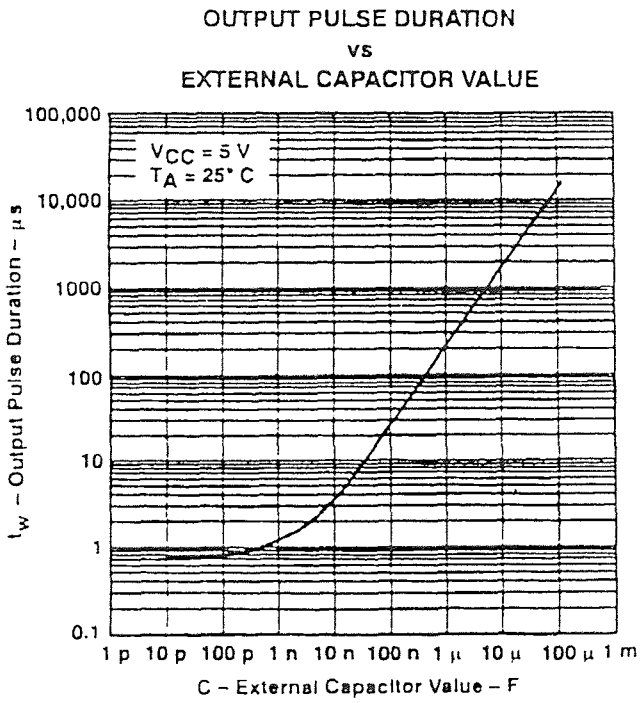


Figure 2.15 Output pulse duration Vs the external capacitor.

The solution to this problem is to shorten the pulse width of the TSL220 signal with a Mono-Stable-Multi-Vibrator (One-Shot). The circuit diagram for the One-Shot is shown in Figure 2.16. The One-Shot shortens the pulse-width of the TSL220 from 3uS to approximately 35nS. Since this is much shorter than the time required for the interrupt service routine to sample and save the DSP timer value, the interrupt correctly triggers only once for every pulse from the TSL220.

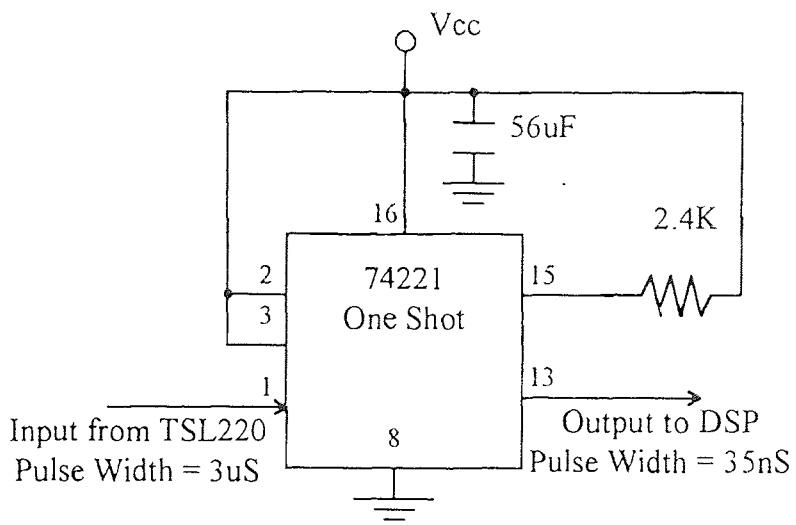


Figure 2.16 Pulse shortening One Shot circuit.

A third problem associated with the Time Sampling scheme is that the sampled data is inversely proportional to the intensity of the incident light on the sensor module. The inverse of the intensity cannot be directly used in the synchronous demodulation process due to the presence of a residue. Specifically, if the inverse of the intensity is used, which is given by $Vin = 1 / [A \sin(\omega t)]$, then an output residue in the form of $\cot(\omega t)$ is created by the demodulation process shown in the equations below (2.5-1) to (2.5-3). The resulting output of the demodulation process will be incorrect because the lowpass filter in

the demodulation process will not properly filter the cot(ωt) term shown in equation (2.5-3). The entire correct demodulation process is described in detail in Chapter 3.

$$Vin = 1 / [A \sin(\omega t)] \tag{2.5-1}$$

$$Vc = \cos(\omega t) / [A \sin(\omega t)] \tag{2.5-2}$$

$$= \cot(\omega t) / A \tag{2.5-3}$$

The solution here is to invert the time signal to get an intensity signal which can be fed into the synchronous demodulation scheme.

This solution method is quite time consuming as the DSP is a fixed point processor and an inversion program is needed in the interrupt service routine to invert the uniformly sampled signal (T_U). The inversion process iteratively calculates an integer frequency value (F) which, when multiplied with the integer time value (T_U) approximately equals a preset target bound (2^N) (2.5-4). The iterative process continues until the error (I_E) given by (2.5-5) is less than a prespecified tolerance (2^n).

$$T_U * F = 2^N \tag{2.5-4}$$

$$I_E = 2^N - T_U * F \tag{2.5-5}$$

The main constraint on the inversion scheme is that it must be completed in less that 12us because the demodulation routine consumes about 13us out of the 25us total allocated loop time. It was determined by experiment that approximately 20 iterations of the inversion loop are required to properly calculate the frequency value (F). This would require 60us, which is several times longer than the available quantum. Table 2.2 gives a description of the variables involved in the inversion loop. The flow chart for the inversion

loop is shown in Figure 2.17 and a comparison of the non-uniform and uniform sampled signals is shown in Figure 2.18.

Table 2.2 Description of the variables involved in the inversion loop.

Variable	Description
I_E	Error at the ith step
F	Frequency value at the ith step
K	Error gain
T_U	Uniformly sampled time value at the ith step
N_m	Maximum number of iterations
2^N	Preset target bound
2^n	Maximum error tolerance

The Time Sampling Scheme was not pursued further due to the excessive time required to resample and invert the signal. In order to increase the available time to properly implement the inversion routine, the sampling frequency would have to be decreased form the desired 10KHz to 3KHz which is too low to properly process the 500Hz laser signal (Sampling at 3KHz will cause aliasing of the higher order harmonics during demodulation).

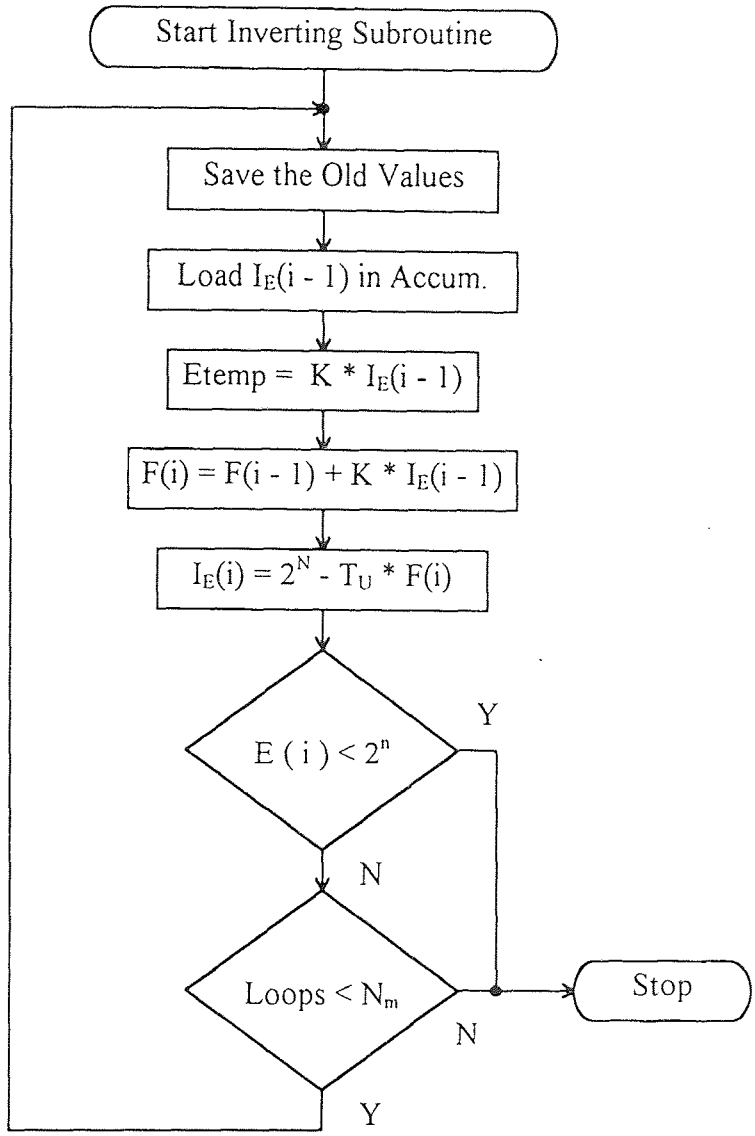


Figure 2.17 Flow chart of the Inversion Loop.

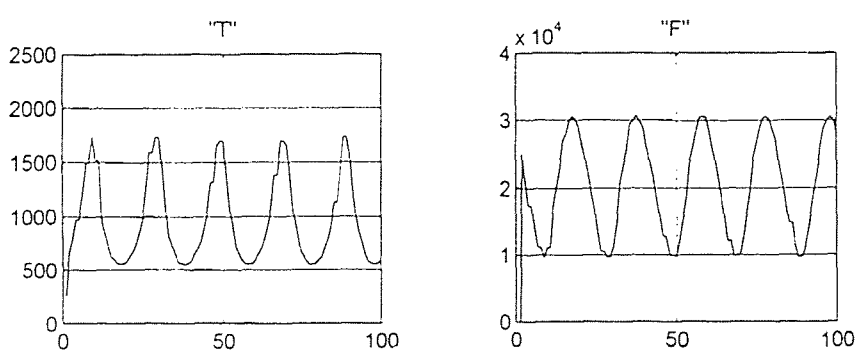


Figure 2.18 Comparison of the non-uniform and uniform sampled signals.

2.5.2 Analog Sampling Scheme

The sampling scheme chosen to replace the Time Sampling scheme was the Analog Sampling scheme. In the Analog Sampling scheme, the TSL220 pulse stream is first converted into an analog sine wave by passing the pulse stream through a one-shot, followed by a 500Hz bandpass filter. Since the amplitude of the resulting sine wave is directly proportional to the intensity of the light illumination on the TSL220's photo diode, no inversion is necessary.

In the Analog Sampling scheme, a one-shot is used to lengthen the pulse width from 3us to approximately 30us so there is sufficient energy in the signal to excite the bandpass filter circuit. The output pulse width of the one-shot is chosen to be approximately 50% of the duty cycle when the laser diode is at maximum brightness. The circuit diagram of the pulse-widening circuit is shown in Figure 2.19. The 10K trim-pot controls the width of the output pulse to prevent over running of adjacent pulses.

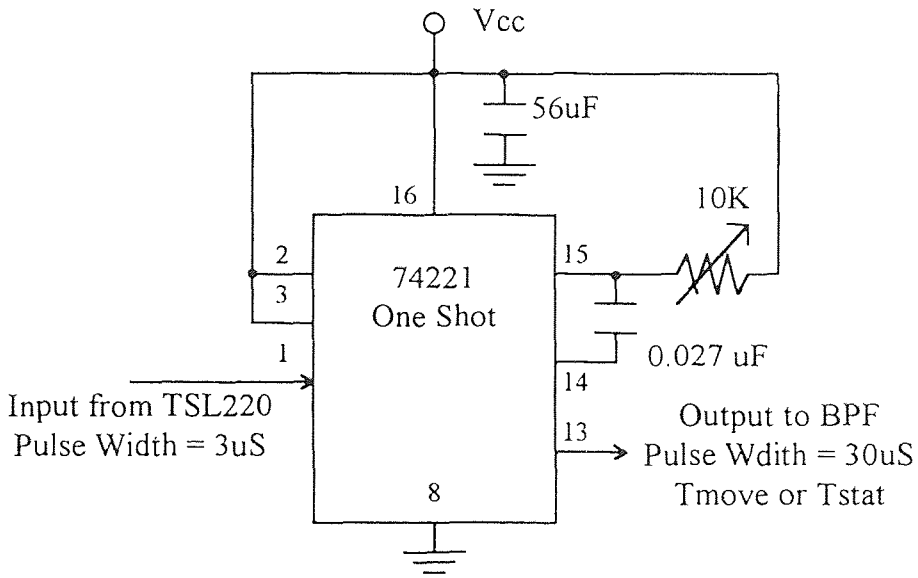


Figure 2.19 Pulse widening One-Shot circuit.

The output of the One Shot (Tstat or Tmove) is then fed into the analog state-variable bandpass filter circuit, shown in Figure 2.20. The bandpass filter has a center frequency of 500Hz and a Q of 50 which produces the 500Hz sine wave by filtering out all but the fundamental frequency of 500Hz. The output of the bandpass filter is then passed through a variable gain circuit so the sensor signal output range is matched to the +/- 5 volt range of the A/D. The output of the gain circuit is represented by Vmove or Vstat. The schematic of the gain circuit is shown in Figure 2.21 and a block diagram of the final sampling scheme is shown in Figure 2.22.

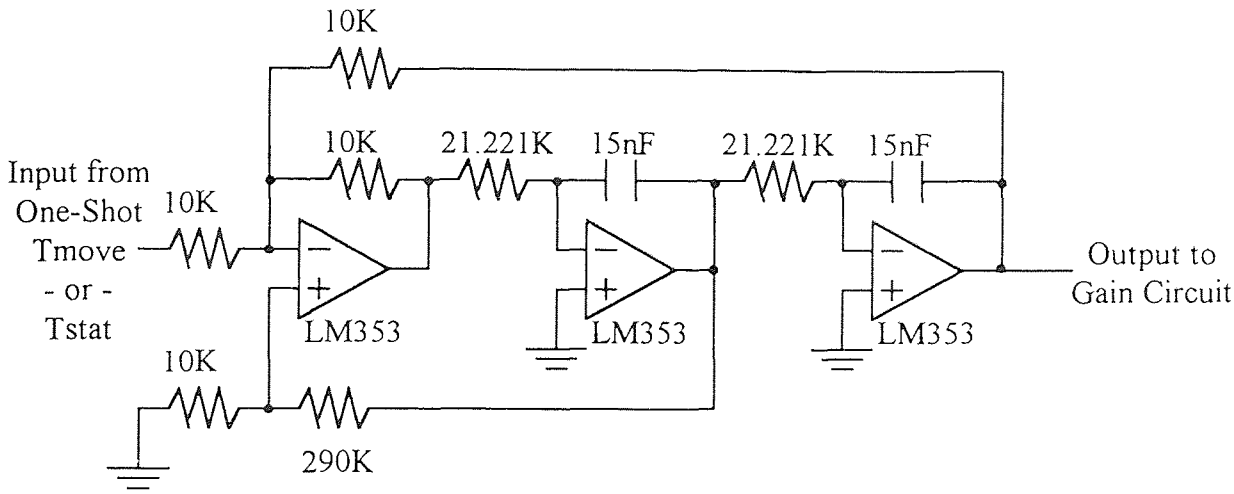


Figure 2.20 Schematic of the State Variable bandpass filter circuit.

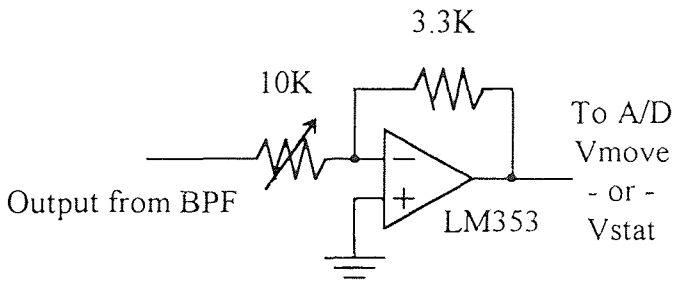


Figure 2.21 Schematic of the Gain circuit.

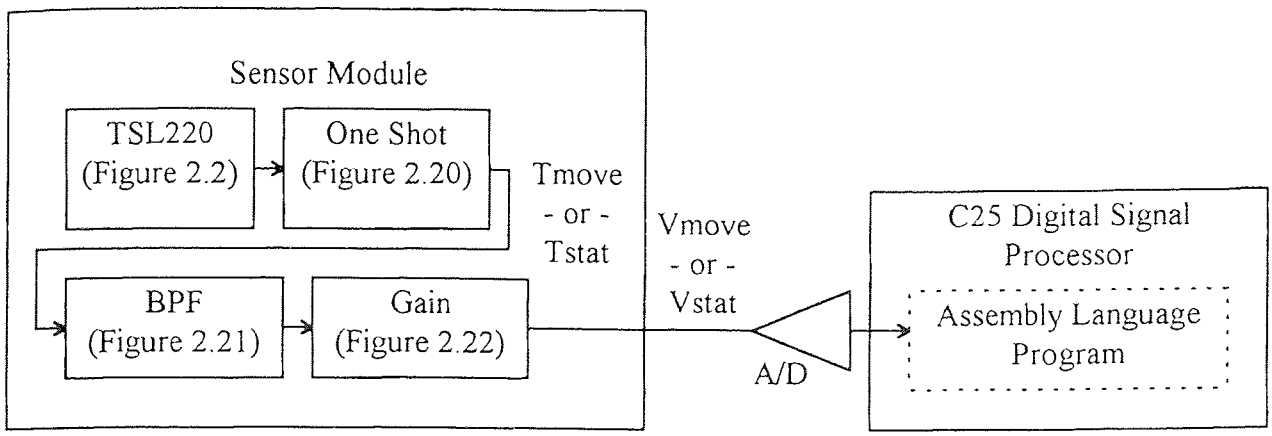


Figure 2.22 Block diagram of the Analog Sampling scheme.

2.5.3 Comparison of Sampling Techniques

Both sampling schemes investigated have relative merits over the other. The main advantage of the Time Sampling scheme is that it samples the time between successive pulses from the TSL220 very accurately with no external analog circuits. However, this method has two disadvantages, non-uniform sampling and signal inversion, which can only be solved by implementing time consuming assembly codes.

The Analog Sampling scheme avoids the above difficulties by using analog circuits to process the data from the TSL220. The analog signal is then sampled by the DSP with the A/D. The main disadvantages of this method are the instability of the analog electronics and the 12-bit resolution of the A/D which limits the resolution of the sensor.

2.6 One-Dimensional Testing Setup Description

A block diagram of the setup that was used to test the sensor system is shown in Figure 2.23.

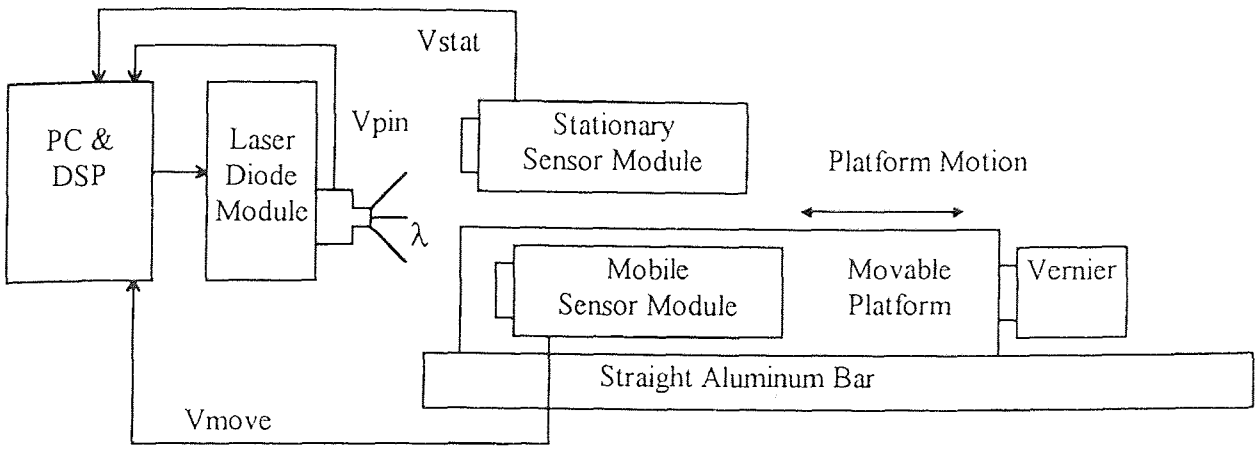


Figure 2.23 Block diagram of the one-dimensional testing setup.

The laser diode, modulated at 500 Hz by the DSP, illuminates two TSL220 sensor modules. One module is setup on a stationary mount which is used to monitor the intensity of the laser diode. The second module is mounted on a platform that can be moved towards and away from the laser diode for either short range or long range tests. For the short range test, the platform can be moved over a range of 25mm with a repeatability of approximately 0.02mm. For the long range test, the platform can be moved over a range of approximately 600mm with a repeatability of approximately 2mm. The purpose of the stationary sensor module is to monitor the laser diode intensity so the signal from the mobile sensor module can be compensated for the drifts in laser.

2.7 Flexible Beam Testing Setup Description

Besides the one-dimension test, the sensor system is also applied to a flexible beam experiment. A sensor module is mounted at the end of a flexible beam which is a type 304

stainless steel beam, approximately 1500 mm long, 25 mm wide, and 6 mm thick. The position of the flexible beam is controlled by a brushless DC motor which is mounted in a vertical position on a platform. The sensor module is illuminated by the laser diode as in the previous test setup.

Refer to Figure 1.1, Figure 1.2 and Table 1.1 for a description of the signals. The DSP program samples the electrical output of the sensor module: Vmove from the mobile sensor module and Vpin from the laser diode. It then calculates Amove and Apin, the amplitudes of the sampled sine waves which represent the non-normalized intensity of the incident light on the mobile sensor module and the output power of the laser diode, respectively. The C-program fetches Amove and Apin from the DSP and calculates the normalized intensity Npin, as $\frac{\text{Apin}}{\text{Amove}}$, which is used as the position signal in the control law. The control signal is finally sent to the power amplifier and the DC motor to generate the proper corrective action. Refer to Appendix 5 for the specifications of the DC motor. Chapter 3 describes the assembly language and C-programs in detail. A block diagram of the flexible beam test setup is shown in Figure 2.24.

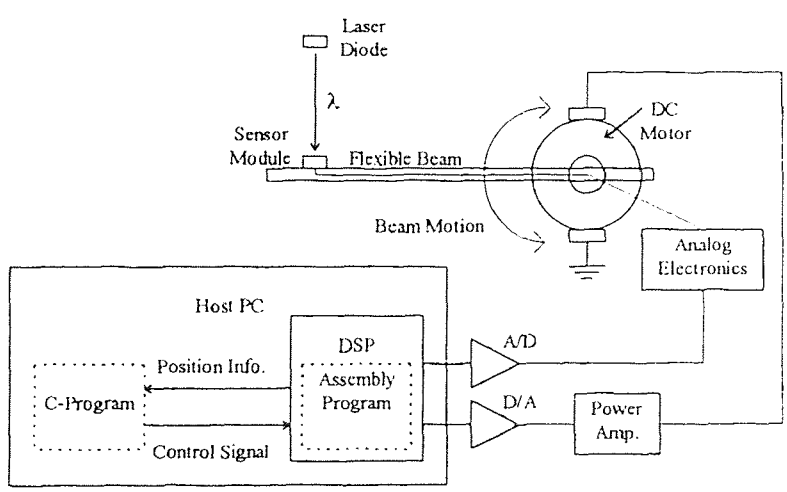


Figure 2.24 Block diagram of the flexible beam setup.

CHAPTER 3

SOFTWARE DEVELOPMENT

3.1 General Description

A block diagram of the general software structure is shown in Figure 3.1. The C25 Digital Signal Processor (DSP) executes the sensor signal demodulation program while synchronously pulsing the laser diode. The PC provides the interface between the user and the DSP by executing the DSP's debugger or a C-program which can access the information in the DSP's external data memory area.

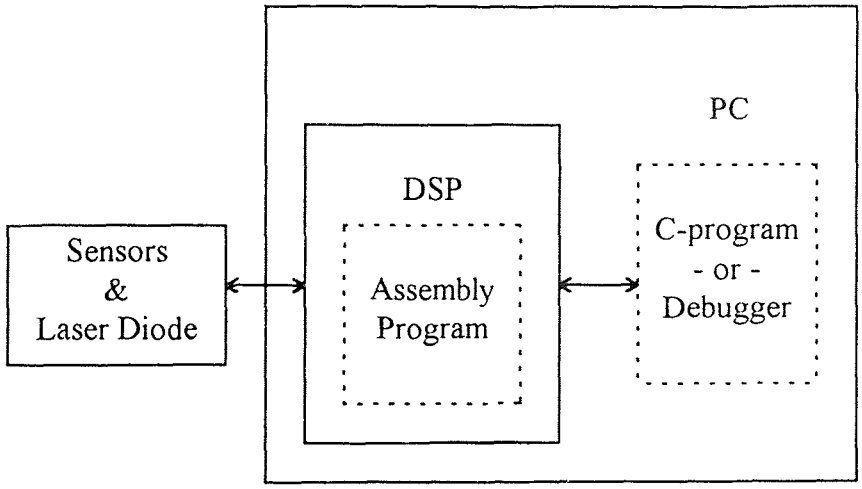


Figure 3.1 Block diagram of the general software structure.

3.1.1 C25 Digital Signal Processor Software

The main assembly language program in the DSP is responsible for modulating the laser diode intensity and demodulating the TSL220 light sensor signals. It is written in Texas Instruments C2x assembly language. A listings of the assembly codes and the link

programs are given in Appendix 6 and 7, respectively. A simplified flow chart of the program's operation is shown in Figure 3.2.

Upon reset, the program sets up all of the necessary initialization and house keeping tasks such as: define and label memory locations; define the interrupt vector table; enable the proper interrupts; set the A/D sampling frequency; set up the sine table and sine table pointers; define the digital filter coefficients; and define the laser modulation voltages.

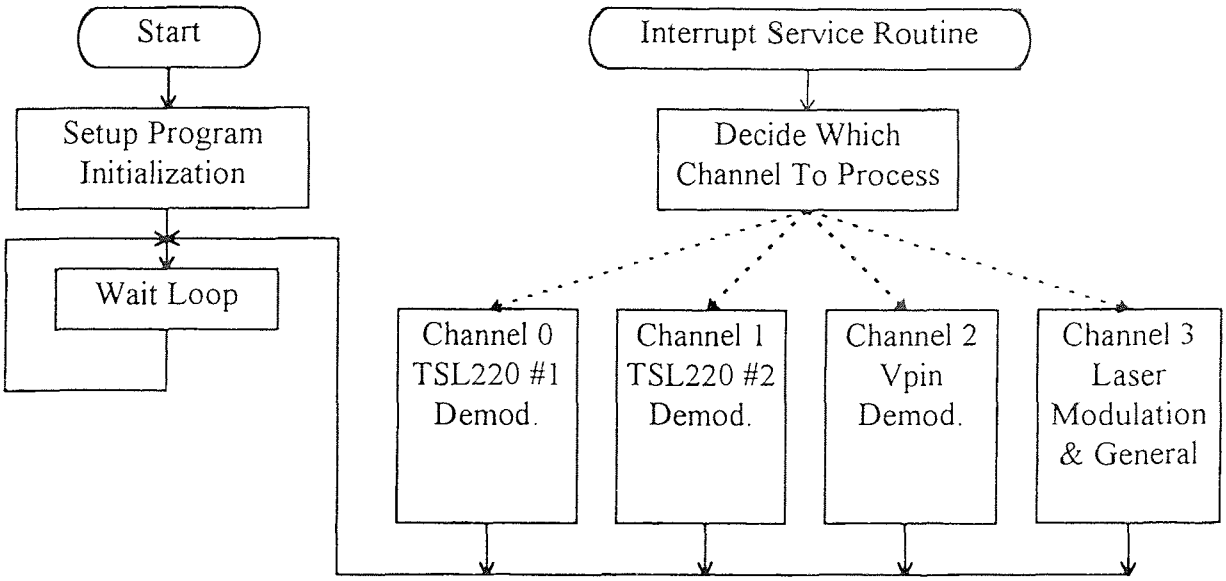


Figure 3.2 Simplified flow chart of the assembly language program.

All of the signal processing takes place in the interrupt service routine. After initialization, the DSP enters into a loop and waits for the interrupt trigger to occur. The interrupt trigger originates from the $\overline{\text{READY}}$ pin of the A/D. Its arrival signals the completion of the conversion process whose conversion rate is governed by an external programmable timer mapped to port zero of the DSP.

There are four channels of program code. Each channel is executed, in succession, at 10KHz. Only one channel can be executed during any one interrupt because the A/D can only be sampled one time during each interrupt. Therefore, the external timer must be set to 40KHz. This sampling rate provides each channel with 25us quantum of loop time.

The first task of the interrupt service routine is to determine which channel to process during the current interrupt. A description of the responsibility of each channel is as follows: Channel Zero and Channel One are responsible for the synchronous demodulation of their respective sensor module signals (Vmove and Vstat). Vmove is the output signal from the mobile sensor module and Vstat is the output from the stationary sensor module. Channel Two is responsible for the synchronous demodulation of Vpin which is the laser power feedback signal from the laser diode's built-in photodiode monitor. Vpin is passed through a highpass filter and a gain circuit to adjust it to properly fill the A/D's +/- 5volt range. Finally, Channel Three is responsible for the modulation of the laser diode and communicating with the host PC.

Functionally, the Channel Zero, One, and Two programs are identical. The only difference is that they sample and perform synchronous demodulation on different input signals. A flow chart for the Channel Zero, One and Two Programs is shown in Figure 3.3.

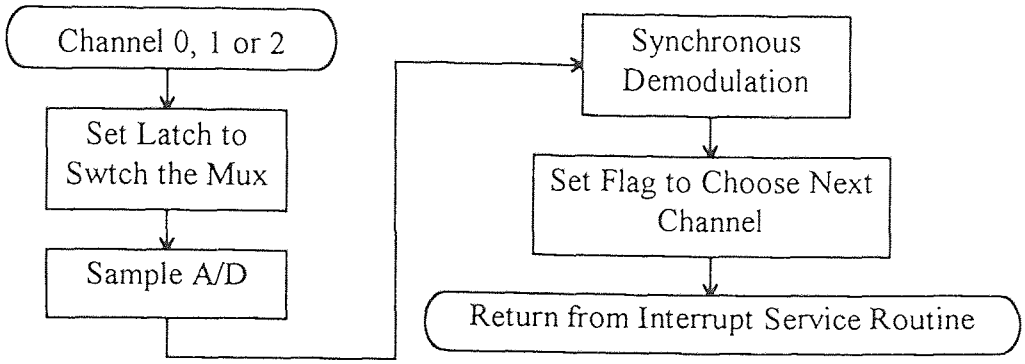


Figure 3.3 Flow chart of Channels Zero, One & Two.

The first operation of each channel is to set the input Multiplexer (Mux) to the next input channel. The Channel Zero program and the Channel One program read in the sampled signals, Vmove and Vstat, via the A/D from the mobile and the stationary sensor modules respectively. The Channel Two program, on the other hand, reads in Vpin, the sampled feedback signal form the laser diode. A block diagram of the A/D's data acquisition system is shown in Figure 3.4.

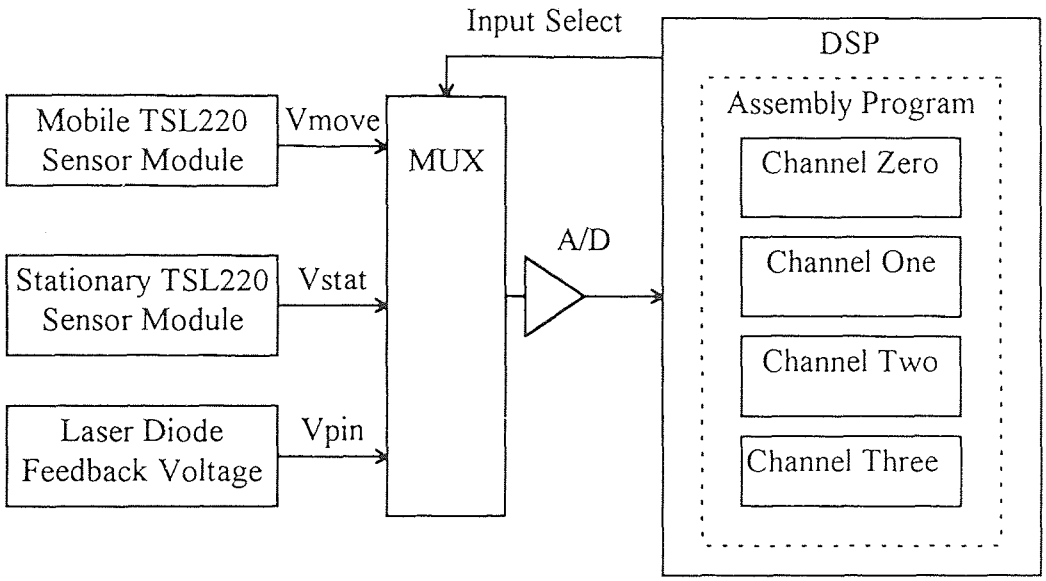


Figure 3.4 Block diagram of the A/D's.

After sampling, synchronous demodulation is performed on each signal. The result is then stored in a memory location where it can be fetched by a C-program running on the host PC and stored on disk. The specific memory locations are described in Section 3.3 and tabulated in Table 3.4. A detailed description of the synchronous demodulation process is given in Section 3.2.

3.1.2 PC Software

The interface between the user and the DSP is the host PC which executes either the DSP's debugger or a C-program. The debugger loads and executes the assembly program and can monitor the external memory of the DSP. The C-program can read from and write to the external memory of the DSP. This dual-porter scheme is advantageous because it is easier to perform complex mathematical calculations concurrently with a PC rather than with the fixed-point DSP. A block diagram of the PC interface is shown in Figure 3.5.

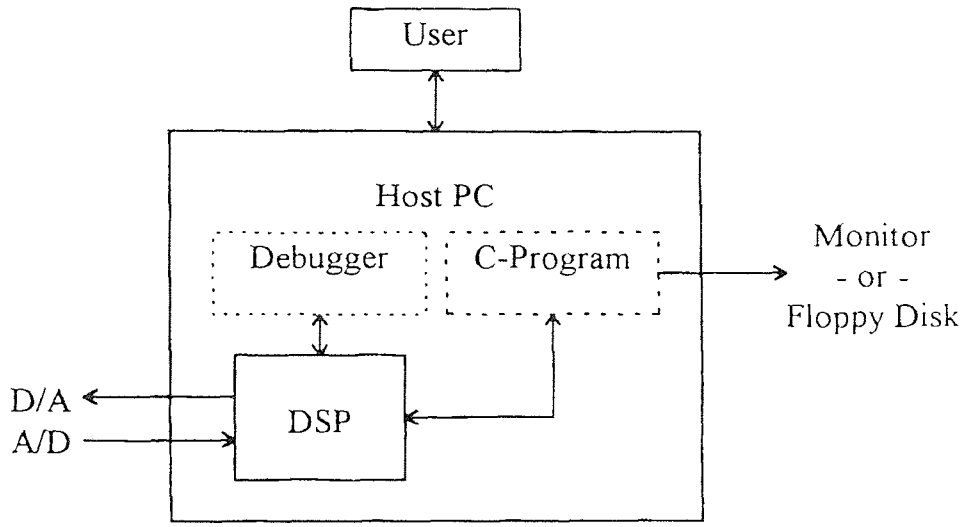


Figure 3.5 General block diagram of the PC interface.

3.2 Synchronous Demodulation Implementation

The demodulation process converts the input signal D_in = B + A sin ($\omega t + \Phi$) to A_out = $\frac{A^2}{4}$ where D_in represents the three sampled signals from the A/D (Dmove, Dstat, and Dpin). Refer to Section 1.2 for a review of the flow of the signals and their descriptions. The amplitude of the input signal is represented by A, the DC bias is represented by B, and the phase shift is represented by Φ. A_out represents the output signals from the three demodulation channels (Amove, Astat, and Apin).

There are five major groups of components in this demodulation implementation: a bandpass filter, two multipliers, two lowpass filters, two elliptic filters, two squarers, and a summer. A flow chart of the synchronous demodulation scheme is shown in Figure 3.6 and a description of the symbols used appears in Table 3.1. Each component will be described in detail in the following sections. Refer to Figure 3.6 and Table 3.1 for a description of the variables through out this chapter.

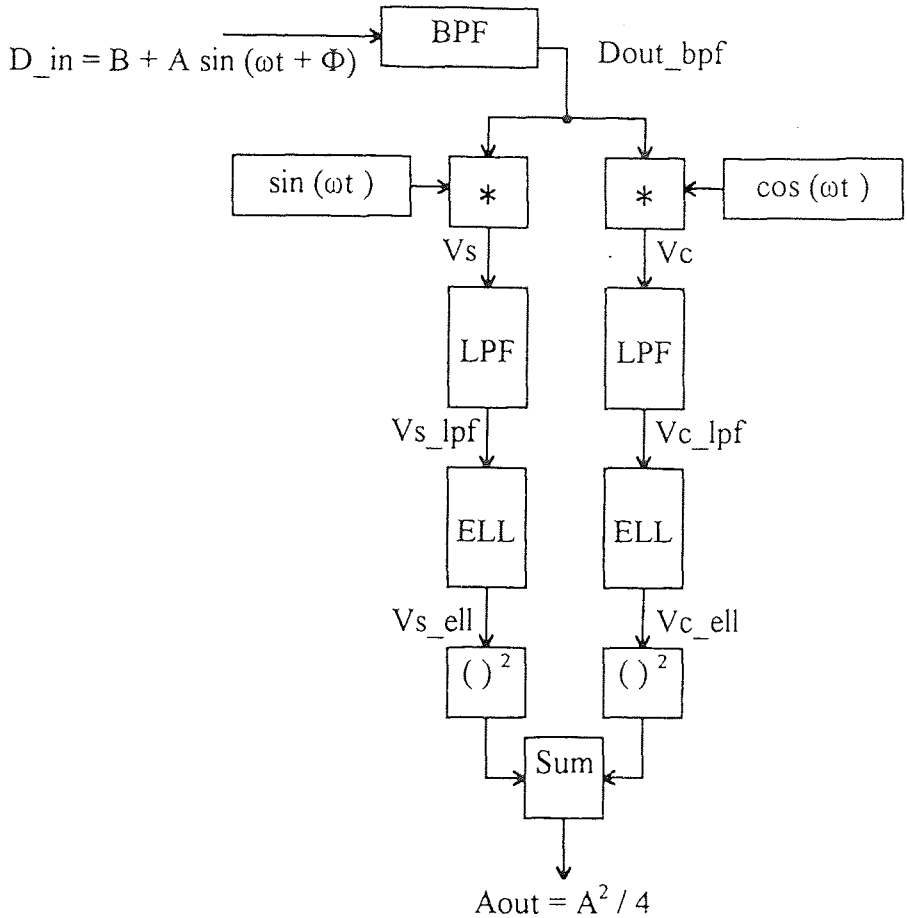


Figure 3.6 Flow chart of synchronous demodulation scheme.

Table 3.1 Description of symbols used in the synchronous demodulation scheme.

Variable	Description
A	Amplitude of the input sine wave D_in
B	DC Bias of the input sine wave D_in
C	DC leakage of the output of the bandpass filter Dout_bpf
Φ	Phase shift of the input sine wave D_in
S	Digital filter scaling factor
D_in	Signal from sensor module or laser feedback sampled at 10KHz
Dout_bpf	Output of bandpass filter
Vs	Output of bandpass filter multiplied by sin(ωt)
Vc	Output of bandpass filter multiplied by cos(ωt)
Vin_lpf	General input of the lowpass filter (Vs or Vc)
Vout_lpf	General output of the lowpass (Vs_lpf or Vc_lpf)
Vs_lpf	Output of lowpass filter in sin(ωt) path
Vc_lpf	Output of lowpass filter in cos(ωt) path
Vin_ell	General input of the elliptic filter (Vs_lpf or Vs_lpf)
Vin_ell	General output of the elliptic (Vs_ell or Vc_ell)
Vs_ell	Output of elliptical filter in sin(ωt) path
Vc_ell	Output of elliptical filter in cos(ωt) path
A_out	The sum of the squares of the outputs of the elliptical filters

3.2.1 Bandpass Filter

The primary function of the bandpass filter is to remove any DC bias on the input signal and to attenuate, if present, any higher harmonics of the 500Hz carrier. The bandpass filter is a second order filter with a center frequency of 500Hz and a bandwidth of 20Hz (Q = 25). Equations (3.2-1) and (3.2-2) describe the operation of the bandpass filter on the input to the demodulation sequence. The need for attenuating the DC component of D_in will be justified in the next section. Dout_bpf represents the output of the bandpass filter.

$$D\_in = B + A \sin(\omega t + \Phi) + \text{higher order harmonics} \tag{3.2-1}$$

$$Dout\_bpf = A \sin(\omega t + \Phi) \tag{3.2-2}$$

Since the DSP is a fixed-point processor, the filter coefficients must be represented by scaled integers. The scaling factor is then removed at the output of the digital filter by division. For convenience, the scaling factor is always chosen to be a value represented by 2^S , where S is an integer, which can be easily divided out by shifting the value in the accumulator. The larger the scaling factor that can be used, the closer the digital filter realization will be to the theoretical digital filter. The scaling factor implemented for all digital filters in the C2x assembly language program is $2^{14} = 16384$.

The transfer function of the bandpass filter is shown in (3.2-3) and the difference equation is shown in (3.2-4). The frequency response of the digital bandpass filter is shown in Figure 3.7.

$$\frac{Dout\_bpf(z)}{D\_in(z)} = \frac{0.006244 - 0.006244\,Z^{-2}}{1 - 1.89027\,Z^{-1} + 0.987511\,Z^{-2}} \tag{3.2-3}$$

$$\begin{aligned} Dout\_bpf(i) = {} & [30970\ Dout\_bpf(i-1) - 16179\ Dout\_bpf(i-2) \\ & + 102\ D\_in(i) - 102\ D\_in(i-2)] / 16384 \end{aligned} \tag{3.2-4}$$

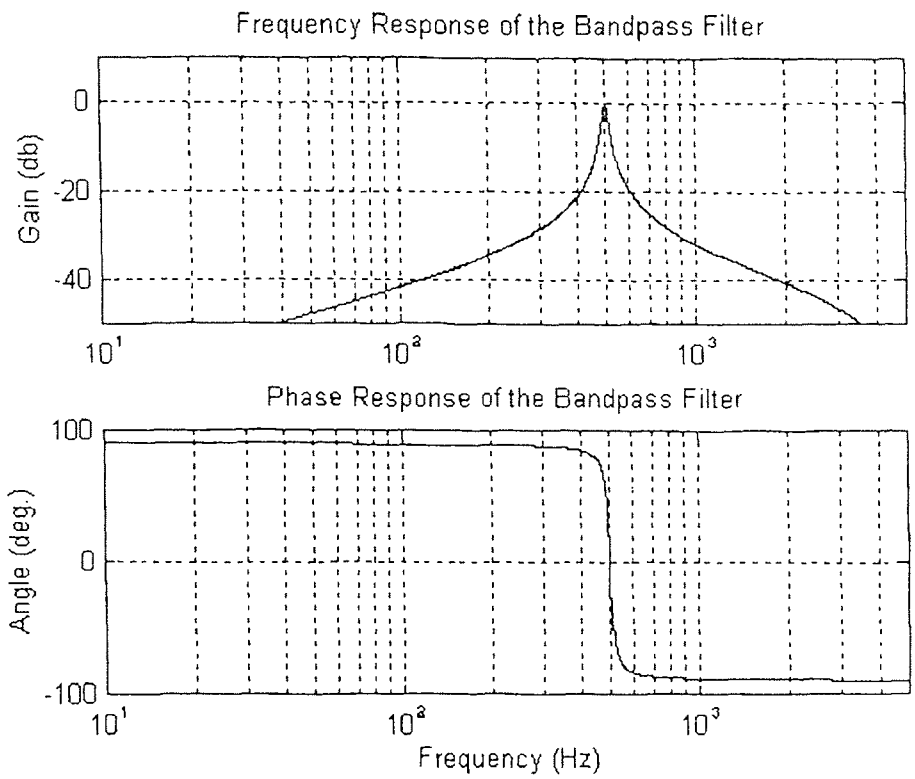


Figure 3.7 Frequency and phase response of the bandpass filter.

3.2.2 Multipliers

The multipliers demodulate the output of the bandpass filter signal Dout_bpf (3.2-2) by multiplying it by cos(ωt) and sin(ωt). The resulting signals consist of a DC component, which is directly proportional to the amplitude and phase shift of the input signal, and the

second harmonic of the carrier frequency (1 KHz). Vs and Vc represent the outputs of the sin(ωt) and cos(ωt) multipliers, respectively. Equations (3.2-7) and (3.2-10) show the result of multiplying Dout_bpf with cos(ωt) and sin(ωt) respectively.

$$Vc = \cos(\omega t)\, Dout\_bpf \tag{3.2-5}$$

$$= A \cos(\omega t) \sin(\omega t + \Phi) \tag{3.2-6}$$

$$= \frac{A}{2} [\, \sin(2\omega t) + \sin \Phi \,] \tag{3.2-7}$$

$$Vs = \sin(\omega t)\, Dout\_bpf \tag{3.2-8}$$

$$= A \sin(\omega t) \sin(\omega t + \Phi) \tag{3.2-9}$$

$$= \frac{A}{2} [\, \cos \Phi - \cos(2\omega t + \Phi) \,] \tag{3.2-10}$$

It is important that the DC component of D_in be eliminated. If there is any DC leakage through the bandpass filter it will cause the primary frequency (500 Hz) to be passed through the synchronous demodulation scheme. To illustrate this effect let the output of the bandpass filter, where the DC component has not been completely attenuated, be given by:

$$C + A \sin(\omega t + \Phi) \tag{3.2-11}$$

The DC component through the bandpass filter is represented by C. When multiplied by the demodulating signals, (3.2-11) produces a carrier frequency component (500 Hz) in addition to the desired double carrier frequency (1 KHz) and DC component,

(3.2-13) and (3.2-15). Compare (3.2-7) and (3.2-10) when the DC bias has been properly removed to (3.2-13) and (3.2-15), respectively.

$$Vc = \cos(\omega t)\,[\,C + A\sin(\omega t + \Phi)\,] \tag{3.2-12}$$

$$= C\cos(\omega t) + \frac{A}{2}\sin\Phi + \frac{A}{2}\sin(2\omega t + \Phi) \tag{3.2-13}$$

$$Vs = \sin(\omega t)\,[\,C + A\sin(\omega t + \Phi)\,] \tag{3.2-14}$$

$$= C\sin(\omega t) + \frac{A}{2}\cos\Phi - \frac{A}{2}\cos(2\omega t + \Phi) \tag{3.2-15}$$

Figure 3.8 is an example of Vc or Vs if there is DC leakage through the bandpass filter and Figure 3.9 shows the Fast Fourier Transform (FFT) of the signal in Figure 3 8. Compare Figure 3.9 to the FFT of Vc or Vs when the DC component has been properly attenuated, which is shown in Figure 3.10.

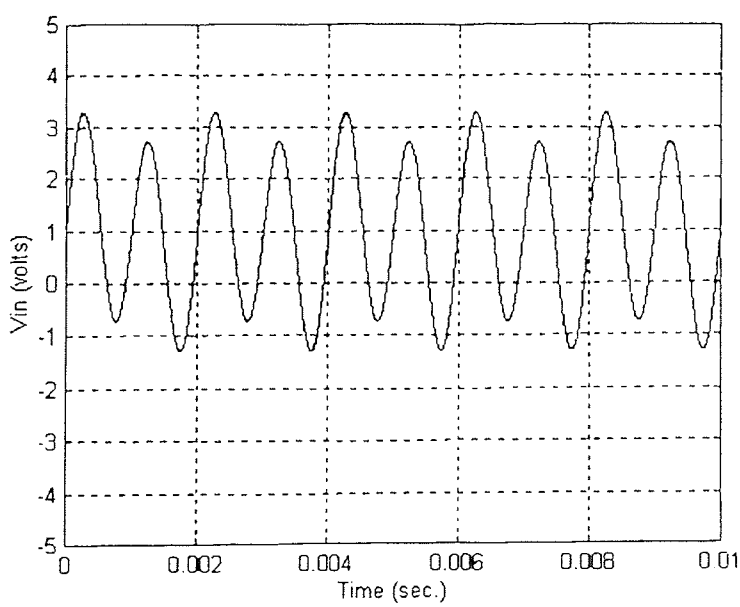


Figure 3.8 Demodulated signal with DC leakage through the bandpass filter.

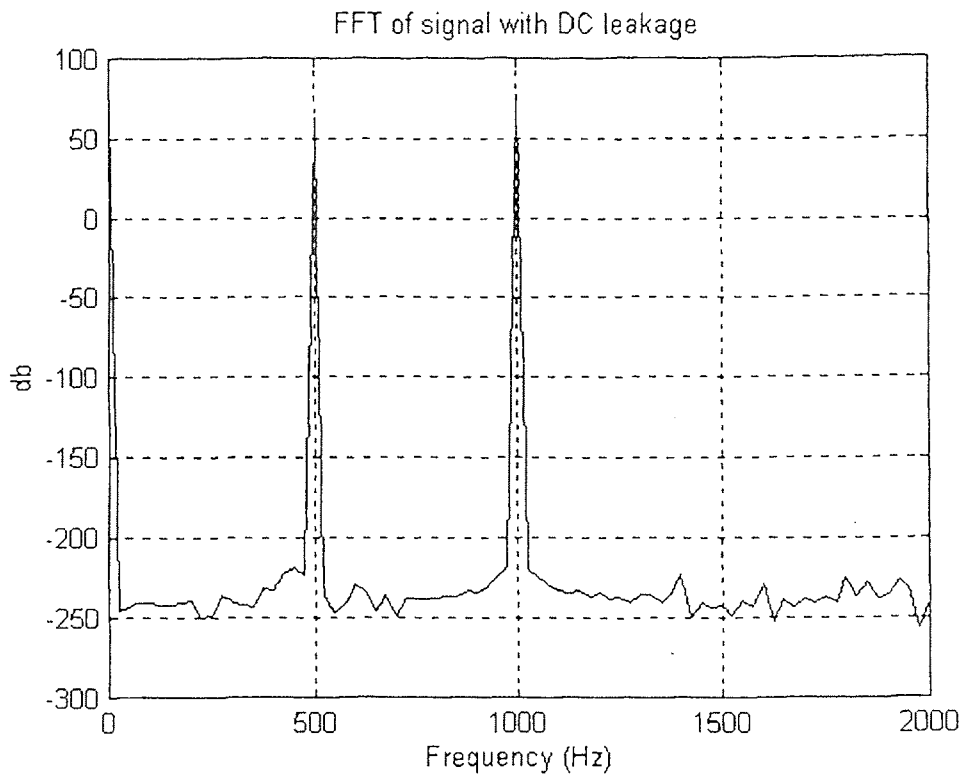


Figure 3.9 FFT of signal with DC leakage (Figure 3.8).

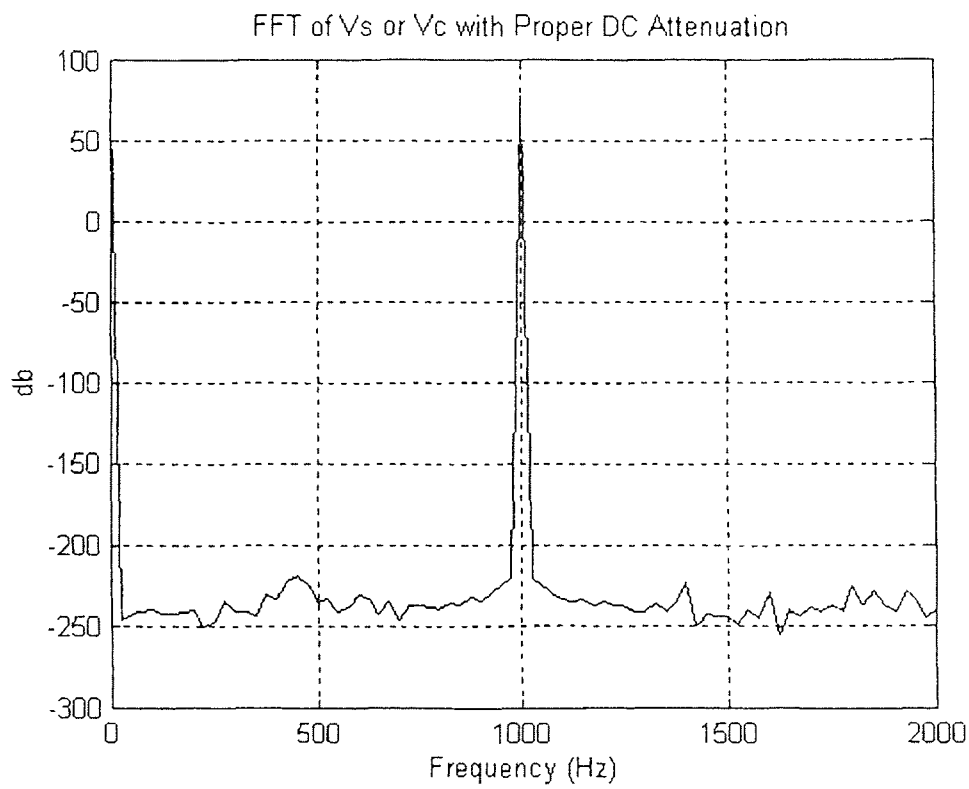


Figure 3.10 FFT of Vc or Vs when the DC component has been properly attenuated.

3.2.3 Lowpass Filter

An elliptic filter is synthesized to remove the double carrier frequency component (1KHz) in Vc (3.2-7) and Vs (3.2-10). However, due to finite word length effects, it is necessary to prefilter Vc and Vs by a lowpass filter to avoid signal saturation in the elliptic filter. This is because the amplitude of the double carrier frequency component of Vc and Vs can be much larger than the DC component which must be extracted to calculate the amplitude of the sampled sine wave.

The lowpass filter is a first order Butterworth filter with a cutoff frequency of 100Hz. The transfer function of the lowpass filter is shown in (3.2-16) and the difference equation is shown in (3.2-17). In (3.2-17), Vin_lpf represents Vs or Vc in Figure 3.6 and Vout_ lpf represents Vs_ lpf, the output of the lowpass filter in the sin(ωt) branch, or Vc_ lpf, the output of the lowpass filter in the cos(ωt) branch. The frequency and phase response of the digital low pass filter is shown in Figure 3.11.

$$\frac{\text{Vout\_lpf}(z)}{\text{Vin\_lpf}(z)} = \frac{0.030468 + 0.030468\,Z^{-1}}{1 - 0.939062\,Z^{-1}} \tag{3.2-16}$$

$$\begin{aligned}\text{Vout\_lpf}(i) = [&15386\ \text{Vout\_lpf}(i-1) + 499\ \text{Vin\_lpf}(i) \\ &+ 499\ \text{Vin\_lpf}(i-1)] / 16384\end{aligned} \tag{3.2-17}$$

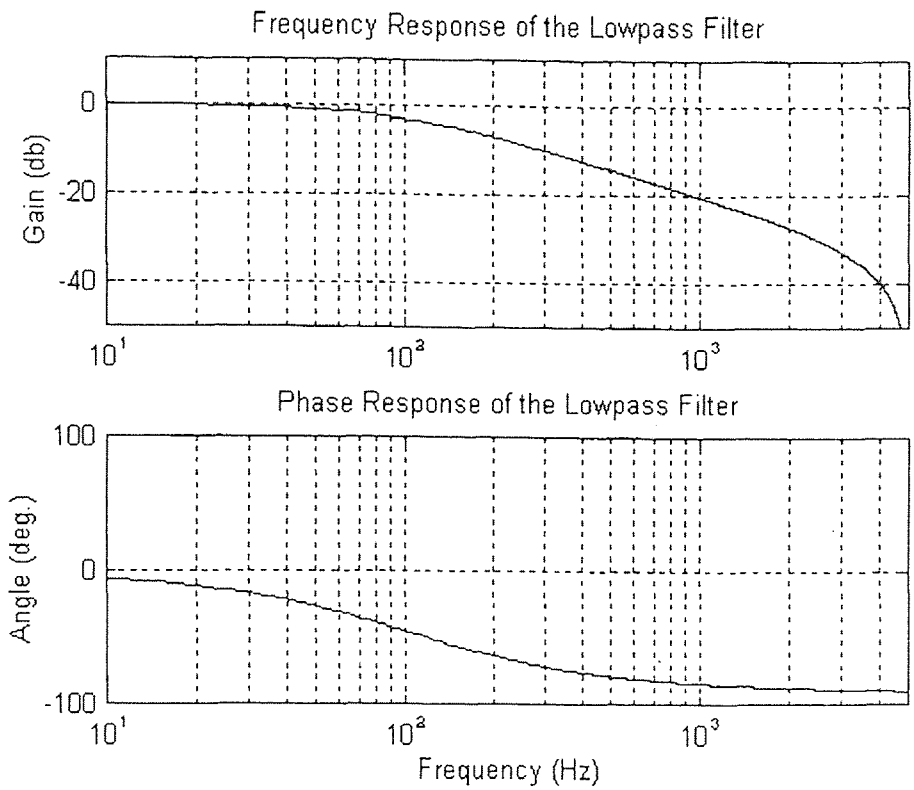


Figure 3.11 Frequency and phase response of the lowpass filter.

3.2.4 Elliptic Filter

At this point in the demodulation scheme, the signal consists of the desired DC component and the second harmonic component (1KHz). The purpose of the elliptic filter is to significantly attenuate the second harmonic component (1KHz). The operation of the elliptic filter is described in (3.2-18) to (3.2-21). The elliptic filter is a second order filter with 3db of ripple in the passband, 20db of attenuation in the stopband and a notch frequency of 1KHz.

$$Vc\_lpf = \frac{A}{2}\sin\Phi + \frac{A}{20}\sin(2\omega t) \tag{3.2-18}$$

$$Vc\_ell = \frac{A}{2}\sin\Phi \tag{3.2-19}$$

$$Vs\_lpf = \frac{A}{2}\cos\Phi - \frac{A}{20}\cos(2\omega t + \Phi) \tag{3.2-20}$$

$$Vs\_ell = \frac{A}{2}\cos\Phi \tag{3.2-21}$$

The transfer function and difference equation of the elliptic filter are shown in (3.2-22) and (3.2-23) respectively. In (3.2-24), Vin_ell represents Vs_lpf or Vc_lpf in Figure 3.5 and Vout_ ell represents Vs_ ell, the output of the elliptic filter in the sin(ωt) branch, or Vc_ ell, the output of the elliptic filter in the cos(ωt) branch. The frequency and phase response of the elliptical filter is shown in Figure 3.12.

$$\frac{Vout\_ell(z)}{Vout\_lpf(z)} = \frac{0.100783 - 0.162863\,Z^{-1} + 0.100783\,Z^{-2}}{1 - 1.795105\,Z^{-1} + 0.849777\,Z^{-2}} \tag{3.2-22}$$

$$\begin{aligned} Vout\_ell(i) = [&-29411\,Vout\_ell(i-1) + 13923\,Vin\_ell(i-2) + \\ &1651\,Vout\_lpf(i) - 2668\,Vout\_lpf(i-1) + \\ &1651\,Vin\_lpf(i-2)] / 16384 \end{aligned} \tag{3.2-23}$$

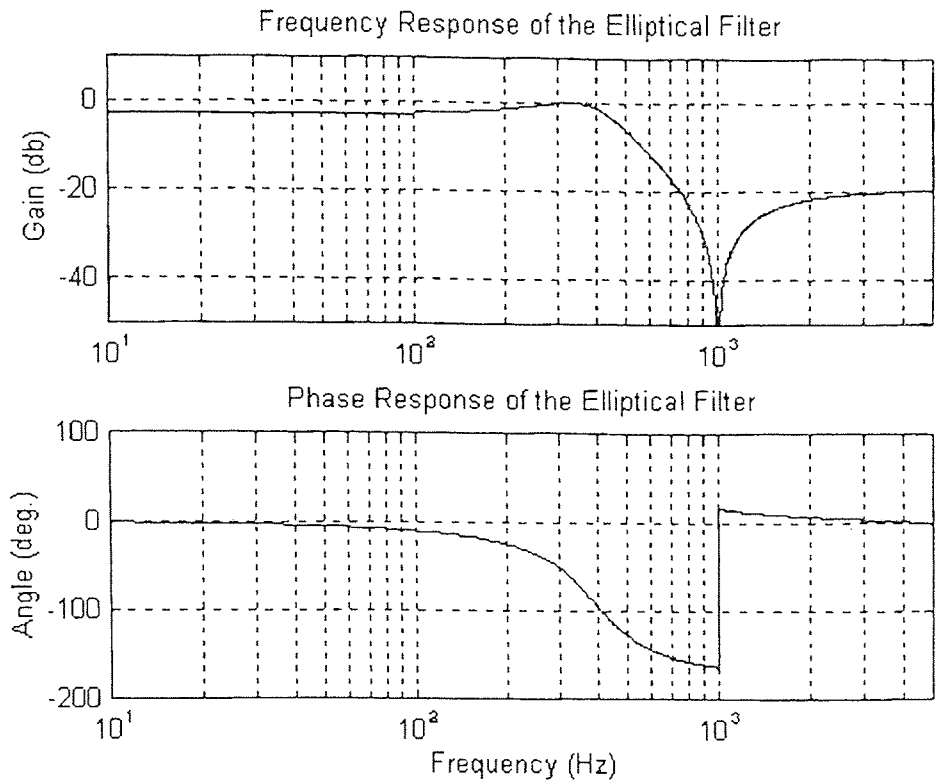


Figure 3.12 Frequency and phase response of the elliptic filter.

3.2.5 Squarer and Summer

The final step in the synchronous demodulation scheme, as shown in Figure 3.6, is performed by the squarer and summer blocks. The outputs of each elliptic filter are squared (3.2-24) and (3.2-25) and then summed together (3.2-26) to remove any dependency on Φ, the signal phase.

$$Vc\_ell^2 = \frac{A^2}{4}\sin^2\Phi \tag{3.2-24}$$

$$Vs\_ell^2 = \frac{A^2}{4}\cos^2\Phi \tag{3.2-25}$$

$$V\_out = Vc\_ell^2 + Vs\_ell^2 = \frac{A^2}{4} \tag{3.2-26}$$

The result is a signal that is a function of the amplitude of the input signal (D_in) only.

3.2.6 Filter Development

During software development each filter was individually tested with both sinusoidal and constant signals. While the processing of sinusoidal signals was correct, it was found that, with a constant input, the software produced a small steady-state error. The problem was traced to both the elliptic filters and the lowpass filters. It was found that the elliptic filter generated a constant offset error for all non-zero initial conditions. The source of the problem was found by executing the filter assembly code manually. The result of this method indicated an error in the way the DSP saves negative numbers after shifting. For Example, suppose that the value to be saved is $\frac{-3068}{4096}$. Dividing by 4096 is equivalent to shifting the accumulator value by four bits to the left and saving the high sixteen bits. The expected answer is zero but the actual answer, as executed by the assembly code, is -1. Figure 3.13 describes the process by which the DSP shifts and saves the values in the accumulator and the resulting error.

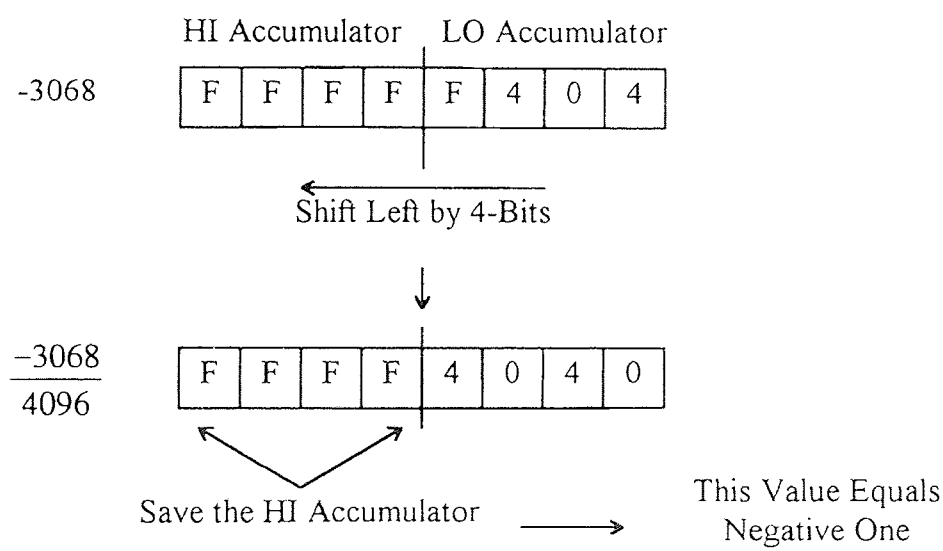


Figure 3.13 Example of the division error in the DSP.

The solution to this problem is the following: If the value in the accumulator to be divided and saved is less than zero, add the denominator minus one to the accumulator before the value is shifted and saved. This will change the hex F's in the high accumulator to zeros. Figure 3.14 shows an example of the division incorporating the correction. To avoid this problem in the other filters the correction for dividing and saving negative numbers was added to all filters through out the synchronous demodulation program.

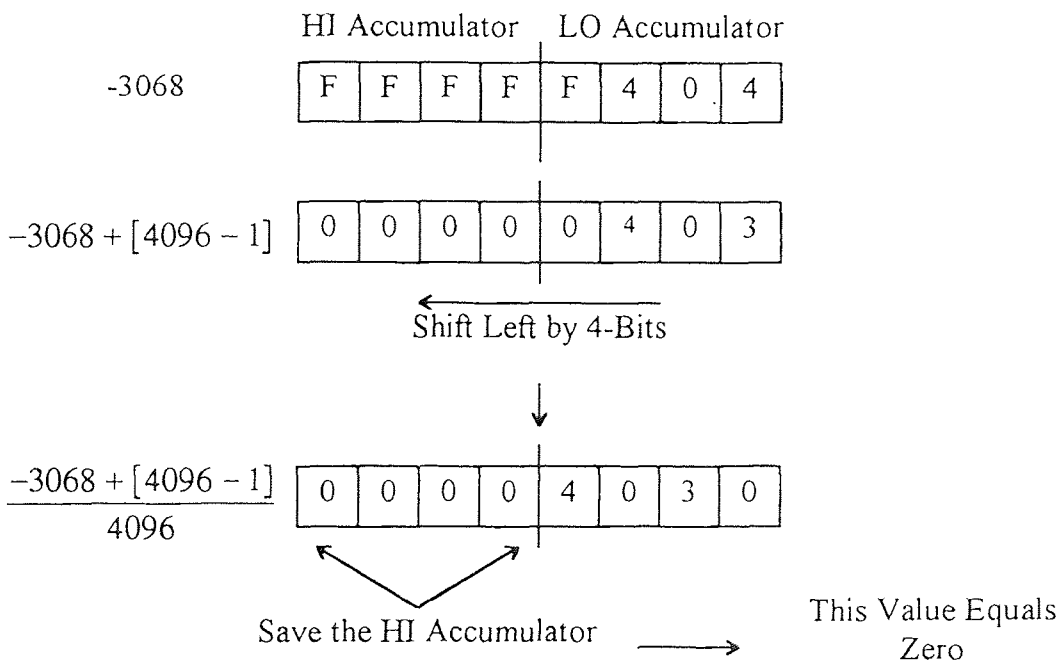


Figure 3.14 Example of the division with the DSP incorporating the correction.

Besides division error in the elliptic filters, it was also found the lowpass filters also produced a steady state offset error as show in Figure 3.15.

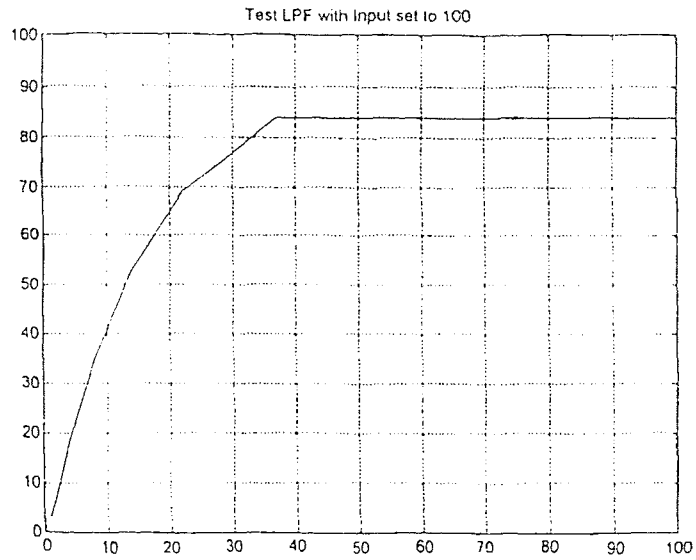


Figure 3.15 Example of the lowpass filter error.

Again, the source of the problem was found by executing the filter assembly code manually. This problem is due to the finite word length effects of the DSP. The output of all of the digital filters must be divided by the scaling factor of the digital filter as described in Section 3.2.1. As the output of the filter approaches the input DC value when testing with a constant input, the change from iteration to iteration becomes smaller and smaller. Eventually, the change from each iteration becomes less than the scaling factor and is lost in the division. The solution of the problem is to gain the input before the lowpass filter.

For example, in a fixed point calculation, when two input values 13000 and 13000 + 3000 are divided by 4096, they are both truncated to 3. With proper scaling (x2 for example) the results are 6 and 7 respectively. The effectiveness of gaining the input before the lowpass filter is shown in the data in Table 3.2.

Table 3.2 Effectiveness of gaining the input of the lowpass filters.

Filter Value	Actual Value	Fixed Point Value	Percent Error
$\frac{13000}{4096}$	3.174	3	5.5
$\frac{13000+3000}{4096}$	3.906	3	23.2
$\frac{2\times[13000]}{4096}$	6.348	6	5.5
$\frac{2\times[13000+3000]}{4096}$	7.813	7	10.4

3.3 Channel Three -- Laser Modulation

A Flow chart of Channel Three is shown in Figure 3.16. The primary responsibility of this channel is to modulate the laser diode at 500 Hz and to copy the outputs of the demodulation routines to the DSP's external memory where it can be accessed by a C-program running on the host PC. This channel can also be used for other additional tasks because the specified responsibilities of Channel Three use less than 3us, which is only a fraction of the 25us allotted for each interrupt service routine.

Since the A/D and D/A share the same external data latch, proper operation of the D/A sample signal requires that the A/D be read each time the interrupt is triggered regardless if the data is needed. This is also critical because the A/D ready signal is used to provide the 25us timing, as described in section 3.1.1. Therefore, the first operation of channel three is to read the A/D.

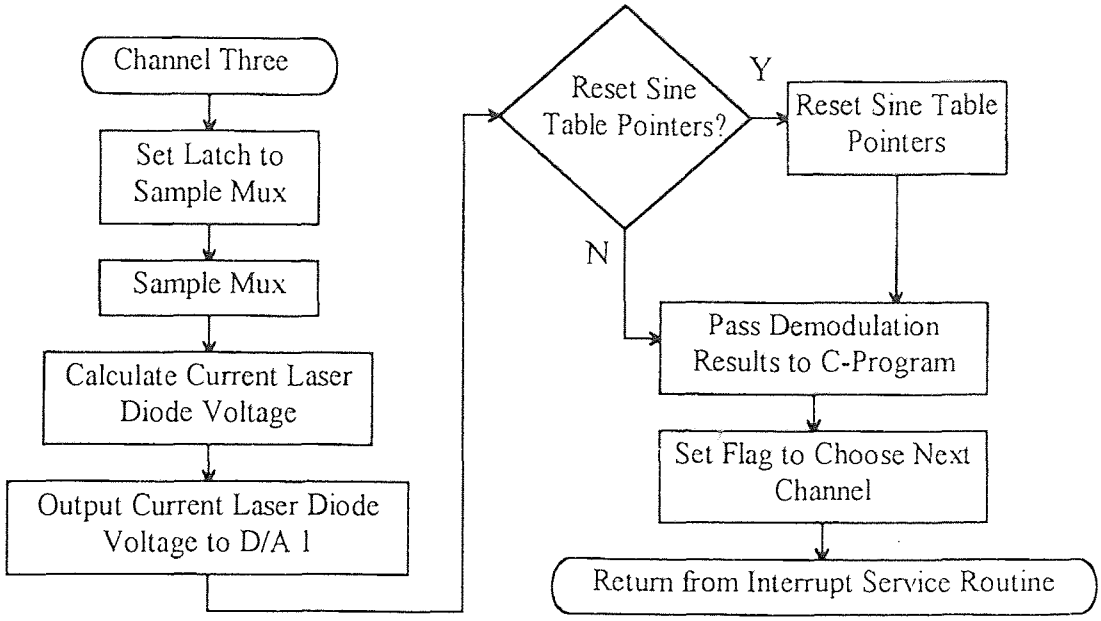


Figure 3.16 Block diagram of Channel Three.

The laser diode is powered by a sine wave voltage with an amplitude of 0.2 volts added to a 2.6 volt DC bias. Since the Channel Three Program is executed at 10KHz and the required modulation frequency of the laser diode is 500Hz, the number of time steps in the sine wave is: $\frac{10\text{KHz}}{500\text{Hz}} = 20$ Steps. Therefore, during the initialization part of the program, a sine table with twenty points equally distributed from zero to 342 degrees is constructed by scaling the values of sin(θ) by 4096: Values in sine table = 4096 * sin(θ). In addition, a pointer which points to the first location in the sine table is initialized in auxiliary register one of the DSP. The program also initializes the amplitude (AMP = 0.2 volts is represented by 80 in the DSP), a DC bias of the sine wave (Bias1 = 2.6 volts is represented by 1077 in the DSP) and another DC bias which is used to properly adjust the

output voltage of the D/A (Bias2 = 5.0 volts is represented by 2048 in the DSP). The specifications for the A/D and D/A may be found in the Dalenco-Spry manual.

During the calculation of the laser diode voltage (V_laser) the program multiplies the preset amplitude (AMP) with the value in the sine table (sin(θ)) indicated by the pointer and divides out the scaling factor (4096) introduced to define the values in the sine table. The result is summed with two bias values (Bias1 and Bias2). Equation (3.3-1) describes how the laser voltage is calculated as a function of θ.

$$V\_laser = \frac{80 \times \sin\theta}{4096} + 1077 + 2048 \qquad (3.3\text{-}1)$$

The value V_laser is then outputted to the D/A and then the amplifier (See Figure 2.5 of Section 2.3 for a description of the laser diode power electronics) which drives the laser diode. If the pointer reaches the end of the table, then it is reset to the top. A block diagram which describes the laser modulation calculation technique is shown in Figure 3.17. Refer to Table 3.3 for a description of the variables involved in laser diode voltage determination.

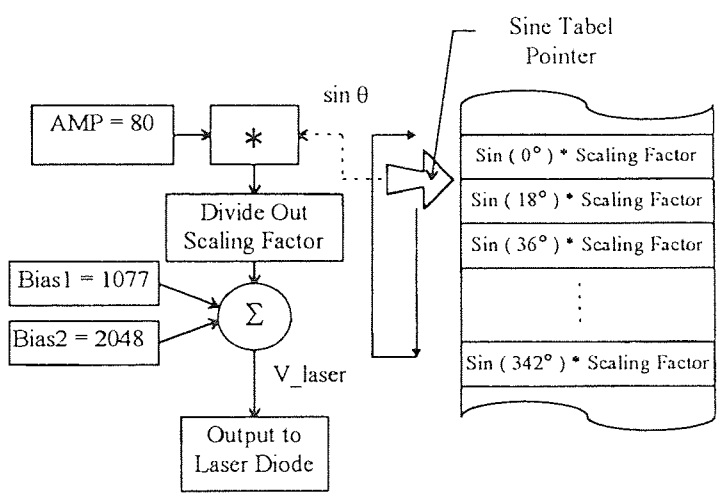


Figure 3.17 Block diagram of the laser diode voltage determination scheme.

Table 3.3 Description of variables involved in laser diode voltage determination.

Variable Name	Value in DSP	Description
sin(θ)		Value representing the sine wave as a function of θ
Amp	80	Amplitude of the sine wave
Scaling Factor	4096	Integer value used to scale sin(θ)
Bias1	1077	DC bias of the sine wave
Bias2	2048	5 volt offset of the D/A

The outputs of the three demodulation channels are saved in variables Amove, Astat, and Apin respectively. After the voltage on the laser diode is calculated, the Channel Three Program copies the memory locations which contain the outputs from the three synchronous demodulation channels from internal memory page 4 to external dual ported memory page 8 on the DSP board where it can be accessed by the host PC. Finally, the channel pointer is updated so that Channel Zero is executed during the next interrupt service routine. Refer to Table 3.4 for the data memory locations of the variables.

Table 3.4: Data memory location summary.

Variable Name	Data Memory Address Page	Memory Location on Page
Amove	4	112
Astat	4	115
Apin	4	119
Amove	8	16
Astat	8	56
Apin	8	96
Flag0	8	20
Flag1	8	60
Flag2	8	100
U	8	120

3.4 PC Interface

The host PC is a 286 computer with a math co-processor. The PC can execute either the DSP's debugger or a C-program which can perform the following tasks: 1) Sample the data from the DSP and save it on a floppy disk for analysis or 2) Execute a control program by sampling position data from the DSP, calculating a control law, and return the control signal (U) to memory location 120 in data memory page 8 in the DSP.

3.4.1 Data Sampling and Saving

The process of taking a data sample was automated as much as possible to save time. The DSP assembly program, which has been described in detail in the previous sections, runs

continuously. A simplified block diagram of the data sampling and saving scheme is shown in Figure 3.18 and a flow chart of the data-saving C-program is shown in Figure 3.19.

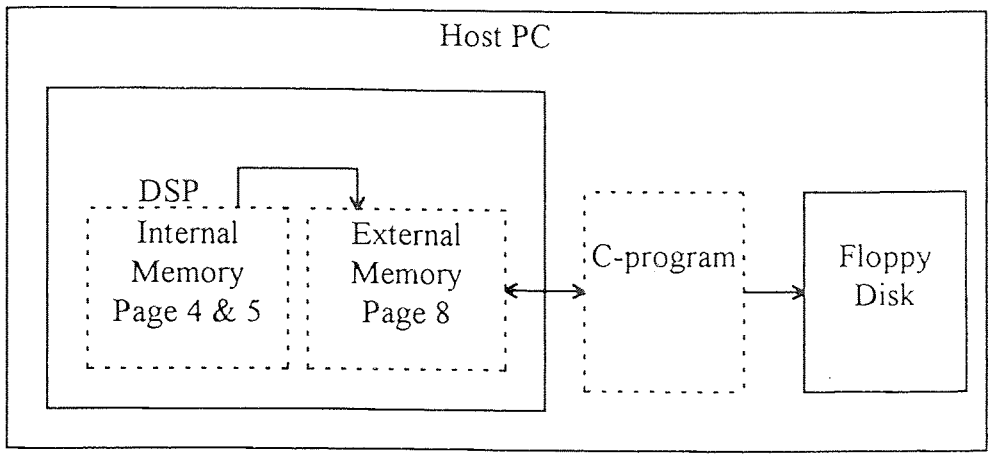


Figure 3.18 Block diagram of the data sampling process.

After the necessary initialization, the C-program continuously examines the data-ready flags (Flag0, Flag1, and Flag2) from the DSP which signal that the demodulation outputs from Channels Zero, One and Two, respectively, are ready. Refer to Table 3.4 for the memory addresses of the data flags. The outputs of the Channel Zero, One and Two programs (Amove, Astat, and Apin) are sampled, when available, and normalized. The normalized variables (Nstat and Npin) and calculated as follows: $\text{Nstat} = \sqrt{\frac{\text{Astat}}{\text{Amove}}}$ and $\text{Npin} = \sqrt{\frac{\text{Apin}}{\text{Amove}}}$. These data are ported to Matlab for analysis. Refer to Appendix 8 for the C-program codes.

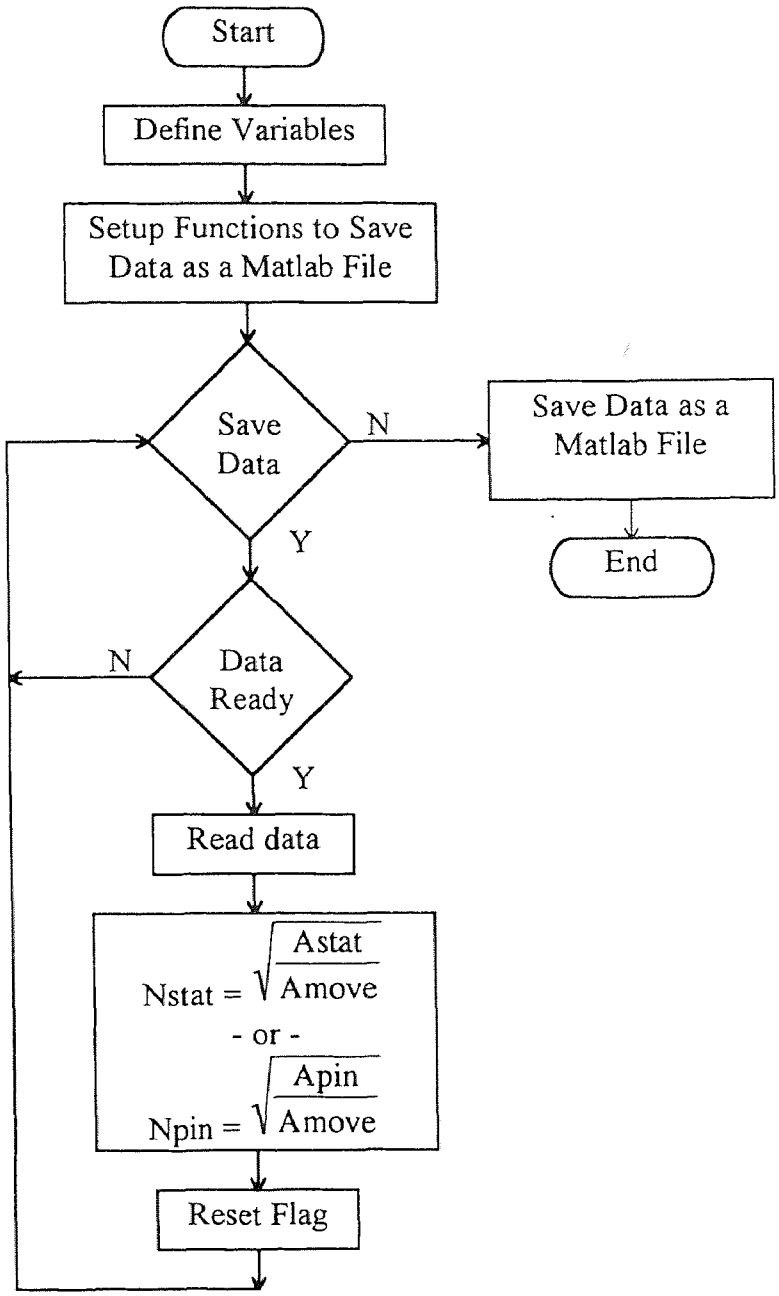


Figure 3.19 Flow chart of the data saving C-program.

3.4.2 Control Program

Once the sensor system has been calibrated and characterized, it is deployed in a flexible beam control experiment. A C-program was written to control the position of the flexible beam which was described in Section 2.7. A block diagram of the control system is shown in Figure 3.20 and a flow chart of the control algorithm is shown in Figure 3.21. See Appendix 9 for the C-program codes. A description of the variables used in the control program is shown in Table 3.5.

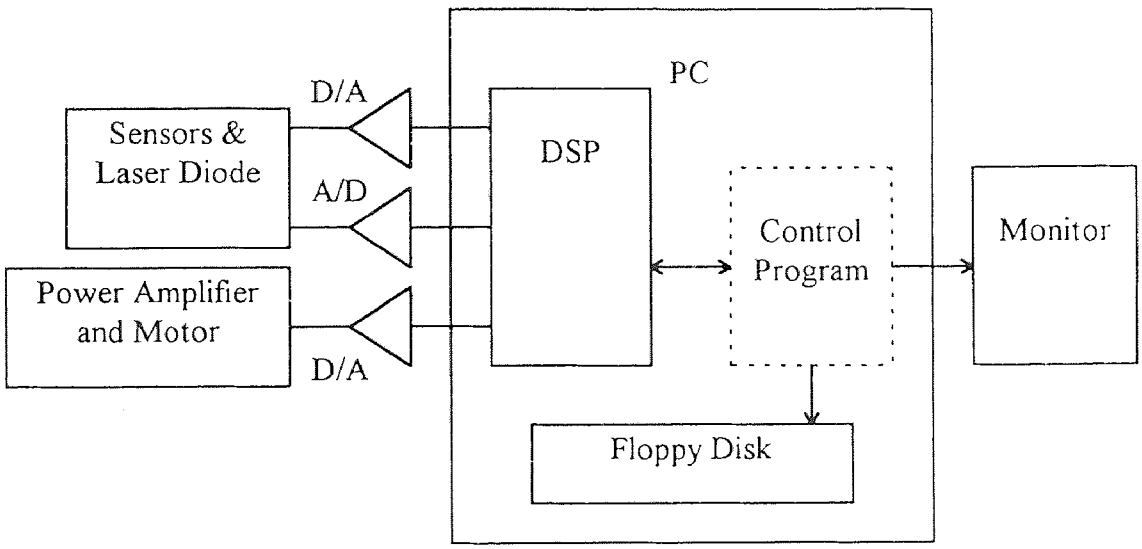


Figure 3.20 Block diagram of the control program scheme.

Table 3.5 Description of control variables.

Variable Name	Description
Y	Position Signal at the ith step
Y_{REF}	Reference Position
E	Error (Y_N - Y_{REF}) at the ith step
U	Command Signal at the ith step
D	Output of LPF at the ith step
INT	Output of Integrator at the ith step
K_P	Position Gain
K_D	Derivative Gain
K_I	Integral Gain

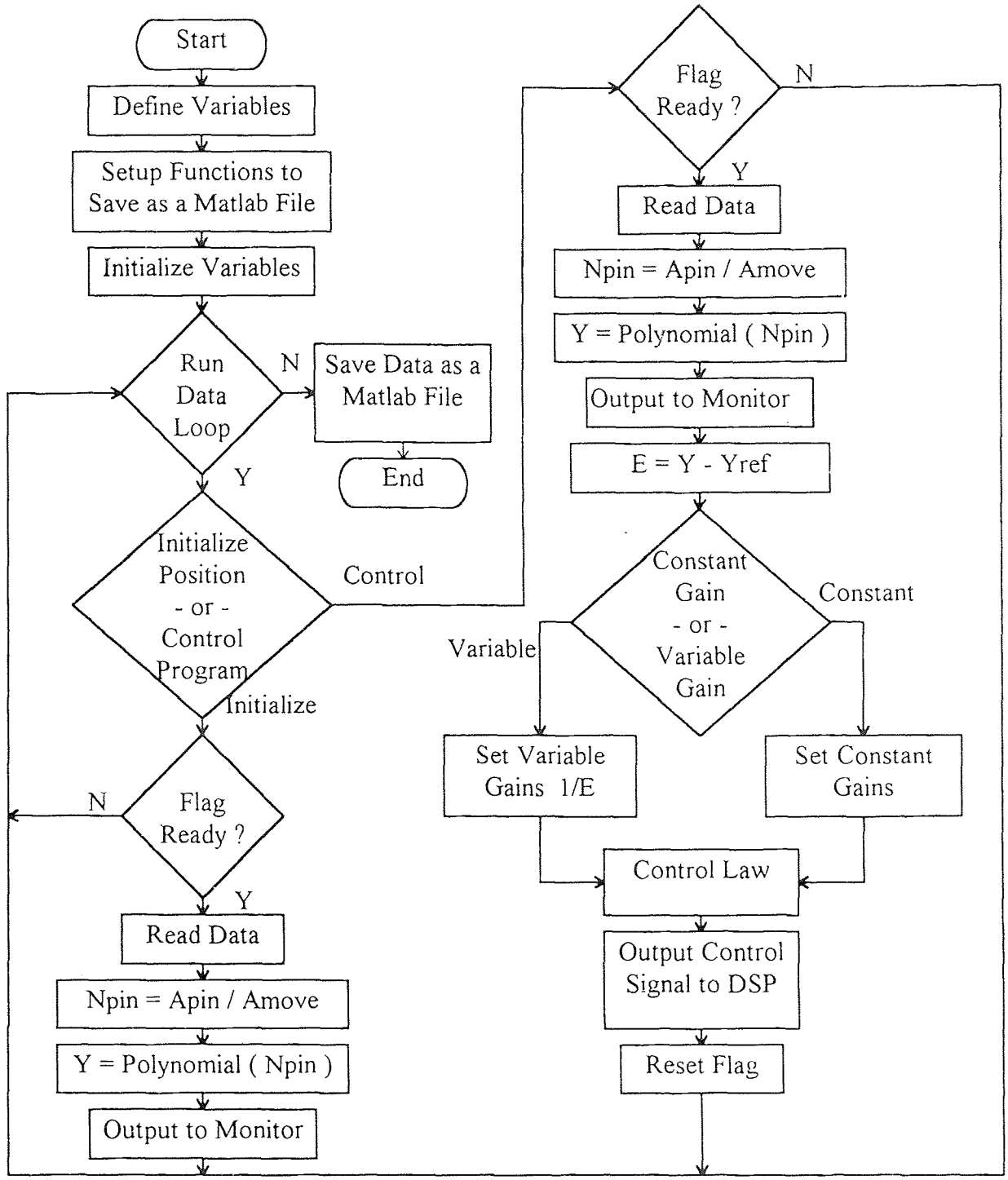


Figure 3.21 Flow chart of the control C-program.

There are two sections in the control program. The first section displays the real-time position of the beam's end point on the monitor continually so the beam can be set to the same initial position before every test for consistency in testing. After the beam is set to the desired starting position, the program runs the control loop which is executed for 20 seconds, and saves the position, error and control signal data on a floppy disk.

A listing of the control variables and their respective descriptions is shown in Table 3.5. The four control laws used are as follows:

1) Position control law (3.4-1), Figure 3.22.

2) Position and Derivative control law (3.4-4), Figure 3.24.

3) Position, Derivative and Integral control law with constant gains (3.4-6), Figure 3.26.

4) Position, Derivative and Integral control law with variable gains (3.4-14), Figures 3.27.

$$U(i) = K_P\,(Y(i) - Y_{REF}) \tag{3.4-1}$$

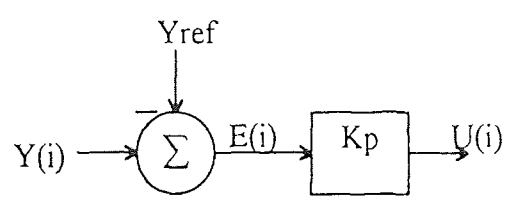


Figure 3.22 Flow chart of the Kp control.

To reduce the noise in the derivative signal, a first order lowpass filter with a cutoff frequency of 40 Hz was synthesized. The transfer function of the lowpass filter is shown in (3.4-2) and the difference equation is given in (3.4-3). The frequency and phase response of the lowpass filter are plotted in Figure 3.23.

$$\frac{D(z)}{E(z)} = \frac{0.5095 + 0.5095\,Z^{-1}}{1 - 0.7548\,Z^{-1}} \tag{3.4-2}$$

$$D(i) = 0.7548\,D(i-1) + 0.5095\,E(i) + 0.5095\,E(i-1) \tag{3.4-3}$$

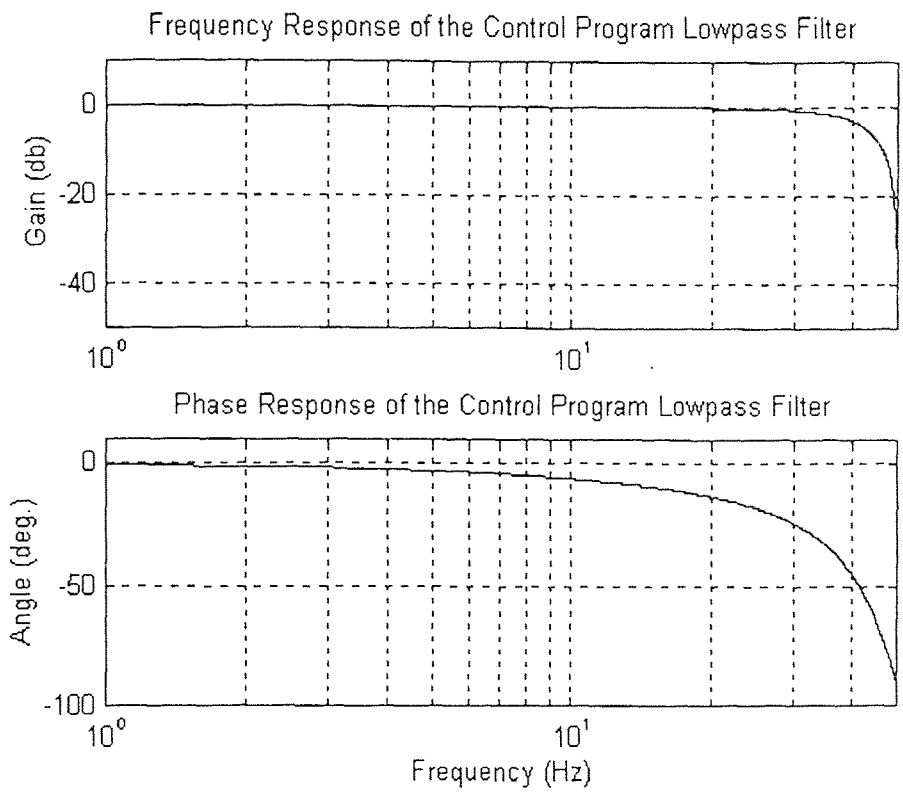


Figure 3.23 Frequency and phase response of the control program's lowpass filter.

$$U(i) = K_P\,[Y(i) - Y_{REF}] + K_D\,[D(i) - D(i-1)] \tag{3.4-4}$$

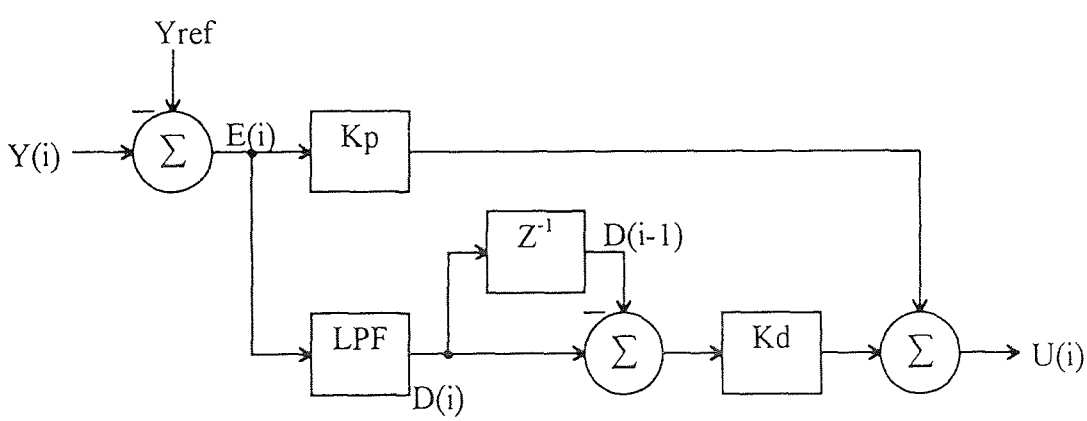


Figure 3.24 Flow chart of the Kp, Kd control.

To overcome the motor friction in the flexible beam setup, an integrator was synthesized to improve the steady state error of the position of the beam. A block diagram of the system which incorporates the integrator is shown in Figure 3.25. The equation for the integrator is shown in (3.4-5).

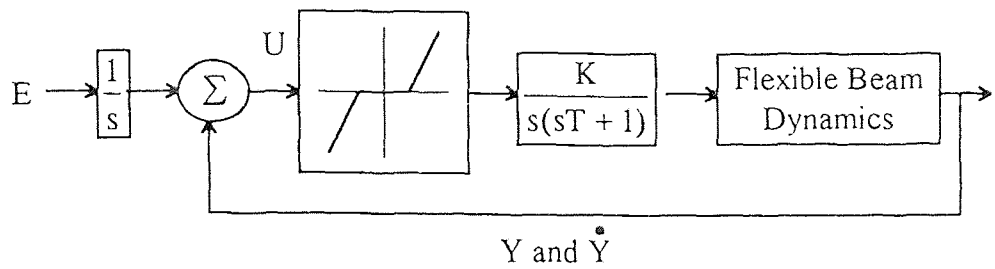


Figure 3.25 Block diagram of the motor control system incorporating the integrator.

$$INT(i) = INT(i-1) + E(i) \tag{3.4-5}$$

$$U(i) = K_P\,[Y(i) - Y_{REF}] + K_D\,[D(i) - D(i-1)] + K_I\,[INT(i)] \tag{3.4-6}$$

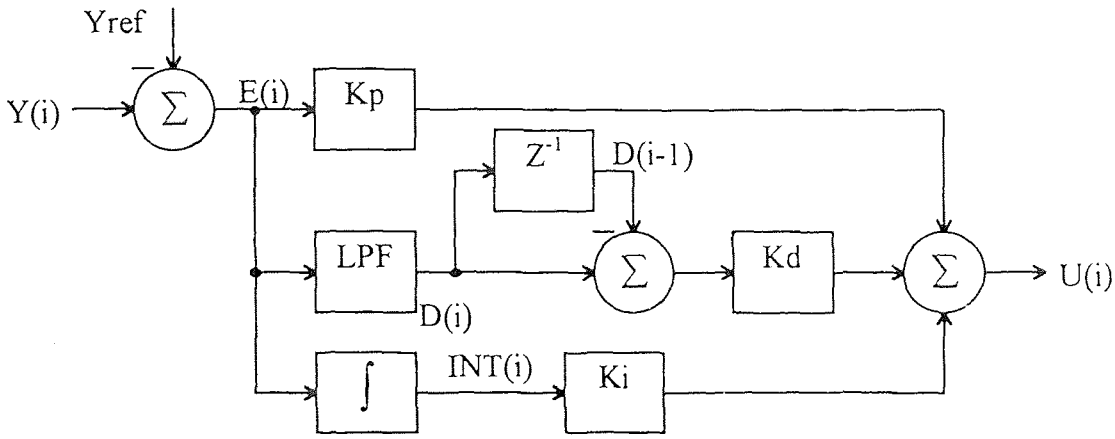


Figure 3.26 Flow chart of the Kp, Kd, Ki control with constant gains.

To improve the dynamic response of the flexible beam system, a variable gain control program was implemented. If the error signal is large, the PID gains will be small while if the error signal is small the PID gains will be large. This implementation will help

prevent oscillation of the flexible beam if the error signal is large but will improve the steady state error of the position of the flexible beam when the error signal is small.

To prevent division by zero in the variable gain equations, the current error value (E(i)) is replaced by the previous error value (E(i-1)) if it is zero. The equations for the variable PID gains are shown in (3.4-8), (3.4-10) and (3.4-12). Each gain is also upper and lower bounded by (3.4-9), (3.4-11) and (3.4-13).

$$\text{If } E(i) = 0 \text{ then } E(i) = E(i-1) \tag{3.4-7}$$

$$K_{PV}(i) = K_P / E(i) \tag{3.4-8}$$

$$K_P / 2 < K_{PV} < 2\, K_P \tag{3.4-9}$$

$$K_{DV}(i) = K_D / E(i) \tag{3.4-10}$$

$$K_D / 2 < K_{DV} < 2\, K_D \tag{3.4-11}$$

$$K_{IV}(i) = K_I / E(i) \tag{3.4-12}$$

$$K_I / 10 < K_{IV} < 10\, K_I \tag{3.4-13}$$

$$U(i) = K_{PV}\,[Y(i) - Y_{REF}] + K_{DV}\,[D(i) - D(i-1)] + K_{IV}\,[INT(i)] \tag{3.4-14}$$

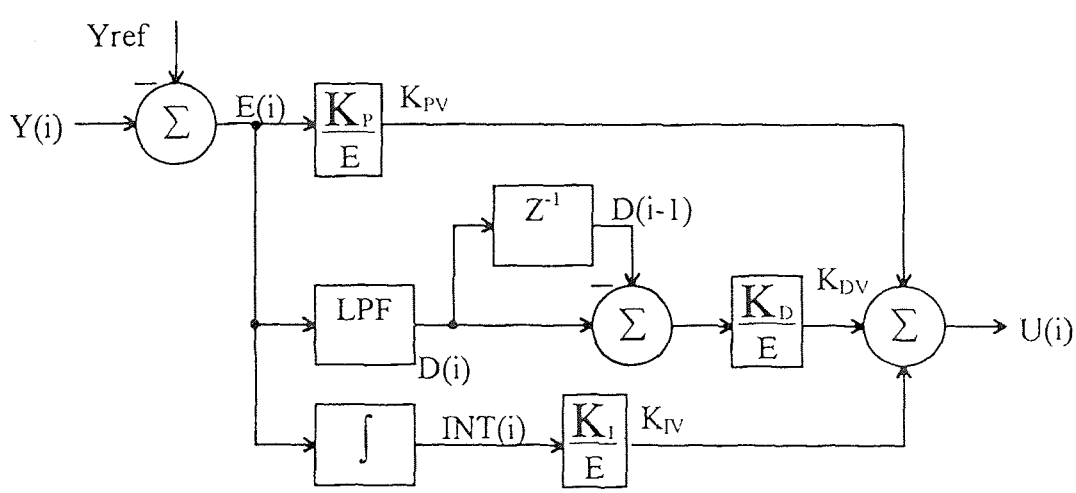


Figure 3.27 Flow chart of the Kp, Kd, Ki control with variable gains.

CHAPTER 4

TESTING AND DATA

4.1 Signal Time Stability

A number of tests were performed to determine, quantitatively, the characteristics of the sensor modules. These tests include: 1) Time stability, 2) Short range accuracy, and 3) Long range accuracy. These test results are now discussed. Refer to Section 1.2 for a review of the signal flow and a description of signal labels.

4.1.1 Time Stability of Pstat

The signal Nstat is created by normalizing the intensity of the incident light on the mobile sensor module (Amove) using the stationary sensor module (Astat). Pstat is the average of the integration of Nstat over one minute. To check the time stability of Pstat, a drift test is performed at the same position over several minutes. Figure 4.1 shows the stability of Pstat at 12.7 mm and Figure 4.2 shows the stability of Pstat at 0 mm. Stability of the signal can be assessed by calculating the trends and the standard deviation.

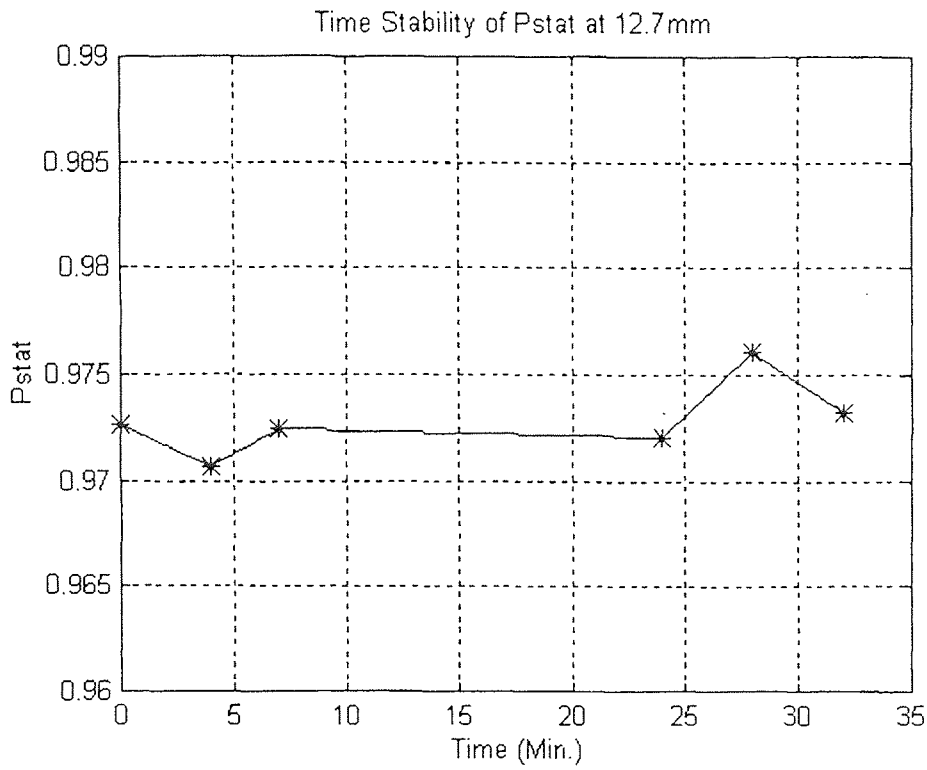


Figure 4.1 Time stability of Pstat at 12.7 mm.

Mean (Pstat) = 0.9728

Std (Pstat) = 0.0018

Position uncertainty = 0.95 mm

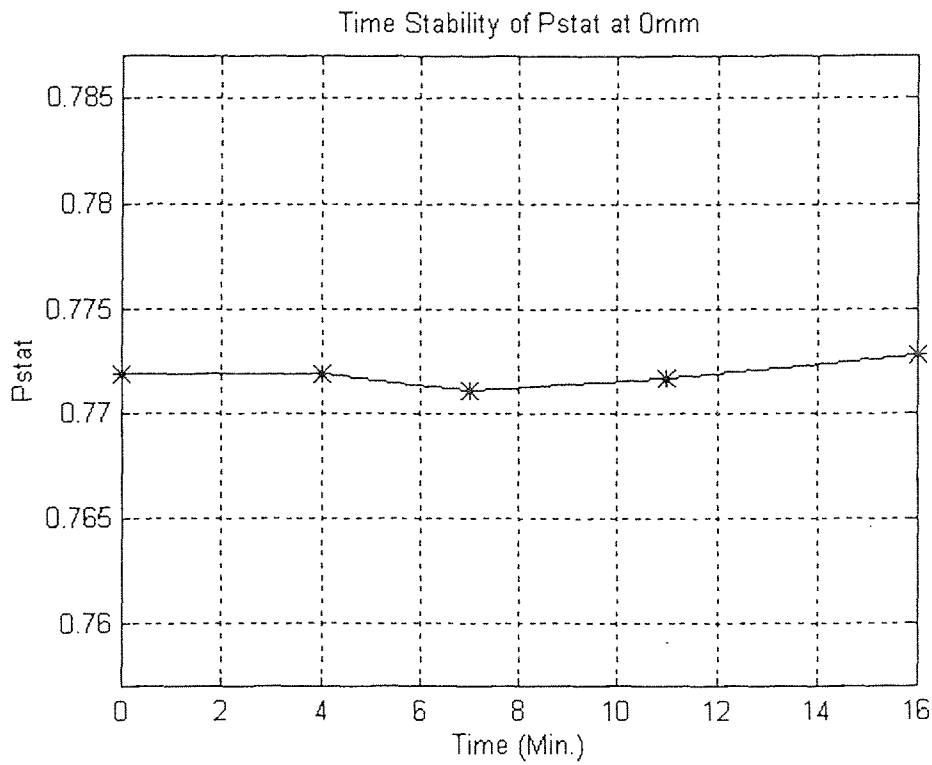


Figure 4.2 Time stability of Pstat at 0 mm.

Mean (Pstat) = 0.7719

Std (Pstat) = 0.0006

Position uncertainty = 0.23 mm

These tests provide a verification of the performance of the sensor: There were no detectable trends in the system during the period of testing.

4.1.2 Time Stability of Dpin & Ppin

The signal Dpin is created by sampling the analog feedback voltage from the laser diode at 10KHz. The values of Dpin are the DSP's representative values of the sampled voltage. To

check the time stability, one minute averages of Dpin were recorded over a period of 16 minutes as shown in Figures 4.3.

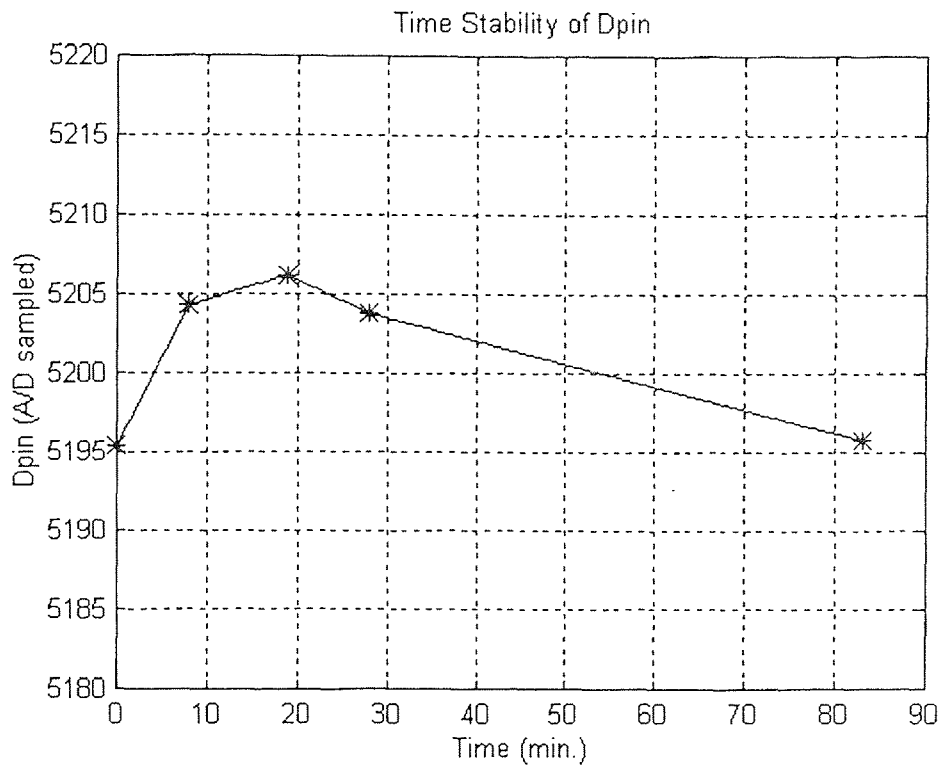


Figure 4.3 Time stability of Dpin.

Mean (Dpin) = 5201.1

Std (Dpin) = 5.1

The signal Npin is created by normalizing the intensity of the incident light on the mobile sensor module (Amove) using the feedback signal from the laser diode (Apin). To check the time stability of Npin, a drift test is performed at the same position over several minutes. Each point plotted in Figure 4.4 is the average of the integration of the Npin over one minute. Stability of the signal can be assessed by calculating the standard deviation.

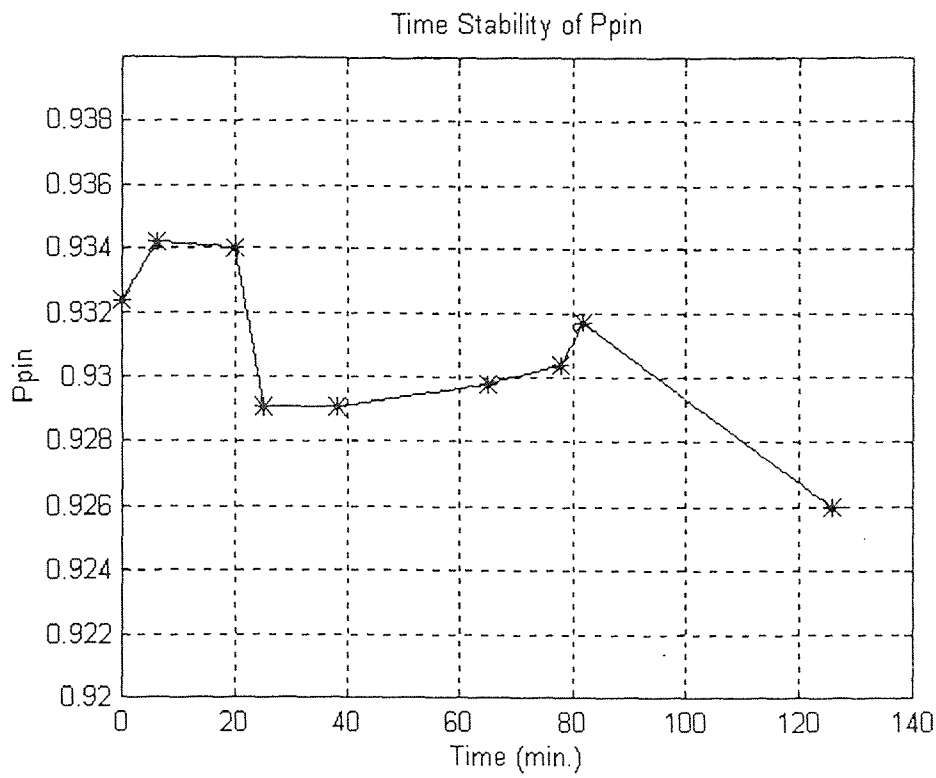


Figure 4.4 Time stability of Ppin.

Mean (Ppin) = 0.9307

Std (Ppin) = 0.0026

Position uncertainty = 1.37 mm

4.2 Calibration and Verification Procedures

The purpose of the test is to determine the ability of the system to predict the position of the mobile sensor module. Table 4.1 describes the symbols used in the calibration and verification procedures.

Table 4.1 Description of symbols used in the calibration and verification procedures.

Variable Name	Description
AL	Actual location (mm)
EL	Error in predicted location (mm)
PL	Predicted location (mm)
P	Pstat or Ppin position signal
P_test	Set of P data to predict location
P_calibration	Set of P data to calculate calibration polynomial
Polynomial	Calibration polynomial

The following steps summarize the calibration and verification procedures:

Step 1) Create a calibration curve: Sample P at several positions (P_calibration) to create a table of known locations versus P. Generate a polynomial using Matlab to fit the calibration data points (4.2-1):

$$\text{Polynomial} = f(\text{AL, P\_calibtation}) \tag{4.2-1}$$

Step 2) Check the quality of the calibration curve: Predict the position using the calibration polynomial and the calibration position data (4.2-2). Plot the known positions Vs the predicted positions and compare. If the curves are approximately the same the calibration polynomial may be used to perform further position prediction tests.

$$\text{PL} = \text{Polynomial (P\_calibration)} \tag{4.2-2}$$

Step 3) Perform prediction test: Take another set of position data (P_test), at known locations, and predict the position using the previously determined calibration polynomial

(4.2-3). To determine the ability of the polynomial to predict the position of the mobile sensor module correctly, plot the difference between the predicted and actual positions (4.2-4).

$$PL = Polynomial(P\_test) \tag{4.2-3}$$

$$EL = AL - PL \tag{4.2-4}$$

Flow charts of the calibration and position predicting steps are shown in Figures 4.5 and 4.6, respectively. The hardware setup used for the testing is described in Section 2.6.

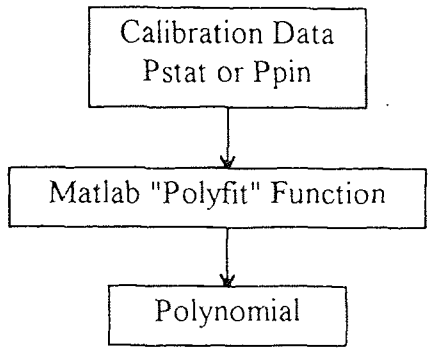


Figure 4.5 Step one of testing -- Calibration.

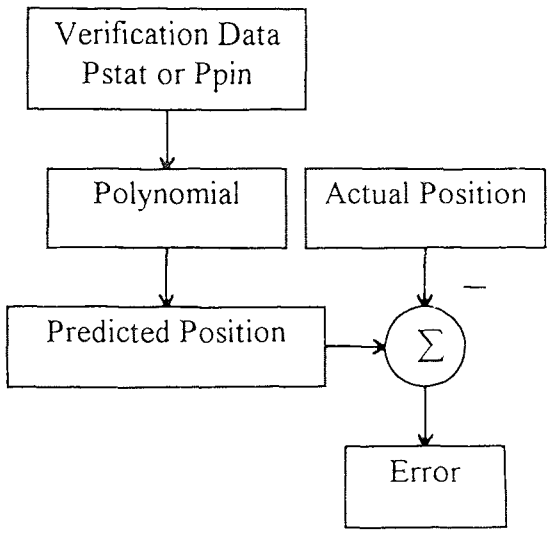


Figure 4.6 Step three of testing -- Position prediction.

4.3 Short-Range Test Using Pstat

The purpose of the short-range test is to determine the system's ability to predict the position of the mobile sensor module over a range of 12.7mm using Pstat, which is the intensity signal normalized by the intensity signal of the stationary sensor module. Refer to section 2.5 for a description of the test setup. The position repeatability of the mobile sensor module is 0.02mm. The calibration and prediction data were taken as described in Section 4.3.

The 2nd order polynomial generated from the calibration data for the short range test is shown in (4.3-1). The calibration curve for the short-range test is shown in Figure 4.7. The solid line is constructed by linearly interpolating the verification data taken at positions indicated by the asterisks. The dashed line, on the other hand, is generated by (4.3-1). The error between the predicted and the actual position, which is calculated using the second set of data P_test as in (4.2-3) and (4.2-4), is plotted in Figure 4.8.

$$\text{Location (mm)} = 101.1071\ \text{Pstat}^2 - 21.2036\ \text{Pstat} - 41.1373 \qquad (4.3\text{-}1)$$

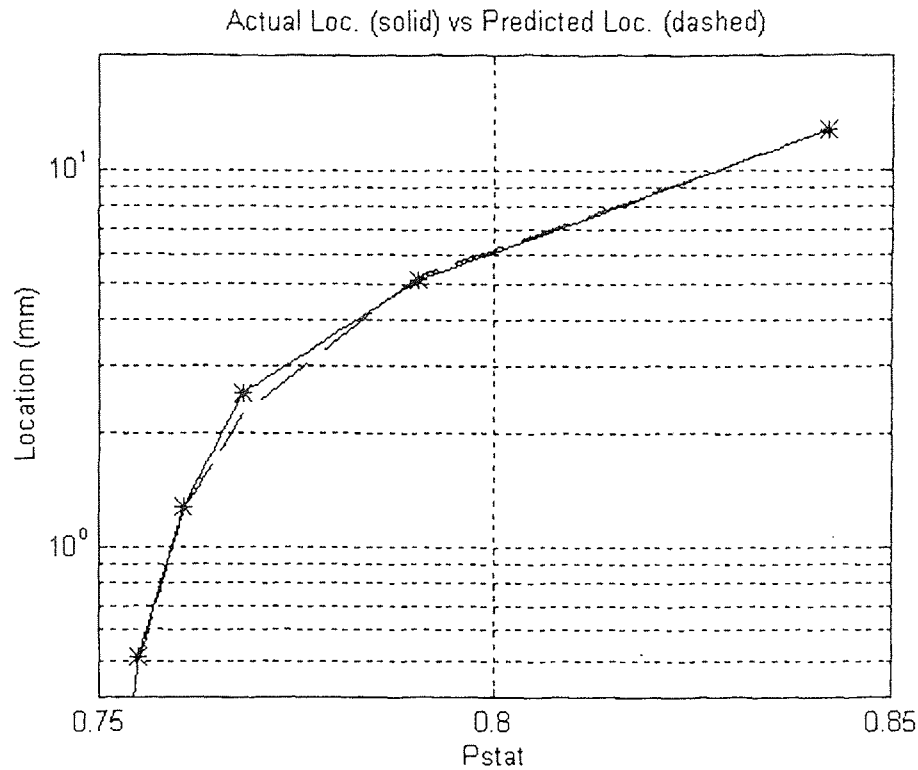


Figure 4.7 Calibration curve for the short-range test.

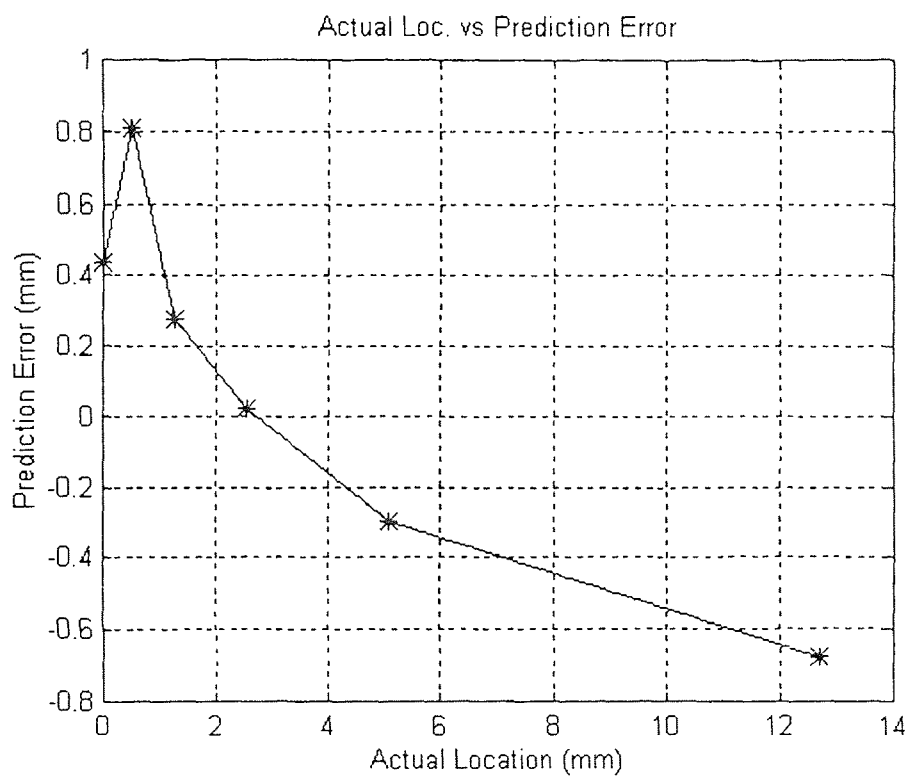


Figure 4.8 Location prediction error for the short-range test.

Mean (Error) = 0.10 mm

Std (Error) = 0.53 mm

From the above standard deviation the location of the mobile sensor module can be predicted to within an uncertainty of approximately 0.5 mm.

4.4 Long-Range Test Using Pstat

The system was also tested for its ability to predict the position of the mobile sensor module over a range of approximately 600mm. The platform with the mobile sensor module mounted on it was adjusted, by hand, along an aluminum bar with distance markings scribed on it. This enabled testing over a greater distance range but increased the uncertainty in the accuracy of positioning the platform to about 2mm. The method for determining the calibration polynomial and prediction data set is the same as for the short range test. However, three calibration data sets were taken for three separate calibration polynomials. The order of the calibration polynomials has also been increased to the 4th power to better fit the data over the expanded operating range.

The three polynomial, generated form the three sets of calibration data shown in (4.4-1), (4.4-2) and (4.4-3). The three calibration curves for the long-range test are shown in Figures 4.9, 4.11 and 4.13. The solid lines are constructed by linearly interpolating the verification data taken at positions indicated by the asterisks. The dashed lines, on the other hand, are generated by (4.4-1), (4.4-2) and (4.4-3) respectively. Figures 4.10, 4.12

and 4.14 show the error between the predicted and the actual position from the three calibration polynomials.

$$\text{Location (mm)} = -0.0226\,\text{Pstat}^4 + 0.8816\,\text{Pstat}^3 - 12.7834\,\text{Pstat}^2 + 110.6928\,\text{Pstat} - 18.0409 \qquad (4.4\text{-}1)$$

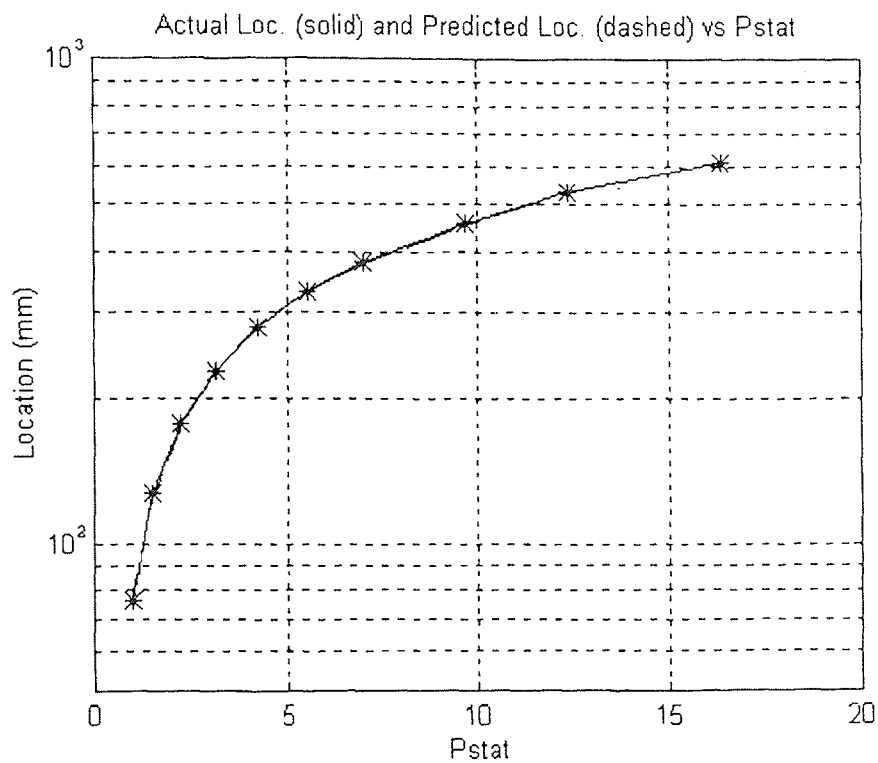


Figure 4.9 Calibration curve #1 for the long-range test using Pstat.

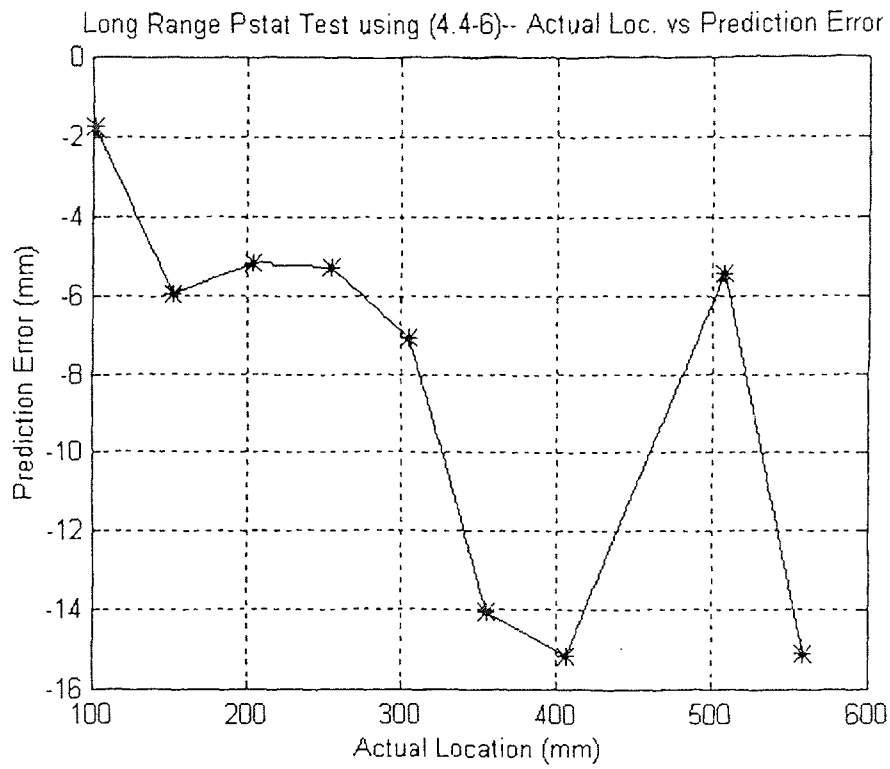


Figure 4.10 Location prediction error from the first calibration polynomial.

Mean (Error) = -8.3 mm

Std (Error) = 5.1 mm

$$\text{Location (mm)} = -0.0164\,\text{Pstat}^4 + 0.6944\,\text{Pstat}^3 - 11.1875\,\text{Pstat}^2 + 107.6895\,\text{Pstat} - 15.9818 \qquad (4.4\text{-}2)$$

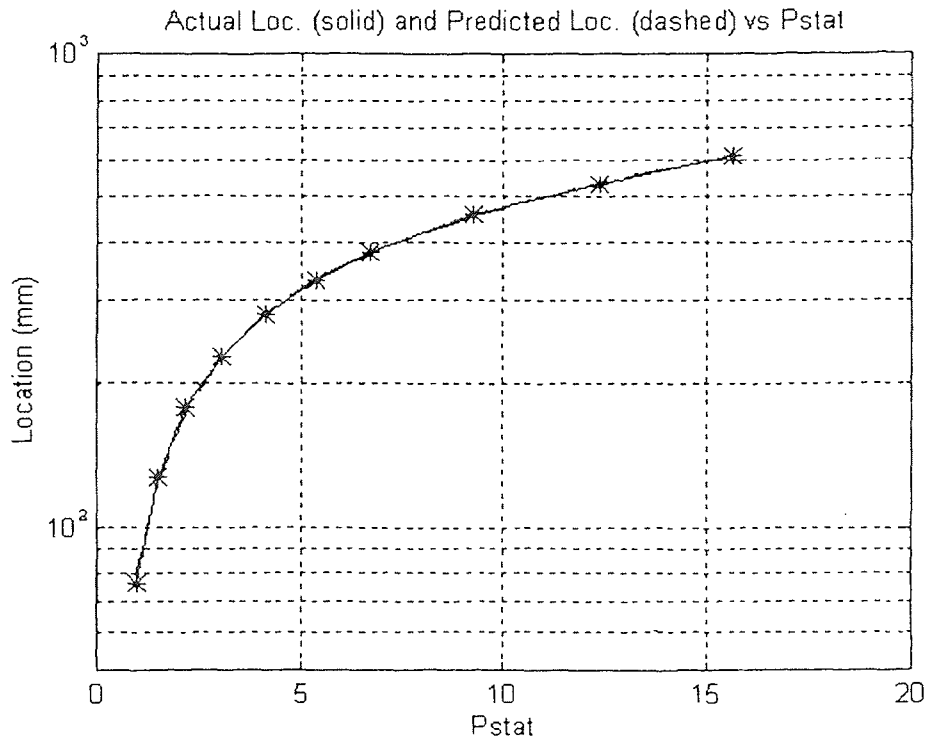


Figure 4.11 Calibration curve #2 for the long-range test using Pstat.

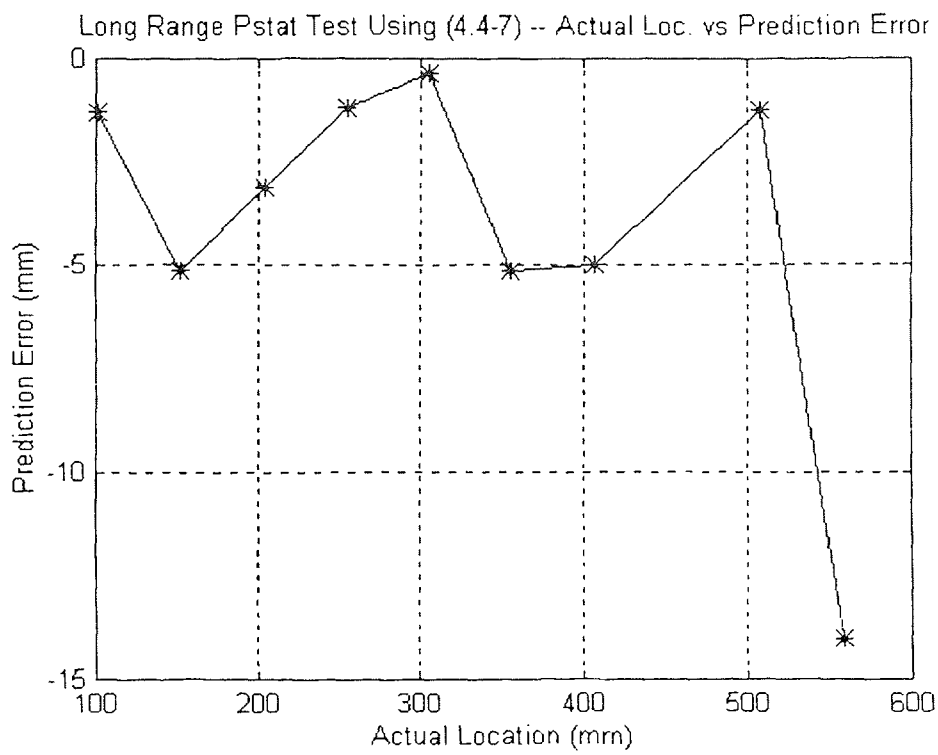


Figure 4.12 Location prediction error from the second calibration polynomial.

Mean (Error) = 4.0 mm

Std (Error) = 4.2 mm

$$\text{Location (mm)} = -0.0175\ \text{Pstat}^4 + 0.7356\ \text{Pstat}^3 - 11.7290\ \text{Pstat}^2 + 110.2947\ \text{Pstat} - 18.6185 \quad (4.4\text{-}3)$$

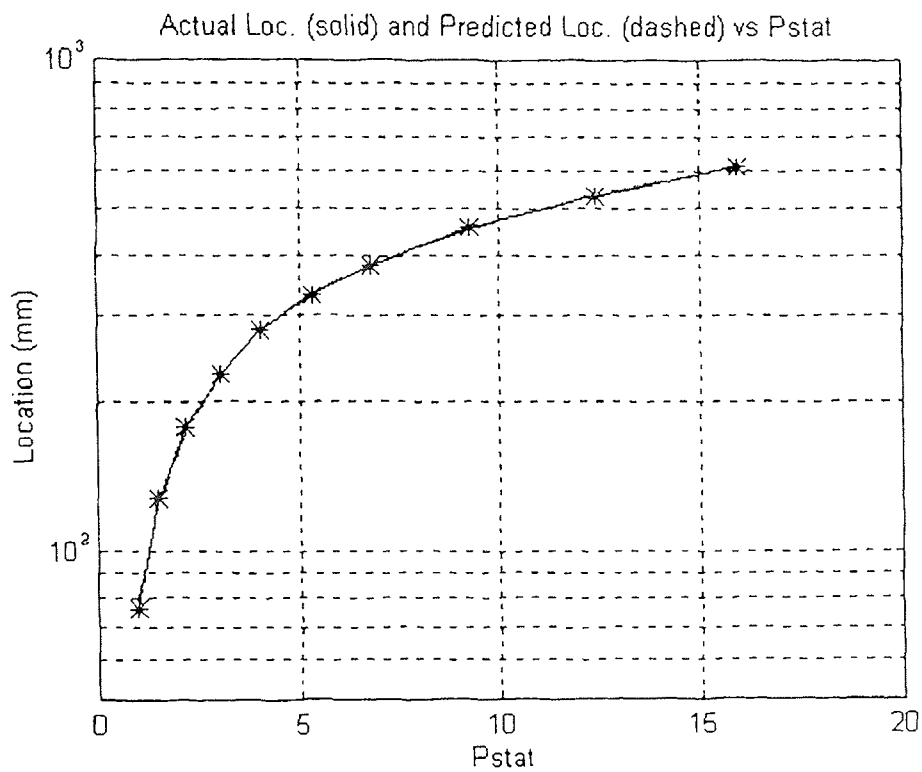


Figure 4.13 Calibration curve #3 for the long-range test using Pstat.

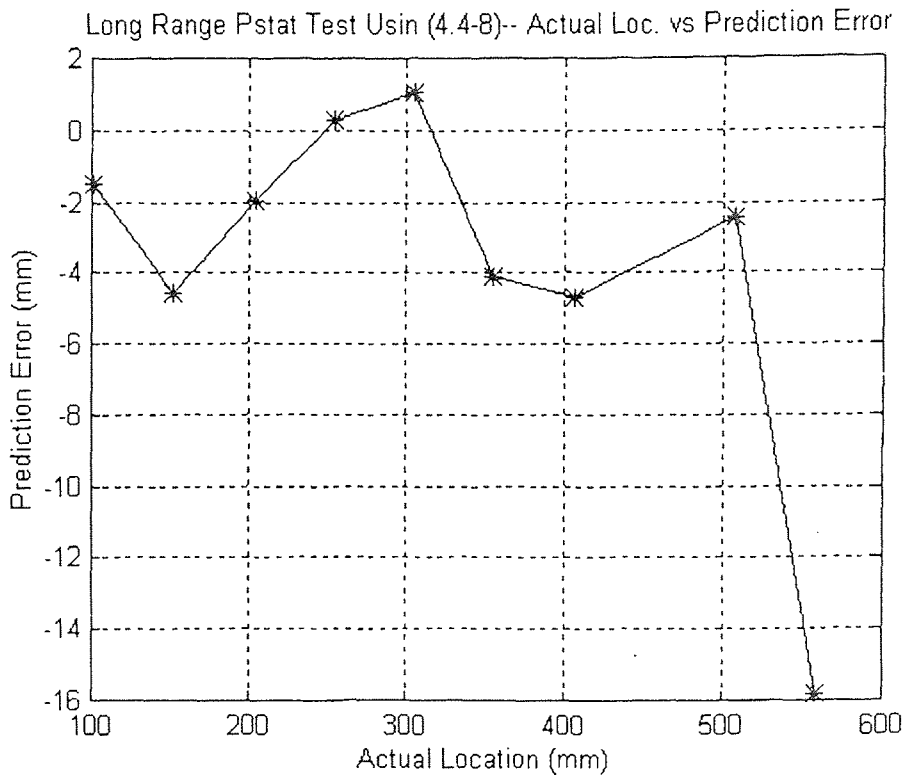


Figure 4.14 Location prediction error from the third calibration polynomial.

Mean (Error) = -3.8 mm

Std (Error) = 5.0 mm

From the above three standard deviations (5.1 mm, 4.2 mm, and 5.0 mm) the location of the mobile sensor module can be predicted to within an uncertainty of approximately 5 mm with the long range test using Pstat.

4.5 Long-Range Test Using Ppin

The long range test is repeated using the laser diode output power signal (Apin) as the normalization signal instead of Amove. This will free up the stationary sensor module

which was previously used as the normalization signal, for use as another mobile sensor module. The test is otherwise the same as the previous long-range test.

For this test, two calibration polynomials were generated, as shown in (4.5-1) and (4.5-2). The tests of the calibration polynomials are shown in Figures 4.15 and 4.17. Note that the coefficients for the calibration polynomials when using Ppin are significantly different from the coefficients for the calibration polynomials when using Pstat: (4.4-1), (4.4-2) and (4.4-3). This is the result of a difference in the amplitudes of the signals used to normalize the intensity of the incident light on the mobile sensor module.

$$\text{Location (mm)} = -0.0012\,\text{Ppin}^4 + 0.0976\,\text{Ppin}^3 - 3.0817\,\text{Ppin}^2 + 55.1808\,\text{Ppin} - 0.7590 \qquad (4.5\text{-}$$

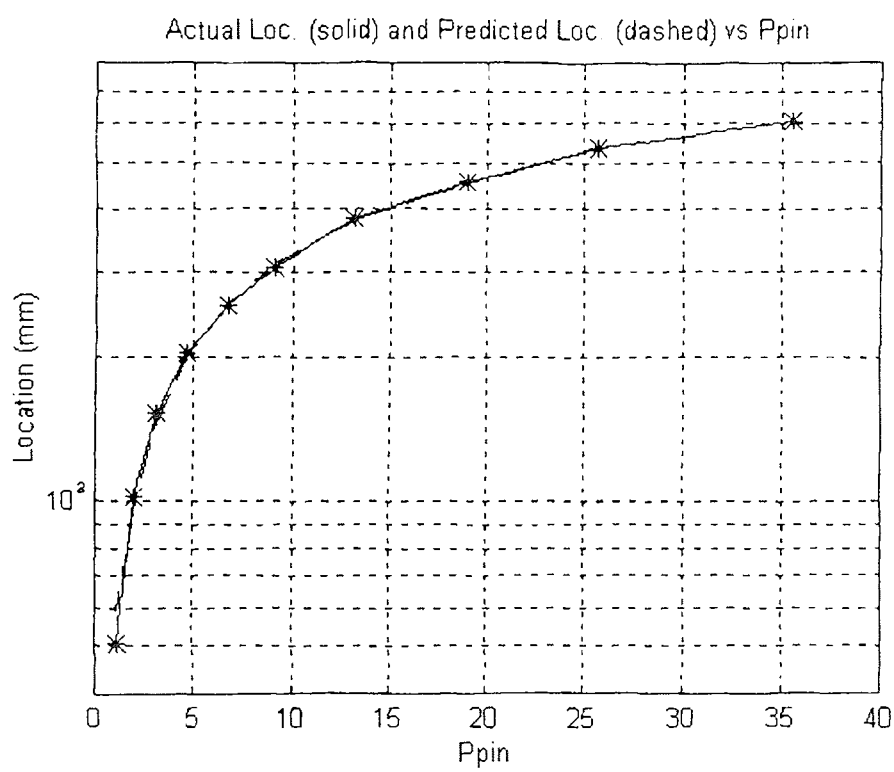


Figure 4.15 Calibration curve #1 for the long-range test using Ppin.

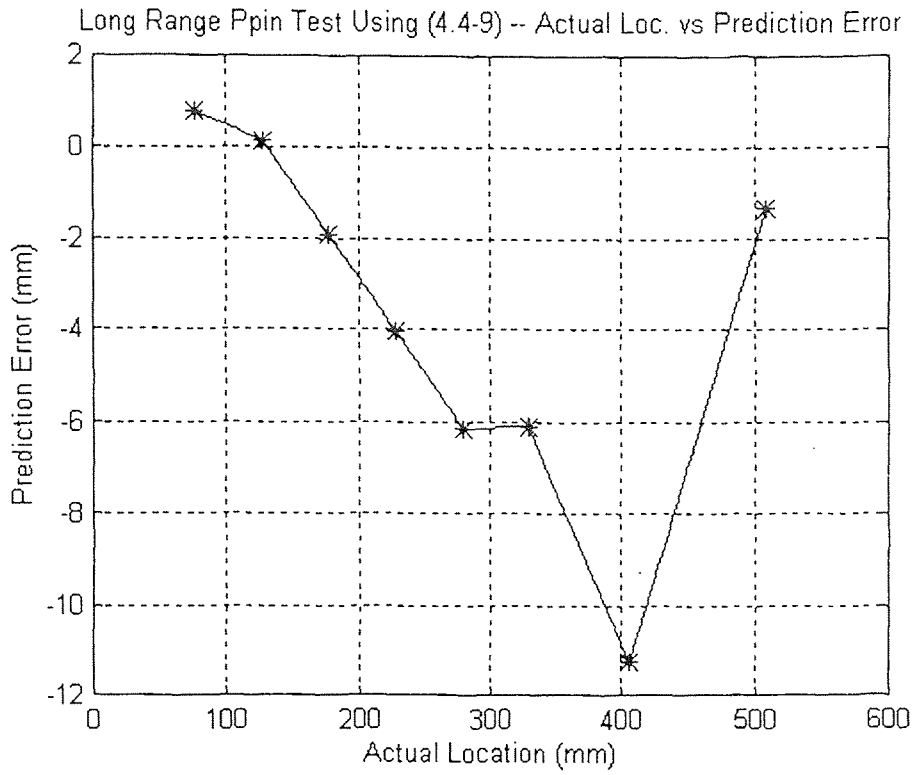


Figure 4.16 Location prediction error from the first calibration polynomial.

Mean (Error) = 3.7 mm

Std (Error) = 4.0 mm

$$\text{Location (mm)} = -0.0013\ \text{Ppin}^4 + 0.1083\ \text{Ppin}^3 - 3.3126\ \text{Ppin}^2 + 56.7829\ \text{Ppin} - 3.7504 \qquad (4.5\text{-}2)$$

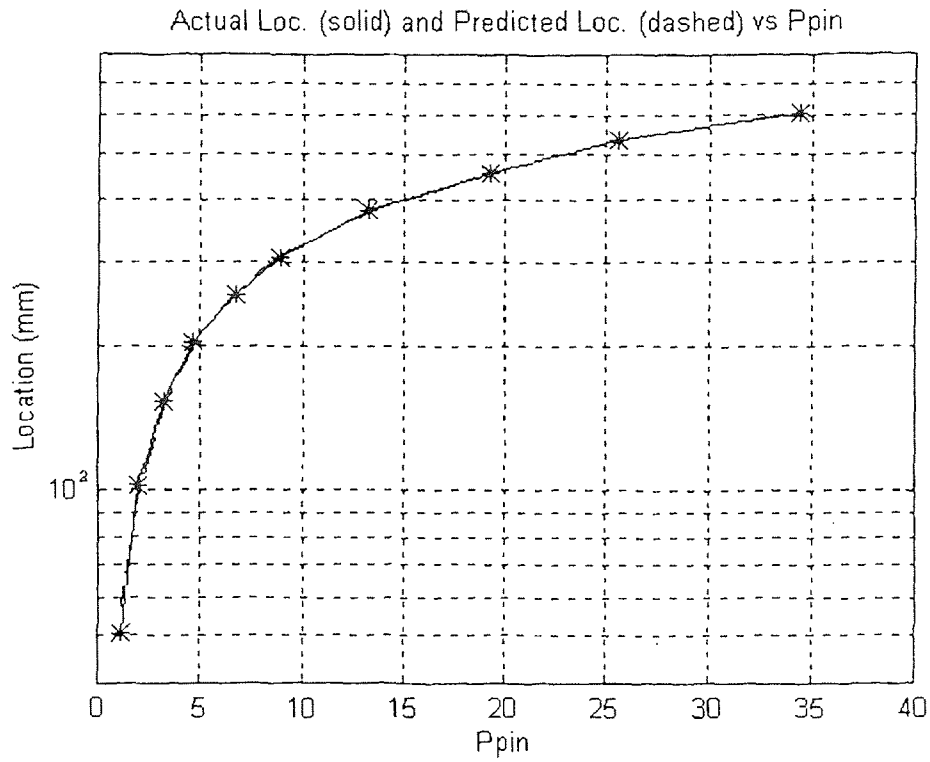


Figure 4.17 Calibration curve #2 for the long-range test using Ppin.

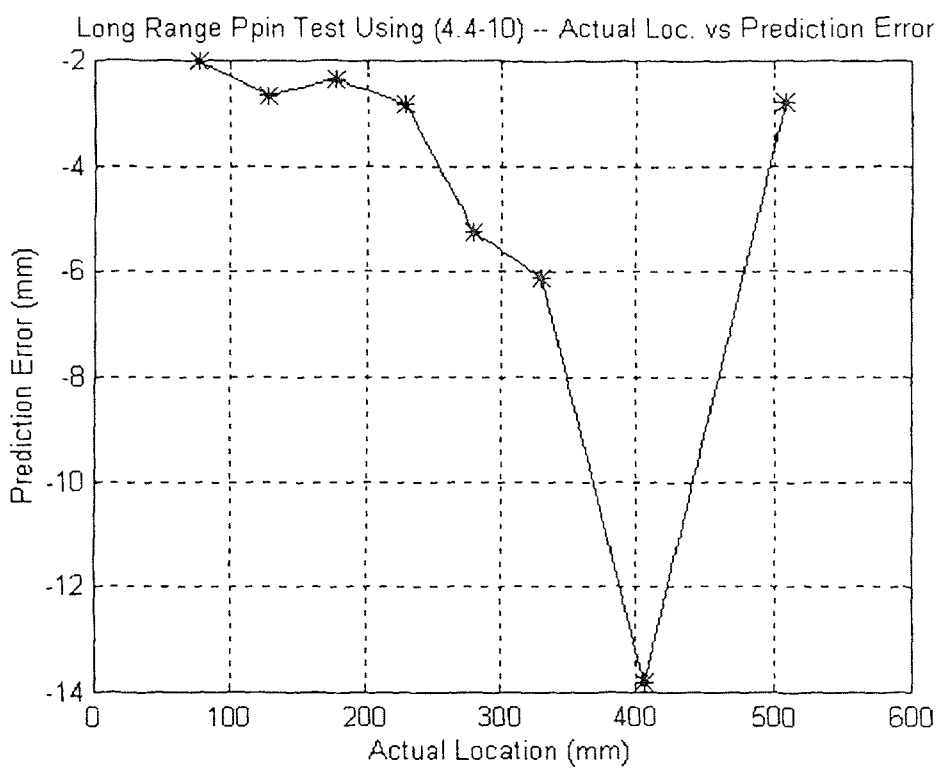


Figure 4.18 Location prediction error from the second calibration polynomial.

Mean (Error) = 4.7 mm

Std (Error) = 3.9 mm

From the above three standard deviations (4.0 mm, and 3.9 mm) the location of the mobile sensor module can be predicted to within an uncertainty of approximately 4 mm with the long range test using Ppin, which is approximately the same as when using Pstat with the long range test.

4.6 Summary of Test Results

Table 4.2 shows a summary of the test results discussed in this chapter.

Table 4.2 Summary of test results.

Test	Location Repeatability (+/- mm)	Range (mm)	Resolution (mm)
Short range w/ Pstat	0.02	25.4	0.5
Long range w/ Pstat	2	600	5
Long range w/ Ppin	2	600	4

From Table 4.2, it is clear that the ability of the system to predict the location of the mobile sensor module is much greater when used in the short range test where the resolution is approximately 0.5 mm that with the long range test where the resolution is approximately 5 mm. Some possible explanations are as follows:

1) The location repeatability for the short range test, when the verner table is used, is 0.02 mm while the position repeatability for the long range test, when the mobile sensor module is positioned by hand, is 2 mm.

2) The amplitude of the position signal from the mobile sensor module is small when at the maximum distance from the laser diode during the long range test (600 mm). This will lower the signal-to-noise ratio and hence the sensor resolution.

3) When the mobile sensor module is in the maximum distance range from the laser diode (200 mm to 600 mm) the amount of background noise signal, the majority of which comes from the room lights, increases.

CHAPTER 5

FLEXIBLE BEAM EXPERIMENT

5.1 Control Implementation

The objective of the flexible beam control experiment is to position the end of the flexible beam at a desired location, in minimum time and with minimum oscillation, using the output position signal from the sensor module as the position feedback. Refer to Chapter 2 for the discussion of the hardware implementation of the sensor module and Section 2.6 for a diagram of the flexible beam setup. The three performance indices used to monitor the relative responses of the control experiment for a variety of control laws and control gains are: 1) $\Sigma E(i)^2$ sum of the squared position error, 2) the final steady state position error of the end of the flexible beam, and 3) the 5% settling time of the flexible beam. Refer to Section 3.4.2 for a description of the four control laws implemented.

To perform each test, the end of the beam is located at +/- 20 cm from the neutral command position, as the initial positions. The beam is then commanded to move to the neutral position (0 mm) by the control program which also records the performance indices previously mentioned.

5.2 Test Results

The results of these four control laws are now summarized.

5.2.1 Position Control with Constant Gain

The first control law implemented was position control. See (3.4-1) and Figure 3.22 for a review of the position control law.

To test the effects of the position control law the position gain K_P was varied and the three performance indices used to monitor the relative responses of the control experiment were recorded, plotted, and compared. The test results are tabulated in Tables 5.2 and 5.3 for, respectively, initial conditions of 20 cm and -20 cm.

The following is a listing of typical time plots of the Y and U signals:

Table 5.1: Listing of typical time plots of the Y and U signals for K_P tests.

Figure	Initial Position (cm)	K_P
5.1	+20	5
5.2	+20	10
5.3	+20	20
5.4	+20	30
5.5	-20	5
5.6	-20	10
5.7	-20	20
5.8	-20	30

Table 5.2 Summary of results: Initial position = +20 cm, K_P control

K_P	Avg $\Sigma E(i)^2$	Std $\Sigma E(i)^2$	Avg Final Error (cm)	Std Final Error (cm)	Avg Settling Time (sec.)	Std Settling Time (sec.)
5	46.8	1.8	-2.18	0.37	N. A.	N. A.
10	108.2	12.4	6.10	0.79	N. A.	N. A.
20	75.2	6.37	-1.78	0.63	N. A.	N. A.
30	N. A.	N. A.	N. A.	N. A.	N. A.	N. A.

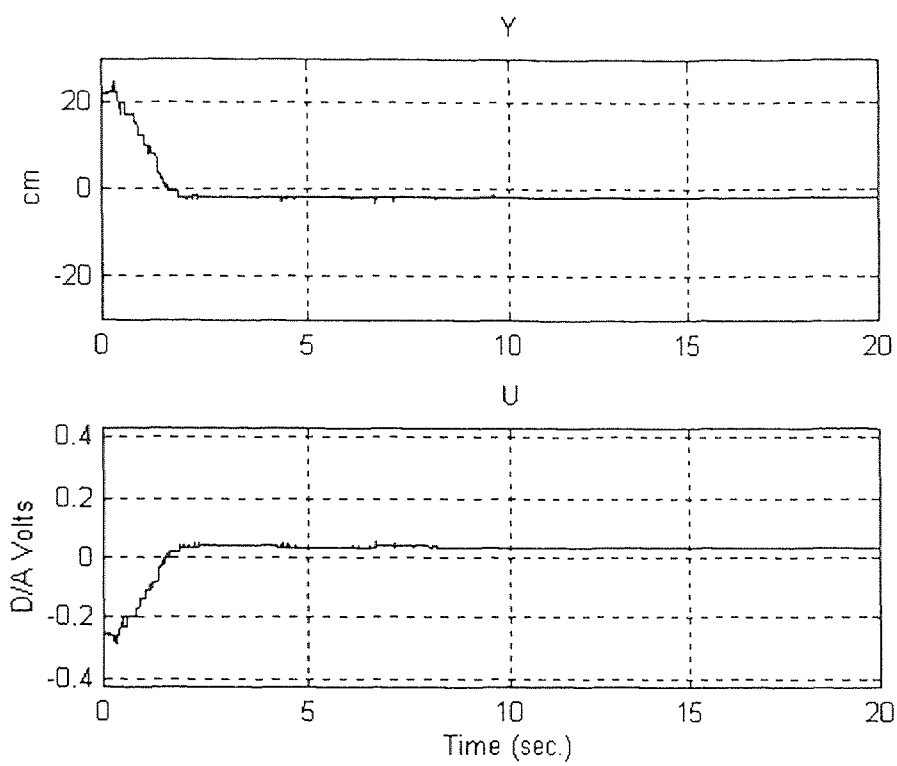


Figure 5.1 Typical Y and U time plots for $K_P = 5$.

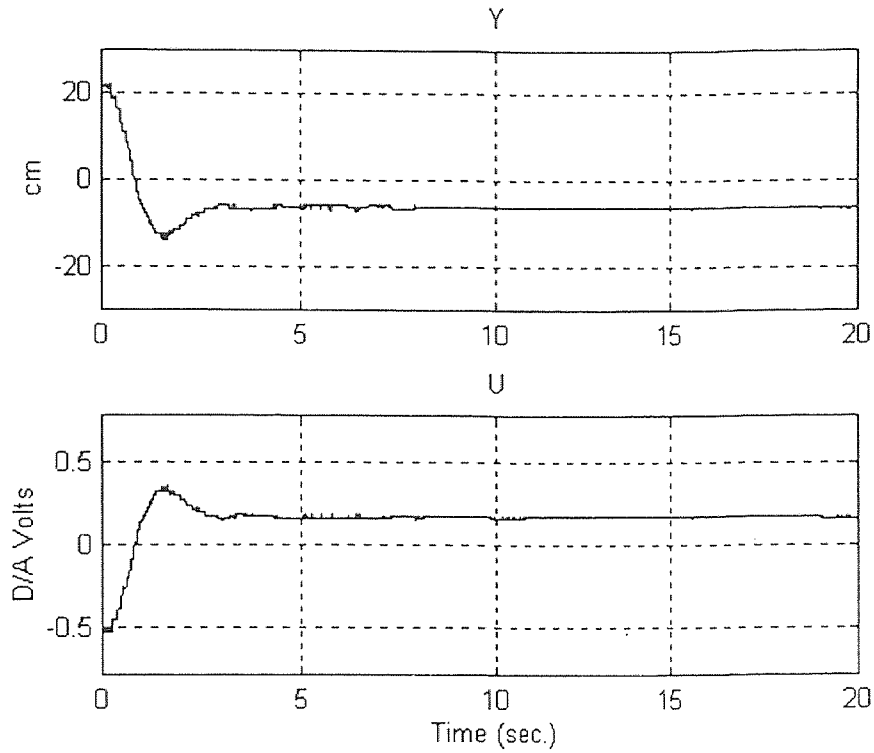


Figure 5.2 Typical Y and U time plots for $K_P = 10$.

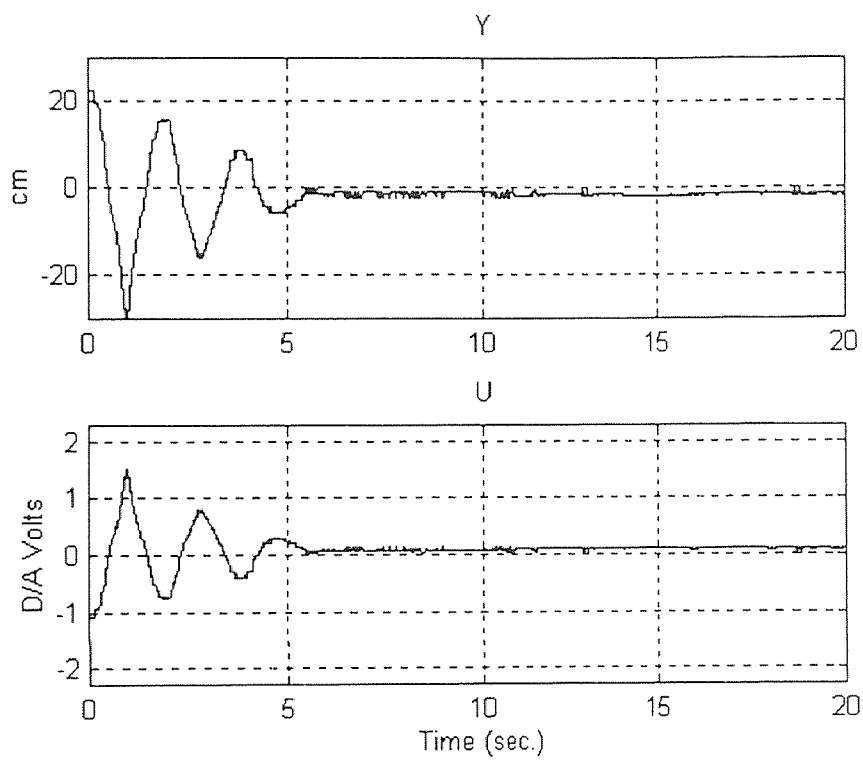


Figure 5.3 Typical Y and U time plots for $K_P = 20$.

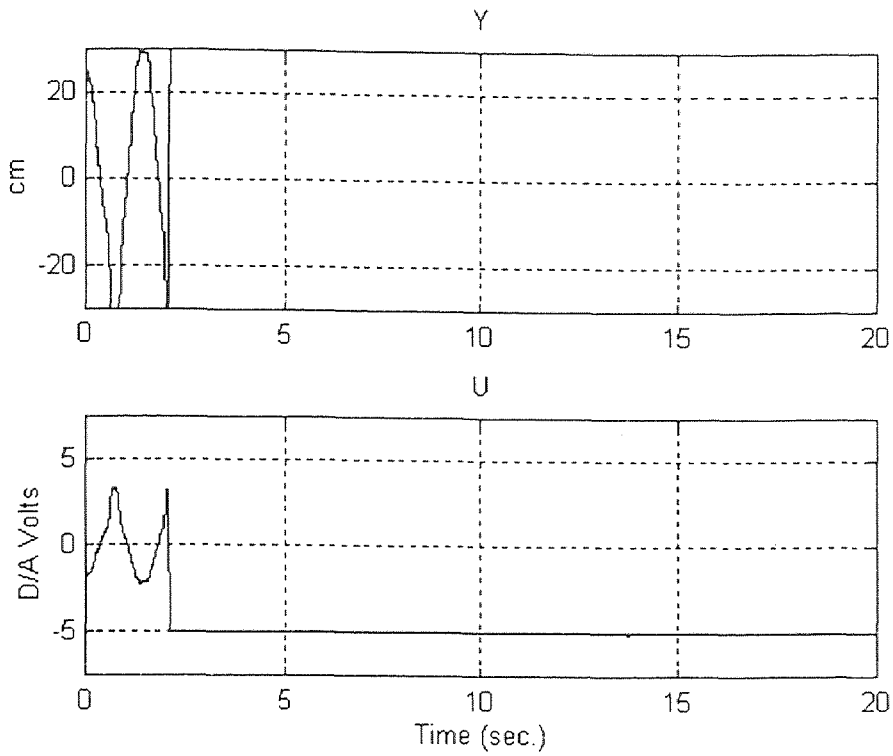


Figure 5.4 Typical Y and U time plots for $K_P = 30$.

From the summary of the K_P test results in Table 5.2, when the initial position is +20 cm, $K_P = 20$ produces the best results when measured with performance indices $\Sigma E(i)^2$ and final position error. Due to the friction in the motor positioning system, no value of K_P, when the initial position is +20 cm will produce a final error within the +/- 5% range (1.0 cm), which can be seen as the lack of settling times in Table 5.2. If K_P is too much greater than 20, the system will become unstable as shown in Figure 5.4.

Table 5.3 Summary of results: Initial position = -20 cm, K_P control

K_P	Avg $\Sigma E(i)^2$	Std $\Sigma E(i)^2$	Avg Final Error (cm)	Std Final Error (cm)	Avg Settling Time (sec.)	Std Settling Time (sec.)
5	314	20.8	12.30	0.41	N. A.	N. A.
10	47.6	2.7	4.12	0.38	N. A.	N. A.
20	29.4	5.8	1.00	0.73	9.0	3.9
30	87.2	9.6	-0.52	1.10	10.5	1.3

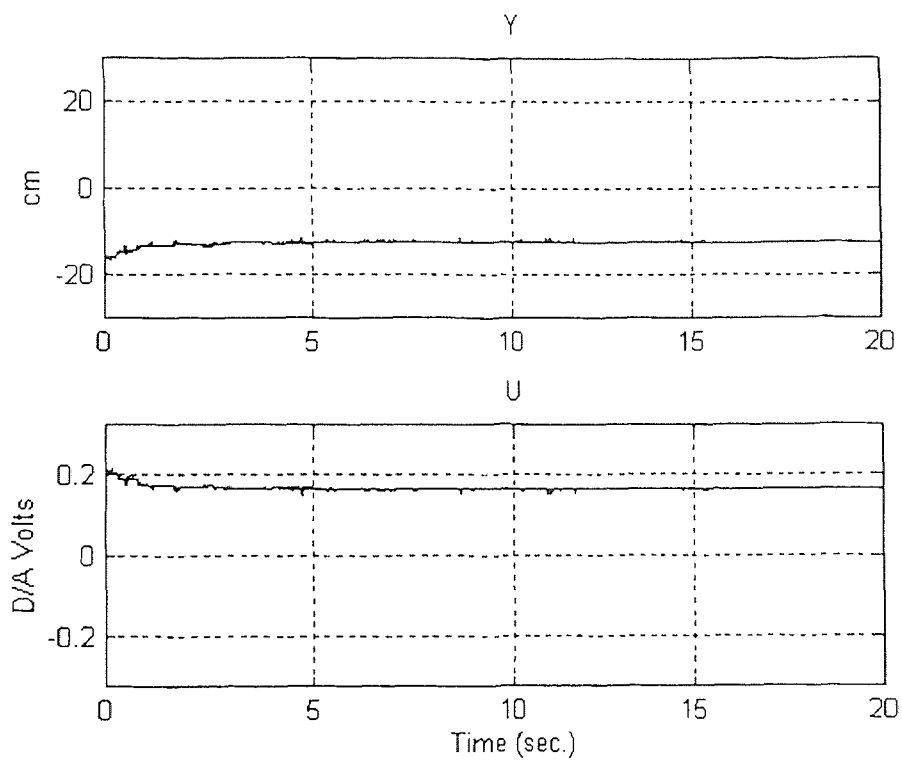


Figure 5.5 Typical Y and U time plots for $K_P = 5$.

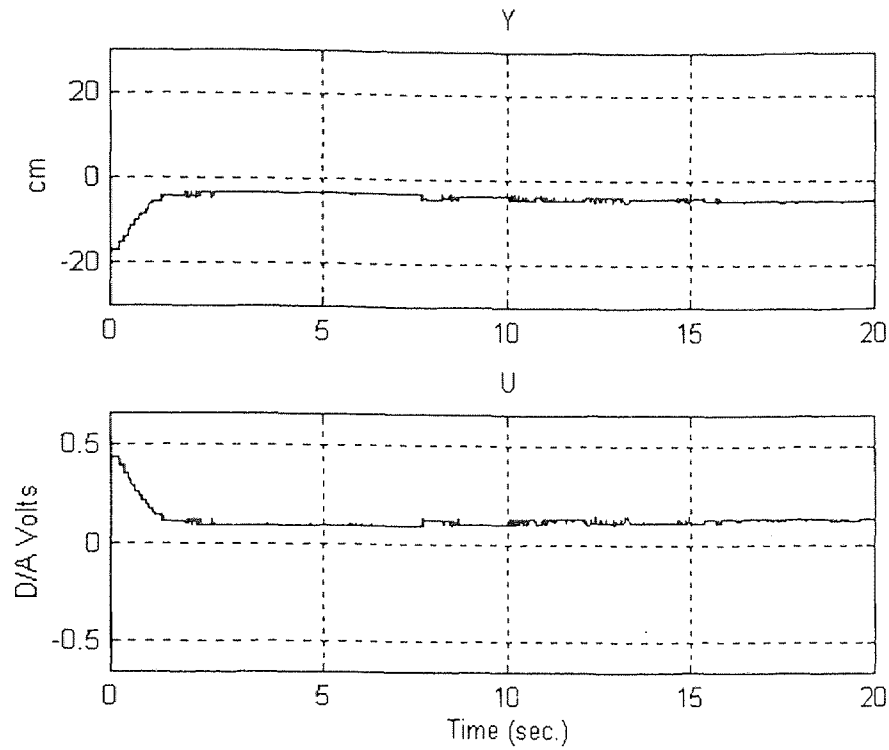


Figure 5.6 Typical Y and U time plots for $K_P = 10$.

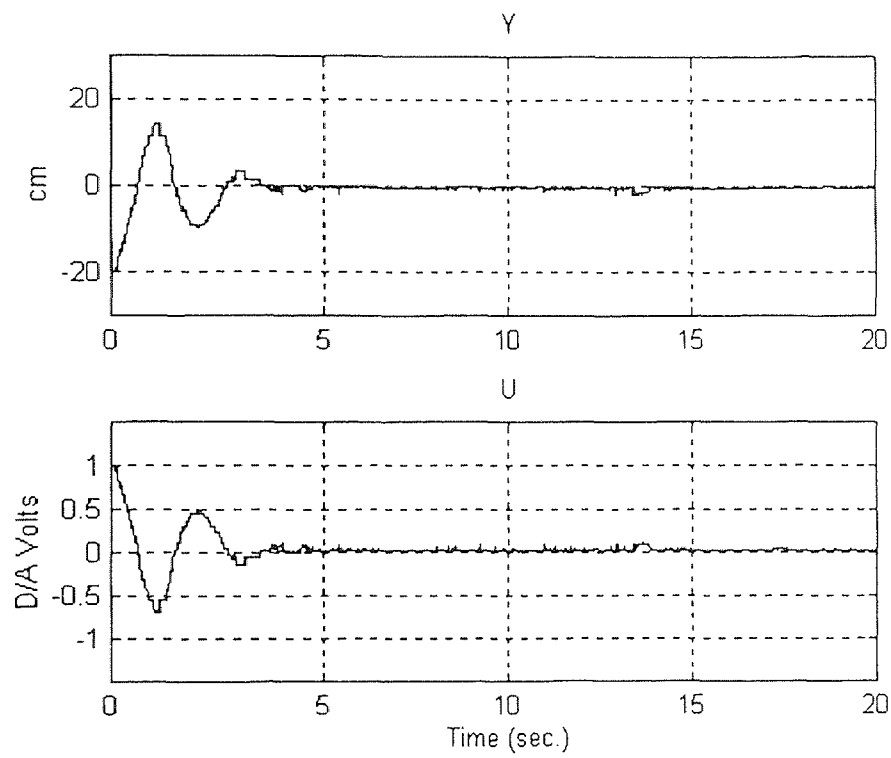


Figure 5.7 Typical Y and U time plots for $K_P = 20$.

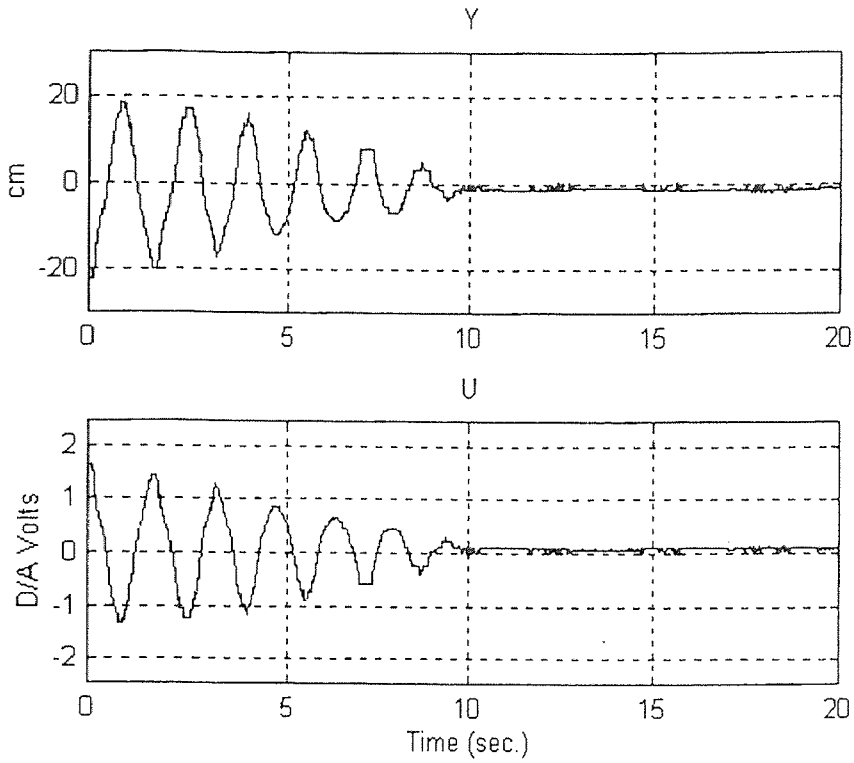


Figure 5.8 Typical Y and U time plots for $K_P = 30$.

From the summary of the K_P test results in Table 5.3, when the initial position is -20 cm, $K_P = 20$ will produce the best results when measured with performance indices $\Sigma E(i)^2$ and settling time, however, $K_P = 30$ produces the best results when measured by the final position error.

5.2.2 Position and Derivative Control with Constant Gain

As discussed in Section 3.4.2, a position and derivative (K_P and K_D) control law was synthesized to improve the response of the flexible beam. See (3.4-4) and Figure 3.24 for a review of the position and derivative control law.

To test the effects of a variety of derivative gains, the K_P gain was set to 10 and the derivative gain K_D was varied. The results of the tests when the initial position of the

flexible beam is +20 cm are shown in Table 5.5 and when the initial position is -20 cm in Table 5.6. The following is a listing of typical time plots of the Y and U signals with K_P to 10:

Table 5.4: Listing of typical time plots of the Y and U signals for K_P K_D tests.

Figure	Initial Position (cm)	K_D
5.9	+20	5
5.10	+20	10
5.11	+20	20
5.12	+20	30
5.13	-20	5
5.14	-20	10
5.15	-20	20
5.16	-20	30

Table 5.5 Summary of results: Initial position = +20 cm, K_P K_D control, K_P = 10

K_D	Avg $\Sigma E(i)^2$	Std $\Sigma E(i)^2$	Avg Final Error (cm)	Std Final Error (cm)	Avg Settling Time (sec.)	Std Settling Time (sec.)
5	62.0	2.3	-4.14	0.43	N. A.	N. A.
10	46.4	12.4	-2.66	1.54	N. A.	N. A.
20	37.2	2.2	-0.31	1.44	2.3	0.2
30	37.0	1.8	-0.27	0.60	2.5	0.3

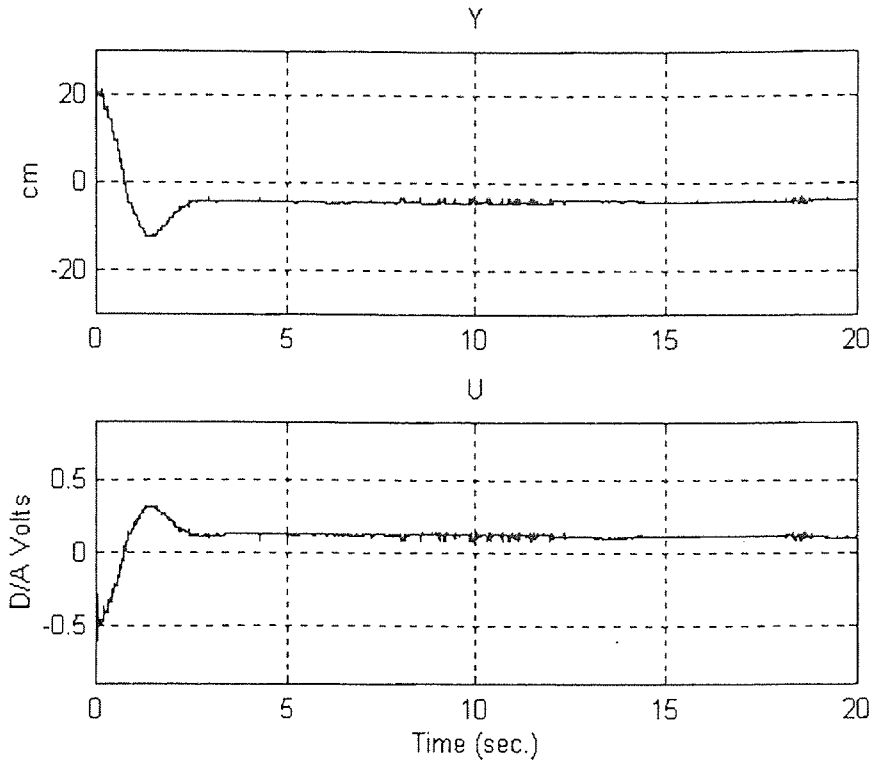


Figure 5.9 Typical Y and U time plots for $K_D = 5$.

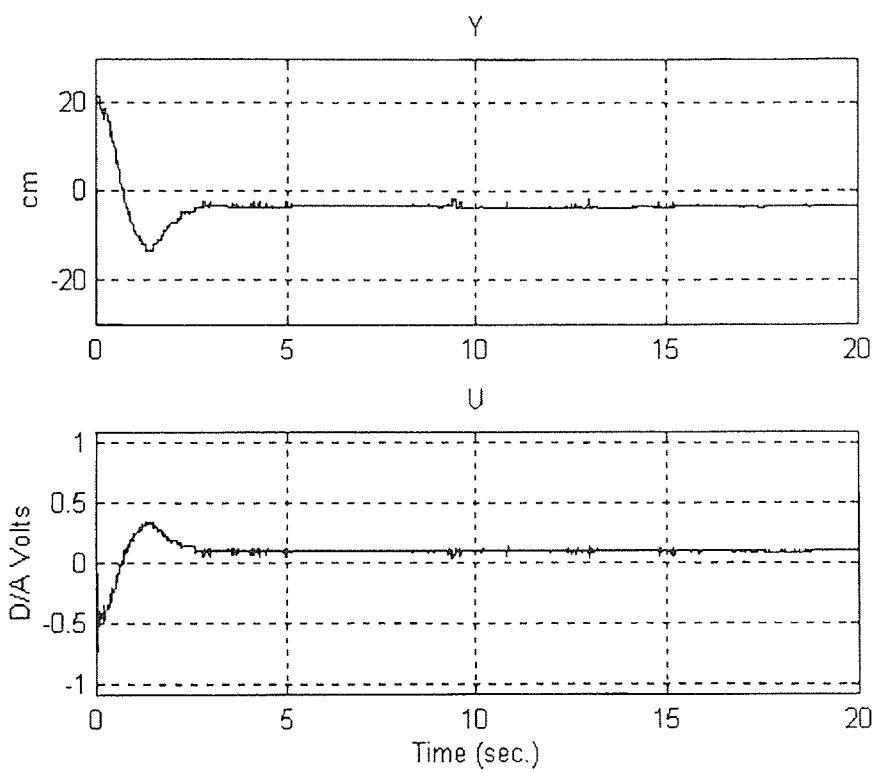


Figure 5.10 Typical Y and U time plots for $K_D = 10$.

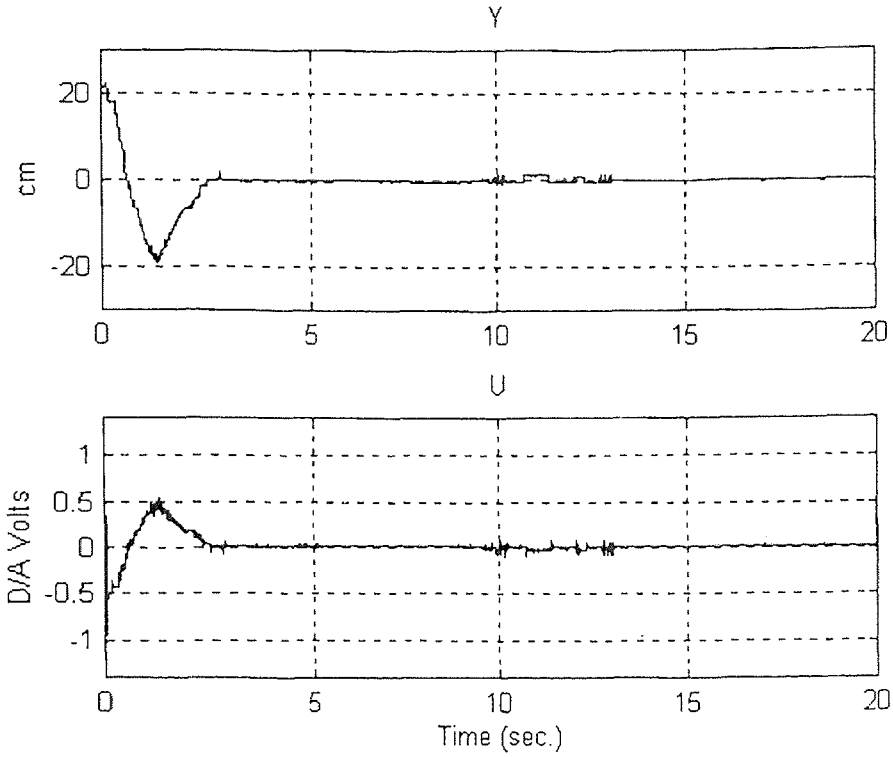


Figure 5.11 Typical Y and U time plots for $K_D = 20$.

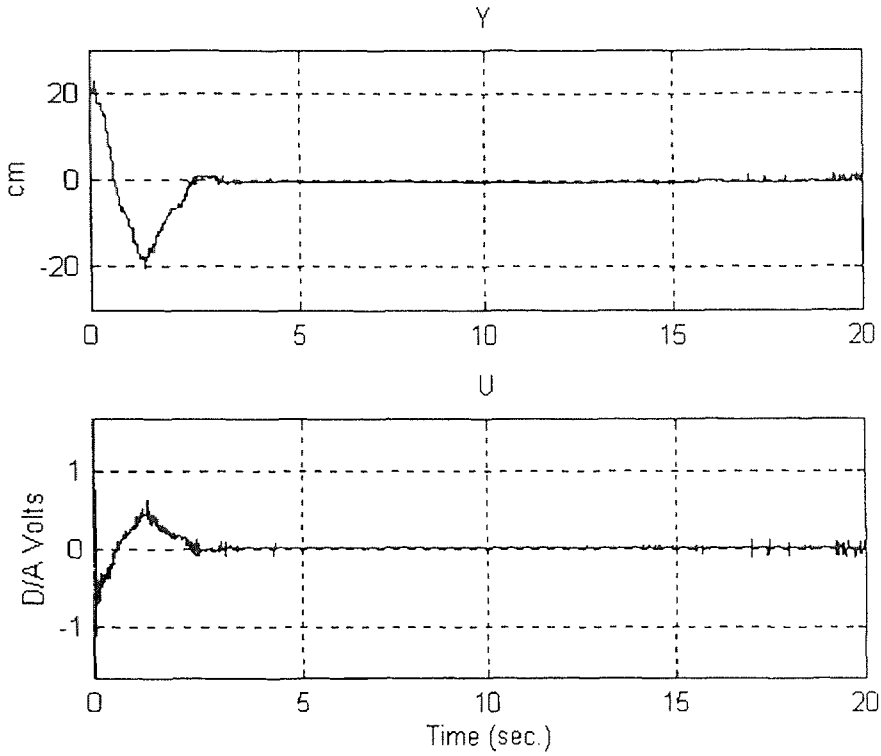


Figure 5.12 Typical Y and U time plots for $K_D = 30$.

From the summary of test results in Table 5.5, $K_D = 30$ produces the best results, which are slightly better than those obtained with $K_D = 20$, when measured with performance indices $\Sigma E(i)^2$ and the final position error. $K_D = 20$ produces the best results when measured by the settling time, but only slightly better than $K_D = 30$. Therefore, it is likely that the optimum K_D, when K_P is to 10 and the initial position is +20 cm, is approximately 25.

Table 5.6 Summary of results: Initial position = -20 cm, K_P K_D control, $K_P = 10$

K_D	Avg $\Sigma E(i)^2$	Std $\Sigma E(i)^2$	Avg Final Error (cm)	Std Final Error (cm)	Avg Settling Time (sec.)	Std Settling Time (sec.)
5	27.6	4.5	-1.94	0.66	N. A.	N. A.
10	21.8	1.7	-1.09	0.26	N. A.	N. A.
20	22.7	1.5	-0.68	0.65	2.2	0.3
30	20.3	1.6	-0.06	0.90	2.2	0.7

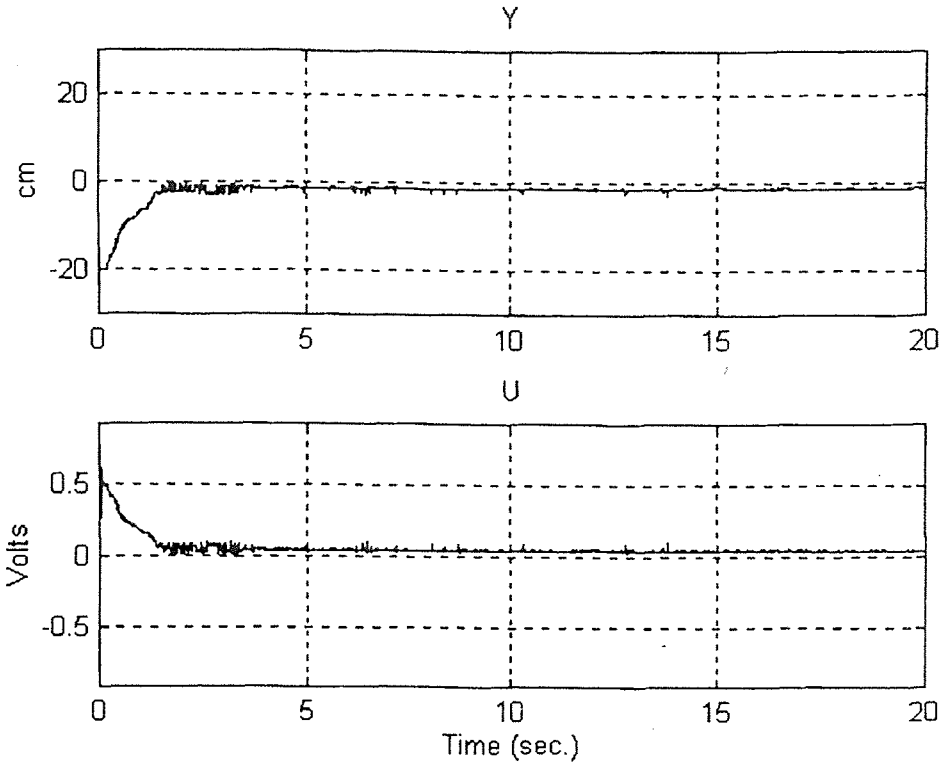


Figure 5.13 Typical Y and U time plots for $K_D = 5$.

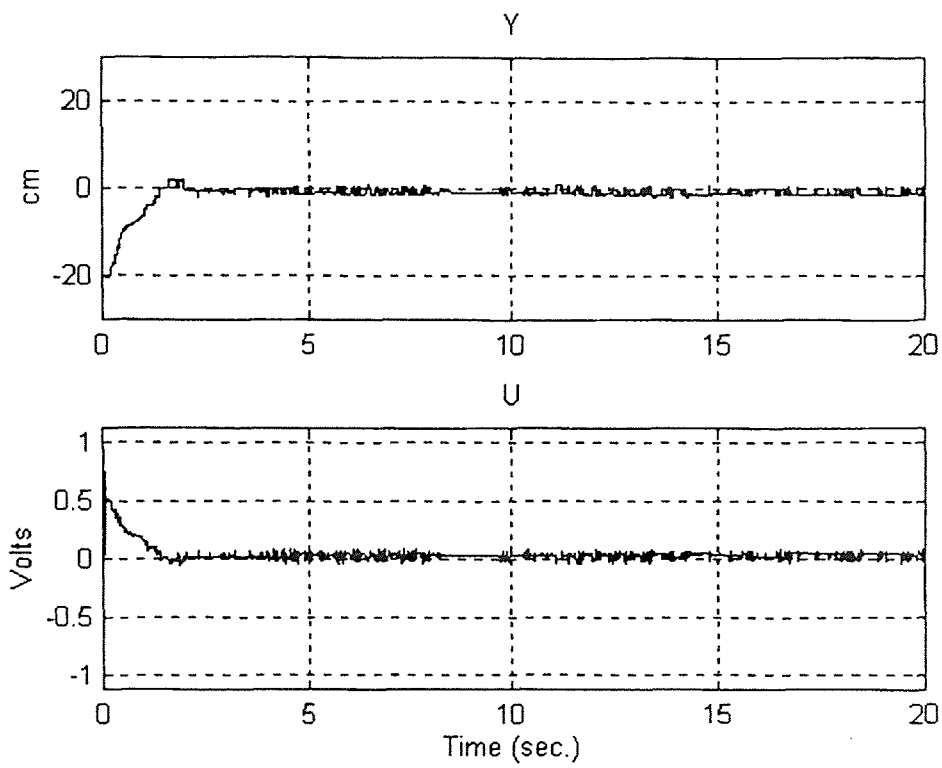


Figure 5.14 Typical Y and U time plots for $K_D = 10$.

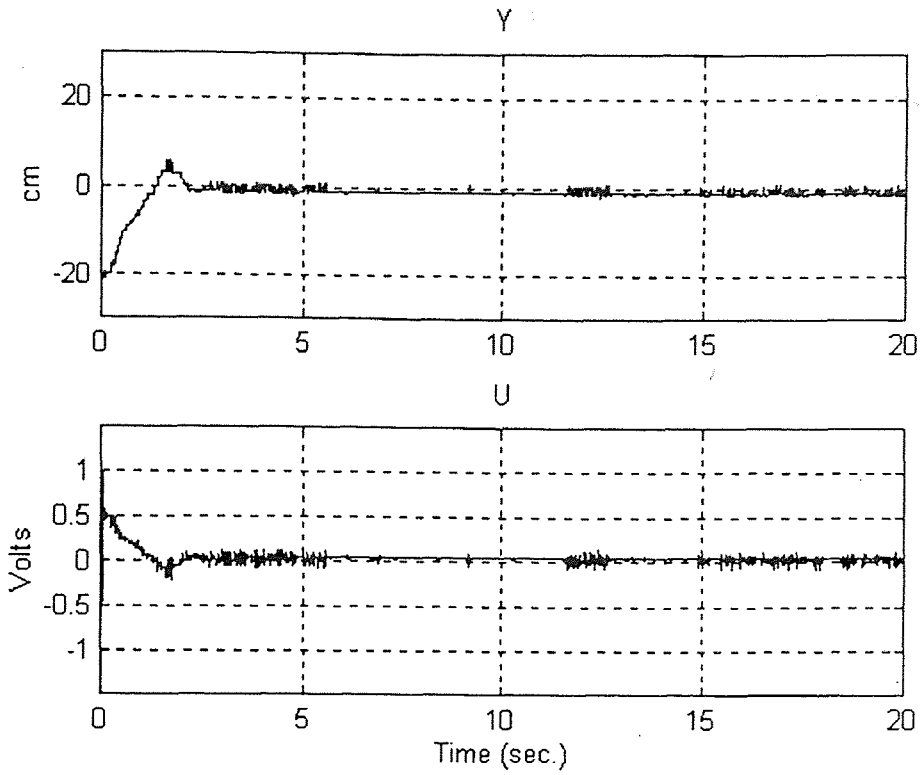


Figure 5.15 Typical Y and U time plots for $K_D = 20$.

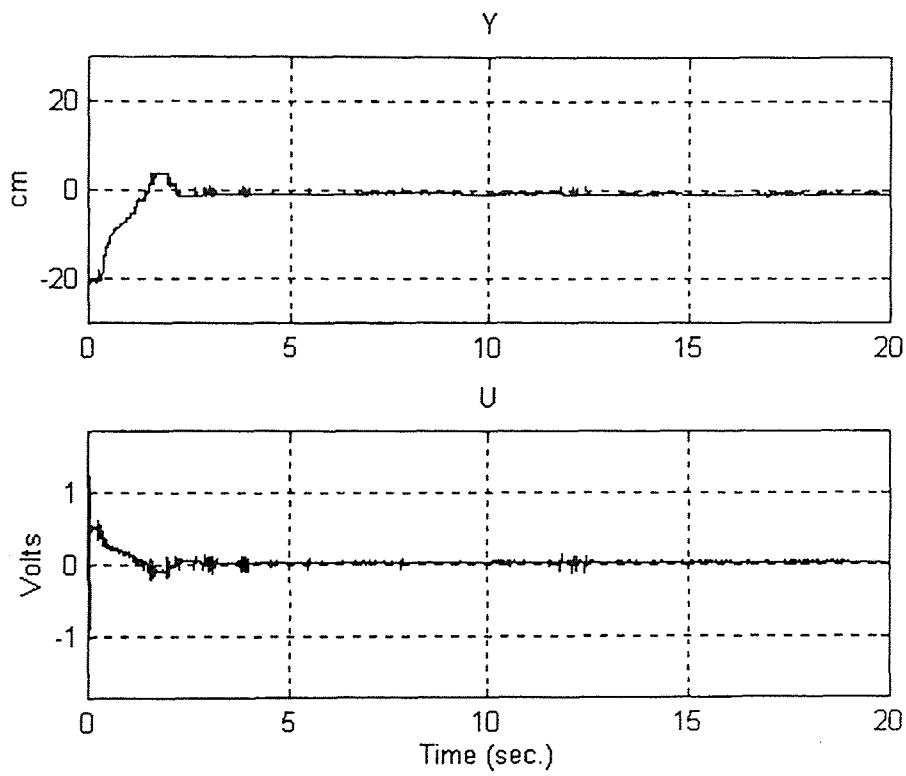


Figure 5.16 Typical Y and U time plots for $K_D = 30$.

From the summary of test results in Table 5.6, $K_D = 30$ produces the best results when measured with performance indices $\Sigma E(i)^2$, the final position error, and the settling time.

5.2.3 Position, Derivative and Integral Control with Constant Gain

As discussed in Section 3.4.2, an integrator was synthesized to overcome the motor friction in the flexible beam setup and to improve the steady state error of the position of the beam. See (3.4-6) and Figure 3.26 for a review of the control law. The equation for the integrator is shown in (3.4-5).

To test the effects of a variety of integrator gains, the K_P and K_D gains were both set to 10 and the integrator gain K_I was varied. The results of the tests when the initial position of the flexible beam is +20 cm is shown in Table 5.8 and when the initial position is -20 cm is shown in Table 5.9.

The following is a listing of typical time plots of the Y and U signals with K_P and K_D set to 10:

Table 5.7: Listing of typical time plots of the Y and U signals for constant gains tests.

Figure	Initial Position (cm)	K_I
5.17	+20	0.005
5.18	+20	0.01
5.19	+20	0.02
5.20	+20	0.04
5.21	-20	0.005
5.22	-20	0.01
5.23	-20	0.02
5.24	-20	0.04

Table 5.8 Summary of results: Initial position = +20 cm,
Constant K_P K_D K_I control -- $K_P = 10$, $K_D = 10$.

K_I	Avg $\Sigma E(i)^2$	Std $\Sigma E(i)^2$	Avg Final Error (cm)	Std Final Error (cm)	Avg Settling Time (sec.)	Std Settling Time (sec.)
0.005	55.5	5.1	0.43	0.34	10.9	1.3
0.01	36.4	2.7	-2.07	0.19	N.A.	N.A.
0.02	32.6	2.1	-0.39	0.29	7.9	7.2
0.04	48.8	9.8	0.21	0.71	8.6	3.1

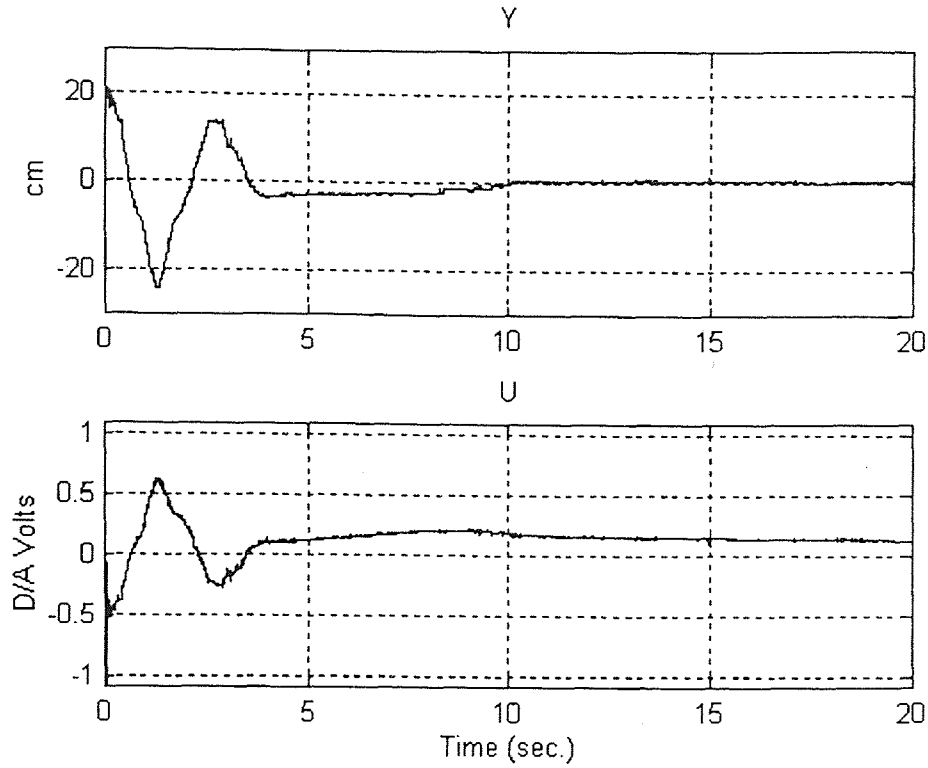


Figure 5.17: Typical Y and U time plots for $K_I = 0.005$ constant gain.

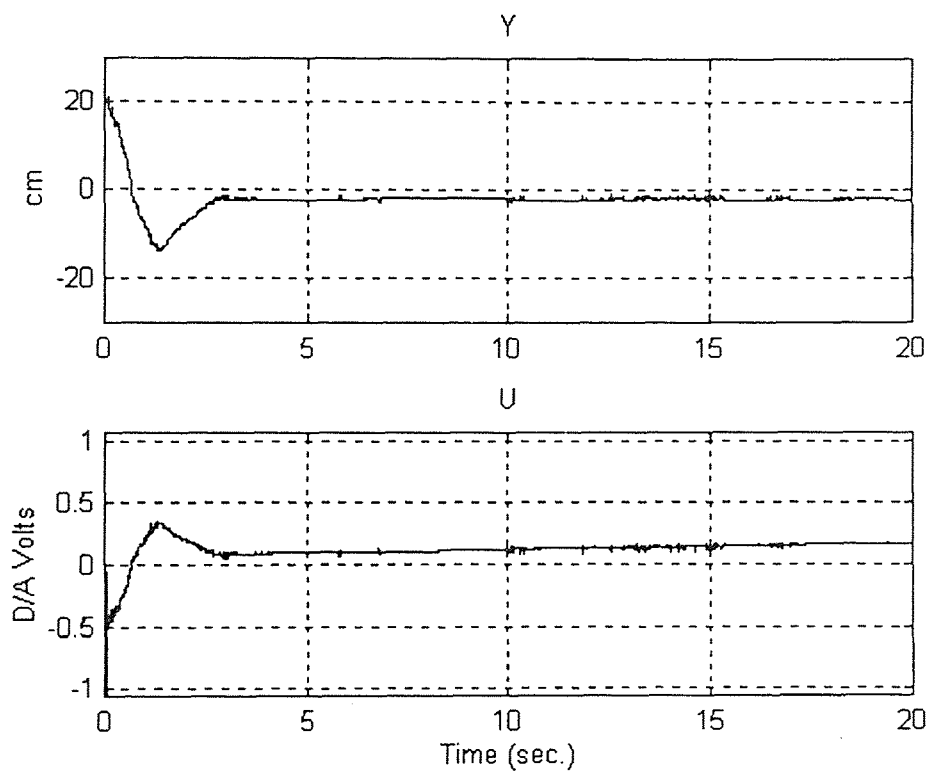


Figure 5.18: Typical Y and U time plots for $K_I = 0.01$ constant gain.

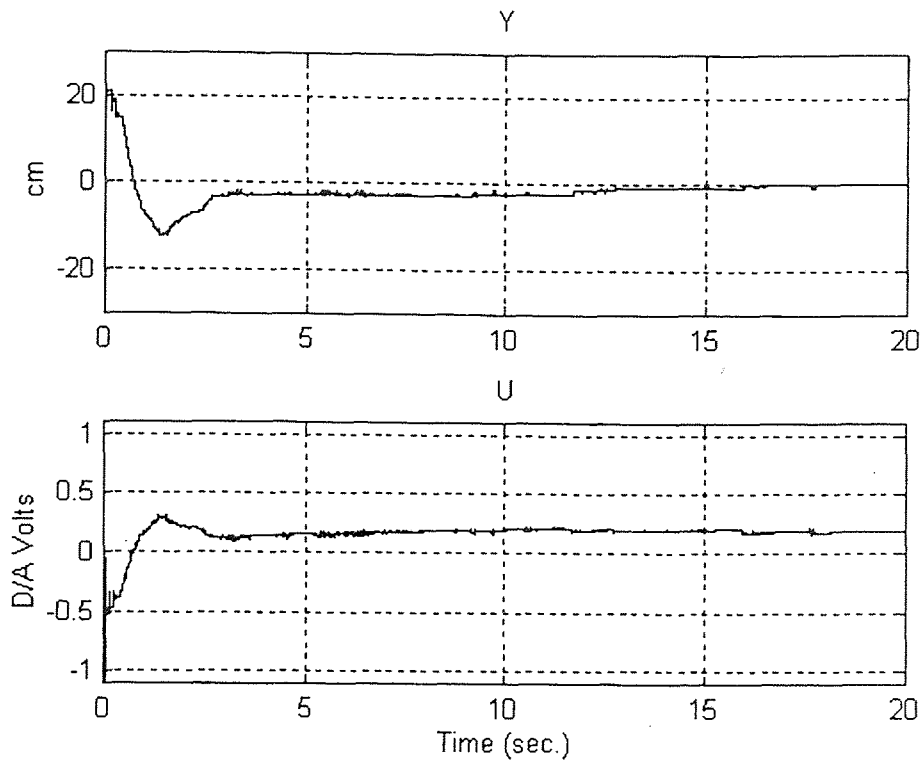


Figure 5.19: Typical Y and U time plots for $K_I = 0.02$ constant gain.

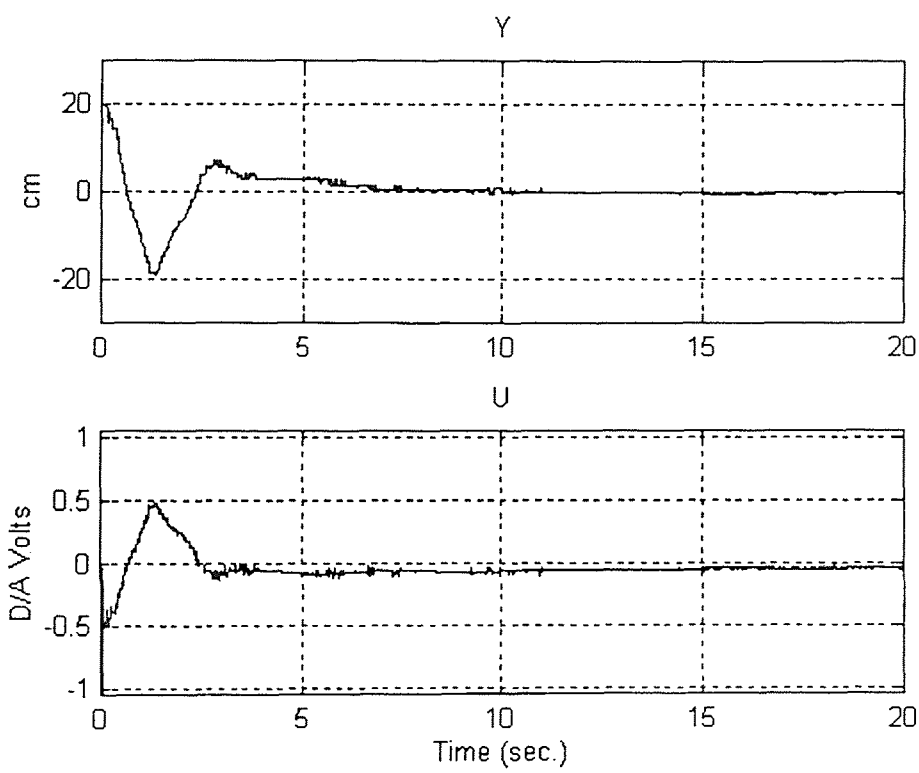


Figure 5.20: Typical Y and U time plots for $K_I = 0.04$ constant gain.

From the summary of test results in Table 5.8, $K_I = 0.02$ produces the best results when measured with performance indices $\Sigma E(i)^2$ and settling time but $K_I = 0.04$ produces the best results when measured by the final position error. Therefore, it is likely that the optimum K_I, when K_P and K_D are set to 10 and the initial position is +20 cm, is approximately 0.03.

Table 5.9 Summary of results: Initial position = -20 cm,
Constant K_P K_D K_I control -- $K_P = 10$, $K_D = 10$.

K_I	Avg $\Sigma E(i)^2$	Std $\Sigma E(i)^2$	Avg Final Error (cm)	Std Final Error (cm)	Avg Settling Time (sec.)	Std Settling Time (sec.)
0.005	22.5	0.6	-0.15	0.85	2.0	0.1
0.01	25.8	0.8	1.16	0.26	N. A.	N. A.
0.02	27.3	1.5	0.36	0.22	15.7	1.5
0.04	29.6	1.7	-0.12	0.34	7.8	4.4

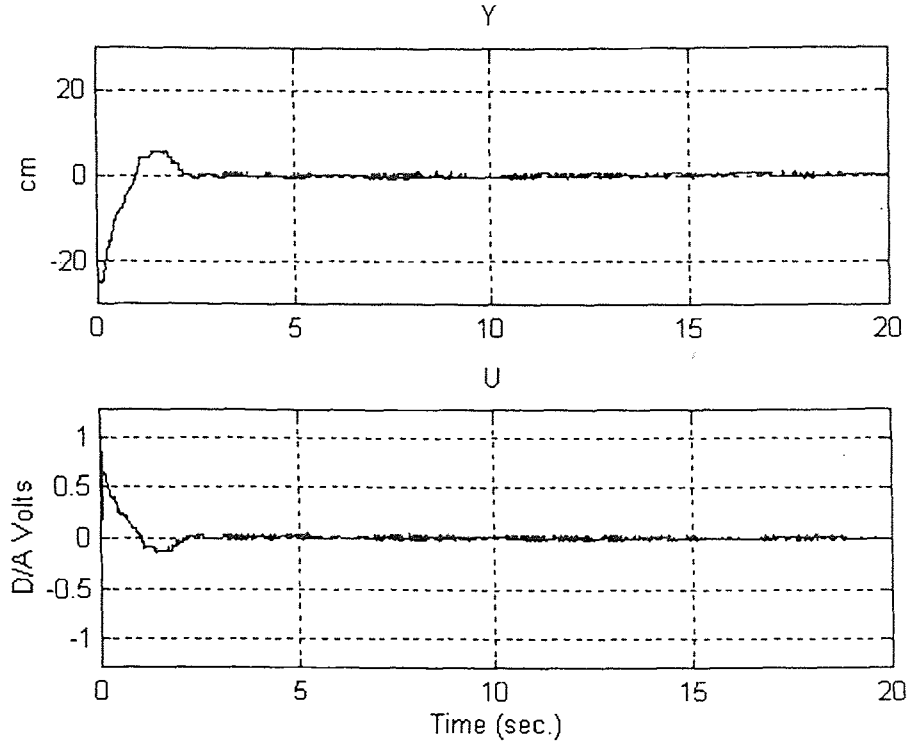


Figure 5.21: Typical Y and U time plots for $K_I = 0.005$ constant gain.

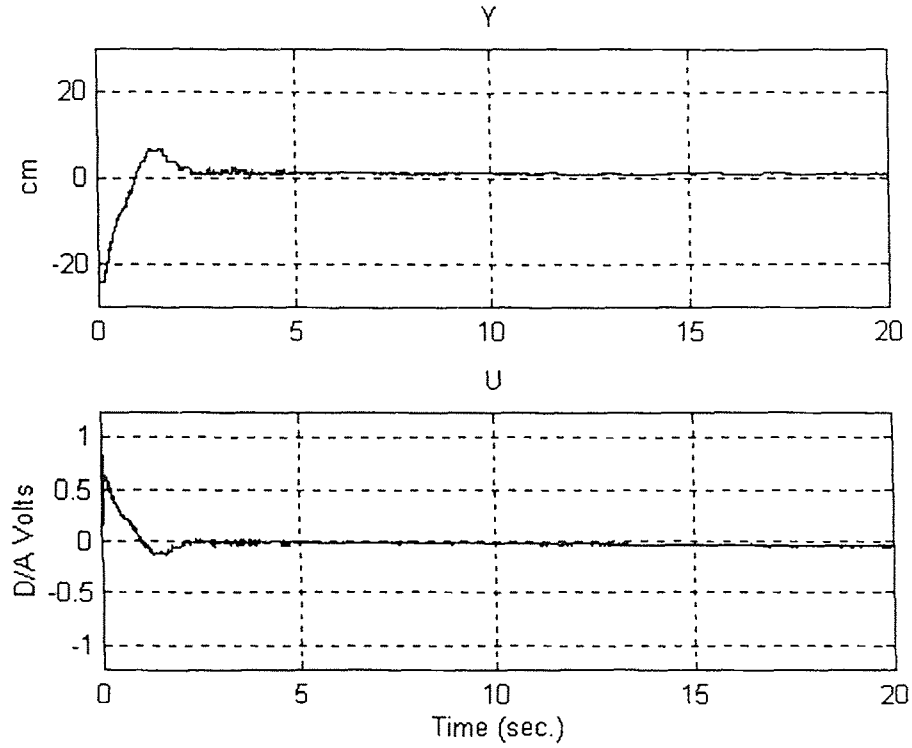


Figure 5.22: Typical Y and U time plots for $K_I = 0.01$ constant gain.

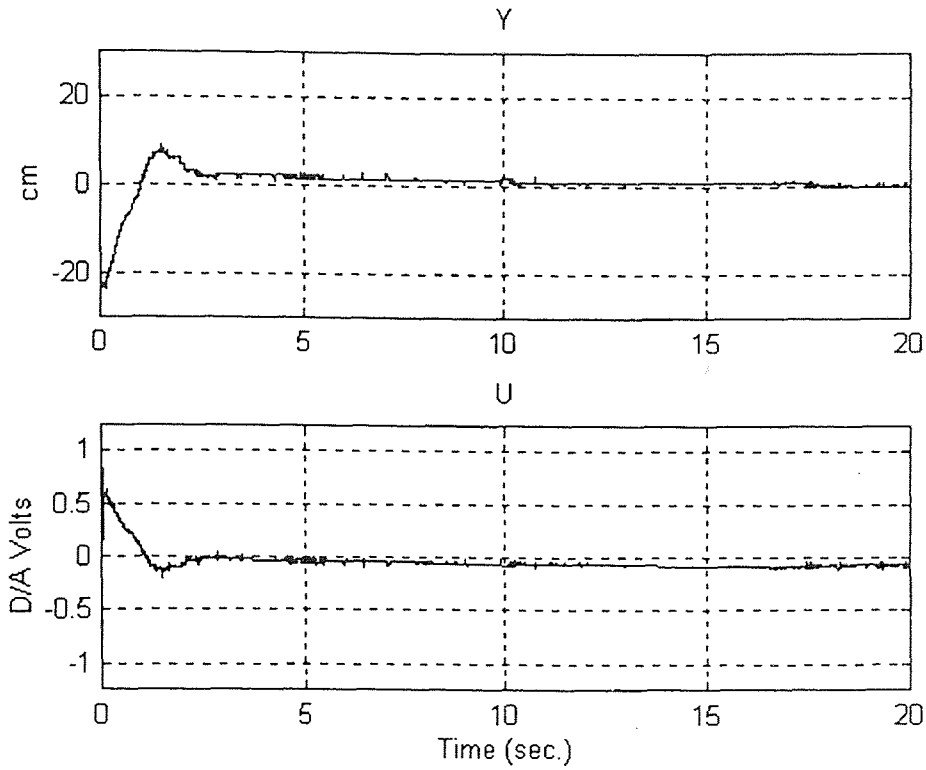


Figure 5.23: Typical Y and U time plots for $K_I = 0.02$ constant gair

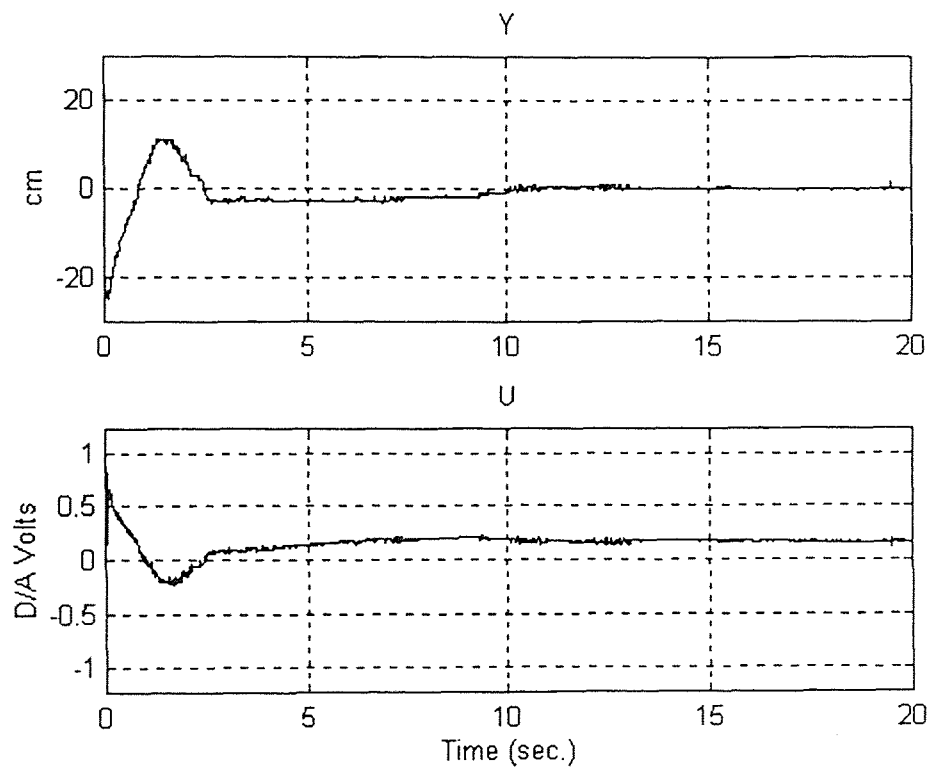


Figure 5.24: Typical Y and U time plots for $K_I = 0.04$ constant gai

From the summary of test results in Table 5.9, when the initial position is -20 cm, $K_I = 0.005$ produces the best results when measured with performance indices $\Sigma E(i)^2$ and settling time. However, due to friction and backlash in the DC motor system, the results when $K_I = 0.005$ may be unusually optimistic.

If the test results when $K_I = 0.005$ are disregarded, $K_I = 0.01$ produces the best results when measured with the performance index $\Sigma E(i)^2$ but $K_I = 0.04$ produces the best results when measured by the final position error and the settling time.

5.2.4 Position, Derivative and Integral Control with Variable Gain

To attempt to improve on the response of the flexible beam system with constant gain, a control law with variable gains was implemented as described in Chapter 3.4.2. The variable gain equations are shown in (3.4-7) to (3.4-14) and the block diagram of the variable gain control law is shown in Figure 3.27. The results of the tests when the initial position of the flexible beam is +20 cm is shown in Table 5.11 and when the initial position is -20 cm is shown in Table 5.12.

The following is a listing of typical time plots of the Y and U signals with K_P and K_D set to 10:

Table 5.10: Listing of typical time plots of the Y and U signals for variable gains tests.

Figure	Initial Position (cm)	K_I
5.25	+20	0.005
5.26	+20	0.01
5.27	+20	0.02
5.28	+20	0.04
5.29	-20	0.005
5.30	-20	0.01
5.31	-20	0.02
5.32	-20	0.04

Table 5.11 Summary of results: Initial position = +20 cm,
Variable K_P K_D K_I control -- $K_P = 10$, $K_D = 10$.

K_I	Avg $\Sigma E(i)^2$	Std $\Sigma E(i)^2$	Avg Final Error (cm)	Std Final Error (cm)	Avg Settling Time (sec.)	Std Settling Time (sec.)
0.005	63.0	5.6	0.26	0.14	9.4	1.1
0.01	44.0	4.6	-2.21	0.19	N.A.	N.A.
0.02	33.8	2.5	-0.31	0.31	3.0	0.1
0.04	45.8	0.8	0.10	0.30	4.1	0.2

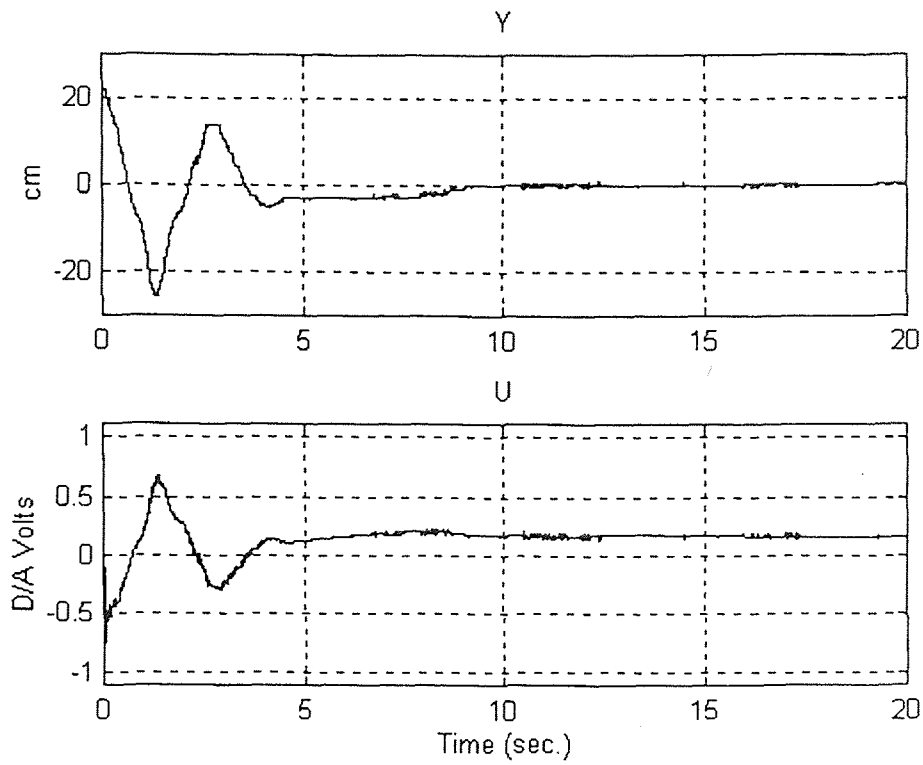


Figure 5.25: Typical Y and U time plots for $K_I = 0.005$ variable gain.

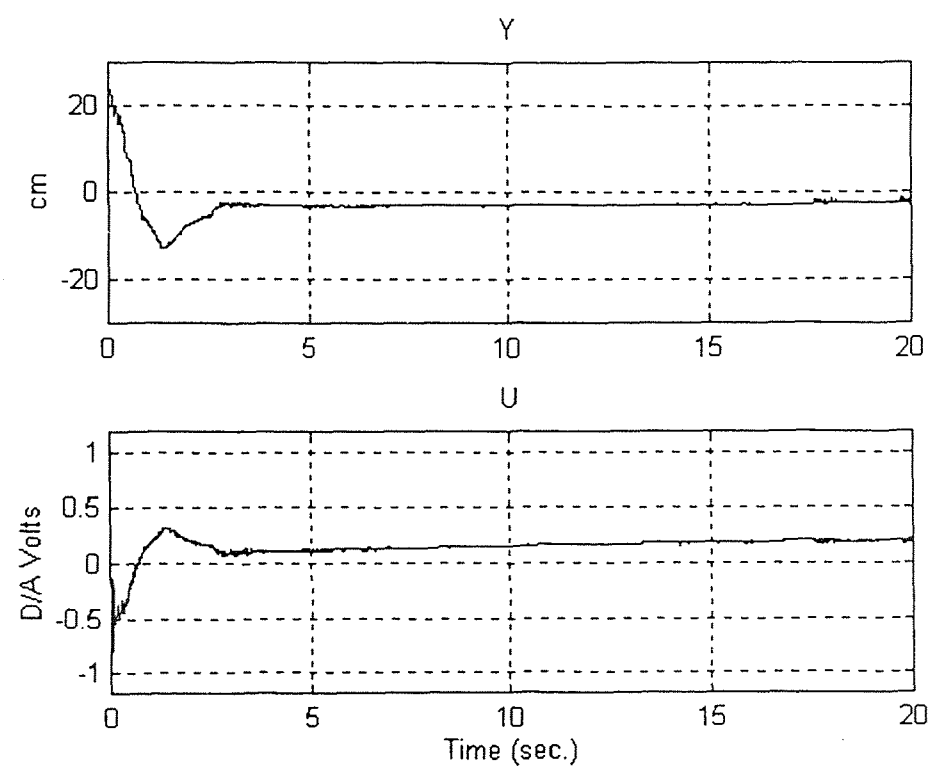


Figure 5.26: Typical Y and U time plots for $K_I = 0.01$ variable gain.

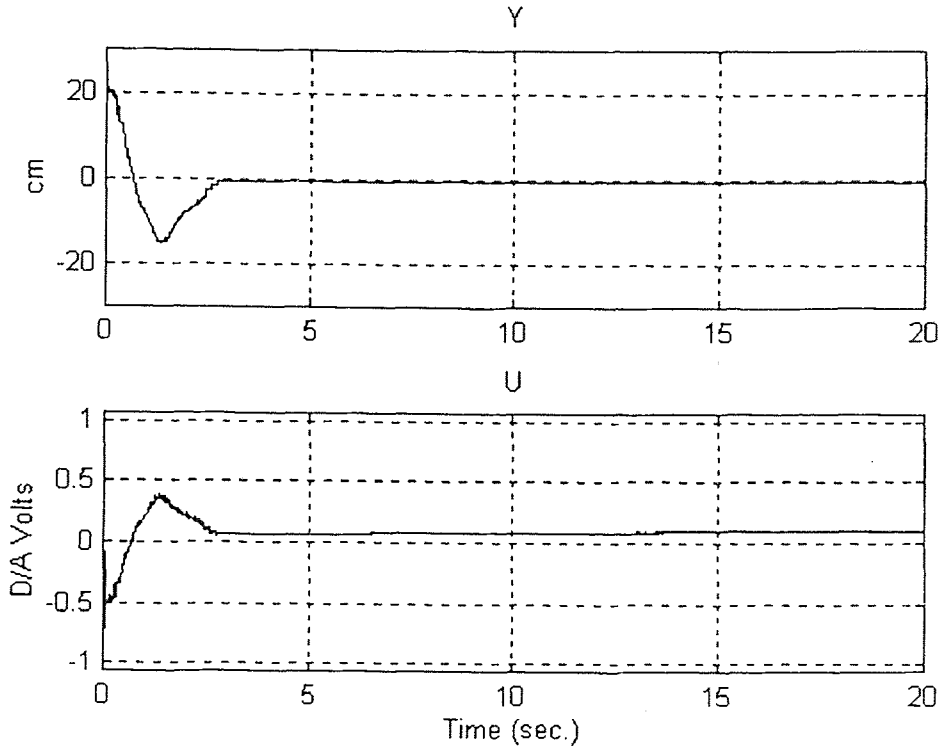


Figure 5.27: Typical Y and U time plots for $K_I = 0.02$ variable gain.

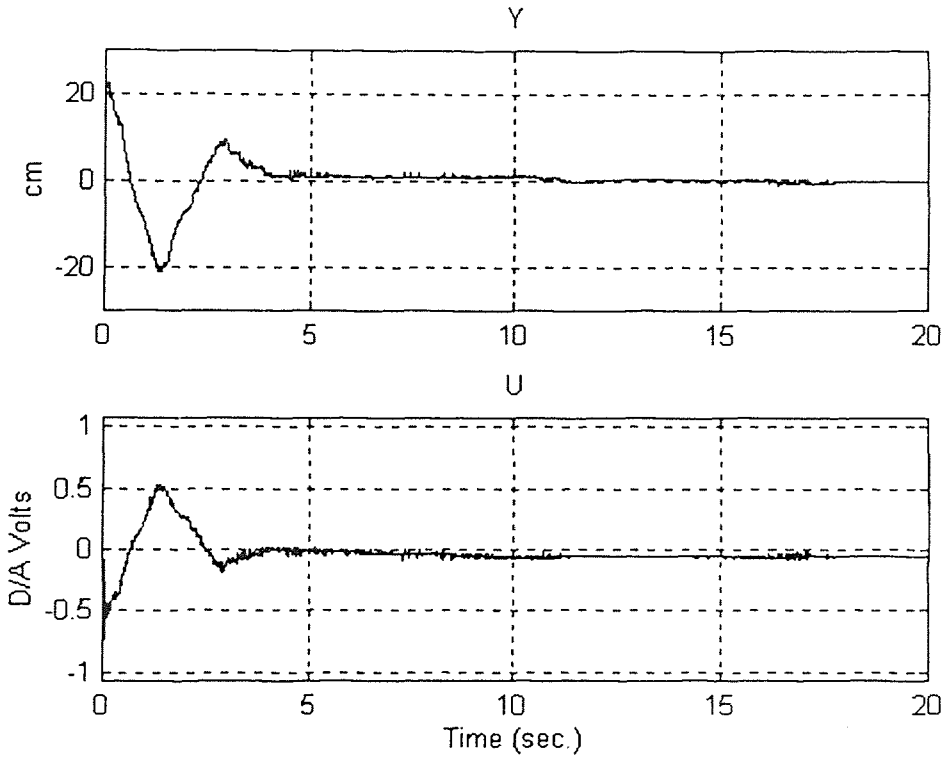


Figure 5.28: Typical Y and U time plots for $K_I = 0.04$ variable gain.

From the summary of test results in Table 5.11, $K_I = 0.02$ produces the best results when measured with performance indices $\Sigma E(i)^2$ and settling time but $K_I = 0.04$ produces the best results when measured by the final position error. Therefore, it is likely that the optimum K_I, when K_P and K_D are set to 10 and the initial position is +20 cm, is approximately 0.03 for the variable gain tests.

Table 5.12 Summary of results: Initial position = -20 cm, Variable K_P K_D K_I control -- $K_P = 10$, $K_D = 10$.

K_I	Avg $\Sigma E(i)^2$	Std $\Sigma E(i)^2$	Avg Final Error (cm)	Std Final Error (cm)	Avg Settling Time (sec.)	Std Settling Time (sec.)
0.005	22.2	3.3	0.74	0.32	2.3	0.4
0.01	24.0	2.0	1.08	0.20	5.3	4.9
0.02	24.6	1.3	0.48	0.26	15.8	2.5
0.04	25.8	1.9	-0.28	0.83	11.7	1.5

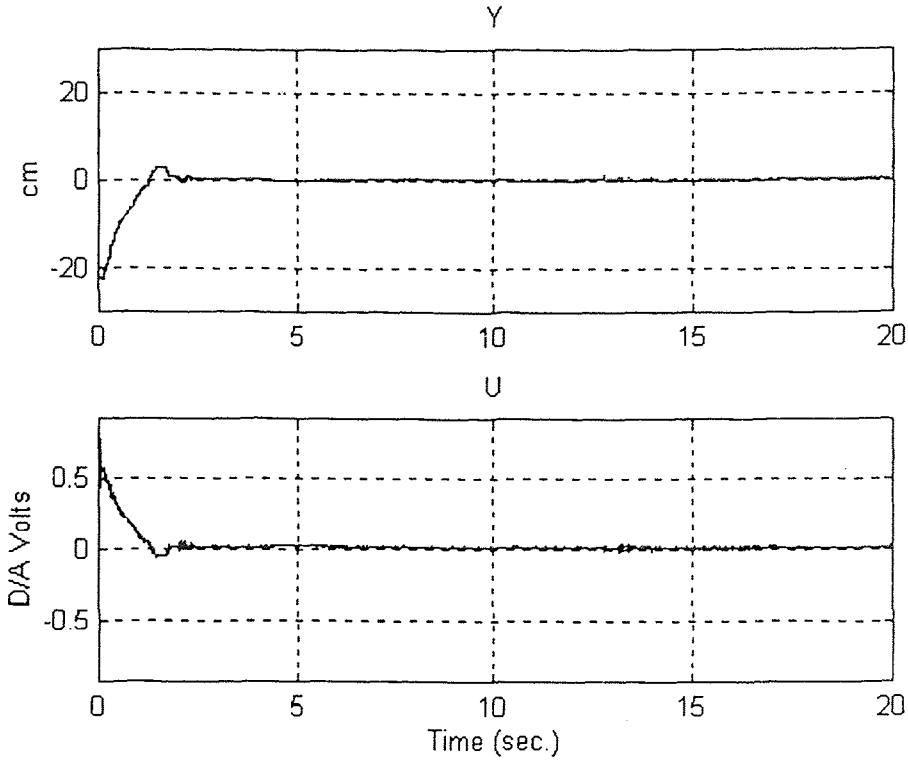


Figure 5.29: Typical Y and U time plots for $K_I = 0.005$ variable gain.

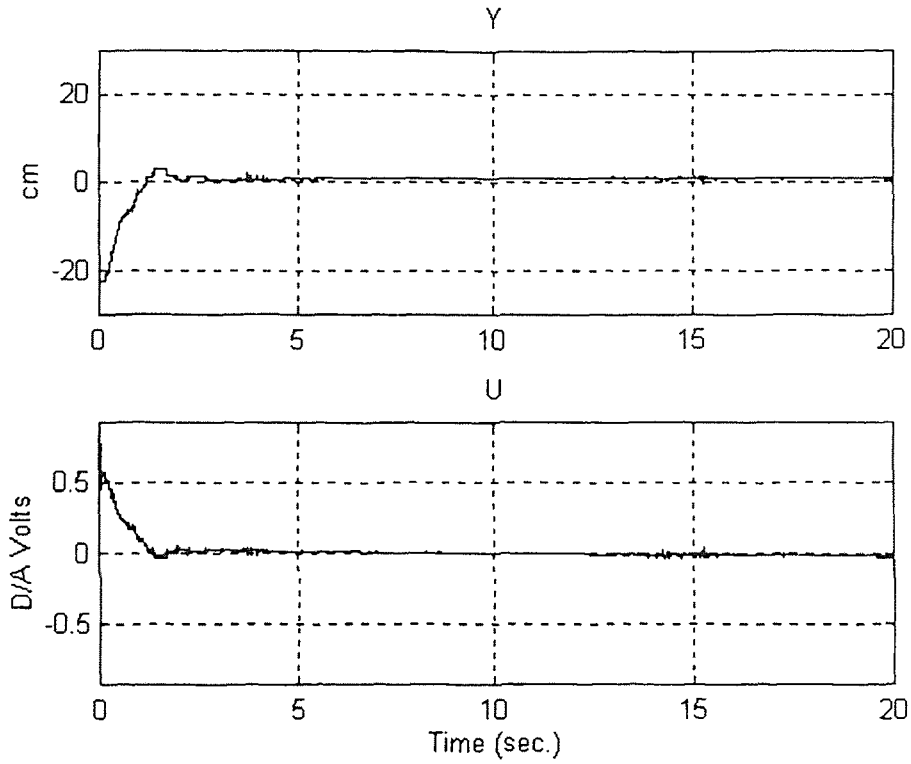


Figure 5.30: Typical Y and U time plots for $K_I = 0.01$ variable gain.

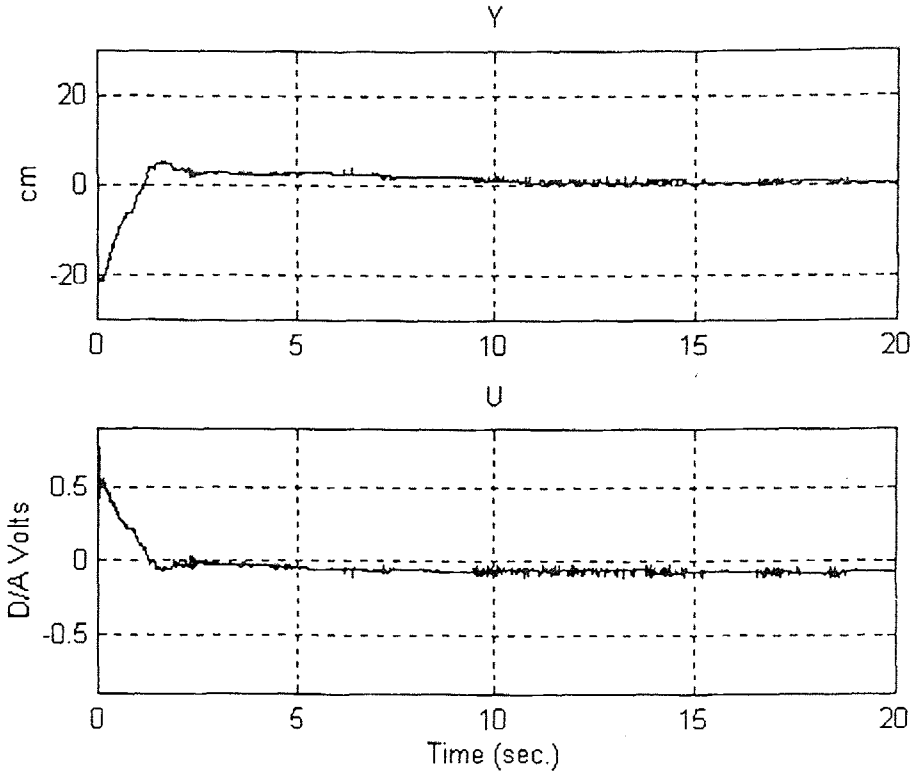


Figure 5.31: Typical Y and U time plots for $K_I = 0.02$ variable gain.

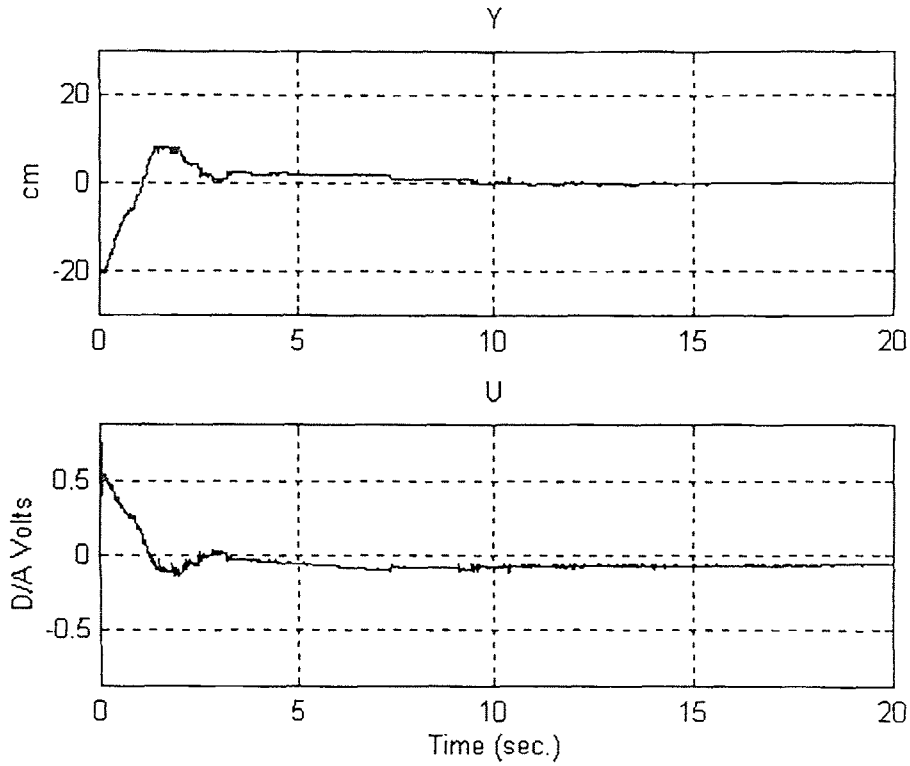


Figure 5.32: Typical Y and U time plots for $K_I = 0.04$ variable gain.

From a comparison of the data obtained during the constant and variable gain tests for the initial position of -20 cm (Tables 5.9 and 5.12), it is determined that the optimum K_I is the same for both conditions: If the test results when $K_I = 0.005$ are disregarded, $K_I = 0.01$ will produce the best results when measured with the performance indices $\Sigma E(i)^2$ and settling time but $K_I = 0.04$ produces the best results when measured by the final position error.

From a comparison of the summary of the test results between the constant and variable gain control laws, Tables 5.8 and 5.11, when the initial position is +20 cm there is no improvement in the performance indices. However, when the initial position is -20 cm, the results of which are summarized in Tables 5.9 and 5.12, there is an approximately 10% improvement in the performance indices. Also, the initial overshoot of the position is decreased by approximately half when the variable gain is used.

CHAPTER 6

CONCLUSIONS

In this thesis work, an optical position sensor has been designed and implemented

6.1 Sensor Module Characterization

The characteristics of the sensor module were analyzed using various standard test procedures such as drift stability, short-range repeatability, and long-range repeatability. A summary of these test results are tabulated in Table 6.1 below. Based on these results, the following conclusions are now drawn:

Table 6.1 Summary of test results (See Chapter 4).

Test	Location Repeatability (+/- mm)	Range (mm)	Resolution (mm)
Short range w/ Pstat	0.02	25.4	0.5
Long range w/ Pstat	2	600	5
Long range w/ Ppin	2	600	4

1) The system can predict the location of the mobile sensor module to an accuracy of 0.5 mm when used in the short-range test where the known position repeatability is 0.02 mm (Refer to Section 4.3 for a review of the short-range test).

2) The system can predict the location of the mobile sensor module to an accuracy of 5 mm when used in the long-range test where the known position repeatability is

approximately 2 mm and the stationary sensor module output is used as the normalization signal (Refer to Section 4.4 for a review of the long-range test when the stationary sensor module output is used as the normalization signal).

3) The system can predict the location of the mobile sensor module to an accuracy of 4 mm when used in the long-range test where the known position repeatability is approximately 2 mm and the feedback signal from the laser diode's output power is used as the normalization signal (Refer to Section 4.5 for a review of the long-range test when the feedback signal from the laser diode's output power is used as the normalization signal).

6.2 Flexible Beam Experiment

Based on the test results of the flexible beam experiment, which tested the ability of the control system (Section 2.7) to position the flexible beam to the command position when the position feedback signal is produced by the mobile sensor module attached to the end of the flexible beam, the following conclusion can be made: The system can consistently position the end of the flexible beam to within 5 mm of the command position in approximately 8 seconds when used with a properly tuned PID controller.

Also note that there is an advantage of using a microprocessor based control system over an analog-circuit based control system for the following reasons: 1) the properties of the control system can be easily changed by editing the control software rather than making changes to analog circuit components, 2) the coefficient values in the control software are stable over time where the component values of an analog circuit can

change due to environmental conditions and over time, and 3) complex logic and arithmetic operations can be easily implemented by a microprocessor.

6.3 Future Improvements

The following improvements are proposed for future work to increase the ability of the system to predict the location of the mobile sensor module:

1) By far the greatest improvement to the system would be to incorporate the Time Sampling scheme, as discussed in Section 2.5.1, which would require the use of a floating point processor. This would eliminate the need for the analog electronics required to pre-process the signal from the sensor modules and enable the DSP to sample to time between the data pulses from the TSL220 light sensor to an accuracy of 0.1uS.

2) The use of a floating point processor would also improve the realization of the digital filters, which can only be approximated with a fixed point processor.

3) Since the output frequency of the TSL220 light sensor is determined by an external capacitor for a given incident light intensity, it is critically important that the value of the capacitor remain very stable. This can either be accomplished with an active monitor and compensator for the capacitance or a forthcoming version of the TSL220 light sensor that no longer requires an external capacitor to control the output frequency.

4) All digital monitoring of the laser diode's output power feedback signal would remove the need for the analog highpass filter circuit (Figure 2.7) and improve the stability of the monitored feedback signal.

5) A variable lens setup which can create either a rapidly diverging cone of light from the laser diode would optimize the system for the short range discussed in Section 4.3 or a slowly diverging cone of light which would optimize the system for the long range tests discussed in Sections 4.4 and 4.5.

6) Further optimization of the control program for the flexible beam experiment and the incorporation of the dynamics of the flexible beam to realize a compensator in the control program would improve the system's dynamics.

7) The use of multiple sensor modules on the flexible beam to monitor and compensate for the flexure in the flexible beam would further improve the system's dynamics.

APPENDIX 1: SPECIFICATION SHEET OF THE TSL220 LIGHT TO FREQUENCY CONVERTER

TSL220
LIGHT-TO-FREQUENCY CONVERTER

D3619, AUGUST 1990–REVISED JUNE 1991

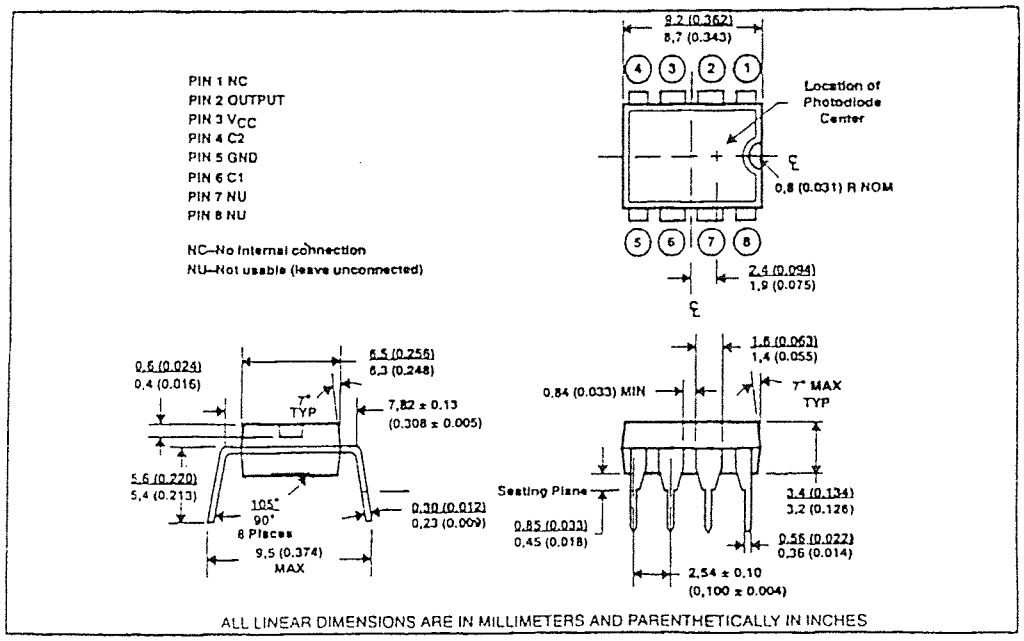
- High-Resolution Conversion of Light Intensity to Frequency
- Wide Dynamic Range . . . 118 dB
- Variable (and Single) Supply Range . . . 5 V to 10 V
- High Linearity . . . Typically Within 2% of FSR (C = 100 pF)
- High Sensitivity . . . Can Detect Change of 0.01% of FSR
- CMOS Compatible Output for Digital Processing
- Minimum External Components
- Microprocessor Compatible

description

The TSL220 consists of a large-area photodiode and a current-to-frequency converter. The output voltage is a pulse train and its frequency is directly proportional to the light intensity (irradiance) on the photodiode. The output is CMOS† compatible and its frequency may be measured using pulse counting, period timing, or integration techniques. The TSL220 is ideal for light-sensing applications requiring wide dynamic range, high sensitivity, and high noise immunity. The output frequency range is determined by an external capacitor; hence, the desired output frequency is adjustable for a given light intensity at the input. The TSL220 is characterized for operation over the temperature range of −25°C to 70°C.

mechanical data

The photodiode and current-to-frequency converter are packaged in a clear plastic 8-pin dual-in-line package. The active chip area is typically 4,13 mm² (0.0064 in²).



ALL LINEAR DIMENSIONS ARE IN MILLIMETERS AND PARENTHETICALLY IN INCHES

† Use of LSTTL logic families may require a 3300-Ω pulldown resistor on the output.

TEXAS INSTRUMENTS

PARAMETER MEASUREMENT INFORMATION

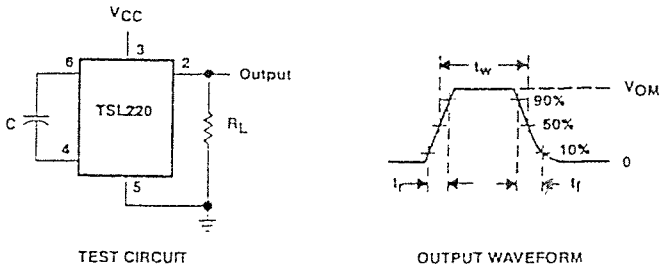


Figure 1. Switching Times

NOTE: Output waveform is monitored on an oscilloscope with the following characteristics: $R_i \geq 1\ M\Omega$, $C_i \leq 6.5$ pF.

TYPICAL CHARACTERISTICS

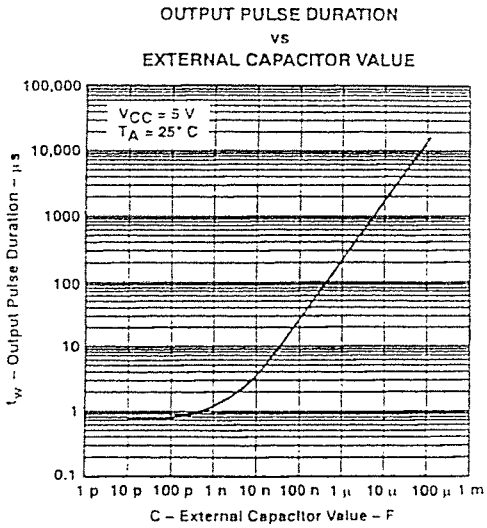


Figure 2

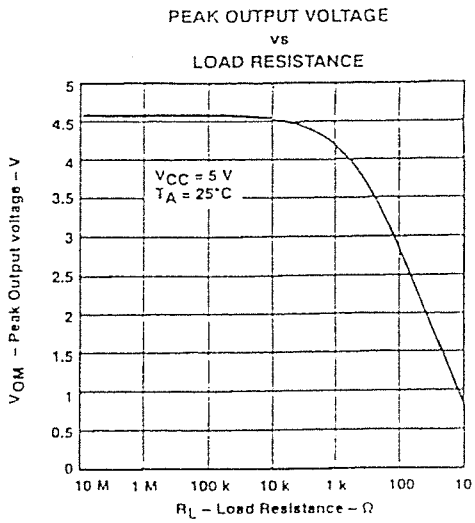
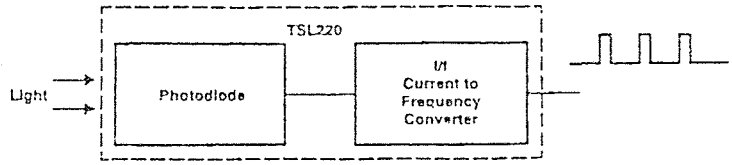


Figure 3

LIGHT-TO-FREQUENCY CONVERTER

functional block diagram



absolute maximum ratings over operating free-air temperature range (unless otherwise noted)

Supply voltage, V_{CC} (see Note 1) ..... 12 V
Operating free-air temperature, T_A ..... −25°C to 70°C
Storage temperature range ..... −25°C to 85°C
Lead temperature 1.6 mm (1/16 inch) from case for 10 seconds ..... 260°C

NOTE 1: All voltage values are with respect to GND (pin 5).

recommended operating conditions

	MIN	NOM	MAX	UNIT
Supply voltage, V_{CC}	4	5	10	V
Output frequency, f_o (C ≤ 100 pF)			750	kHz
Operating free-air temperature range, T_A	−25		70	°C

electrical characteristics at V_{CC} = 5 V, T_A = 25°C (see Figure 1)

PARAMETER		TEST CONDITIONS	MIN	TYP	MAX	UNIT
V_{OM}	Peak output voltage	R_L = 50 kΩ	3	4		V
I_{CC}	Supply current	C = 100 pF, E_e = 0		7.5	10	mA

operating characteristics at V_{CC} = 5 V, T_A = 25°C (see Figure 1)

PARAMETER		TEST CONDITIONS	MIN	TYP	MAX	UNIT
f_o	Output frequency	E_e = 125 μW/cm², λ = 880 nm, C = 100 pF	50	150	250	kHz
		E_e = 0, C = 100 pF	0	1	50	Hz
t_w	Output pulse duration	C = 470 pF		1		μs
t_r	Output pulse rise time	C = 100 pF		20		ns
t_f	Output pulse fall time	C = 100 pF		120		ns

TYPICAL CHARACTERISTICS

OUTPUT FREQUENCY
vs
IRRADIANCE

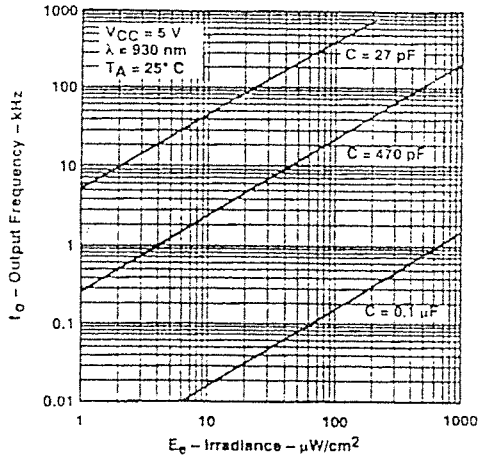


Figure 4

NORMALIZED OUTPUT FREQUENCY
vs
LOAD RESISTANCE

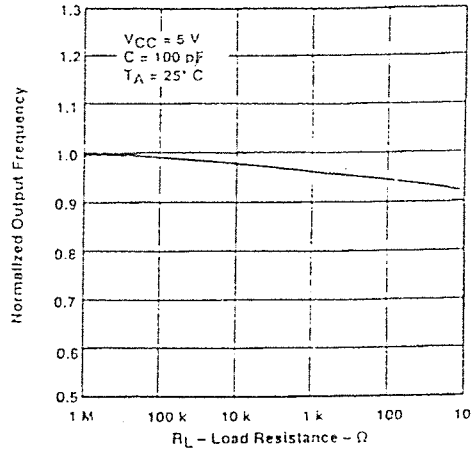


Figure 5

OUTPUT FREQUENCY
vs
EXTERNAL CAPACITOR VALUE

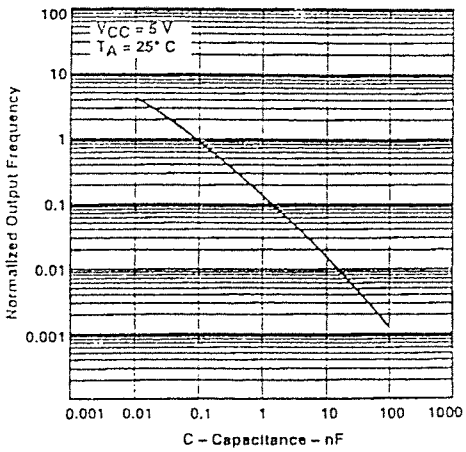


Figure 6

NORMALIZED OUTPUT FREQUENCY
vs
FREE-AIR TEMPERATURE

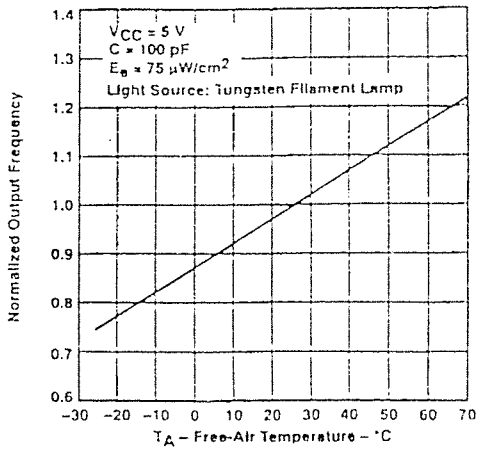


Figure 7

TYPICAL CHARACTERISTICS

SUPPLY CURRENT
vs
SUPPLY VOLTAGE

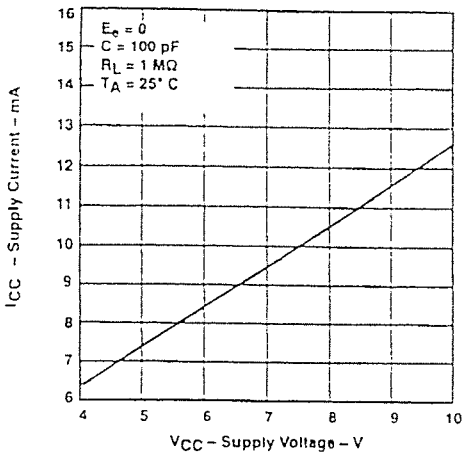


Figure 8

SUPPLY CURRENT
vs
FREE-AIR TEMPERATURE

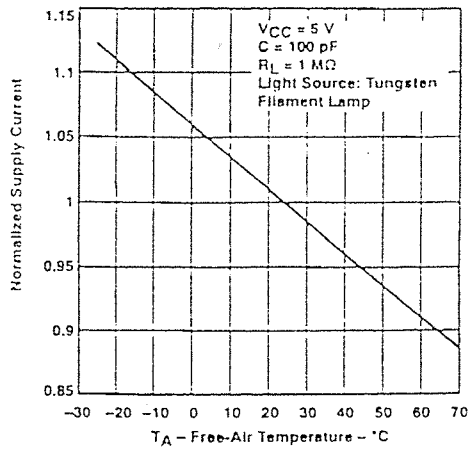


Figure 9

PHOTODIODE SPECTRAL RESPONSE

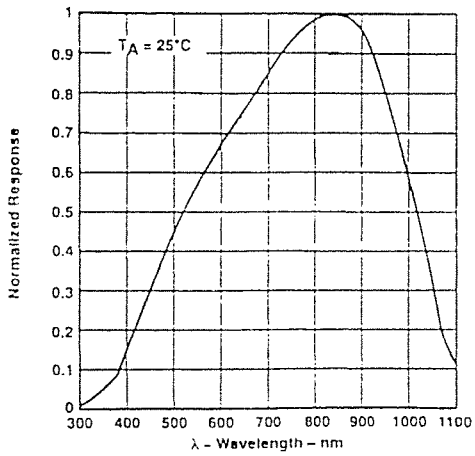


Figure 10

APPENDIX 2: SPECIFICATION SHEET OF THE PANASONIC LN9705 LASER DIODE

SPECIFICATION

LN9R05M (9mm)
LN9R05MS (5.6mm)
P459

- o Structure
- o Absolute maximum ratings
- o Electro-optical characteristics
- o Drawings
- o Safety and handling consideration

Presented
by
T.TAKIDA
Manager
Okayama Factory Discrete Device Division
Business Group-I

Date	Nov.10.'92					Signature	Page
							1 of 6

MATSUSHITA ELECTRONICS CORPORATION
BIZEN. OKAYAMA. JAPAN

SPECIFICATION
LN9R05H

(7 mm)

1. Structure

InGaAlP double hetero-structure visible laser diode.

Terminal connection is shown in Figure 1.

Outline is shown in Figure 2.

2. Absolute Maximum Ratings (T_c=25±2°C)

Parameter		Symbol	Value	Unit
Power Output		Po	5	mW
Reverse Voltage	Laser	Vr	2	V
	PIN	Vr(PIN)	30	V
Power Dissipation(PIN)		Pd(PIN)	60	mW
Operating Temperature		Top	-10 ~ 55	°C
Storage Temperature		Tstg	-40 ~ 85	°C

Date	Nov.10.'92					Signature	Page
							2 of 6

MATSUSHITA ELECTRONICS CORPORATION
BIZEN, OKAYAMA, JAPAN

SPECIFICATION

LN9R05H

3. Electro-optical Characteristics (Test Condition Ta=25±2°C)

Test Items	Symbol	Note	Conditions	Ratings			Unit
				Min.	Typ.	Max.	
Threshold current	Ith	1	CW	15	25	30	mA
Operating current	Iop		CW Po=3mW	20	35	40	mA
Operating voltage	Vop		CW Po=3mW	2.0	2.5	3.0	V
Wavelength	λp		CW Po=3mW	660	670	680	nm
Radiation Angle	θ⊥	3	CW Po=3mW	6	8	11	deg.
Radiation Angle	θ∥	3	CW Po=3mW	25	30	40	deg.
Differential Effi.	η	4	CW	0.2	0.4	0.8	W/A
Astigmatic Distance	ΔS	2	CW Po=3mW	-	-	10	μm
PIN dark current	Id		Vr(PIN)=15V	-	-	0.1	μA
PIN photo current	Ip		CW Po=3mW,Vr(PIN)=5V	0.2	0.5	1.0	mA
Emission Angle	θx		CW Po=3mW	-2	-	2	deg.
Emission Angle	θy		CW Po=3mW	-3	-	3	deg.
Emission Point	Δx	2		-100	-	100	μm
Emission Point	Δy	2		-100	-	100	μm
Emission Point	Δz	2		-100	-	100	μm

Note 1. Ith is defined at the cross point of current axis and the line connecting the points of 1mW and 3mW.

Note 2. Sampling inspection by lot.

Note 3. Angle of 50% peak intensity (FWHM).

Note 4. η is defined by the slope of the line connecting the point of 1mW and 3mW.

Date	Nov.10.'92					Signature	Page
							3 of 6

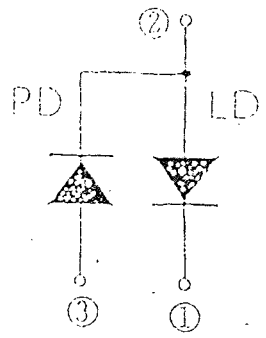
MATSUSHITA ELECTRONICS CORPORATION

BIZEN, OKAYAMA, JAPAN

SPECIFICATION

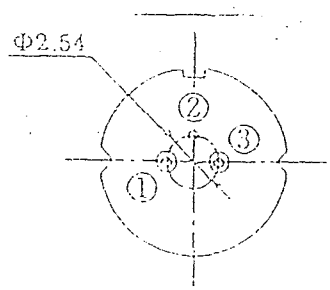
LN7105M

Figure 1 Terminal Connection



Terminal Connection

1. LD (Cathode)
2. Common (Case)
3. PD (Anode)



Date	Feb.11,'92					Signature	Page
							4 of 6

MATSUSHITA ELECTRONICS CORPORATION

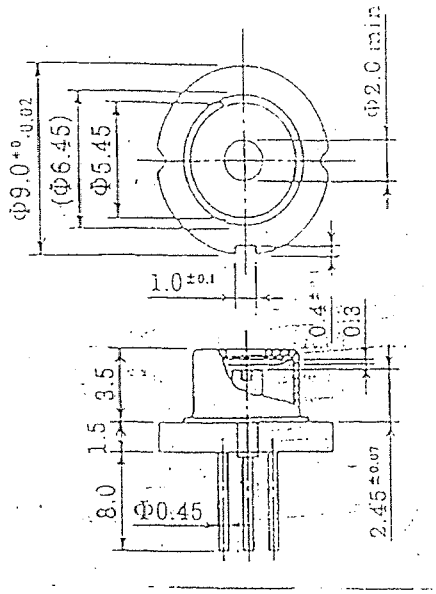
BIZEN, OKAYAMA, JAPAN

SPECIFICATION

LN9R05H

Figure 3 Outline

(9 mm)



Dimensions in mm

Date	Feb.11,'92					Signature	Page
							5 of 6

MATSUSHITA ELECTRONICS CORPORATION

BIZEN, OKAYAMA, JAPAN

APPENDIX 3: TIME SAMPLING ASSEMBLY CODE

Table A3.1: Description of memory locations in the Time Sampling code.

Memory Location	Memory Page	Description
2	0	DSP Timer
	8	Saved Timer Values

```
int0_ser

        ;interrupt 0 handeler

        ;the interrupt zero is triggered whenever the TSL220
        ; outputs a pulse [time proportional to light intensity]

        ;timer stuff
        ;save the old timer value in "timer"

        ldpk >0
        lac >2
        ldpk >8
        larp >1
        sacl *+

        ;reset the timer

        ldpk >0
        lalk >ffff
        sacl >2

        ;jump out of the interrupt routine

        eint
        ret
```

APPENDIX 4A

DESCRIPTION OF INVERSION ASSEMBLY LANGUAGE PROGRAM VARIABLES

Table A4.1 Symbol name cross reference.

Symbol in Figure 2.17	Equivalent variable in DSP program
I_E	ec_hi & ec_lo
F	fc
T_U	T
2^N	N

Table A4.2 Description of assembly program variables.

Program Variable	Description
ticker	Number of inversion iterations executed
fc	Current frequency value
fp	Previous frequency value
ec_lo	Low two bytes of the current error
ec_hi	Hi two bytes of the current error
ep_lo	Low two bytes of the previous error
ep_hi	Hi two bytes of the previous error
e_temp	Temporary error signal
N	Target bound
T	Time signal to be inverted

APPENDIX 4B

INVERSION ASSEMBLY PROGRAM

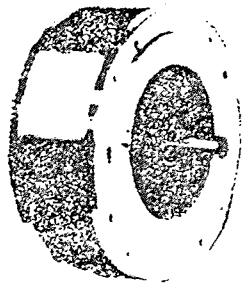
```
;i.asm

t_in        .set    1011
f_last      .set    16352
loops       .set    20
Ki          .set    6
Edivide     .set    7

            ;declare memory locations

        .bss    ticker,1
        .bss    fc,1
        .bss    fp,1
        .bss    ec_lo,1
        .bss    ec_hi,1
        .bss    ep_lo,1
        .bss    ep_hi,1
        .bss    e_temp,1
        .bss    N,1
        .bss    T,1
        .bss    x1,1
        .bss    x2,1

        ;interrupt flow table
        ;the "vectors" section is a label to link with the link.cmd
        ; file to load the interrupt flow table at address 0h

        .sect "vectors"

        b start
        b int0_ser
        b int1_ser

        ;the "text" section is a label to link with the link.cmd
        ; file to load the main program at address 50h

        .text
```

```
start:  ;start of program

        dint
        ldpk 8

        lrlk 1, 1280

        zac
        lrlk 2, 1024
        lrlk 3, 512
again:  larp 2
        sacl *+
        larp 3
        banz again

        lack loops
        sacl ticker

        ;N = 2^24

        lalk 256
        sacl N

        lalk t_in
        sacl T

        lark 7, 20

        lalk f_last
        sacl fc

loop:   save the old values

        lac fc
        sacl fp
        larp 1
        sacl *+

        lac ec_lo
        sacl ep_lo
        lac ec_hi
        sacl ep_hi

        load the previous error in the acc
```

```
        lac ep_lo
        addh ep_hi

        ;"multiply" by Ki
        ;Ki = 2^(-n)
        ;shift left by 16 - n bits
        ;Ki = 2^(-13)

        sach e_temp, Ki

        ;calculate f(i) = f(i-1) + Ki  * e(i-1)
        ;e_temp = Ki*e(i-1)

        lac e_temp
        add fp
        sacl fc

        ;load the acc with 2^N

        zac
        addh N

        ;calculate e(i) = 2^N - T*f(i)

        lt fc
        mpy T
        spac

        ;save the current error

        sacl ec_lo
        sach ec_hi

        ;check the see if error is sufficiently small to stop
        ; if error < 2^12 the loop will stop

        bgz no_neg
        neg
no_neg: sach e_temp, Edivide
        lac e_temp
        bz stop_loop

        ;check to see if max loops reached
```

```
        lac ticker
        subk 1
        sacl ticker

        larp 7
        banz loop

int0_ser:
int1_ser:

stop_loop:

        b stop_loop
        .end
```

APPENDIX 5: SPECIFICATION SHEET OF THE DC MOTOR



PMI MODEL #U12M4 DC PAN-CAKE MOTOR WITH DUAL SHAFT

Peak Torque: 1315.5 oz-in
Continuous Torque: 121.5 oz-in
Peak Torque: 84.5 amps
Continuous Current: 8.13 amps
Torque Constant: 15.63 oz-in/amp
No Load Speed: 3000 RPM
Terminal Voltage: 43.4 VDC
Inertia: 0.019 oz-in/sec/sec
Inductance: <100 mh
Dimensions: 5.5" dia. x 2.1"L., Shaft Front 0.5" dia. x 1-1/4"L., Rear 0.5" dia. x 1-1/4"L.
Weight: 8 Lbs. USED

Stock No. DM-258 . . . . $175.00

APPENDIX 6A

DESCRIPTION OF ASSEMBLY LANGUAGE PROGRAM VARIABLES

Table A6.1 Symbol name cross reference.

Symbol in Figure 1.1	Equivalent variable in DSP program
Dmove	bpf_in_nm0_0
Dstat	bpf_in_nm0_1
Dpin	bpf_in_nm0_2
Amove	a_squ_out_0
Astat	a_squ_out_1
Apin	a_squ_out_2

Table A6.2 Description of assembly program variables.

Program Variable	Description
amplitude	amplitude of the laser diode modulation signal.
bias	DC Bias of the laser diode modulation signal.
v_laser	DSP laser diode modulation signal voltage
h800	800 hex storage location
dummy	dummy variable
dummy1	dummy variable

Table A6.2
(Continued)

Program Variable	Description
flag	dummy loop decision flag
ad_in_0	A/D 0 storage variable
ad_in_1	A/D 1 storage variable
ad_in_2	A/D 2 storage variable
bpf_in_nm0_0	input(i) of the Channel 0 bandpass filter
bpf_in_nm1_0	input(i-1) of the Channel 0 bandpass filter
bpf_in_nm2_0	input(i-2) of the Channel 0 bandpass filter
bpf_out_nm0_0	output(i) of the Channel 0 bandpass filter
bpf_out_nm1_0	output(i-1) of the Channel 0 bandpass filter
bpf_out_nm2_0	output(i-2) of the Channel 0 bandpass filter
bpf_in_nm0_1	input(i) of the Channel 1 bandpass filter
bpf_in_nm1_1	input(i-1) of the Channel 1 bandpass filter
bpf_in_nm2_1	input(i-2) of the Channel 1 bandpass filter
bpf_out_nm0_1	output(i) of the Channel 1 bandpass filter
bpf_out_nm1_1	output(i-1) of the Channel 1 bandpass filter
bpf_out_nm2_1	output(i-2) of the Channel 1 bandpass filter
bpf_in_nm0_2	input(i) of the Channel 2 bandpass filter
bpf_in_nm1_2	input(i-1) of the Channel 2 bandpass filter
bpf_in_nm2_2	input(i-2) of the Channel 2 bandpass filter

Table A6.2
(Continued)

rogram Variable	Description
bpf_out_nm0_2	output(i) of the Channel 2 bandpass filter
bpf_out_nm1_2	output(i-1) of the Channel 2 bandpass filter
bpf_out_nm2_2	output(i-2) of the Channel 2 bandpass filter
bpf_coeff_in_nm0	input coefficient(i) of the bandpass filters
bpf_coeff_in_nm1	input coefficient(i-1) of the bandpass filters
bpf_coeff_in_nm2	input coefficient(i-2) of the bandpass filters
bpf_coeff_out_nm0	output coefficient(i) of the bandpass filters
bpf_coeff_out_nm1	output coefficient(i-1) of the bandpass filters
bpf_coeff_out_nm2	output coefficient(i-2) of the bandpass filters
s_out_0	output of the Channel 0 sin multiplier
c_out_0	output of the Channel 0 cos multiplier
s_out_1	output of the Channel 1 sin multiplier
c_out_1	output of the Channel 1 cos multiplier
s_out_2	output of the Channel 2 sin multiplier
c_out_2	output of the Channel 2 cos multiplier
lpfcos_in_nm0_0	input(i) of the Channel 0 cos lowpass filter
lpfcos_in_nm1_0	input(i-1) of the Channel 0 cos lowpass filter
lpfcos_out_nm0_0	output(i) of the Channel 0 cos lowpass filter
lpfcos_out_nm1_0	output(i-1) of the Channel 0 cos lowpass filter

Table A6.2
(Continued)

Program Variable	Description
lpfsin_in_nm0_0	input(i) of the Channel 0 sin lowpass filter
lpfsin_in_nm1_0	input(i-1) of the Channel 0 sin lowpass filter
lpfsin_out_nm0_0	output(i) of the Channel 0 sin lowpass filter
lpfsin_out_nm1_0	output(i-1) of the Channel 0 sin lowpass filter
lpfcos_in_nm0_1	input(i) of the Channel 1 cos lowpass filter
lpfcos_in_nm1_1	input(i-1) of the Channel 1 cos lowpass filter
lpfcos_out_nm0_1	output(i) of the Channel 1 cos lowpass filter
lpfcos_out_nm1_1	output(i-1) of the Channel 1 cos lowpass filter
lpfsin_in_nm0_1	input(i) of the Channel 1 sin lowpass filter
lpfsin_in_nm1_1	input(i-1) of the Channel 1 sin lowpass filter
lpfsin_out_nm0_1	output(i) of the Channel 1 sin lowpass filter
lpfsin_out_nm1_1	output(i-1) of the Channel 1 sin lowpass filter
lpfcos_in_nm0_2	input(i) of the Channel 2 cos lowpass filter
lpfcos_in_nm1_2	input(i-1) of the Channel 2 cos lowpass filter
lpfcos_out_nm0_2	output(i) of the Channel 2 cos lowpass filter
lpfcos_out_nm1_2	output(i-1) of the Channel 2 cos lowpass filter
lpfsin_in_nm0_2	input(i) of the Channel 2 sin lowpass filter
lpfsin_in_nm1_2	input(i-1) of the Channel 2 sin lowpass filter
lpfsin_out_nm0_2	output(i) of the Channel 2 sin lowpass filter

Table A6.2
(Continued)

Program Variable	Description
lpfsin_out_nm1_2	output(i-1) of the Channel 2 sin lowpass filter
lpf_coeff_in_nm0	input coefficient(i) of the lowpass filters
lpf_coeff_in_nm1	input coefficient(i-1) of the lowpass filters
lpf_coeff_out_nm0	output coefficient(i) of the lowpass filters
lpf_coeff_out_nm1	output coefficient(i-1) of the lowpass filters
ellcos_in_nm0_0	input(i) of the Channel 0 cos bandpass filter
ellcos_in_nm1_0	input(i-1) of the Channel 0 cos bandpass filter
ellcos_in_nm2_0	input(i-2) of the Channel 0 cos bandpass filter
ellcos_out_nm0_0	output(i) of the Channel 0 cos bandpass filter
ellcos_out_nm1_0	output(i-1) of the Channel 0 cos bandpass filter
ellcos_out_nm2_0	output(i-2) of the Channel 0 cos bandpass filter
ellcos_in_nm0_1	input(i) of the Channel 1 cos bandpass filter
ellcos_in_nm1_1	input(i-1) of the Channel 1 cos bandpass filter
ellcos_in_nm2_1	input(i-2) of the Channel 1 cos bandpass filter
ellcos_out_nm0_1	output(i) of the Channel 1 cos bandpass filter
ellcos_out_nm1_1	output(i-1) of the Channel 1 cos bandpass filter
ellcos_out_nm2_1	output(i-2) of the Channel 1 cos bandpass filter
ellcos_in_nm0_2	input(i) of the Channel 2 cos bandpass filter
ellcos_in_nm1_2	input(i-1) of the Channel 2 cos bandpass filter

Table A6.2
(Continued)

Program Variable	Description
ellcos_in_nm2_2	input(i-2) of the Channel 2 cos bandpass filter
ellcos_out_nm0_2	output(i) of the Channel 2 cos bandpass filter
ellcos_out_nm1_2	output(i-1) of the Channel 2 cos bandpass filter
ellcos_out_nm2_2	output(i-2) of the Channel 2 cos bandpass filter
ellsin_in_nm0_0	input(i) of the Channel 0 sin bandpass filter
ellsin_in_nm1_0	input(i-1) of the Channel 0 sin bandpass filter
ellsin_in_nm2_0	input(i-2) of the Channel 0 sin bandpass filter
ellsin_out_nm0_0	output(i) of the Channel 0 sin bandpass filter
ellsin_out_nm1_0	output(i-1) of the Channel 0 sin bandpass filter
ellsin_out_nm2_0	output(i-2) of the Channel 0 sin bandpass filter
ellsin_in_nm0_1	input(i) of the Channel 1 sin bandpass filter
ellsin_in_nm1_1	input(i-1) of the Channel 1 sin bandpass filter
ellsin_in_nm2_1	input(i-2) of the Channel 1 sin bandpass filter
ellsin_out_nm0_1	output(i) of the Channel 1 sin bandpass filter
ellsin_out_nm1_1	output(i-1) of the Channel 1 sin bandpass filter
ellsin_out_nm2_1	output(i-2) of the Channel 1 sin bandpass filter
ellsin_in_nm0_2	input(i) of the Channel 2 sin bandpass filter
ellsin_in_nm1_2	input(i-1) of the Channel 2 sin bandpass filter
ellsin_in_nm2_2	input(i-2) of the Channel 2 sin bandpass filter

Table A6.2
(Continued)

Program Variable	Description
ellsin_out_nm0_2	output(i) of the Channel 2 sin bandpass filter
ellsin_out_nm1_2	output(i-1) of the Channel 2 sin bandpass filter
ellsin_out_nm2_2	output(i-2) of the Channel 2 sin bandpass filter
ell_coeff_in_nm0	input coefficient(i) of the elliptic filters
ell_coeff_in_nm1	input coefficient(i-1) of the elliptic filters
ell_coeff_in_nm2	input coefficient(i-2) of the elliptic filters
ell_coeff_out_nm0	output coefficient(i) of the elliptic filters
ell_coeff_out_nm1	output coefficient(i-1) of the elliptic filters
ell_coeff_out_nm2	output coefficient(i-2) of the elliptic filters
a_squ_hi_0	hi 16 bits of the output of the Channel 0 squarer
a_squ_lo_0	lo 16 bits of the output of the Channel 0 squarer
a_squ_out_0	output of the Channel 0 squarer ported to D/A
a_squ_hi_1	hi 16 bits of the output of the Channel 1 squarer
a_squ_lo_1	lo 16 bits of the output of the Channel 1 squarer
a_squ_out_1	output of the Channel 1 squarer ported to D/A
a_squ_hi_2	hi 16 bits of the output of the Channel 2 squarer
a_squ_lo_2	lo 16 bits of the output of the Channel 2 squarer
a_squ_out_2	output of the Channel 2 squarer ported to D/A
aux_laser_pointer	temp aux reg storage for the laser diode voltage pointer

Table A6.2
(Continued)

Program Variable	Description
aux_laser_reset	temp aux reg storage for the laser diode reset pointer
aux_cos_pointer	temp aux reg storage for the cos pointer
aux_cos_reset	temp aux reg storage for the reset cos pointer
aux_sin_pointer	temp aux reg storage for the sin pointer
aux_sin_reset	temp aux reg storage for the reset sin pointer
out_lo_old_0	hi 16 bits of the output average of the Channel 0 squarer
out_hi_old_0	lo 16 bits of the output average of the Channel 0 squarer
out_avg_0	output avg of the Channel 0 squarer ported to host PC
avged_out_0	output avg of the Channel 0 squarer ported to host PC
out_lo_old_1	hi 16 bits of the output average of the Channel 1 squarer
out_hi_old_1	lo 16 bits of the output average of the Channel 1 squarer
out_avg_1	output avg of the Channel 1 squarer ported to host PC
avged_out_	output avg of the Channel 1 squarer ported to host PC
out_lo_old_2	hi 16 bits of the output average of the Channel 2 squarer
out_hi_old_2	lo 16 bits of the output average of the Channel 2 squarer
out_avg_2	output avg of the Channel 2 squarer ported to host PC
avged_out_2	output avg of the Channel 2 squarer ported to host PC
avg_out_lo_old_0	previous lo 16 bits of the output avg of Channel 0
avg_out_hi_old_0	previous hi 16 bits of the output avg of Channel 0

Table A6.2
(Continued)

Program Variable	Description
avged_out_avg_0	output avg of the Channel 0 squarer ported to host PC
avged_avged_out_0	output avg of the Channel 0 squarer ported to host PC
avg_out_lo_old_1	previous lo 16 bits of the output avg of Channel 1
avg_out_hi_old_1	previous hi 16 bits of the output avg of Channel 1
avged_out_avg_1	output avg of the Channel 1 squarer ported to host PC
avged_avged_out_1	output avg of the Channel 1 squarer ported to host PC
avg_out_lo_old_2	previous lo 16 bits of the output avg of Channel 2
avg_out_hi_old_2	previous hi 16 bits of the output avg of Channel 2
avged_out_avg_2	output avg of the Channel 2 squarer ported to host PC
avged_avged_out_2	output avg of the Channel 2 squarer ported to host PC
command	flexible beam command signal from C-program

APPENDIX 6B

DSP ASSEMBLY LANGUAGE PROGRAM

```
;assembly program
;assembly.asm

;Set up labels

f_ext_timer          .set   -2571        ;40k Hz

;Laser Voltage *** DO NOT CHANGE THESE VALUES!!!!!! ***
amp                  .set   80
bias_                .set   3125
number_average       .set   99           ;number must be one less than desired !!!!

;Sin wave logistics variables
sin_start            .set   900h
reset_ticker         .set   19
display_tick_start   .set   117

;Gains & Divisors
hpfgain              .set   1            ;gain = 2^x
hpfdiv               .set   2            ;4 = divide by 4096 for unscaling
shift                .set   4            ;4 = divide by 4096 for unscaling
lpfgain              .set   0            ;gain = 2^X
lpfdiv               .set   2            ;4 = divide by 4096 for unscaling
ellgain              .set   0            ;gain = 2^X
elldiv               .set   2            ;4 = divide by 4096 for unscaling
outdiv               .set   5

;Interrupt Enable Variable
int_on               .set   2            ;enable interrupt(s)

;A/D & D/A Mux Controls
out_0                .set   0            ;latch value for D/A chan zero
out_1                .set   8            ;latch wavue for D/A chan one
in_0                 .set   0
in_1                 .set   1
in_2                 .set   2
```

```
;Port map
ext_timer       .set    0       ;timer = port 0
latch           .set    1       ;latch = port 1
input           .set    2       ;input a/d = port 2
timer_loc       .set    2       ;timer location
output          .set    3       ;output d/a = port 3
timer_per       .set    3       ;timer starting value
int_mask        .set    4       ;interrupt mask = port 4

;Define Variables and memory locations
;Page 4

;Laser variables and stuff
amplitude       .set    0
bias            .set    1
v_laser         .set    2
h800            .set    3
dummy           .set    4
dummy1          .set    5
flag            .set    6

;A/D Input variables
ad_in_0         .set    7
ad_in_1         .set    8
ad_in_2         .set    9

;HPF variables
bpf_in_nm0_0    .set    10
bpf_in_nm1_0    .set    11
bpf_in_nm2_0    .set    12
bpf_out_nm0_0   .set    13
bpf_out_nm1_0   .set    14
bpf_out_nm2_0   .set    15

bpf_in_nm0_1    .set    16
bpf_in_nm1_1    .set    17
bpf_in_nm2_1    .set    18
bpf_out_nm0_1   .set    19
bpf_out_nm1_1   .set    20
bpf_out_nm2_1   .set    21

bpf_in_nm0_2    .set    22
bpf_in_nm1_2    .set    23
bpf_in_nm2_2    .set    24
bpf_out_nm0_2   .set    25
```

```
bpf_out_nm1_2        .set   26
bpf_out_nm2_2        .set   27

bpf_coeff_in_nm0     .set   28
bpf_coeff_in_nm1     .set   29
bpf_coeff_in_nm2     .set   30
bpf_coeff_out_nm0    .set   31
bpf_coeff_out_nm1    .set   32
bpf_coeff_out_nm2    .set   33

;Sin & Cos Multiplication
s_out_0              .set   34
c_out_0              .set   35
s_out_1              .set   36
c_out_1              .set   37
s_out_2              .set   38
c_out_2              .set   39

;LPF variables
lpfcos_in_nm0_0      .set   40
lpfcos_in_nm1_0      .set   41
lpfcos_out_nm0_0     .set   42
lpfcos_out_nm1_0     .set   43
lpfsin_in_nm0_0      .set   44
lpfsin_in_nm1_0      .set   45
lpfsin_out_nm0_0     .set   46
lpfsin_out_nm1_0     .set   47

lpfcos_in_nm0_1      .set   48
lpfcos_in_nm1_1      .set   49
lpfcos_out_nm0_1     .set   50
lpfcos_out_nm1_1     .set   51
lpfsin_in_nm0_1      .set   52
lpfsin_in_nm1_1      .set   53
lpfsin_out_nm0_1     .set   54
lpfsin_out_nm1_1     .set   55

lpfcos_in_nm0_2      .set   56
lpfcos_in_nm1_2      .set   57
lpfcos_out_nm0_2     .set   58
lpfcos_out_nm1_2     .set   59
lpfsin_in_nm0_2      .set   60
lpfsin_in_nm1_2      .set   61
lpfsin_out_nm0_2     .set   62
lpfsin_out_nm1_2     .set   63
```

```
lpf_coeff_in_nm0      .set    64
lpf_coeff_in_nm1      .set    65
lpf_coeff_out_nm0     .set    66
lpf_coeff_out_nm1     .set    67

;ELL variables
ellcos_in_nm0_0       .set    68
ellcos_in_nm1_0       .set    69
ellcos_in_nm2_0       .set    70
ellcos_out_nm0_0      .set    71
ellcos_out_nm1_0      .set    72
ellcos_out_nm2_0      .set    73

ellcos_in_nm0_1       .set    74
ellcos_in_nm1_1       .set    75
ellcos_in_nm2_1       .set    76
ellcos_out_nm0_1      .set    77
ellcos_out_nm1_1      .set    78
ellcos_out_nm2_1      .set    79

ellcos_in_nm0_2       .set    80
ellcos_in_nm1_2       .set    81
ellcos_in_nm2_2       .set    82
ellcos_out_nm0_2      .set    83
ellcos_out_nm1_2      .set    84
ellcos_out_nm2_2      .set    85

ellsin_in_nm0_0       .set    86
ellsin_in_nm1_0       .set    87
ellsin_in_nm2_0       .set    88
ellsin_out_nm0_0      .set    89
ellsin_out_nm1_0      .set    90
ellsin_out_nm2_0      .set    91

ellsin_in_nm0_1       .set    92
ellsin_in_nm1_1       .set    93
ellsin_in_nm2_1       .set    94
ellsin_out_nm0_1      .set    95
ellsin_out_nm1_1      .set    96
ellsin_out_nm2_1      .set    97

ellsin_in_nm0_2       .set    98
ellsin_in_nm1_2       .set    99
ellsin_in_nm2_2       .set    100
```

```
ellsin_out_nm0_2      .set    101
ellsin_out_nm1_2      .set    102
ellsin_out_nm2_2      .set    103

ell_coeff_in_nm0      .set    104
ell_coeff_in_nm1      .set    105
ell_coeff_in_nm2      .set    106
ell_coeff_out_nm0     .set    107
ell_coeff_out_nm1     .set    108
ell_coeff_out_nm2     .set    109

;Square & Summer variables
a_squ_hi_0            .set    110
a_squ_lo_0            .set    111
a_squ_out_0           .set    112

a_squ_hi_1            .set    113
a_squ_lo_1            .set    114
a_squ_out_1           .set    115

a_squ_hi_2            .set    117
a_squ_lo_2            .set    118
a_squ_out_2           .set    119

;Auvilary Register [Pointers & Resets] Load & Save Locations
aux_laser_pointer     .set    120
aux_laser_reset       .set    121
aux_cos_pointer       .set    122
aux_cos_reset         .set    123
aux_sin_pointer       .set    124
aux_sin_reset         .set    125

; Page 5

;Average variables
out_lo_old_0          .set    0
out_hi_old_0          .set    1
out_avg_0             .set    2
avged_out_0           .set    3

out_lo_old_1          .set    4
out_hi_old_1          .set    5
out_avg_1             .set    6
avged_out_            .set    7
```

```
out_lo_old_2        .set    8
out_hi_old_2        .set    9
out_avg_2           .set    10
avged_out_2         .set    11

avg_out_lo_old_0    .set    12
avg_out_hi_old_0    .set    13
avged_out_avg_0     .set    14
avged_avged_out_0   .set    15

avg_out_lo_old_1    .set    16
avg_out_hi_old_1    .set    17
avged_out_avg_1     .set    18
avged_avged_out_1   .set    19

avg_out_lo_old_2    .set    20
avg_out_hi_old_2    .set    21
avged_out_avg_2     .set    22
avged_avged_out_2   .set    23

command             .set    24

        ;interrupt flow table
        ;the "vectors" section is a label to link with the link.cmd
        ; file to load the interrupt flow table at address 0h

        .sect "vectors"

        b start                         ;external reset interrupt
        b int0_ser                      ;external interrupt zero
        b int1_ser                      ;external interrupt one

        ;the "text" section is a label to link with the link.cmd
        ; file to load the main program at address 50h

        .text

start:  ;start of program

        ;sets the interrupt mask - turn on INT1

        dint
        ldpk 0
        lack int_on
        sacl int_mask
```

```
;set up external timer for adc operation
;this interrupt sets the "resampling" rate

ldpk 4
lalk f_ext_timer
sacl dummy
out dummy, ext_timer

;set up latch to choose A/D = 0  D/A = 0

lalk
sacl dummy
out dummy, latch

;clear the memory locations

call clear

;call the set up sin & cos look-up routine

call setup_sin

;set up sin & cos auxilary registers and tickers

call sin_logistics

;load the HPF coefficients

call hpf_coeff

;load the LPF coefficients

call lpf_coeff

;load the ELL coefficients

call ell_coeff

;load laser power values

call laser_setup
;load 800 hex for D/A outputs

lalk 2048
```

```
        sacl h800

        ;setup "flag" to choose which channel to run

        ;If Acc == 0 --- will execute channel zero
        ;If Acc == 1 --- will execute channel one
        ;If Acc == 2 --- will execute channel two
        ;If Acc == 3 --- will execute channel three

        lark 0, 1000h
        lark 7, number_average

        zac
        sacl flag

        larp 1
        eint

main_loop:  ;if flag == 0 than go into demod loop
        ;if flag == 1 than don't go into demod loop

        idle

        b main_loop


;*******************
;*******************
;*******************

int0_ser:

        eint
        ret

;*******************
;*******************
;*******************

; The demodulation loops and the laser modulation calculations are
;   all performed in interrupt number one server.

int1_ser:

        ;Load auxilary pointer to 1 for pointers and tickers
```

```
        larp 1

        ;Decide which channel to run

        ldpk 4
        lac flag
        bz channel_0
        subk 1
        bz channel_1
        lac flag
        subk 2
        bz channel_2
        b channel_3

;******************

; Program Channel 0

channel_0:

        ;set up latch to choose A/D = 1  D/A = 0

        lalk 10
        sacl dummy
        out dummy, latch

         Read in data from A/D

        in ad_in_0, input
        lac ad_in_0
        sub h800
        sacl ad_in_0

;******************

        ;Band Pass Filter

bpf_0:  ;load the "input"
        ; add gain to the input

        lac ad_in_0, hpfgain
        sacl bpf_in_nm0_0

        zac
```

```
        ;execute the difference equation

        lt bpf_out_nm2_0
        mpy bpf_coeff_out_nm2

        ltd bpf_out_nm1_0
        mpy bpf_coeff_out_nm1

        apac

        lt bpf_in_nm2_0
        mpy bpf_coeff_in_nm2

        ltd bpf_in_nm1_0
        mpy bpf_coeff_in_nm1

        ltd bpf_in_nm0_0
        mpy bpf_coeff_in_nm0
        apac

        bgz no_add_hpf_0
        addk 4095
no_add_hpf_0: sach bpf_out_nm0_0, hpfdiv

        dmov bpf_out_nm0_0

;*******************

mult_cos_0:

        ;figure out cos

        lar 1, aux_cos_pointer
        lt bpf_out_nm0_0
        mpy *
        pac
        sach c_out_0, shift

;*******************

mult_sin_0:

        ;figure out sin
        ;the sub_in_val is from the cos signal
```

```
        lar 1, aux_sin_pointer
        lt bpf_out_nm0_0
        mpy *
        pac
        sach s_out_0, shift

;*******************

lpf_cos_0:

        ;load the "input"
        ; add gain to the

        lac c_out_0, lpfgain
        sacl lpfcos_in_nm0_0

        zac

        ;execute the difference equation

        lt lpfcos_out_nm1_0
        mpy lpf_coeff_out_nm1

        apac

        lt lpfcos_in_nm1_0
        mpy lpf_coeff_in_nm1

        ltd lpfcos_in_nm0_0
        mpy lpf_coeff_in_nm0
        apac

        bgz noadd_lpfcos_0
        addk 4095
noadd_lpfcos_0: sach lpfcos_out_nm0_0, lpfdiv

        dmov lpfcos_out_nm0_0

;*******************

lpf_sin_0:

        ;load the "input"
        ; add gain to the
```

```
        lac s_out_0, lpfgain
        sacl lpfsin_in_nm0_0

        zac

        ;execute the difference equation

        lt lpfsin_out_nm1_0
        mpy lpf_coeff_out_nm1

        apac

        lt lpfsin_in_nm1_0
        mpy lpf_coeff_in_nm1

        ltd lpfsin_in_nm0_0
        mpy lpf_coeff_in_nm0
        apac

        bgz noadd_lpfsin_0
        addk 4095
noadd_lpfsin_0: sach lpfsin_out_nm0_0, lpfdiv

        dmov lpfsin_out_nm0_0

;*******************

ell_cos_0:  ;(125e0 + 4071e1 + 4045f1) / 4096

        ;load the "input"
        ; add gain to the

        lac lpfcos_out_nm0_0, ellgain
        sacl ellcos_in_nm0_0

        zac

        ;execute the difference equation

        lt ellcos_out_nm2_0
        mpy ell_coeff_out_nm2

        ltd ellcos_out_nm1_0
        mpy ell_coeff_out_nm1
```

```
        apac

        lt ellcos_in_nm2_0
        mpy ell_coeff_in_nm2

        ltd ellcos_in_nm1_0
        mpy ell_coeff_in_nm1

        ltd ellcos_in_nm0_0
        mpy ell_coeff_in_nm0
        apac

        bgz noadd_ellcos_0
        addk 4095
noadd_ellcos_0: sach ellcos_out_nm0_0, elldiv

        dmov ellcos_out_nm0_0

;*******************

ell_sin_0:  ;(125g0 + 4071g1 + 4045h1) / 4096

        ;load the "input"
        ; add gain to the

        lac lpfsin_out_nm0_0, ellgain
        sacl ellsin_in_nm0_0

        zac

        ;execute the difference equation

        lt ellsin_out_nm2_0
        mpy ell_coeff_out_nm2

        ltd ellsin_out_nm1_0
        mpy ell_coeff_out_nm1

        apac

        lt ellsin_in_nm2_0
        mpy ell_coeff_in_nm2

        ltd ellsin_in_nm1_0
```

```
        mpy ell_coeff_in_nm1

        ltd ellsin_in_nm0_0
        mpy ell_coeff_in_nm0
        apac

        bgz noadd_ellsin_0
        addk 4095
noadd_ellsin_0: sach ellsin_out_nm0_0, elldiv

        dmov ellsin_out_nm0_0

;*******************

summer_0:

        ;NOTE -- when outputting the answer to the DAC the value
        ; must be less that 2 ^ 12 because of the limit of the DAC
        ;HOWEVER -- the internal answer can remain at 2 ^ 16 resolution

        sqra ellcos_out_nm0_0
        zac
        sqra ellsin_out_nm0_0
        apac

        sach a_squ_hi_0
        sacl a_squ_lo_0

        sach a_squ_out_0, outdiv

        lac a_squ_out_0

;       out a_squ_out_0, output

skip_0:

;******************

        ;Set flag to implement next channel

        lack 1
        sacl flag

        eint
        ret
```

```
;*******************

;Program Channel 1

channel_1:

        ;set up latch to choose A/D = 2  D/A = 0

        lalk 11
        sacl dummy
        out dummy, latch

        ;Read in data from the A/D

        in ad_in_1, input
        lac ad_in_1
        sub h800
        sacl ad_in_1

;******************

        ;Band Pass Filter

bpf_1:  ;load the "input"
        ; add gain to the

        lac ad_in_1, hpfgain
        sacl bpf_in_nm0_1

        zac

        ;execute the difference equation

        lt bpf_out_nm2_1
        mpy bpf_coeff_out_nm2

        ltd bpf_out_nm1_1
        mpy bpf_coeff_out_nm1

        apac

        lt bpf_in_nm2_1
        mpy bpf_coeff_in_nm2
```

```
        ltd bpf_in_nm1_1
        mpy bpf_coeff_in_nm1

        ltd bpf_in_nm0_1
        mpy bpf_coeff_in_nm0
        apac

        bgz no_add_hpf_1
        addk 4095
no_add_hpf_1: sach bpf_out_nm0_1, hpfdiv

        dmov bpf_out_nm0_1

;*******************

mult_cos_1:

        ;figure out cos

        lar 1, aux_cos_pointer
        lt bpf_out_nm0_1
        mpy *
        pac
        sach c_out_1, shift

;*******************

mult_sin_1:

        ;figure out sin
        ;the sub_in_val is from the cos signal

        lar 1, aux_sin_pointer
        lt bpf_out_nm0_1
        mpy *
        pac
        sach s_out_1, shift

;*******************

lpf_cos_1:

        ;load the "input"
        ; add gain to the
```

```
        lac c_out_1, lpfgain
        sacl lpfcos_in_nm0_1

        zac

        ;execute the difference equation

        lt lpfcos_out_nm1_1
        mpy lpf_coeff_out_nm1

        apac

        lt lpfcos_in_nm1_1
        mpy lpf_coeff_in_nm1

        ltd lpfcos_in_nm0_1
        mpy lpf_coeff_in_nm0
        apac

        bgz noadd_lpfcos_1
        addk 4095
noadd_lpfcos_1: sach lpfcos_out_nm0_1, lpfdiv

        dmov lpfcos_out_nm0_1

;*******************

lpf_sin_1:

        ;load the "input"
        ; add gain to the

        lac s_out_1, lpfgain
        sacl lpfsin_in_nm0_1

        zac

        ;execute the difference equation

        lt lpfsin_out_nm1_1
        mpy lpf_coeff_out_nm1

        apac

        lt lpfsin_in_nm1_1
```

```
        mpy lpf_coeff_in_nm1

        ltd lpfsin_in_nm0_1
        mpy lpf_coeff_in_nm0
        apac

        noadd_lpfsin_1
        addk 4095       ;add to acc short immediate
noadd_lpfsin_1: sach lpfsin_out_nm0_1, lpfdiv

        dmov lpfsin_out_nm0_1

;*******************

ell_cos_1:  ;(125e0 + 4071e1 + 4045f1) / 4096

        ;load the "input"
        ; add gain to the

        lac lpfcos_out_nm0_1, ellgain
        sacl ellcos_in_nm0_1

        zac

        ;execute the difference equation

        lt ellcos_out_nm2_1
        mpy ell_coeff_out_nm2

        ltd ellcos_out_nm1_1
        mpy ell_coeff_out_nm1

        apac

        lt ellcos_in_nm2_1
        mpy ell_coeff_in_nm2

        ltd ellcos_in_nm1_1
        mpy ell_coeff_in_nm1

        ltd ellcos_in_nm0_1
        mpy ell_coeff_in_
        apac

        bgz noadd_ellcos_1
```

```
        addk 4095
noadd_ellcos_1: sach ellcos_out_nm0_1, elldiv

        dmov ellcos_out_nm0_1

;*******************

ell_sin_1:  ;(125g0 + 4071g1 + 4045h1) / 4096

        ;load the "input"
        ; add gain to the

        lac lpfsin_out_nm0_1, ellgain
        sacl ellsin_in_nm0_1

        zac

        ;execute the difference equation

        lt ellsin_out_nm2_1
        mpy ell_coeff_out_nm2

        ltd ellsin_out_nm1_1
        mpy ell_coeff_out_nm1

        apac

        lt ellsin_in_nm2_1
        mpy ell_coeff_in_nm2

        ltd ellsin_in_nm1_1
        mpy ell_coeff_in_nm1

        ltd ellsin_in_nm0_1
        mpy ell_coeff_in_nm0
        apac

        bgz noadd_ellsin_1
        addk
noadd_ellsin_1: sach ellsin_out_nm0_1, elldiv

        dmov ellsin_out_nm0_1

;*******************
```

```
summer_1:

        ;NOTE -- when outputting the answer to the DAC the value
        ; must be less that 2 ^ 12 because of the limit of the DAC
        ;HOWEVER -- the internal answer can remain at 2 ^ 16 resolution

        sqra ellcos_out_nm0_1
        zac
        sqra ellsin_out_nm0_1
        apac

        sach a_squ_hi_1
        sacl a_squ_lo_1
        sach a_squ_out_1, outdiv

skip_1:

;******************

        Set flag to implement next channel

        lack 2
        sacl flag

        eint
        ret

;********************

;Program Channel 2

channel_2:

        ;set up latch to choose A/D = 0  D/A = 1

        lalk 8
        sacl dummy
        out dummy, latch

        ;Read in data from the A/D

        in ad_in_2, input
        lac ad_in_2
        sub h800
        sacl ad_in_2
```

```
;*******************

        ;Band Pass Filter

bpf_2:  ;load the "input"
        ; add gain to the

        lac ad_in_2, hpfgain
        sacl bpf_in_nm0_2

        zac

        ;execute the difference equation

        lt bpf_out_nm2_2
        mpy bpf_coeff_out_nm2

        ltd bpf_out_nm1_2
        mpy bpf_coeff_out_nm1

        apac

        lt bpf_in_nm2_2
        mpy bpf_coeff_in_nm2

        ltd bpf_in_nm1_2
        mpy bpf_coeff_in_nm1

        ltd bpf_in_nm0_2
        mpy bpf_coeff_in_nm0
        apac

        bgz no_add_hpf_2
        addk 4095
no_add_hpf_2: sach bpf_out_nm0_2, hpfdiv

        dmov bpf_out_nm0_2

;*******************

mult_cos_2:

        ;figure out cos
```

```
        lar 1, aux_cos_pointer
        lt bpf_out_nm0_2
        mpy *+
        sar 1, aux_cos_pointer
        pac
        sach c_out_2, shift

        ;check weather to reset the cos_ref pointer

        lar 1, aux_cos_reset
        banz skip_c

        ;Reset the Pointers

        lalk sin_start
        sacl aux_cos_pointer
        lalk reset_ticker
        sacl aux_cos_reset
        b dont_reset_cos_again

skip_c: sar 1, aux_cos_reset
dont_reset_cos_again:

;*******************

mult_sin_2:

        ;figure out sin
        ;the sub_in_val is from the cos signal

        lar 1, aux_sin_pointer
        lt bpf_out_nm0_2
        mpy *+
        sar 1, aux_sin_pointer
        pac
        sach s_out_2, shift

        ;check weather to reset the sin_ref pointer

        lar 1, aux_sin_reset
        banz skip_s

        ;Reset the pointers

        lalk sin_start
```

```
        sacl aux_sin_pointer
        lalk reset_ticker
        sacl aux_sin_reset
        b dont_reset_sin_again

skip_s: sar 1, aux_sin_reset
dont_reset_sin_again:

;*******************

lpf_cos_2:

        ;load the "input"
        ; add gain to the

        lac c_out_2, lpfgain
        sacl lpfcos_in_nm0_2

        zac

        ;execute the difference equation

        lt lpfcos_out_nm1_2
        mpy lpf_coeff_out_nm1

        apac

        lt lpfcos_in_nm1_2
        mpy lpf_coeff_in_nm1

        ltd lpfcos_in_nm0_2
        mpy lpf_coeff_in_nm0
        apac

        bgz noadd_lpfcos_2
        addk 4095
noadd_lpfcos_2: sach lpfcos_out_nm0_2, lpfdiv

        dmov lpfcos_out_nm0_2

;*******************

lpf_sin_2:

        ;load the "input"
```

```
; add gain to the

lac s_out_2, lpfgain
sacl lpfsin_in_nm0_2

zac

;execute the difference equation

lt lpfsin_out_nm1_2
mpy lpf_coeff_out_nm1

apac

lt lpfsin_in_nm1_2
mpy lpf_coeff_in_nm1

ltd lpfsin_in_nm0_2
mpy lpf_coeff_in_nm0
apac

bgz noadd_lpfsin_2
addk 4095
noadd_lpfsin_2: sach lpfsin_out_nm0_2, lpfdiv

dmov lpfsin_out_nm0_2

lpfsin_out_nm0_2
;*******************

ell_cos_2:  ;(125e0 + 4071e1 + 4045f1) / 4096

;load the "input"
; add gain to the

lac lpfcos_out_nm0_2, ellgain
sacl ellcos_in_nm0_2

zac

;execute the difference equation

lt ellcos_out_nm2_2
mpy ell_coeff_out_nm2
```

```
        ltd ellcos_out_nm1_2
        mpy ell_coeff_out_nm1

        apac

        lt ellcos_in_nm2_2
        mpy ell_coeff_in_nm2

        ltd ellcos_in_nm1_2
        mpy ell_coeff_in_nm1

        ltd ellcos_in_nm0_2
        mpy ell_coeff_in_nm0
        apac

        bgz noadd_ellcos_2
        addk 4095
noadd_ellcos_2: sach ellcos_out_nm0_2, elldiv

        dmov ellcos_out_nm0_2

;*******************

ell_sin_2:  ;(125g0 + 4071g1 + 4045h1) / 4096

        ;load the "input"
        ; add gain to the

        lac lpfsin_out_nm0_2, ellgain
        sacl ellsin_in_nm0_2

        zac

        execute the difference equation

        lt ellsin_out_nm2_2
        mpy ell_coeff_out_nm2

        ltd ellsin_out_nm1_2
        mpy ell_coeff_out_nm1

        apac

        lt ellsin_in_nm2_2
        mpy ell_coeff_in_nm2
```

```
        ltd ellsin_in_nm1_2
        mpy ell_coeff_in_nm1

        ltd ellsin_in_nm0_2
        mpy ell_coeff_in_nm0
        apac

        bgz noadd_ellsin_2
        addk 4095
noadd_ellsin_2: sach ellsin_out_nm0_2, elldiv

        dmov ellsin_out_nm0_2

;*******************

summer_2:

        ;NOTE -- when outputting the answer to the DAC the value
        ; must be less that 2 ^ 12 because of the limit of the DAC
        ;HOWEVER -- the internal answer can remain at 2 ^ 16 resolution

        sqra ellcos_out_nm0_2
        zac
        sqra ellsin_out_nm0_2
        apac

        sach a_squ_hi_2
        sacl a_squ_lo_2

        sach a_squ_out_2, outdiv

skip_2:

;*******************

        ;Set flag to implement next channel

        lack 3
        sacl flag

        eint
        ret

;********************
```

```
;Program Channel 3

channel_3:

        ;set up latch to choose A/D = 1  D/A = 1

        lalk 9
        sacl dummy
        out dummy, latch

        ;Read in data from the A/D

        n dummy, input

laser:  ;routine to power the laser
        ;calculate the voltage on the laser

        lar 1, aux_laser_pointer
        lt amplitude
        mpy *+
        sar 1, aux_laser_pointer
        pac
         sach v_laser, shift
        lac v_laser
        add bias
        sacl v_laser

        out v_laser, output

        ;check weather to reset the sin_ref pointer

        lar 1, aux_laser_reset
        banz skip_laser_reset

        ;Reset the pointers

        lalk sin_start
        sacl aux_laser_pointer
        lalk reset_ticker
        sacl aux_laser_reset
        b dont_reset_laser_again

skip_laser_reset:
```

```
        sar 1, aux_laser_reset
dont_reset_laser_again:

        ;$$$$$$$$$$$$$$$$$$$$$
        ;$$$$$$$$$$$$$$$$$$$$$
        ;$$$$$$$$$$$$$$$$$$$$$

        ;set up latch to choose A/D = 0  D/A = 0

        lalk 1
        sacl dummy
        out dummy, latch

        lalk 2048
        sacl dummy

        ldpk 8
        out 120, output ;should be 120

        ;$$$$$$$$$$$$$$$$$$$$$
        ;$$$$$$$$$$$$$$$$$$$$$
        ;$$$$$$$$$$$$$$$$$$$$$

        ;**********************
        ;**********************

        ;AVERAGE OUTPUTS

        ;Average Channel Zero

        ldpk 5
        zalh out_hi_old_0
        adds out_lo_old_0
        ldpk 4
        adds a_squ_lo_0
        addh a_squ_hi_0

        ldpk 5
        sacl out_lo_old_0
        sach out_hi_old_0

        ;Average Channel One

        zalh out_hi_old_1
        adds out_lo_old_1
```

```
        ldpk 4
        adds a_squ_lo_1
        addh a_squ_hi_1

        ldpk 5
        sacl out_lo_old_1
        sach out_hi_old_1

        ;Average Channel Two

        zalh out_hi_old_2
        adds out_lo_old_2
        ldpk 4
        adds a_squ_lo_2
        addh a_squ_hi_2

        ldpk 5
        sacl out_lo_old_2
        sach out_hi_old_2

        larp 7

        banz no_divide

divide:

        zalh out_hi_old_0
        adds out_lo_old_0

        sach avged_out_0

        zalh out_hi_old_1
        adds out_lo_old_1
        sach avged_out_1

        zalh out_hi_old_2
        adds out_lo_old_2
        sach avged_out_2

        lac avged_out_0
        ldpk 8
         sacl 16
        lalk 1
        sacl 20
        ldpk 5
```

```
lac avged_out_1
ldpk 8
sacl 56
lalk 1
sacl 60
ldpk 5

lac avged_out_2
ldpk 8
sacl 96
lalk 1
sacl 100
ldpk 5

zac
sacl out_hi_old_0
sacl out_lo_old_0
sacl out_hi_old_1
sacl out_lo_old_1
sacl out_hi_old_2
sacl out_lo_old_2

lark 7, number_average

;**********************
;**********************

;AVERAGE OUTPUTS -- AGAIN!!!

        ;Average Channel Zero AGAIN

        zalh avg_out_hi_old_0
        adds avg_out_lo_old_0
        adds avged_out_0

        sacl avg_out_lo_old_0
        sach avg_out_hi_old_0

        ;Average Channel One AGAIN

        zalh avg_out_hi_old_1
        adds avg_out_lo_old_1
        adds avged_out_1
```

```
 avg_out_lo_old_1
sach avg_out_hi_old_1

;Average Channel Two AGAIN

zalh avg_out_hi_old_2
adds avg_out_lo_old_2
adds avged_out_2

sacl avg_out_lo_old_2
sach avg_out_hi_old_2

larp 6

banz no_divide

zalh avg_out_hi_old_0
adds avg_out_lo_old_0
rptk 7
sfl
sach avged_avged_out_0

zalh avg_out_hi_old_1
adds avg_out_lo_old_1
rptk 7
sfl
sach avged_avged_out_1

zalh avg_out_hi_old_2
adds avg_out_lo_old_2
rptk 7
sfl
sach avged_avged_out_2

lac avged_avged_out_0
ldpk 8
sacl 24
lalk 1
sacl 28
ldpk 5

lac avged_avged_out_1
ldpk 8
sacl 64
lalk 1
```

```
            sacl 68
            ldpk 5

            lac avged_avged_out_2
            ldpk 8
            sacl 104
            lalk 1
            sacl 108
            ldpk 5

            ;Zero all the sums

            zac
            sacl avg_out_hi_old_0
            sacl avg_out_lo_old_0
            sacl avg_out_hi_old_1
            sacl avg_out_lo_old_1
            sacl avg_out_hi_old_2
            sacl avg_out_lo_old_2

            lark 6, number_average

no_divide:
no_avg:

        b skip_monitor_laser

        ;Monitor the varriables

        ldpk 4
        lac a_squ_hi_0
        ldpk 8
        sacl 0
        lalk 1
        sacl 4
        ldpk 4
        lac a_squ_lo_0
        ldpk 8
        sacl 8
        lalk 1
        sacl 12
        ldpk 4

        lac a_squ_hi_1
        ldpk 8
```

```
        sacl 40
        lalk 1
        sacl 44
        ldpk 4
        lac a_squ_lo_1
        ldpk 8
        sacl 48
        lalk 1
        sacl 52
        ldpk 4

        lac a_squ_hi_2
        ldpk 8
        sacl 80
        lalk 1
        sacl 84
        ldpk 4
        lac a_squ_lo_2
        ldpk 8
        sacl 88
        lalk 1
        sacl 92
        ldpk 4

        larp 1

skip_monitor_laser:

skip_laser:

;*******************
;

        ;Set "flag" to execute channel zero next

        lack 0
        sacl flag

        eint
        ret

;********************
;********************
;********************
;

clear:  ;load aux register one with the number of mem to clear
```

```
        lark 1, 200h
        lark 2, 400h

        zac

clear1: larp 1
        sacl *+

        larp 2
        banz clear1

        lark 1, 0
        lark 2, 0

        ret

;*****************

setup_sin:

        ;current output frequency set to 500 Hz

        ;sin table scaled by 4096
        ;shift left by four bits

        lark 5, sin_start
        larp 5

        lalk 0
        sacl *+
         1266
        sacl *+
        lalk 2408
        sacl *+
        lalk 3314
        sacl *+
        lalk 3896
        sacl *+
        lalk 4096
        sacl *+
        lalk 3896
        sacl *+
        lalk 3314
        sacl *+
```

```
        lalk 2408
        sacl *+
        lalk 1266
        sacl *+
        lalk 0
        sacl *+
        lalk -1266
        sacl *+
        lalk -2408
        sacl *+
        lalk -3314
        sacl *+
        lalk -3896
        sacl *+
        lalk -4096
        sacl *+
        lalk -3896
        sacl *+
        lalk -3314
        sacl *+
        lalk -2408
        sacl *+
        lalk -1266
        sacl *+

        ret

;*******************

sin_logistics:

        ;memory location 900h is start of sine table

        lalk 902h
        sacl aux_laser_pointer
        lalk 17
        sacl aux_laser_reset

        lalk 900h
        sacl aux_sin_pointer
        lalk 19
        sacl aux_sin_reset

        lalk 905h
        sacl aux_cos_pointer
```

```
        lalk 14
        sacl aux_cos_reset

        ret

;*******************

hpf_coeff:

        ;multiplier = 2 ^ 14
        ;cutoff low = at ??? Hz
        ;cutoff hi = at ??? Hz

        ldpk 4

        lalk 102
        sacl bpf_coeff_in_nm0
        lalk 0
        sacl bpf_coeff_in_nm1
        lalk -102
        sacl bpf_coeff_in_nm2

        lalk -16384
        sacl bpf_coeff_out_nm0
        lalk 30970
        sacl bpf_coeff_out_nm1
        lalk -16179
        sacl bpf_coeff_out_nm2

        ret

;*******************

lpf_coeff:

        ;coefficients set to a low pass filter
        ;multiplier = 2 ^ 12 = 4096
        ;cutoff at 500 Hz

        ldpk 4

        lalk 499
        sacl lpf_coeff_in_nm0
        lalk 499
        sacl lpf_coeff_in_nm1
```

```
        lalk -16384
        sacl lpf_coeff_out_nm0
        lalk 15386
        sacl lpf_coeff_out_nm1

        ret

;*******************

ell_coeff:

        ;coefficients set to an elliptical filter
        ;multiplier = 2 ^ 12 = 4096
        ;cutoff at ??? Hz
        ;dip at ??? Hz

        ldpk 4

        lalk 1651
        sacl ell_coeff_in_nm0
        lalk -2668
        sacl ell_coeff_in_nm1
        lalk 1651
        sacl ell_coeff_in_nm2

        lalk -16384
        sacl ell_coeff_out_nm0
        lalk 29411
        sacl ell_coeff_out_nm1
        lalk -13923
        sacl ell_coeff_out_nm2

        ret

;*******************

laser_setup:

        lalk amp
        sacl amplitude
        lalk bias_
        sacl bias

        ret
        end;
```

APPENDIX 7

PROGRAM TO LINK THE ASSEMBLY PROGRAM

```
/* link.cmd */
h.obj           /* object file for the test1.asm */
-o h.out        /* option to name the output file */

/* see page 8-21 for memory stuff */
/* "TMS320 Fixed-Point Assembly Language Tools" */
/* PAGE 0: ROM: == program space */
/* PAGE 1: RAM: == data space */

MEMORY
{
        PAGE 0: ROM1:  origin = 0h,        length = 30h
        PAGE 0: ROM2:  origin = 050h,      length = 1000h
        PAGE 1: RAM:   origin = 200h,      length = 1000h
}

/* see page 8-23 for section stuff */
/* "TMS320 Fixed-Point Assembly Language Tools" */
/* the "vectors" label is linked to the command file by '.sect "vectors"*/
/*  in the assembly program to load the interrupt flow table at 0h*/
/* the ".text" label is linked to the assembly program to loaf the*/
/*  main program at 50h*/

SECTIONS
{
        vectors:        load = ROM1
        .text:    load = ROM2

}
```

Batch file to compile DSP assembly language program:
File Name = h.asm

```
dspa h.asm
dsplnk -h link.cmd
dsphex -t h.out
```

APPENDIX 8A

DESCRIPTION OF DATA SAVING C-PROGRAM VARIABLES

Table A8.1: Symbol cross reference in data saving C-program.

Symbol in Figure 1.2	Equivalent variable in C-program
Amove	data_0
Astat	data_1
Apin	data_1
Nstat	data_out
Npin	data_out

Table A8.2 Description of data saving C-program variables.

C-program variable	Type	Description
data_out[5000]	double	array to save the output data
data_0	unsigned	Nmove signal from the DSP
data_1	unsigned	Nstat or Npin signal from the DSP
flag0	unsigned	data ready flag for data_0
flag1	unsigned	data ready flag for data_1
length	int	total number of samples to read
count	int	current number of samples taken
go_in	int	program sample flag
clear	int	data sampled flag DSP
pointer_addr0	unsigned	DSP memory address of Nmove
pointer_addr1	unsigned	DSP memory address of Nstat or Npin
flag_addr0	unsigned	DSP memory address of Nmove data flag
flag_addr1	unsigned	DSP memory address of Nstat or Npin data flag

APPENDIX 8B

DATA SAVING C-PROGRAM

```
/*  dsp1.c */

#include <stdio.h>
#include <math.h>
#include <conio.h>

main()
{
        /* Define Variables */
        double data_out[5000];
        unsigned data_0, data_1, flag0, flag1;
        int length, count, go_in, clear;
        unsigned pointer_addr0, pointer_addr1;
        unsigned flag_addr0, flag_addr1;

        FILE *fp;

        /* Define variables to save data in Matlab format */
        typedef struct
        {
                long type;   /* type */
                long mrows;  /* row dimension */
                long ncols;  /* column dimension */
                long imagf;  /* flag indicating imag part */
                long namlen; /* name length (including NULL) */
        }

        Fmatrix;
        Fmatrix x;

        /* Initialize variables */

        pointer_addr0 = 0x438;
        pointer_addr1 = 0x460;

        flag_addr0 = pointer_addr0 + 4;
        flag_addr1 = pointer_addr1 + 4;
```

```
length = 2344;


/* Initialization to store data in Matlab file. */
x.type = 0;
x.mrows = 1;
x.ncols = length;
x.imagf = 0;
x.namlen = strlen("a") + 1;

/* Initialize relevant values to zero. */
count = 0;
flag0 = 0;
go_in = 0;
clear = 0;

clrscr();

/* Enter data taking loop. */
while(go_in == 0)
{
        /* Read "data ready" flag from DSP. */
        outport(0x302, flag_addr0);
        flag0 = inport(0x300);

        if(flag0 == 1)
        {
                /* Read data from DSP if data is ready. */
                /* Read Nmove */
                outport(0x302, pointer_addr0);
                data_0 = inport(0x300);

                /* Read Nstat or Npin */
                outport(0x302, pointer_addr1);
                data_1 = inport(0x300);

                /* Normalize data */
                data_out[count] = sqrt((double)data_1 / (double)data_0);

                gotoxy(1,1);
                printf("%lf",data_out[count]);

                /* Reset the flag to zero for the next set of data */
                outport(0x302, flag_addr0);
```

```
                outport(0x300, clear);

                count++;
                if(count == length)
                {
                        go_in = 1;
                }
        }
}

/* Save the data into a Matlab readable file */
fp = fopen("b:data.mat","w+b");
fwrite(&x, sizeof(Fmatrix), 1, fp);
fwrite("a", sizeof(char), (int)x.namlen, fp);
fwrite(&data_out, sizeof(double), count, fp);
fclose(fp);

printf("\nData have been written to b:data.mat, name: a\n");

}
```

APPENDIX 9A

DESCRIPTION OF CONTROL C-PROGRAM VARIABLES

Table A9.1: Symbol cross reference in control C-program.

Symbol in Figure 1.2	Equivalent variable in C-program
Astat	y_raw
Apin	Vpin
Npin	yd

Table A9.2 Description of control C-program variables.

C-program variable	Type	Description
data[7000]	double	array to save the output data
pointer_addr	unsigned	DSP memory address of the position signal
flag_addr	unsigned	DSP address of the position signal ready flag
Vpin_addr	unsigned	DSP memory address of the normalization signal
upper	int	upper limit of the D/A output value
lower	int	lower limit of the D/A output value
count	int	number of 100 Hz loops executed
flag	int	data ready flag

Table A9.2
(Continued)

C-program variable	Type	Description
go_in	int	program sample flag
U	int	command
length	int	number of program loops to execute
save_length	int	total length of data vector
mode	int	instantaneous position or run loop flag
run	int	sample data flag
scan	int	values of control gains input from keyboard
y	float	current position
y_last	float	previous position
y_ref	float	reference position
Kp	float	position gain
Kd	float	derivative gain
delta	float	derivative signal
u_temp	float	command signal before correction for D/A output
e	float	current error
z	float	current output of lowpass filter
z_last	float	previous output of lowpass filter
a1	float	lowpass filter output coefficient

Table A9.2
(Continued)

b1	float	lowpass filter input coefficient
Vcorrection	float	correction value to correct D/A offset error
Vpin	float	normalization signal from laser diode
y_raw	float	non-normalized pos. signal from sensor module
Kpo	float	position gain input from keyboard
Kp_min	float	minimum value for the position gain
Kp_max	float	maximum value for the position gain
C-program variable	Type	Description
e_last	float	previous error
e_temp	float	temporary error signal
Kdo	float	derivative gain input from keyboard
Kd_min	float	minimum value for the derivative gain
Kd_max	float	maximum value for the derivative gain
Kp_range	float	position gain multiplication and division factor
e_sum	float	integral signal
Ki	float	integral gain
Ki_min	float	minimum value for the integral gain
Ki_max	float	maximum value for the integral gain
Ki_range	float	integral gain multiplication and division factor

Table A9.2
(Continued)

Kio	float	integral gain input from keyboard
L1	float	4th order coefficient of calibration polynomial
L2	float	3rd order coefficient of calibration polynomial
L3	float	2nd order coefficient of calibration polynomial
L4	float	1st order coefficient of calibration polynomial
L5	float	constant coefficient of calibration polynomial
yd	float	normalized position signal
sum	float	sum of all error signals
avg	float	average of all error signals

APPENDIX 9B

CONTROL C-PROGRAM

```
/* control.c */
/* control C-program. */

#include <stdio.h>
#include <math.h>
#include <conio.h>

void main(int argc, char *argv[])

{
        /* Define variables. */
        double data[7000];
        unsigned pointer_addr, flag_addr, Vpin_addr;
        int upper, lower, count, flag, go_in, clear, U, length, save_length;
        int Vreset, mode, run, scan;
        int Kp_power, Kd_power;
        float y, y_last, y_ref, Kp, Kd, delta, u_temp, e, z, z_last, a1, b1;
        float Vcorrection, Vpin, y_raw, Kpo, Kp_min, Kp_max, e_last, e_temp;
        float Kdo, Kd_min, Kd_max, Kp_range, Kd_range, e_sum, Ki;
        float Ki_min, Ki_max, Ki_range, Kio;
        float L1, L2, L3, L4, L5, yd, sum, avg;

        FILE *fp;

        /* Define variables to save data in Matlab format. */
        typedef struct
        {
                long type;              /* type */
                long mrows;             /* row dimension */
                long ncols;             /* column dimension */
                long imagf;             /* flag indicating imag part */
                long namlen;            /* name length (including NULL) */
        }

        Fmatrix;
        Fmatrix x;
```

```
/* Initialize variables. */
mode = 0;                       /* mode = 0 for constant gain,  = 1 for variable gain */
length = 2000;                  /* number of data samples */
y_ref = 70.0;                   /* command position  */
a1 = 0.7548;                    /* lpf input gain */
b1 = 0.5095;                    /* lpf output gain */
Kpo = (float) atof(argv[1]);    /* position gain */
Kdo = (float) atof(argv[2]);    /* derivative gain */
Kio = (float) atof(argv[3]);    /* integration gain */
Kp_range = 2.0;                 /* position gain range multiplier */
Kd_range = 2.0;                 /* derivative gain range multiplier */
Ki_range = 10.0;                /* integration gain range multiplier */
upper = 4095;                   /* command upper limit */
lower = 0;                      /* command lower limit */
pointer_addr = 0x438;           /* DSP address to read feedback signal */
flag_addr = pointer_addr + 4; /* DSP address of data ready flag */
Vpin_addr = 0x410;              /* DSP address to read Vpin signal */
save_length = length * 3;       /* length of data vector to save */
Vcorrection = 0x6;              /* hex correction value to fix D/A */

/* Coefficient values for the curve fit.*/
L1 = 3346;
L2 = -5630;
L3 = 3262;
L4 = -835;
L5 = 141.8;

/* Initialization to store data in Matlab file. */
x.type = 0;
x.mrows = 1;
x.ncols = save_length;
x.imagf = 0;
x.namlen = strlen("a") + 1;

/* Calculate the ranges of the gains. */
Kp_max = Kp_range * Kpo;
Kp_min = Kpo / Kp_range;
Kd_max = Kd_range * Kdo;
Kd_min = Kdo / Kd_range;
Ki_max = Ki_range * Kio;
Ki_min = Kio / Ki_range;

/* Initialize relevant values to zero. */
count = 0;
flag = 0;
```

```
go_in = 0;
clear = 0;
y = 0;
y_last = 0;
z = 0;
z_last = 0;
data[0] = 0;
data[length] = 0;
data[length*2] = 0;
e_sum = 0;
run = 0;
scan = atoi(argv[4]);

/* Ensure that e is not zero to start. */
e = (Kp_max + Kp_min)/2.0;

clrscr();

/* Enter data-taking loop. */
while (run == 0)
{
        if (kbhit())
        /* Press any key to take data. */
        {
                scan = 1;
        }

        if (scan == 0)
        /* Output the position on the screen to set. */
        {
                /* Read non-normalized mobile sensor module signal. */
                outport (0x302, pointer_addr);
                y_raw = inport(0x300);

                /* Read laser diode power feedback signal for normalization. */
                outport (0x302, Vpin_addr);
                Vpin = inport(0x300);

                /* Normalize mobile sensor module signal. */
                yd = y_raw / Vpin;
                /* Calculate current position. */
                y = L1*yd*yd*yd*yd + L2*yd*yd*yd + L3*yd*yd + L4*yd +L5;

                /* Output current position to monitor. */
                gotoxy(1,1);
```

```
        printf("%lf",y);
}

if (scan == 1)
{
        while(go_in == 0)
        {
                outport(0x302, flag_addr);
                flag = inport(0x300);

                if(flag == 1)
                {
                        count++;

                        if( count == length-1 )
                        {
                                go_in = 1;
                        }

                        y_last = y;

                        /* Read non-normalized mobile sensor module signal. */
                        outport (0x302, pointer_addr);
                        y_raw = inport(0x300);

                        /* Read laser diode power feedback signal for
                            normalization. */
                        outport (0x302, Vpin_addr);
                        Vpin = inport(0x300);

                        /* Normalize mobile sensor module signal. */
                        yd = y_raw / Vpin;
                        /* Calculate current position. */
                        y = L1*yd*yd*yd*yd + L2*yd*yd*yd + L3*yd*yd +
                         L4*yd +L5;
                        data[count] = y;

                        /* Output current position to monitor. */
                        gotoxy(1,1);
                        printf("%lf",y);

                        /* Calculate the error. */
                        e_last = e;
                        e = y - y_ref;
                        e_sum = e_sum + e;
```

```
data[count+length] = e;
e_temp = e;

if ( mode == 0 )
{
        /* Mode Zero -- Constant Gain. */

        Kp = Kpo;
        Kd = Kdo;
        Ki = Kio;
}

if( mode == 1 )
{
        /* Mode One -- Variable Gain. */

        if (e_temp == 0)
        {
                e_temp = e_last;
        }

        if ( e_temp < 0)
        {
                e_temp = -e_temp;
        }

        Kp = Kpo / (e_temp);

        if ( Kp > Kp_max )
        {
                Kp = Kp_max;
        }

        if ( Kp < Kp_min )
        {
                Kp = Kp_min;
        }

        Kd = Kdo / (e_temp);

        if ( Kd > Kd_max )
        {
                Kd = Kd_max;
        }
```

```
        if ( Kd < Kd_min )
        {
                Kd = Kd_min;
        }

        Ki = Kio / (e_temp);

        if ( Ki > Ki_max )
        {
                Ki = Ki_max;
        }

        if ( Kio < Ki_min);
        {
                Ki = Ki_min;
        }
}

/* Put error through lowpass filter. */
z_last = z;
z = a1 * ( y + y_last ) - b1 * z_last;
delta = z - z_last;

/* Calculate command signal. */
u_temp = Kp * e + Kd * delta + Ki * e_sum;
u_temp = u_temp + 0x800 + Vcorrection;

/* Clip command signal to +/- 5volts. */
if(u_temp > upper)
{
        u_temp = upper;
}

if(u_temp < lower)
{
        u_temp = lower;
}

data[count+length*2] = u_temp;
U = u_temp;

/* Output command signal to DSP. */
outport(0x302, 0x478);
outport(0x300, U);
```

```
                        /* Reset the flag to zero for the next set of data. */
                        outport(0x302, flag_addr);
                        outport(0x300, clear);
                }
        }

        run = 1;

    }
}

/* Set the D/A to zero volts. */
Vreset = 0x800 + Vcorrection;

outport(0x302, 0x478);
outport(0x300, Vreset);

sum = 0;
go_in = 0;
count = 0;


/* calculate ΣE² */
while( go_in == 0)
{

    sum = sum + data[count+length] * data[count+length];

    if( count == length-1)
    {
        go_in = 1;
    }

    count = count +1;

}

avg = sum/length;
printf("\n%lf",sum);

/* Save the data into a Matlab readable file. */
fp = fopen("b:data.mat","w+b");
fwrite(&x, sizeof(Fmatrix), 1, fp);
fwrite("a", sizeof(char), (int)x.namlen, fp);
fwrite(&data, sizeof(double), save_length, fp);
```

```
fclose(fp);

printf("\nData have been written to b:d.mat, name: a\n");
```

REFERENCES

Chang, T. *Decentralized Robust Control of Interconnected Resonators.*

Davison, E. and T. Chang. *Decentralized Controller Design Using Parameters Optimization Methods.* Control-Theory and Advanced Technology. Vol. 2, No. 2, 131 - 154, 1986.

Davison, E. and T. Chang. *Decentralized Stabilization and Pole Assignment for General Proper Systems.* IEEE Transactions on Automatic Control, June 1990, pp 652 - 664.

Dorf, Richard. *Modern Control Systems, 5th Edition.* New York, NY: Addison-Wesley, 1989.

Johnson, Johnny. *Introduction to Digital Signal Processing.* Englewook Cliffs, NJ: Prentice-Hall, 1989.

Kwong, R. and T. Chang. *Development of Control; System Hardware Demonstration for Third Generation Spacecrafts.* Department of Electrical Engineering, University of Toronto, Toronto, Ontario. June, 1986.

Lewis, Frank. *Applied Optimal Control and Estimation: Digital Design and Implementation.* Englewook Cliffs, NJ: Prentice-Hall, 1992.

Phillips, Charles and Royce Harbor. *Feedback Control Systems, 2nd Edition.* Englewood Cliffs, NJ: Prentice-Hall, 1991.

Ziemer, R. and W. Tranter. *Principles of Communication, Systems Modulation and Noise, 3rd Edition.* Boston, MA: Houghton-Mifflin Co, 1990.