LABVIEW SOFTWARE DEVELOPMENT FOR INPUT AND OUTPUT
MEASUREMENT AND CONTROL OF FLEXLAB

A thesis presented to
the faculty of
the Russ College of Engineering and Technology of Ohio University

In partial fulfillment
of the requirements for the degree
Master of Science

Carrie L. Saunders
November 2006
This thesis entitled
LABVIEW SOFTWARE DEVELOPMENT FOR INPUT AND OUTPUT
MEASUREMENT AND CONTROL OF FLEXLAB

by

CARRIE L. SAUNDERS

has been approved for
the School of Electrical Engineering and Computer Science
and the Russ College of Engineering and Technology by

Jerrel R. Mitchell
Cheng Professor of Engineering and Technology

Dennis Irwin
Dean, Russ College of Engineering and Technology
Abstract

SAUNDERS, CARRIE L., M.S., November 2006, Electrical Engineering

LABVIEW SOFTWARE DEVELOPMENT FOR INPUT AND OUTPUT MEASUREMENT AND CONTROL OF FLEXLAB (122 pp.)

Director of Thesis: Jerrel R. Mitchell

Flexlab’s purpose is to model a real-world flexible structure environment while allowing for a wide range of modeling and control techniques. This thesis discusses how Flexlab was further expanded with LabVIEW software to become an even more adaptable environment for ease of real-world testing.

The first chapter of this thesis discusses the history of Flexlab. Flexlab’s main component is a 3/8 inch, 12 foot long aluminum rod. The purpose of the size of the rod is to provide a very flexible structure to perform experiments, modeling and control.

The second chapter of this thesis describes all of the hardware and software components that make up Flexlab. Understanding the input and output hardware components is critical to ensure that effective software is written.

The third chapter of this thesis describes the software design and development. How the software was designed and how it meets the goals of this thesis is discussed.

Approved:

Jerrel R. Mitchell
Cheng Professor of Engineering and Technology
Acknowledgments

I would like to thank and recognize all of the people who have made this work possible. First I would like to thank my two advisors on this project, Dean R. Dennis Irwin who lead me through many years of graduate work and Dr. Jerrel Mitchell who so kindly helped me finish my thesis. I would also like to thank all of the faculty and staff of the School of Electrical Engineering and Computer Science at Ohio University for their support and education throughout my schooling.

A special thanks goes to Tony Adami, for his help in familiarizing me in Flexlab and for his patience and help showing me how to use the Matlab scripts written to create models and controllers for Flexlab.

Finally, I would like to thank my family, friends, and employees who helped make this possible. Without your help in taking over my tasks as a mother and business owner, I wouldn’t have been able to finish writing my thesis.
# Table of Contents

Abstract .................................................................................................................................................................................. 3
Acknowledgments ........................................................................................................................................................................ 4
List of Figures ............................................................................................................................................................................. 6

1 Introduction .............................................................................................................................................................................. 7
   1.1 History of Flexlab ............................................................................................................................................................ 7
   1.2 Motivation ......................................................................................................................................................................... 11

2 System Definition ........................................................................................................................................................................ 12
   2.1 Personal Computer Platform ........................................................................................................................................... 12
   2.2 Accelerometers ............................................................................................................................................................... 13
   2.3 Position-Sensitive Detector ............................................................................................................................................ 14
   2.4 Motors ............................................................................................................................................................................... 17
   2.5 Floor PSD system ............................................................................................................................................................ 18
   2.6 Mirror PSD system ........................................................................................................................................................ 19
   2.7 Static Mass ...................................................................................................................................................................... 20
   2.8 Dynamic Mass ............................................................................................................................................................... 20
   2.9 LabVIEW Software ....................................................................................................................................................... 23

3 Software Design & Development ........................................................................................................................................... 27
   3.1 Hardware Repair ............................................................................................................................................................ 27
   3.2 System Block Diagram ................................................................................................................................................... 28
   3.3 Software Sampling Rate ............................................................................................................................................... 29
   3.4 LabVIEW Implementation ........................................................................................................................................... 30
      3.4.1 LabVIEW Front Panel ........................................................................................................................................... 30
      3.4.2 LabVIEW Diagram ................................................................................................................................................. 35

4 Demonstration ............................................................................................................................................................................ 47

5 Conclusions and Future Work .................................................................................................................................................. 51

References ..................................................................................................................................................................................... 54

Appendix A – LabVIEW Software .................................................................................................................................................. 55
   Appendix A.1 – LabVIEW Front Panel .................................................................................................................................. 56
   Appendix A.2 – LabVIEW Initialization Routine .................................................................................................................. 57
   Appendix A.3 - Moveable Mass Routine ................................................................................................................................ 63
   Appendix A.4 - Disturbance Routine ................................................................................................................................... 97
   Appendix A.5 - Closed Loop Routine .................................................................................................................................. 101
   Appendix A.6 - Open Loop Routine .................................................................................................................................... 102
   Appendix A.7 - Shutdown Routine ...................................................................................................................................... 105

Appendix B – PIC Program ......................................................................................................................................................... 107

Appendix C – LabVIEW Flow Charts ......................................................................................................................................... 111
List of Figures

Figure 1.1 Flexlab Structure Overview [Strahler] .............................................................. 8
Figure 1.2 Flexlab System with Mirror PSD Implementation [Strahler].......................... 10
Figure 2.1 PSD S-1200 Position Detectability [PSD Manual] ............................................ 15
Figure 2.2 Dimensional Outline and Pin Out of S-1200 PSD [PSD Manual]..................... 16
Figure 2.3 DC Motors and Motor Mount ......................................................................... 17
Figure 2.4 μFlash876 [Neill] ........................................................................................... 21
Figure 2.5 μS-DD4 Darlington Driver Board [Neill] .......................................................... 22
Figure 2.6 μS-IT8 Input Terminator Board [Neill] ........................................................... 23
Figure 2.7 Example Multiply and Divide Program ............................................................ 25
Figure 2.8 Example Multiply and Divide Block Diagram ................................................. 26
Figure 3.1 System Block Diagram .................................................................................. 28
Figure 3.2 Main Configuration Area of the LabVIEW Program ......................................... 31
Figure 3.3 Disturbance Configuration Area of the LabVIEW Program............................... 32
Figure 3.4 Closed Loop Configuration and Display ......................................................... 34
Figure 3.5 Open Loop Display ...................................................................................... 35
Figure 3.6 Flexlab LabVIEW Overview Flow Chart ....................................................... 36
Figure 3.7 Initialization Flow Chart ................................................................................ 37
Figure 3.8 Moveable Mass Flow Chart Overview ............................................................ 38
Figure 3.9 Disturbance Flow Chart ................................................................................ 40
Figure 3.10 Open Loop Control ..................................................................................... 42
Figure 3.11 Closed Loop Control ................................................................................... 44
Figure 3.12 LabVIEW Program to Calculate Matrix Equations ....................................... 45
Figure 3.13 Control Calculation Flow Chart ................................................................... 46
Figure 4.1 A Control Matrix ......................................................................................... 48
Figure 4.2 B Control Matrix ......................................................................................... 49
Figure 4.3 C Control Matrix ......................................................................................... 49
Figure 4.4 Graph of Open Loop vs Closed Loop ......................................................... 50
Figure 5.1 Results of Proof of Concept Example ................................................................ 52
1 Introduction

Real-world flexible structures require complicated control methods that are usually oversimplified in a test environment. Flexlab’s purpose is to model a real-world flexible structure environment while allowing for a wide range of modeling and control techniques. This thesis discusses how Flexlab was further expanded with LabVIEW software to become an even more adaptable environment for ease of real-world testing.

1.1 History of Flexlab

Figure 1.1 shows the schematic concept of the Flexlab structure in its original form. The main component of the structure is a 3/8 inch, 12 foot long aluminum rod. The purpose of the small diameter and long length of the rod is to provide a very flexible structure to perform experiments, modeling and control. The rod is suspended from the ceiling and is attached to a motor bracket. The motor bracket has two permanent magnet DC motors. These motors serve as the system’s disturbance and control actuators. They are mounted orthogonally to each other to provide disturbance and control along the x-axis and y-axis. The top motor (the one closest to the ceiling) has its shaft mounted parallel to the x-axis and is referred to as the x-axis motor. The bottom motor has its shaft mounted parallel to the y-axis and is referred to as the y-axis motor.
As seen in Figure 1.1, the orientation of the system is positive Z is from the floor up to the motors, positive Y is to the right, and positive X is into the. This is the orientation that is assumed throughout this thesis. This orientation was determined from the perspective that the end of the devised is being looked at from the floor PSD’s perspective.
To allow for structure characteristic changes, two movable masses are mounted on the rod. Most recently, one of the masses was mounted with stepper motors to allow for movement while the structure is being controlled. This allows for non-linear control to be studied. Having a test bed for non-linear control study on a flexible structure allows for study of a real-world situation, providing valuable learning information to students.

The four accelerometers depicted in Figure 1.1 are Kistler K-Beam accelerometers. They provide feedback of the movement of the rod at four areas on the rod. These accelerometers can be positioned anywhere along the rod to accommodate different data feedback needs. Usually the accelerometers are positioned orthogonally to each other in close proximity to allow for data collection along the x-axis and y-axis at a point in the rod.

Depicted in Figure 1.1 is a PSD (position sensitive detector) located on the floor below the flexible structure. An infrared light emitting diode (IR-LED) is mounted on the end of the rod. The combination of these two electrical devices provide a two-dimensional position of the end of the structure.

In the thesis “Integration of an Active Optical System for Flexlab” written by Jeremy A. Strahler in March 2000, a second PSD device was added to the system. This PSD device was used in conjunction with a steering mirror and diode laser as seen in Figure 1.2. The purpose of Jeremy A. Strahler’s work was to expand Flexlab’s test facility to provide a method of testing optical system stabilization and control. Several applications of optical path stabilization include the Stellar Interferometer Mission at the
Jet Propulsion Laboratory of the California Institute of Technology as well as the airborne telescope SOFIA (Stratospheric Observatory for Infrared Astronomy). [Strahler]
In order to facilitate an adaptable environment at a reasonable cost, a PC based control system has been developed for Flexlab. The original data acquisition system was a Keithley/Metrabyte/Asyst model DAS-20. Currently Flexlab uses National Instruments data acquisition cards to interface between the Flexlab system and the PC. Each component in Flexlab will be discussed in more detail in Chapter 2.

1.2 Motivation

Graphical development software, such as LabVIEW, allows researchers to rapidly create control applications that are flexible and easy to manipulate. [LabVIEW website] In a University lab environment, flexibility and ease of manipulation are key to allowing minimal downtime and minimal learning curve. The natural flow of students in and out of the lab, when a new student arrives or a student leaves, makes having a system that is custom built very hard to start working to make new changes right away. Using software such as LabVIEW, allows students to quickly become productive in Flexlab by reducing the time it takes to get familiar with the system.

Previously, C and Java were used to develop software to control the flexible structure in Flexlab. It was found through this work, that code created in LabVIEW is able to control the structure at a significantly higher frequencies. A modular controller component is added with the LabVIEW system. This allows one to plug in different control algorithms easily, facilitating testing out different control methodologies. This thesis shows that a graphical development software, like LabVIEW, can be used to control a complex flexible system, such as the structure in Ohio University’s Flexlab.
2 System Definition

Flexlab is composed of various hardware and software components. Understanding the input and output hardware components is critical to ensure that effective software is written. This chapter will discuss each component of Flexlab in detail. Flexlab’s hardware consists of a five input, eight output system. Two of the inputs are direct current (DC) motors that can control and disturb the system. The other three inputs are a 3 axis steering mirror used in optical control experiments. Four accelerometers and two PSD’s (position sensitive devices) are used for position and motion detection of the system. Each PSD has an X and Y output component. These six devices make up the eight outputs in the system. All inputs and outputs are controlled and monitored by a Personal Computer (PC) system. There are also two movable masses that can be used to change the dynamics of the system. Each component will be discussed in more detail in the following sections.

2.1 Personal Computer Platform

The Flexlab system is monitored and controlled by a Personal Computer (PC) based system. Currently, Flexlab’s PC is an Intel Pentium III 500MHz with 256Mb of RAM. The PC system has two National Instrument control boards, a PCI-MIO-16E-4 and a AT-AO-10, to be able to read outputs and write to inputs.

The PCI-MIO-16E-4 board is a high-performance analog, digital and timing input/output (I/O) board for PCI bus computers. The current configuration of Flexlab has the PCI-MIO-16E-4 board reading the output sensors of the system. The functions of this
board include analog input, analog output, digital input, digital output and timing I/O. This board is completely switchless and jumperless allowing them to be easily software-configured and calibrated. [PCI E series User Manual]

The AT-AO-10 National Instruments control board is a high-performance analog output and digital I/O board for the PC. This board is used as the analog output board for Flexlab. Its capability for waveform generation makes it very useful to use for the DC motors. The AT-AO-10 has 10 double-buffered, multiplying, 12-bit digital to analog converters providing 10 analog output channels. The 10 output channels can be unipolar or bipolar voltage outputs. They can provide 4 to 20mA current output or a voltage output with an onboard digital to analog reference voltage of 10V. The analog output channels also have internal timer and external signal update capability for waveform generation. An onboard 1,024-word first in first out buffer is also included for the 10 outputs with transfer rates up to 2000 ksamples/second per channel. Onboard analog output auto calibration circuitry is also included in the AT-AO-10 board. It also has eight digital I/O lines able to sink up to 24 mA of current. More features of the AT-AO-10 board include timer-generated and externally generated interrupts. [LabVIEW AT-AO6/10 User Manual]

2.2 Accelerometers

The current Flexlab system has 4 Klister 8304B10 K-beam Accelerometers positioned along the flexible structure. The accelerometers are usually positioned as an orthogonal pair to be able to measure the X and Y acceleration at a point along the structure, but for the purpose of this work, their position is not significant. These
accelerometers have a +- 10g acceleration range with a sensitivity factor of 200mV/g. They will operate at a temperature range of -40 to 205 F. [Klister manual] More details, including details on high pass filters for each accelerometer, are discussed in Blinn’s thesis.

### 2.3 Position-Sensitive Detector

Flexlab uses two Hamamatsu S-1200 position-sensitive detectors (PSDs) to measure the movement of the structure in two different places. One PSD is used to measure the movement at the end of the structure near the floor. The other PSD is used in conjunction with a steering mirror for optical path stabilization experiments. [Strahler]

The Hamamatsu PSD receives position information from a light source and is capable of measuring 13mm in the X direction and 13mm in the Y direction. Figure 2.1 shows the S-1200 position detectability. The dimensional outline and the pin out for the S-1200 PSD is given in Figure 2.2. The S-1200 PSD has an operating temperature of -10 to 60°C. The use of each PSD will be discussed in further detail in Sections 2.5 and 2.6.
Figure 2.1 PSD S-1200 Position Detectability [PSD Manual]
Figure 2.2 Dimensional Outline and Pin Out of S-1200 PSD [PSD Manual]
2.4 Motors

The disturbance and control motors used in Flexlab are two 48 Volt Indiana General Motor DC permanent magnet motor, model number 4020D-105B. The motors are positioned orthogonally to each other. The top motor (the one closest to the ceiling) has its shaft mounted parallel to the x-axis and is referred to as the x-axis motor. The top motor controls the movement of the rod in the Y direction. The bottom motor has its shaft mounted parallel to the y-axis and is referred to as the y-axis motor. The bottom motor controls the movement of the rod in the X direction. [Blinn] A picture of the motors and motor mount can be seen in Figure 2.3.

![DC Motors and Motor Mount](image)

Since control of the motors is implemented using software, simple hardware is all that is needed to interface to the motors. A voltage-to-current converter for each motor is
implemented. More details, including a circuit diagram in Figure 2.4, is included in Blinn’s thesis.

2.5 Floor PSD system

The floor PSD receives the position information from an infrared light source mounted on the end of the flexible structure. To increase its range of “view,” a Fujinon model CF25B closed circuit television lens is mounted over the PSD. [Blinn] This PSD uses a C4758 signal processing circuit specifically designed for the tetra-lateral Hamamatsu 2-dimensional position-sensitive detectors. It is designed to provide position data independent of the light intensity of the light source. The C4758 provides position data in terms of V/mm with respect to the center of the 2-D PSD. [instruction manual of c4758]

The C4758 signal processing circuit characteristics include a variable output voltage amplitude of ±2.05 V minimum and ±6.75 V maximum based on the reference voltage. The Flexlab system is set up with an output voltage amplitude of ±6.5V. The response time of the C4758 is 30μs with an output noise of 5mV peak-to-peak. When a S-1200 PSD is connected to then, then for every 1mm the light source is moved from the orientation (middle) of the PSD, there is a 1V reading on the output voltage. For example, if the PSD is moved 3mm in the positive X direction, then the X output voltage would read 3V.

The main purpose of the floor PSD is to measure how far off from center the end of the aluminum rod has moved. This is useful when collecting data from the system to be able to model the behavior of the end of the flexible rod due to a disturbance to it. It
then also can be used to help control the end of the rod. Data received from the end of
the structure is put into the control system to know the current position of the end of the
rod so that the controller can correct the structure and center the rod.

2.6 **Mirror PSD system**

The main purpose of the mirror PSD system is to have a test environment for
optical path modeling and control. Optical paths mounted on a flexible structure, such as
Flexlab, are susceptible to disturbances. These disturbances can be a result of the entire
optical system moving relative to the light source as well as movement between the
mirror and the PSD. The closed-loop control studied in Strauler’s thesis attenuates these
disturbances by adjusting the orientation of the mirror. The end goal is to keep the light
image held at a constant position on the PSD device. When this goal is reached, then
stabilization of the optical path is achieved. [Strahler]

The second PSD is used in the optical system of Flexlab. This PSD is referred to
as the mirror PSD. The optical system of Flexlab uses a diode laser as the light source
for the mirror PSD. The power from the diode laser was out of the acceptable power
input of the PSD. The neutral density filter used brings the power level down to an
acceptable range, allowing proper operation of the PSD. [Strahler]

The mirror PSD system uses a different signal processing board than the floor
PSD. The mirror system uses two Hamamatsu C3683-01 signal processing boards, one
for each axis. The C3683-01 signal processing board has different characteristics than
the previously mentioned C4758 board used for the floor PSD. The output voltage range
for the C3683-01 board is ±10V with a response time of 300μs minimum. The typical output noise for the C3683-01 is 25mV peak-to-peak. [Hamamatsu C3683-01]

The last main component of the Mirror PSD system is a Garman Systems Inc. high speed steering mirror. The steering mirror has three piezoelectric actuators that allow for movement at three different positions on the mirror. The actuators have a maximum stroke potential of 20μm. The mirror system has a mechanical amplification mechanism used to increase the stroke of the piezoelectric stack to approximately 100μm. The actuators have a voltage range of -20 to +130V. With this large range of working voltage, an amplifier is needed to drive the actuators. A Garman Systems LV-15 amplifier is used in this system. It has a three-channel amplifier for each of the three actuators which offer a -50 to +150V output range. This amplifier as an adjustable 0 to 100V DC offset. The input voltage range for the Garman Systems amplifier is -5 to +15V as currently setup. [Strahler]

2.7 Static Mass

Originally, two static masses were added to the system to be able to change the dynamics of the system. Each mass was manually movable in between experiments. Currently only one of these masses is static and the 2nd can be dynamically movable as will be discussed in the next section.

2.8 Dynamic Mass

As mentioned in section 2.7, one of the static masses can be converted to a dynamic mass. This mass can be mounted on a moveable bracket run by 2 stepper
motors, allowing moving of the mass while experiments are being run. This allows for study of time varying control.

The stepper motors used for the dynamic mass are Oriental Motors, part number PK-744-03A. They are 2 phase DC stepper motors with 1.8 degree step rotation operating at 0.4A with a resistance of 30 ohm. A μFlash876 by Pond Electronics, as seen in Figure 2.4, is the heart of the driver circuitry for the stepper motors. It contains a PIC16f876 as well as the sub components needed to program and run the PIC. The PIC on this board is self programming, meaning that a separate PIC programmer is not needed to load programs into its memory.

Figure 2.4 μFlash876 [Neill]

Along with the μFlash876 board is a μS-DD4 Darlington Driver Board and a μS-IT8 input terminal board as seen in Figure 2.5 and Figure 2.6, respectively. The driver board is used to drive the two stepper motors. It is capable of driving loads of up to 24V.
The μS-IT8 input terminator board allows interface to many sensors including up to 8 input channels per board. Each input is protected to +/- 20V with filtering to reduce high frequency noise.

Figure 2.5 μS-DD4 Darlington Driver Board [Neill]
LabVIEW Software

LabVIEW is a graphical programming language using icons instead of lines of text to create applications and programs. LabVIEW uses dataflow programming, much like a circuit wiring diagram. In contrast, text-based programming uses instructions to determine the program execution. LabVIEW contains sets of tools and objects which can immediately be used and connect together to create applications. [LabVIEW user manual]

LabVIEW has two main areas in which an application is created. The first area is called the front panel. This is where architecture of the application is designed. Controls and indicators are placed on the front panel. They are the interactive input and output terminals of the application. Common controls of the front panel include knobs, push
buttons, and dials. Common indicators include graphs, slide devices, and digital displays. [LabVIEW user manual]

The second main area in which an application is created is the block diagram. Once the front panel is developed, code is created in a block diagram to connect and control the front panel objects. Terminals, nodes, wires and structures make up the block diagram. The data type of the control or indicator is represented by a terminal. For example, a “DBL” terminal block represents a double-precision, floating-point numeric control or indicator. Information is exchanged between the front panel and block diagram through terminals. The data from a terminal can be an input to a node. [LabVIEW user manual]

Objects on the block diagram that have inputs and/or outputs are called nodes and they perform operations on data. Example nodes would be a multiply or divide function in an application. Wires transfer data between nodes and terminals. Each wire can only be connected to a single data source but can be connected to many other objects to read information from the single data source. Structures are a graphical equivalent of loops and case statements and are used to control conditional code execution, repeat blocks of code or to have code executed in a specific order. [LabVIEW user manual]

Figure 2.7 shows a very simple LabVIEW program with four pairs of numeric input controls labeled A, B, D, and E with two indicators labeled C and F. This example software program shows two basic math operations, multiply and divide. Figure 2.8 shows the block diagram that corresponds to Figure 2.7’s front panel.
In Figure 2.8, nodes A, B, C, D, E, and F can be seen. These terminals correspond to the control input and indicators seen in Figure 2.7. The multiply and divide nodes shown in the figure are connected to two control inputs. Input A and B are connected to the multiply node, while input D and E are connected to the divide node. The indicator C is connected to the output of the multiply node. Indicator C will then display the result of A times B. Similarly, the indicator F is connected to the output of the divide node. Indicator F will display the result of D divided by E.
Figure 2.8 Example Multiply and Divide Block Diagram
3 Software Design & Development

The major goal of this thesis is to convert the old program functionality to LabVIEW. The LabVIEW program should be easy to use and easy to extend. The new LabVIEW program must have the same or better sampling rate as the old software. Another goal of this thesis is to develop software to control the motions of the moveable mass. Before any software design could be started, there were several hardware components that needed repaired.

3.1 Hardware Repair

The first component requiring repair was an op-amp in one of the voltage-to-current converter circuits. One of the trim pod resistors was found to be at 180k ohm on the circuit board when it should be 0.8 ohm. This variable resistor was rated to be 0-40 ohms, so a smaller variable resistor was put in its place to help increase the accuracy of the resistance. A 0-2 ohm variable resistor was put in its place. Zener diodes in the circuit were also blown and were replaced. Once the diodes, resistor and new Op-Amp were in place, the motor X circuitry functioned again.

New PIC programming was written to control the movable mass. The mass can move up or down a set number of steps, or it can go to the home (bottom) position, or it can continually move up and down the system while a disturbance is applied and/or during open or closed loop control. This allows for the study of time varying systems. Refer to Appendix B for the PIC program.
3.2 System Block Diagram

To properly understand what needs to be coded into the new LabVIEW system, first an understanding of the system block diagram is needed. Figure 3.1 shows the system block diagram. This diagram represents all three possible modes of operation of the Flexlab system. When collecting disturbance data, $r(k)$ is a uniform white noise signal. In this mode, the controller is removed and data is collected at $u(k)$. This allows for reading of the system response to the disturbance applied to the system.

When performing open loop experiments, first the disturbance is sent to it, with $r(k)$ containing uniform white noise with the feedback loop disconnected. Once the system is reasonably excited $r(k)$ is set to zero and the system records the reaction of the system after the disturbance is turned off in the open loop case, i.e., the feedback loop is not connected.

When closed loop experiments are desired, then first the disturbance is sent to it for a user specified amount of time with $r(k)$ containing uniform white noise while the
feedback loop is disconnected. Then closed loop is started. The input r(k) is set to zero and the feedback loop is connected through the controller. The system records the response with the feedback controller in the loop and appropriate data is collected for later analysis.

3.3 Software Sampling Rate

One requirement of the program was to have at least a 200Hz sampling rate, which was typically the maximum attainable using the old C, C++ and Java programs. It was found that hardware sampling provided the greater sampling rate. A hardware control example was used as the basis of the closed loop portion of the program. One very useful feature of the written program is notification to the operator if the program fell behind and was not sampling at the frequency specified. The program will warn the operator if it is not running at the frequency specified.

File reading and writing is also a key to being able to sample fast enough. A major need of the output files are that they can be read by mathematical software like Matlab for later review and analysis. Matlab is the current software used for analysis and can read tab delimited files easily. Tab delimited files can also be read by almost any other mathematical or analytical software. The first natural choice for file writing then was to a spreadsheet file. However, file input and output for spreadsheet types of files was found to be too slow. When this type of file input and output was used, the system could not run at the minimum sample rate. A much faster way of writing to files in LabVIEW is to write to a plain text type of file. To solve the need for speed and tab
delimited data, an array to spreadsheet string conversion is done before writing to the plain text file. This satisfies both needs.

The number of input and outputs recorded during operation limits the sample rate at which the system can run without losing data. It has been found that the system can run at 400-500Hz with a small number of inputs and outputs being recorded. The system can consistently run at 300Hz with all of the current inputs and outputs being recorded.

### 3.4 LabVIEW Implementation

The new LabVIEW software has several different operations. The system can record the disturbance input to the system and the output of the system to the recorded disturbance. Data is collected in this operation is used in system identification. The system also has an open loop mode, where the program can disturb the system and record the natural decay of the system after the disturbance is removed. The last main operation of the system is closed loop mode, where the program can disturb the system and record the decay of the system after the disturbance has been removed. All data is recorded in a tabbed delimited file for ease of importing into analysis software such as Matlab for further analysis.

#### 3.4.1 LabVIEW Front Panel

There are three main areas in the front panel. The first section is for general configuration as seen in Figure 3.2. Here the different modes of operation can be selected as well as values used for each of the modes of operation. For example, disturbance, closed loop and open loop operations can be turned on and off. The
sampling frequency is also set in this section as well as the number of samples to take when performing open or closed loop. In this section, the outputs of the system to be recorded are also chosen. All 8 outputs can be chosen to be recorded or none can be chosen.

![Figure 3.2 Main Configuration Area of the LabVIEW Program](image)

Figure 3.2 Main Configuration Area of the LabVIEW Program

The next section in the front panel with the label “Disturbance” is where disturbance specific options are selected and where a real-time display of the selected outputs are shown. This can be seen in Figure 3.3. The first choice is to get the disturbance data from a tab-delimited file or to create a random disturbance from a white noise generator. If the data source for the disturbance is a file, then the column for motor X and motor Y needs to be specified.
Figure 3.3 Disturbance Configuration Area of the LabVIEW Program

If a disturbance file is not provided, then the program needs several other pieces of data. The program needs to know the disturbance duration in seconds. One or both motor X and motor Y can be selected for creating the disturbance. Output device is on the front panel, but its default value of “2” should not need to be changed. This is required by LabVIEW to be available and its purpose is to tell LabVIEW which control board to use (PCI-MIO-16E-4 or AT-AO-10). An amplitude for the waveform, as well as seeds for motor X and Y, can be selected. Changing the seed value allows for a different noise sample sequence to be generated. If the value is less than or equal to zero, then the noise generator will not be reseeded and will resume producing noise samples as a continuation of the seed. [LabVIEW Measurements Manual]

In the disturbance section, there is also a waveform display. This display shows the signal being sent to the motor X and motor Y. There is also a “scan rate OK”
indicator. This will turn red if the program is unable to sample the outputs at the frequency defined in the main section of the front panel.

The third section of the front panel can be seen in Figure 3.4. The Closed loop section is active only if closed loop was selected in the main configuration area. Here one of the inputs is selected to control the output in the “AI channel” section. The value here needs to correspond with the column of the array from the main section. For example, if the desired input to control with is accelerometer #1, as seen in Figure 3.2, then the value to put here is 0. Since 0 is the first element when an array starts with 0 as it’s first element. Next, an output channel to control is determined. Usually the input channel and the output channel match. For example, if it is desired to control with the PSD X direction, then control with motor X would be selected. A real-time representation of the inputs as the system is controlling it is provided in the “Closed-Loop Chart” provided in this section of the front panel. Similar to the “scan rate OK” of the disturbance section, this one will also turn red if the program is unable to read the inputs at the rate specified in the main section.
The last section of the front panel is the open loop section as seen in Figure 3.5. The open loop section is the simplest of each of the sections. It displays in real-time the response of the inputs to any disturbance applied to it. If the program is unable to read the information at the rate specified in the main section, then the “scan rate OK” button will turn red.
The LabVIEW program has five different main sections. Each part operates one right after another in a sequence structure. The flow chart seen in Figure 3.6 shows the overall flow of the program. Each of the five different sections and their order of operation can be seen. The first section sets up any necessary file inputs and creates any necessary files for writing data. For example, if closed loop is selected, then the program prompts the user for the A, B, C, and D matrices needed to control the system. It also creates a control output file to record the information from the selected sensors for later analysis. This section also turns on the laser relay if disturbance, open or closed loop routines are selected. There is a 5 second delay after it turns the laser on to allow for proper time for the laser light to warm up. (See to Figure 3.7 for detail program flow for this section.)
Is Both Open and Closed Loop Selected?  

No  

Initialization Routine  

Moveable Mass Routine  

Disturbance Routine  

Is Disturbance Selected?  

No  

Closed Loop Routine  

Is Closed Loop Selected?  

No  

Open Loop Routine  

Is Open Loop Selected?  

No  

Shut down System Routine  

End Program  

Error Output to User to select either Open Loop or Closed Loop, not both

Figure 3.6 Flexlab LabVIEW Overview Flow Chart
Initialization Routine Start

Is Disturbance, Open, or Closed Loop Selected?

Yes

Turn on Laser Relay

Wait for 5 seconds (allows laser to warm up)

No

Is Disturbance Selected?

Yes

Select a Disturbance File to read the disturbance data in

Select a Disturbance File to record the disturbance data in

No

Is Closed Loop Selected?

Yes

Create File to record control data

Create File to record sensor data while the system is being disturbed

Create File to record sensor data while the system is in closed loop

No

Create Array data structure with contents of A, B, C, D matrix files

Create File to record sensor data while the system is in open loop

Is Closed Loop Selected?

Yes

Create File to record sensor data while the system is in closed loop

Create File to record sensor data in open loop

Waiting for all operations to be completed

Store File creation and initialization results

Make sure input and output notations are the same as in thesis text

Initialization Routine end

Figure 3.7 Initialization Flow Chart
If the movable mass is to be used, the second section controls it. The program first checks to see if the motor speed and number of steps to take is greater than zero. If both are greater than zero then the program will run the movable mass routine. The desired movement of the mass is determined and an appropriate section is run for each case. The four options are to move the mass up, down, continual or to the home position. An overview flow chart of the moveable mass section is show in Figure 3.8 for this section of the program. More details on each of the sub processes in this flow chart can be found in Appendix C.
The third section in the program runs the disturbance routines. After the program checks for the proper information to run the disturbance, it then checks to see if the user selected to have file inputs for the disturbance or disturbance needs to be generated. The basic operation is the same, no matter what the source of the disturbance data. First, the software configures the analog input. Next the software reads the information from the selected input devices, writes that information to a file, and then sends the waveform data to the analog output. This sequence is repeated until the desired number of data points have been read. In the case where a waveform is generated, then a uniform white noise waveform program is used to generate a waveform with the amplitude and seed specified by the user. When a disturbance file is selected instead, then the program will open that file and read it into an array structure which is passed onto the analog output. Details on this can be found in Figure 3.9.
Send Error to user that all appropriate file names were not selected

Are all necessary files set up?

Yes

Is Disturbance File Selected?

Yes

Read in disturbance file and create an output array for Motor X and Motor Y

Create Sensor Data file to write to and set up Analog Input

Create Uniform White noise data with amplitude given and seed given for Motor Y

Create Uniform White noise data with amplitude given and seed given for Motor X

Write data to Motor X and Motor Y

Record sensor data to the data file

Is there more Data to read and is the timing not too slow?

Yes

No

Close Sensor Data file and output to user any errors

End Disturbance Routine

No

Is Disturbance File Selected?

Yes

Create Sensor Data file, disturbance data file, and Set up Analog Input

Send Error to user that all appropriate file names were not selected

Is Motor X Selected?

Yes

Write data to Motor X and Motor Y

Record sensor data to the data file

Is there more Data to read and is the timing not too slow?

Yes

No

Close Sensor Data file and output to user any errors

End Disturbance Routine

No

Is Motor Y Selected?

Yes

Create Uniform White noise data with amplitude given and seed given for Motor Y

Create Uniform White noise data with amplitude given and seed given for Motor X

Write data to Motor X and Motor Y

Record sensor data to the data file

Is there more Data to read and is the timing not too slow?

Yes

No

Close Sensor Data file and output to user any errors

End Disturbance Routine

No

Is there more Data to read and is the timing not too slow?

Yes

No

Send Error to user that all appropriate file names were not selected

End Disturbance Routine

No

Are all necessary files set up?

Yes

Is Disturbance File Selected?

Yes

Read in disturbance file and create an output array for Motor X and Motor Y

Create Sensor Data file to write to and set up Analog Input

Create Uniform White noise data with amplitude given and seed given for Motor Y

Create Uniform White noise data with amplitude given and seed given for Motor X

Write data to Motor X and Motor Y

Record sensor data to the data file

Is there more Data to read and is the timing not too slow?

Yes

No

Close Sensor Data file and output to user any errors

End Disturbance Routine

No

Are all necessary files set up?

Yes

Is Disturbance File Selected?

Yes

Read in disturbance file and create an output array for Motor X and Motor Y

Create Sensor Data file to write to and set up Analog Input

Create Uniform White noise data with amplitude given and seed given for Motor X

Write data to Motor X and Motor Y

Record sensor data to the data file

Is there more Data to read and is the timing not too slow?

Yes

No

Close Sensor Data file and output to user any errors

End Disturbance Routine

No

Are all necessary files set up?
The fourth section in the program performs either closed loop or open loop control. The program tests to see if open loop or closed loop is selected. If open loop is selected, then it executes the open loop routine. In the open loop routine, first the analog input is configured. Also the open loop file to record the input data is opened for writing. The next step the program performs is the reading of all of the user selected inputs. It then displays the results in the open loop chart and writes the data to the open loop file. The program continually repeats this process of reading in data and sending it to the display and output file until the number of samples specified by the user specified to have been taken, or it will stop if the program is unable to read the information at the operating frequency specified. After the loop is finished, it then closes the analog input connection and closes the data file. The process flow of the open loop routine is seen in Figure 3.10.
Begin Open Loop Routine

Configure the Analog input and open the Open Loop file for writing

Read from the Analog Input

Display Input Data on the Graph and Write to the Open Loop File

Is there more Data to read and is the timing not too slow?

Yes

No

Close the Analog Input Communication and Output any Errors to User

End Open Loop Routine

Figure 3.10 Open Loop Control
If closed loop is selected, then in the forth section it performs the closed loop routines as described in Figure 3.11. First the data files and analog input are set up. The program then reads all of the selected analog inputs. This data is then displayed in the closed loop control chart on the front panel and it is also stored into the input data file. The A, B, C, and D matrices are sent to the control calculation as well as the $u(k)$, and $x(k)$. Figure 3.12 is the LabVIEW program that calculates the matrix recurrence equations that is implemented to realize a digital controller. A flow chart of the control calculation can be seen in Figure 3.13. The output sequence $y(k)$ is calculated by the control routine seen in Figure 3.13 and is then sent to the analog output device selected by the operator and is also written to the output data file. If the program has more samples to take, and the program is running at the specified rate, then it starts over again at reading another scan of analog inputs. If the program has completed taking data or it is running too slow, it closes the input communication and data files and performs the shut down routine.
Begin Closed Loop Routine

Set up data files and Analog Input

Read one scan of the analog inputs

Display the Analog Input scan in the Closed Loop Chart & write the data to the input data file

Send the A, B, C and D matrices to the Control Calculation as well as the u(k), and x(k)

Send the y(k) results to the Analog output device and write this data to the output file

Is there more samples to take and is the timing not too slow?

Yes

x(k) element is initialized with zeros the first time the loop runs. After that, x(k) is the x(k+1)

No

Close Input communication and data files

End Closed Loop Routine

Figure 3.11 Closed Loop Control
Figure 3.12 LabVIEW Program to Calculate Matrix Equations
The last main section of the program does some simple but very important tasks. One of the tasks it performs is to send a stop command to the stepper motors, in case they are running in continual mode. This section also turns off the laser relay. Finally this section sets the output to the motors to zero before ending. This task is very important. If this is not done, then the next time the program runs, it will start the output at a non-zero value. The output line retains the previous voltage sent. This is causes a constant current to be sent to the motors while they are not operating if the hardware is left on. This can potentially cause damage to the driver circuitry by causing over heating conditions as well as possible over heating and damage to the wiring. It also can interfere with expected future results since the system will start at an unknown, non-zero state.
4 Demonstration

Several proof of concept experiments were performed to verify the usability of the new LabVIEW software developed. This chapter shows the step by step of one of these experiments.

Data was collected from LabVIEW in the disturbance mode. For the example in this thesis, only X data was collected. The X axis of the floor PSD was recorded while the system was disturbed in the X direction. The X direction was disturbed with uniform white noise sent to motor X. This uniform white noise has a mean of zero with a distribution between +/- 5V. This data was collected in a tab delimited file for analysis in Matlab. The data was recorded at 300Hz and 100,000 samples were collected.

To read the tab delimited information into Matlab, the “dlmread” Matlab command was used. This command allows the user to read in any kind of delimited file while also allowing the user to define which rows and columns to read. For this example, the disturbance data is stored to a tab delimited file. This file contains information sent to the motor that controls the X disturbance as well as the Y disturbance. For this example, all of the Y disturbance data is zero. The “dlmread” command in Matlab is used to read only the X column of data into a Matlab variable.

Once the disturbance data is read into a variable and once the response to this disturbance is read in, the Matlab Spectrum function can be used to estimate the frequency response of the transfer function between the input to the x-axis motor and the output of the x-axis PSD. Next an inverse Fourier transform is performed which gives an estimate of the impulse response of the system. A popular algorithm for producing state
space models for systems is called the Eigen Realization Algorithm or ERA., which requires impulse response data to produce the state space A, B, and C matrices. ERA has been programmed in Matlab as a function named “era.” This function was used to create a model between the x-axis motor and the x-axis PSD of Flexlab.

Once a model was created using the ERA method, Matlab m-files created by previous students were used to create the control matrices shown in Figure 4.1, Figure 4.2, and Figure 4.3. The m-file used produced an LQR controller design. The results of running these m-files produced a tenth order controller in which A was 10x10, B was 10x1, C was 1x10 and D was 1x1. The four control matrices were then written to four files, one for each matrix, using the “dlmwrite” Matlab command. These matrix files were then directly used in the new LabVIEW system to test the effectiveness of the controller created. The new LabVIEW system then was run with the same disturbance used when collecting the information for system identification but with closed loop control is also selected.

\[
\begin{array}{cccccccccc}
0.439113 & -0.61625 & 0.291931 & 0.568133 & -0.1667 & 0.39905 & 0.187303 & 0.308429 & -0.26719 & -0.23563 \\
-0.42218 & 0.164587 & 0.289753 & 0.735352 & -0.08444 & 0.542124 & 0.281622 & 0.515507 & -0.15669 & -0.2441 \\
-0.17974 & -0.14994 & 1.004175 & 0.151773 & -0.06079 & 0.095023 & 0.042269 & 0.095153 & -0.05932 & -0.06586 \\
0.126432 & -0.05243 & -0.04276 & 1.015653 & 0.050045 & 0.01574 & 0.023054 & 0.014458 & 0.031855 & 0.025055 \\
-0.47891 & -0.62572 & 0.267515 & 0.659259 & 0.872958 & 0.477533 & 0.198607 & 0.391642 & -0.18749 & -0.21981 \\
-0.25997 & -0.20151 & 0.117453 & 0.201947 & -0.15179 & 1.129211 & 0.054039 & 0.116243 & -0.06536 & -0.0931 \\
0.238597 & 0.411493 & -0.15207 & -0.35713 & 0.04541 & -0.28389 & 0.858308 & -0.36498 & 0.107983 & 0.127072 \\
-0.26793 & -0.52392 & 0.184131 & 0.451345 & -0.04086 & 0.355556 & 0.27802 & 1.311711 & -0.11785 & -0.14208 \\
-0.45061 & -0.55657 & 0.243624 & 0.504633 & -0.12309 & 0.360448 & 0.170105 & 0.354734 & 0.82597 & -0.25085 \\
0.082093 & 0.031356 & -0.0306 & -0.03858 & 0.034962 & -0.02003 & 0.0051 & -0.03369 & 0.075059 & 1.020948 \\
\end{array}
\]

Figure 4.1 A Control Matrix
The process used to implement the closed loop control is as follows. Before the program starts running, it prompts the user for the A, B, C and D matrices to use for the control. The LabVIEW systems data for analysis of the effectiveness of the controller. For this example, the PSD data in the X direction was stored for later comparison. A separate LabVIEW program was created for ease of plotting the disturbance on top of the open loop control, on top of the closed loop control. Figure 4.4 shows the results of the A, B, C and D control matrices. The green waveform is a plot of the disturbance data sent to motor X. The graph is zoomed in to show details though of the open and closed loop performance and the details of the disturbance is not shown in this graph. The white waveform is the open loop response and it is placed on top of the closed loop response shown in red. From Figure 4.4, one can see that the closed loop control brought the system to a still state in a very short time period in comparison to the open loop.
Figure 4.4 Graph of Open Loop vs Closed Loop
5 Conclusions and Future Work

The goal of the work presented in this thesis was to further expand Flexlab with LabVIEW software to become an even more adaptable environment for ease of real-world testing. Chapter 1 discusses the history of Flexlab as well as the motivation for the work presented in this thesis. Chapter 2 describes all of the hardware and software components of Flexlab for a better understanding of the needs of the software developed. Chapter 3 discusses the software design and development done through the work of this thesis including a full description of the software implementation.

For the purpose of this thesis, SISO (single input single output) controllers were created using Matlab controller creation methods developed by previous students. These controllers were created as a proof of concept, to ensure that the program written met the needs for proper modeling and control of the system. The data put into this Matlab system was taken from data collected using the new LabVIEW system. The controller matrices created from the Matlab scripts were successfully input to the new LabVIEW system. These matrices were tested in LabVIEW for their effectiveness in controlling the flexible structure. The developed LabVIEW program records the effectiveness of the control system to a tab delimited file. This allows for later analysis by software such as Matlab. Figure 5.1 shows the results of one experiment in testing the proof of concept for the LabVIEW program written.
Figure 5.1 Results of Proof of Concept Example

Figure 5.1 shows three waveforms overlapped, which represent the movement of the floor PSD in the X direction. The disturbance waveform is in green and the amplitude of this waveform is out of the scope of this graph. The graph is zoomed in to show the open loop, shown in white, and the closed loop, shown in red, in detail. This data was taken at 300Hz. This graph shows that the LabVIEW program designed in this thesis can effectively implement a controller operating at 300Hz. LabVIEW greatly simplifies controller implementation and the collection of data for analysis or system identification.

When performing these proof of concept experiments some undesired signals were recorded from the PSD device. When experiments for collecting floor PSD data were performed, sometimes it was found that the floor PSD would go into what appeared
to be saturation mode. The signal from the floor PSD would go from an expected waveform, to a complete scattered, noisy signal. This signal would return to normal if the PSD was covered up to block all light sources to it.

When researching information about the PSD systems, including the mirror PSD system, it was found through Strauler’s work that the laser light source would cause a PSD to go into saturation. The laser light source was too powerful for the S-1200 PSD. The way this was prevented was to put a neutral density filter on the mirror PSD to reduce the light power from the laser light source. The floor PSD, however, does not have a neutral density filter on it since it uses a LED light source. It appears that when experiments are run, there is a chance that the laser from the mirror PSD system could be reflected off of the structure or the mirror itself and bounce back to the floor PSD. This appears to be what was causing this behavior of the floor PSD. A future improvement would be in preventing any stray laser light beams from being able to fall upon the floor PSD system.

In the process of testing the functionality of the moveable mass, it was found that the moveable mass sometimes would rotate about the flexible aluminum rod. The moveable mass system is not symmetrical about both X and Y axis, it is only symmetrical about one axis. This rotation can cause changes in the characteristics of the system. Future improvement could include preventing this rotation when non-linear control is studied.
References


Appendix A.1 – LabVIEW Front Panel
Appendix A.2 – LabVIEW Initialization Routine
Appendix A.3 - Moveable Mass Routine
Appendix A.4 - Disturbance Routine
Appendix A.5 - Closed Loop Routine
Appendix A.6 - Open Loop Routine
Appendix A.7 - Shutdown Routine
Appendix B – PIC Program

#include <uF876.h>
#include <stdlib.h>
#include <input.c>

#BYTE port_b = 0x06

BYTE const POSITIONS[4] = {0b0011, 0b0110, 0b1100, 0b1001};

boolean Home_Switch = FALSE;
boolean Top_Switch = FALSE;
boolean Stop = FALSE;
BYTE Stepper_State = 0;

void drive_stepper(int step_delay, char dir, int steps);
void go_to_home(int step_delay);
void continual(int step_delay);
void is_home(void);
void is_top(void);

#int_rda
void serial_isr()
{
    char received;
    received=getc();
    printf("Int...");
    if(received=="s")
        STOP=TRUE;
}

void main()
{
    byte step_delay, steps;
    char command, dir;

    set_tris_b(0xf0);

    port_b = 0;

    while(TRUE)
{ command=getc(); // get user's command

STOP=FALSE;

if(command == 'h')
{
  is_home();
  is_top();
  step_delay=get_int();

  go_to_home(step_delay);
}
if(command == 'm')
{
  is_home();
  is_top();

  step_delay=get_int();

  printf("StepDelay:%u ", step_delay);

dir=getc();

  while(dir!='u' && dir!='d')
  {
    dir=getc();
  }

  steps=get_int();

  printf("Steps:%u ",steps);

  drive_stepper(step_delay, dir, steps);
}
if(command == 'c') // continual movement
{
  is_home();
  is_top();
  step_delay=get_int();

  continual(step_delay);
}
drive_stepper(int step_delay, char dir, int steps)
{
    int i;

    for(i=0; i<steps; i++)
    {
        is_home();
        is_top();

        if((Home_Switch && dir != 'u') || (Top_Switch && dir == 'u'))
            printf("%u=%u", i, steps);
        return;
    }
    else
    {
        delay_ms(step_delay);

        port_b = POSITIONS [Stepper_State];
        if(dir == 'u')
            Stepper_State = (Stepper_State +1) & (sizeof(POSITIONS)-1);
        else
            Stepper_State = (Stepper_State -1) & (sizeof(POSITIONS)-1);
    }
}

void go_to_home(int step_delay)
{
    while(!Home_Switch)
    {
        delay_ms(step_delay);

        port_b = POSITIONS [Stepper_State];
        Stepper_State = (Stepper_State -1) & (sizeof(POSITIONS)-1);
        is_home();
    }
}
void continual(int step_delay)
{
    enable_interrupts(global);
    enable_interrupts(int_rda);

    while(!Stop)
    {
        while(!Home_Switch && !Stop) // go down until hit Home Switch or Stop command
        {
            delay_ms(step_delay);

            port_b = POSITIONS [Stepper_State];
            Stepper_State = (Stepper_State -1) & (sizeof(POSITIONS)-1);
            is_home();
        }
        while(!Top_Switch && !Stop) // go up until hit Top Switch or Stop command
        {
            delay_ms(step_delay);

            port_b = POSITIONS [Stepper_State];
            Stepper_State = (Stepper_State +1) & (sizeof(POSITIONS)-1);
            is_top();
        }
        is_home();
        is_top();
    }
    disable_interrupts(global);
}

void is_home()
{
    Home_Switch = !bit_test(port_b,4);
    if(Home_Switch)
        putc('b'); // let PC know stepper motors are at home (bottom)
}

void is_top()
{
    Top_Switch = !bit_test(port_b,5);
    if(Top_Switch)
        putc('t'); // let PC know stepper motors are at top
Appendix C – LabVIEW Flow Charts
Overview Routine

Start

Is Both Open and Closed Loop Selected?

No

Initialization Routine

Yes

Error Output to User to select either Open Loop or Closed Loop, not both

Moveable Mass Routine

Is Disturbance Selected?

No

Yes

Is Closed Loop Selected?

No

Is Open Loop Selected?

No

Yes

Shut down System Routine

End Program
Initialization Routine Routine

Initialization Routine Start

Is Disturbance, Open, or Closed Loop Selected?
  Yes
  Turn on Laser Relay
  Wait for 5 seconds (allows laser to warm up)

Is Closed Loop Selected?
  Yes
  Create a file to record control data
  Select A, B, C, D matrix input files
  Create Array data structure with contents of A, B, C, D matrix files

Is Closed Loop Selected?
  Yes
  Create File to record sensor data while the system is being disturbed
  Create File to record sensor data while the system is in closed loop
  Create File to record sensor data while the system is in open loop

Is Disturbance Selected?
  Yes
  Select a Disturbance File to read the disturbance data in
  Select a Disturbance file to record the disturbance data in

Is Closed Loop Selected?
  Yes
  Create a file to record control data

Waiting for all operations to be completed

Store File creation and initialization results

Initialization Routine end

Make sure input and output notations are the same as in thesis text
Begin Moveable Mass Routine

Is the Motor Speed > 0 and Number of steps > 0

Yes

What is the Value of "Move Motor" input?

0 – Move motor Up
1 – Move Motor Down
2 – Home
3 - Continual

Do Nothing since no speed or number of steps were entered

Moveable Mass "Up" routine
Moveable Mass "Down" routine
Moveable Mass "Home" routine
Moveable Mass "Continual" routine

End

Moveable Mass Routine Overview Routine
Begin Moveable Mass Up

Output to the user to select a speed < 255

Is the motor speed > 255?

No

Set up the serial port and baud rate for communication with the PIC board

Is the number of steps > 255?

No

Set up the serial port and baud rate for communication with the PIC board

Yes

Send the “m” character and wait 3 ms for command to be received

Send the motor speed and wait 3 ms for command to be received

Send a “carriage return” character followed by the “u” character and wait 3 ms

Send the number of steps to take and wait 3 ms

Send a “carriage return” character, close the serial line and wait 3 ms

End Moveable Mass Up

Number of steps / 255 Store Quotient & Remainder

Send a “carriage return” character

Send the number of steps to take and wait 3ms

Send a “carriage return” character and wait 3 ms

Has the loop executed Quotient times?

Yes

Send a “u” character and wait 3 ms

Send the “remainder” number of steps to take, wait 5 ms

Send a “carriage return” and wait 3ms

No

Send the motor speed and wait 3 ms for command to be received

Send 255 (number of steps to take) and wait 5 ms

Send a “carriage return” and wait 3 ms

Send the “m” character and wait 3 ms

Send the Motor speed and wait 3 ms

Send the “carriage return” and wait 3 ms

Send a “u” character and wait 3 ms

Send the “remainder” number of steps to take, wait 5 ms

Send a “carriage return” and wait 3ms

Moveable Mass Up Routine
Moveable Mass Down Routine

Begin Moveable Mass Down

Output to the user to select a speed < 255

Is the motor speed > 255?

No

Set up the serial port and baud rate for communication with the PIC board

No

Is the number of steps > 255?

Yes

Number of steps / 255 Store Quotient & Remainder

Set up the serial port and baud rate for communication with the PIC board

Send the "m" character and wait 3ms for command to be received

Send the motor speed and wait 3ms for command to be received

Send a "carriage return" character and wait 3ms

Send the number of steps to take and wait 3ms

Send a "carriage return" character, close the serial line and wait 3ms

End Moveable Mass Down

Yes

Send 255 (number of steps to take) and wait 5ms

Send a "carriage return" character and wait 3ms

Send the "m" character and wait 3ms

Send the Motor speed and wait 3ms

Send the "carriage return" and wait 3ms

Has the loop executed Quotient times?

Yes

Send a "d" character and wait 3ms

Send the "remainder" number of steps to take, wait 5ms

Send a "carriage return" and wait 3ms

Send the motor speed and wait 3ms for command to be received

Send a "carriage return" character followed by the "d" character and wait 3ms

Send the number of steps to take and wait 3ms

Send a "carriage return" character and wait 3ms

Send 255 (number of steps to take) and wait 5ms

Send a "carriage return" character and wait 3ms

Send the "m" character and wait 3ms

Send the Motor speed and wait 3ms

Send the "carriage return" and wait 3ms

Send 255 (number of steps to take) and wait 5ms

Send a "carriage return" character and wait 3ms
Begin Moveable Mass Home

Set up the Serial port and Baud rate for communication with the PIC board

Send the character ‘h’ to the PIC program, wait 2ms

Format the motor speed input into a string

Send the motor speed to the PIC program, wait 2ms

Send the “carriage return” character to the PIC

Close the serial port communications, wait 2ms

End Moveable Mass Home

The character ‘h’ sent to the PIC microprocessor tells the PIC to move down till the home limit switch is triggered.

Moveable Mass Home Routine
Set up the Serial port and Baud rate for communication with the PIC board

Send the character 'c' to the PIC program, wait 2ms

Format the motor speed input into a string

Send the motor speed to the PIC program, wait 2ms

Send the "carriage return" character to the PIC

Close the serial port communications, wait 2ms

End Moveable Mass Continuous

Moveable Mass Continuous Routine
Begin Disturbance Routine

Are all necessary files set up? Yes No

Send Error to user that all appropriate file names were not selected

Is Disturbance File Selected? Yes No

Create Sensor Data file, disturbance data file, and Set up Analog Input

Is Motor Y Selected? Yes No

Create Uniform White noise data with amplitude given and seed given for Motor Y

Is Motor X Selected? Yes No

Create Uniform White noise data with amplitude given and seed given for Motor X

Write data to Motor X and Motor Y

Record sensor data to the data file

Is there more Data to read and is the timing not too slow? Yes No

Close Sensor Data file and output to user any errors

End Disturbance Routine

Disturbance Routine
Begin Closed Loop Routine

Set up data files and Analog Input

Read one scan of the analog inputs

Display the Analog Input scan in the Closed Loop Chart & write the data to the input data file

Yes

Send the A, B, C and D matrices to the Control Calculation as well as the u(k), and x(k)

Send the y(k) results to the Analog output device and write this data to the output file

Is there more samples to take and is the timing not too slow?

No

Close Input communication and data files

End Closed Loop Routine

x(k) element is initialized with zeros the first time the loop runs. After that, x(k) is the x(k+1)
Open Loop Routine

Begin Open Loop Routine

Configure the Analog input and open the Open Loop file for writing

Read from the Analog Input

Display Input Data on the Graph and Write to the Open Loop File

Is there more Data to read and is the timing not too slow?

Yes

No

Close the Analog Input Communication and Output any Errors to User

End Open Loop Routine
Begin Shut Down Routine

Set the Output Line to Zero for Motor X and Motor Y

Turn off Laser Relay

Send the stop command to the moveable mass incase they are in continual mode

End Shut Down Routine

Shut Down Routine