3-2004

# Microprocessor-Based Systems Control for the Rigidized Inflatable Get-Away-Special Experiment

David C. Moody


# MICROPROCESSOR-BASED SYSTEMS CONTROL FOR THE RIGIDIZED INFLATABLE GET-AWAY-SPECIAL EXPERIMENT

THESIS

David C. Moody, 1st Lieutenant, USAF

AFIT/GE/ENG/04-17

**DEPARTMENT OF THE AIR FORCE**
**AIR UNIVERSITY**

## AIR FORCE INSTITUTE OF TECHNOLOGY

**Wright-Patterson Air Force Base, Ohio**

The views expressed in this thesis are those of the author and do not reflect the official policy or position of the United States Air Force, Department of Defense, or the U.S. Government.

AFIT/GE/ENG/04-17

# MICROPROCESSOR-BASED SYSTEMS CONTROL FOR THE RIGIDIZED INFLATABLE GET-AWAY-SPECIAL EXPERIMENT

THESIS

Presented to the Faculty

Department of Electrical and Computer Engineering

Graduate School of Engineering and Management

Air Force Institute of Technology

Air University

Air Education and Training Command

In Partial Fulfillment of the Requirements for the

Degree of Master of Science in Electrical Engineering

David C. Moody, BS

1st Lieutenant, USAF

March 2004

AFIT/GE/ENG/04-17

# MICROPROCESSOR-BASED SYSTEMS CONTROL FOR THE RIGIDIZED INFLATABLE GET-AWAY-SPECIAL EXPERIMENT

David C. Moody, BS

1st Lieutenant, USAF

Approved:

| | |
|---|---|
| ___//signed//_____ | _____ |
| Richard A. Raines, Ph.D (Chairman) | Date |
| ___//signed//_____ | _____ |
| Richard G. Cobb, Maj, USAF (Member) | Date |
| ___//signed//_____ | _____ |
| Anthony N. Palazotto, Ph.D (Member) | Date |

**Acknowledgments**

I would like to thank my God and Savior for giving me the strength to make it work through this research effort. I would also like to thank my wife for her love and support. To my daughter, who has shown me just how much you can actually accomplish on little to no sleep.

I want to give my appreciation to my thesis advisor, Dr. Raines. You have been a great help and motivation. To Maj. Cobb, thanks for keeping me motivated and focused.

David C. Moody

## Table of Contents

Page

## List of Figures

Page

## List of Tables

Page

**Abstract**

As the demand for space based communications and faster data throughput increase, satellites are becoming larger. Larger satellite antennas help to provide the needed gain to increase communications in space. Compounding the performance and size trade-offs are the payload weight and size limit imposed by the launch vehicles. Inflatable structures offer a cost saving opportunity since the structure is significantly lighter and has a reduced storage volume. This allows for smaller launch vehicles and/or increased performance capabilities. Inflatable structures offer possibilities for increased satellite lifetimes, increased communications capacity, and reduce launch costs.

This thesis develops and implements the computer control system and power system to support the Rigidized Inflatable Get-Away-Experiment. The autonomous computer system controls the flow of the experiment while at the same time collecting and recording temperature, pressure, vibration, and image data.

The computer system consists of two processors, one for experiment control and sensor data collection and the second for image data collection. These two systems can work simultaneously to control the flow of the experiment and meet the experiment objectives. Examples of the data collection include heating curves, pressure, tube transfer function plots and images. This thesis also develops the Matlab® tools required to analyze the data collected by the computers for post-flight data processing.

This thesis lays the groundwork for a microprocessor-based architecture for autonomous space experiments. This pioneering effort has been selected for flight testing on-board the U.S. Space Shuttle.

# MICROPROCESSOR-BASED SYSTEMS CONTROL FOR THE RIGIDIZED INFLATABLE GET-AWAY-SPECIAL EXPERIMENT

## I. Introduction

### *1.1 Motivation*

Over the recent decades the demand for satellite communications has increased dramatically. Among the demands are telephony, television, and the new computer networks in the sky. As the demand rose, more and more geostationary satellites have been placed into orbit with longitudinal spacing of only 2º [PrB03]. The frequency spectrum is also becoming very limited. New satellites are being designed to optimize use of orthogonal antenna polarizations and the use of spot beams to provide for spatial division multiple access (SDMA) [PrB03]. SDMA allows for the frequency spectrum to be reused [Skl00]. To provide tight (small beamwidth) beams, the antennas must have relatively large apertures. The problems with large antennas are their weight and the amount of space needed for launch.

One solution to the weight/space problem of mechanically built satellite antennas is the use of inflatable structures. These inflatable structures can be used for building large antennas for communications and radar purposes. They can also be used for building large trusses for manned space stations. The primary benefit of the inflatable structures is the cost savings. Folded tubes can be packed tighter and lighter than solid beams that must be folded and hinged together for launch. Also, compressed gas can be used to produce the force needed to form the structure instead of relying on motors.

The Air Force Institute of Technology (AFIT) has developed an experiment called *Rigidized Inflatable Get-Away-Special Experiment* (RIGEX). RIGEX is the first step towards a future of space inflatable structures. The data collected from RIGEX will help further the development of rigidizable inflatable structures. RIGEX will also be the first experiment involving the deployment of a rigidizable inflatable structure.

### *1.2 RIGEX Objectives*

As the RIGEX effort has progressed, AFIT has given the following mission statement to the effort [DiS01]:

> *To verify and validate ground testing of inflation and rigidization methods for inflatable space structures against zero-gravity space environment*

The results of the ground testing of the inflatable tubes can be found in [Phi03] and [SiT02]. The results from this thesis effort will be used to determine whether testing on the ground in a gravity environment provides a good approximation to inflation in a space environment.

### *1.3 Thesis Objectives*

The purpose of this thesis comprises the following two objectives:

- *Design and construct an embedded computer system to control the experiment and collect temperature, pressure, vibration, and image data.*

*- Develop the necessary analysis tools to analyze the data produced during the experiment.*

The RIGEX system is a structure containing three folded tubes as seen in Figure 1.1. The tubes are rigid at room temperature; but once heated to 125ºC, they become flexible. The heating process allows them to be inflated to form a tube. When they cool down in the inflated state, they become rigid again. After they are inflated, their modal characteristics are measured. This simple routine is sequentially repeated for each tube. Figure 1.2 shows a conceptual drawing of what the experiment should look like [DiS01].

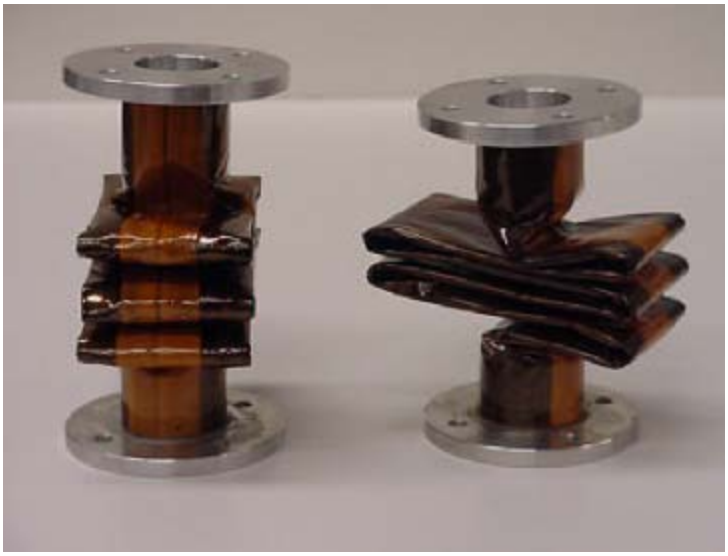


Figure 1.1: Folded rigidizable inflatable tubes

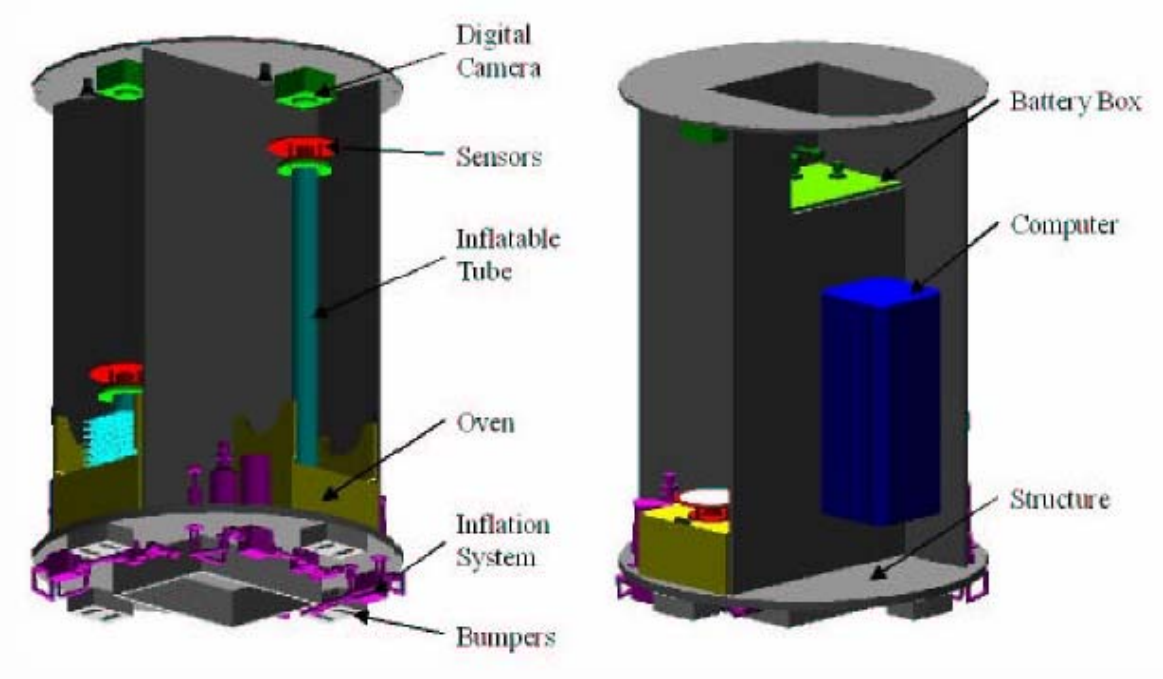



Figure 1.2: Conceptual design of RIGEX structure with its equipment

### *1.4 Thesis Summary*

The thesis is set up in five chapters. The second chapter gives background on the history of inflatable satellites and current efforts in the development of inflatable antennas for use in space. The chapter also gives detail on what has already been accomplished for the RIGEX effort. Chapter Three describes the methodology of how the RIGEX electrical system is developed. The chapter gives details on how data is collected, how the modal characteristics of the inflated tube are measured, how the experiment flows, and what experiments and tests will be conducted to initiate the operation of the RIGEX computers. Chapter Four gives results of the experiments and

tests carried out from Chapter Three. Chapter Five summarizes the thesis effort and makes recommendations for future work in the electronic area of RIGEX.

# II. Literature Review

## *2.1 Introduction*

In this chapter, a review of RIGEX topics will be discussed. The first topic will be the inflatable structures. In the section on inflatable structures, the motivation behind the development of inflatables, historical achievements, and some current inflatable structure experiments will be discussed. The next section will discuss the previous work accomplished for the RIGEX effort. The third section covers some information about the Get-Away-Special (GAS) container the RIGEX system will be carried in. The final section will cover some details about the PC/104 computer system.

## *2.2 Inflatable Structures*

*2.2.1 Motivation* Man's desires and needs to use space for telecommunications and surveillance have continued to grow since the emergence of Earlybird (first geostationary communications satellite). As these needs have grown, so have the demands on the electrical and mechanical components used in these systems. Specific needs such as increased power, larger antennas, and stronger structure have pushed existing technologies to the limits given size, weight, and cost constraints for orbital vehicles. Table 2.1 shows representative launch limitations and costs associated with current space vehicles. To help overcome some of these limitations, inflatable structures are being investigated for possible use in this environment. If inflatable systems could be used in space, the available launch payload weight and space could be used more efficiently. A large inflated antenna, for example, could also be packed into the same amount of space used by a smaller mechanically constructed antenna. If these

Table 2.1: Current Space Vehicle Limitations and Costs [DiS01]

| Launch System | Payload LEO (kg) | Payload GEO (kg) | Payload Diameter (m) | Fairing Length (m) | Costs to LEO FY00 (dollars/kg) |
|---|---|---|---|---|---|
| Atlas II | 8640 | 1050 | 4.2 | 12.0 | 11.6-12.7 |
| Delta II | 5089 | 3890 | 2.9 | 8.5 | 9.8-10.8 |
| STS | 24400 | n/a | 4.5 | 18.0 | 16.4 |
| Titan IV | 21645 | 18600 | 4.5 | 18.9 | 9.9 |
| Ariane 5 (ESA) | 18000 | 12000 | 4.5 | 12.0 | 7.2 |
| H-2 (Japan) | 10500 | 6600 | 4.6 | 5.0 | 15.2-19.5 |
| Long March (China) | 13600 | 2250 | 3.8 | 6.0 | 5.5 |
| Proton (Russia) | 20900 | 2500 | 4.1 | 15.6/7.5 | 2.6-3.6 |

inflatable antennas could be deployed efficiently and sustain a reasonable lifetime, the concept of inflatability could be projected to solar arrays and structures for larger space stations.

*2.2.2 Historic Achievements* In the 1950s, the idea of inflatables was conceived. NASA developed a passive communications satellite known as Echo I. Echo I was a 100 ft diameter sphere. It was constructed from thin sheets of mylar coated with vapor deposited aluminum. These sheets were then bonded together. The final weight and package size of Echo I was 136lbs and fit into a 26 inch diameter spherical container. Echo I was launched on a Delta rocket. It was successful in its inflation and provided adequate reflection of signals for several days. NASA later developed Echo II as a follow-on for Echo I. Echo II was a 135ft sphere. Echo II was the first example of an inflatable rigidizable structure. Once it was inflated and all the wrinkles are removed, the

sphere became rigid due to the aluminum coating. Following Echo II, a smaller series of inflatables known as the Explorer series was developed ([Fr98]).

The use of inflatables declined due to concern over collisions with asteroids and other natural space debris. At that time, the study of meteors showers was not very well known. The estimates of the size of meteor showers were often in error by a factor of 1000. This worry, coupled with the concern of how to keep the inflatables pressurized, caused a decrease in the desire to use inflatable structures ([Th92]).

In the 1970s, inflatable structures began to surface again. The reason for their resurfacing included lower launch weight , smaller packing volume, ease in making curved surfaces, and finally, tolerance to nuclear blasts ([Th92]). Again, problems were encountered with inflatable structures. System designers were having difficulty finding ways to keep the inflatables in orbit for more than a few days or hours. Also, advancements in rocket engine power decreased the need of inflatable structures. Powerful engines were being developed that could carry the loads that inflatables were to replace. This pushed the study of inflatable structures to the side once more.

*2.2.3 Current Developments* Several inflatable structures have been developed in recent years. The primary development has been in the area of antennas for remote-sensing and communications. The following highlights examples of these systems.

*2.2.3.1 Inflatable Antenna Experiment (IAE)* The IAE was developed in the mid-1990s by L'Garde (manufacturer of inflatable structures) to test the validity of innovative

space technologies for NASA. The experiment was to deploy a 14-meter inflatable parabolic reflector, measure the reflector's accuracy for use as an antenna, and investigate structural damping under operational conditions. This experiment demonstrated that a large inflatable structure could be developed to be flight quality and also provide lower cost and smaller storage volume. The IAE was flown in 1996. Once the experiment was activated, it showed some unexpected movements during inflationl; nevertheless, it was successful. The details of the IAE can be found in [FrBi92] and [Fr97].

*2.2.3.2 Inflatable Antenna Arrays* The Jet Propulsion Laboratory (JPL) in California is working to develop three types of inflatable antenna arrays [Hu01]. All three antennas consist of a tubular inflatable frame. The first structure being developed is an inflatable L-band Synthetic Aperture Radar (SAR) array for imaging. The array design calls for the structure to be 10m x 3m. JPL has succeeded in developing a 1/10 sized model. The antenna's final results from ground tests were "a bandwidth of 80 MHz, aperture efficiency of 74 % and a total mass of 15 kg." [Hu01] The antennas 3 dB beamwidths were approximately 5º along the long axis and 12º along the short axis.

The second antenna being developed is an X-band inflatable Reflectarray. This antenna is a 1m reflectarray consisting of a copper surface etched with 1000 isolated patch antennas with a feedhorn attached above the center point [Hu01]. The X-band reflectarray's ground tests resulted in a "37 % aperture efficiency, good radiation patterns, and a total mass of 1.2 kg (excluding inflation system)." The approximate 3 dB beamwidth of the antenna is 5º.

The third antenna being developed is a 3m Ka-band reflectarray. This antenna performs the same functions as the X-band reflectarray but at higher frequencies. The ground test results for the Ka-band antenna were "a surface-flatness of 0.1mm RMS, good radiation patterns and a total mass of 12.8 kg (excluding inflation system)." The final 3 dB beamwidth of the Ka-band antenna is $< 2^{o}$.

*2.2.3.3 Possible Uses of Inflatable Structures* As shown above, the uses of inflatable structures are ever expanding. The use of inflatable antenna arrays will dramatically improve communications in space. It will allow for dynamic control and restriction of broadcast communications. It will allow for higher satellite transmitter power and possibly prevent hostile jamming. Inflatable structures are not restricted to antenna design. L'Garde and colleagues are developing inflatables to help protect optical telescopes and build trusses for space stations. These other developments in inflatable structures can be found in [DiS01] and [Phi03].

### *2.3 Previous RIGEX Work*

The previous RIGEX work ([DiS01], [Phi03]) has identified several areas that require specialty in the electrical and signal acquisition domain.

DiSebastian [DiS01] used a systems engineering approach to provide a foundational design for the RIGEX project. Conceptually, RIGEX is required to be fully autonomous while aboard the space shuttle. To fulfill the autonomous requirement, [DiS01] identified the following needs: a computer system for command and control, various sensors for gathering the required data, video system for taking images of tube

deployment, and the DC power system. Environmental control for the computer and the cameras was also determined to be needed.

As a follow-up to DiSibastian's work, Philley [Phi03] created a prototype single tube version of the RIGEX system. The primary goal of Philley's investigation was to design and fabricate hardware for ground testing of the RIGEX tube deployment. Appendix A contains the prototype setup for initial ground tests of the inflatable tubes. Philley [Phi03] also reiterated the specialized development of the computer, sensors, camera system, and power supply.

*2.3.1 Computer System* The need for RIGEX to be autonomous requires the controlling computer to be embedded in the experiment. This computer must be capable of being initiated by an outside signal, activating the experiment, collecting all data and then shutting down [DiS01]. The computer must also mark fail-safe points within its routine. These fail-safe points are used for experiment shutdown by the shuttle crew in the event of an emergency. These points also allow for restart of the experiment at the shutdown point. These fail-safe points can be seen in different locations within the computer's algorithm. The initial algorithm for the computer can be found in [DiS01].

The PC/104 computer system was chosen to implement the RIGEX control system. It was chosen ([DiS01]) because of its compact size. The computer consists primarily of a small motherboard with an embedded Intel x86 series processor. It has a data bus that allows other circuit boards to be stacked with the processor board. The PC/104 architecture has design flexibility and is an industry standard. Job specific modules can be created and stacked together to create a fully capable embedded system

([PCCon96]). The Tri-M Engineering MZ104+® was the chosen model for the experiment. More specifics on the PC/104 can be found in Section 2.5.

*2.3.2 Sensors* The focus of this experiment is to gather structural data on the inflatable tubes and the deployment of the inflatable tube. This requires the use of pressure sensors, accelerometers, and temperature and voltage sensors. Table 2.2 will provide a summary of the sensors needed.

*2.3.2.1 Pressure Sensors* Pressure sensors are required for two purposes [DiS01]. The first purpose is to monitor the environment. The environment is supposed to be a vacuum, and any pressure build up in the GAS container is vented through a relief valve. Only a single pressure sensor is required to monitor the environmental pressure. For this experiment, the required sensitivity is 0.001 atmospheres (atm). The sensor location on the structure is non-specific due to its general use.

The second pressure sensing requirement is inside the inflation system. DiSebastian [DiS01] states that it is important to monitor the gas pressure within the tubes. This data provides information on how well the inflation process performs and how well the gas is vented after inflation and rigidization. These sensors also require a sensitivity of 0.001 atm and must be placed inside the inflation system. For more on the inflation system, see [Phi03].

*2.3.2.2 Accelerometers* To test the structural properties of the rigidized tubes, a modal analysis must be performed on the tubes. The modal analysis is performed by externally exciting the tubes using a piezo-electric device. This device transfers frequency energy into the tube. The tube's response is then measured at the free end of the tube. To measure this response, a tri-axial accelerometer is placed at the free end of the tube. An accelerometer is an electronic device that measures changes in acceleration. A tri-axial accelerometer measures acceleration in 3-dimensional Euclidean space.

There are two specific location requirements for the accelerometers. The first requirement, as mentioned above, is to have one placed on the top flange of each of the three tubes. The sensitivity for these MEMS devices is 10 millivolts per g (g is acceleration of gravity). The second requirement is for an accelerometer to be placed on the structure to measure the vibrations of the space shuttle. This data is needed to decouple the measured tube data with that of the space shuttle. The required sensitivity is 20 millivolts per g.

*2.3.2.3 Temperature* The inflation and hardening (or rigidization) processes of the tubes are based solely on temperature. To allow the tubes to inflate, the tubes must be heated over 100 degrees C [Phi03]. For the tubes to harden, they must cool. Since the success of both processes is highly dependent on temperature, appropriate temperature sensors must be employed. There are three specific uses for these sensors. The first is monitoring the tubes, the second is monitoring the environment, and the third is monitoring such temperature sensitive devices as the computer and cameras.

The first set of temperature sensors are used to monitor the heating and cooling of the tubes. These sensors allow a temperature profile of the tubes to be developed. Philley [Phi03] used two thermocouples to measure the tube temperature. One was located at the base of the tube and the other was located within the first fold of the uninflated tube.

The second need of the temperature sensors is to measure the environmental temperature within the GAS container. This is a requirement to help the user to better understand the overall experimental environment.

The last function needed for the temperature sensors is to help maintain adequate operating temperatures for the electronics. The PC/104 and cameras are unable to operate below -20º C [TriM01]. The experiment's operational environment can drop significantly below the manufacturer's specified operating temperature. Each of these temperature sensors signals a temperature-regulated heater that is located with the computer or cameras.

*2.3.2.4 Voltage* The requirement for a voltage sensor is not a strict one. The voltage sensor is used to measure the voltages coming from the battery cells. This data is used to calculate the power used by the system and the lifetime of the battery cells.

*2.3.3 Camera System* The camera system is used to measure the static position of the inflated tube. The use of a camera is a cost-effective alternative to using laser displacement sensors. The camera is to be placed at the top of the structure directly

above the inflated tube. This placement gives the best view of inflation as well as accuracy for measuring height [DiS01]. Since there are three experimental tubes, three cameras are required to support the experiment.

The camera images are used to measure the height of inflation during various stages of the process. Comparing pre- and post-tube deployment images determines the height and inflation angle measurements. A ratio is taken between the image area of the targets, with proper adjustments due to tilt angle. This ratio gives a percentage of height achieved. The targets in the images are the top flanges of the tubes.

The images also allow for a third-person perspective of the inflation process. During the actual space flight, no one will be inside the GAS container watching the experiment as it is being conducted. The experimental images allow for a qualitative analysis of the tubes. This video "data" gives the users a way of "sitting in the shuttle." To make these images worthwhile, DiSebastian [DiS01] identified the need for light to be available for the camera. During the entire conduct of the experiment, the GAS containers are sealed off with no ambient light available. Philley [Phi03] mounted two 24

Table 2.2: Summary of Required Sensors [DiS01].

| Sensor | Location | Sensitivity | Size |
|---|---|---|---|
| Pressure | Tubes | 0.001 atm | ¼ inch fitting |
| | Environment | 0.001 atm | n/a |
| Acceleration | Tubes | 10 mV/g | less than 1 cubic in. |
| | Environment | 20 mV/g | n/a |
| Voltage | Power Supply | 0.5 V | n/a |
| Temperature | Tubes | 0.5º C | 0.5 inch square |
| | Environment | 0.5º C | 1 inch square |
| | Components | 1º C | internal |

volt incandescent lights per tube to the inside of the structure. These lights provide sufficient lighting for the cameras.

DiSebastian [DiS01] set the required resolution for the images to be 1000x1000 pixels. This resolution gives a reference target resolution of 700x700 pixels. The reported accuracy of this resolution yields distance measurements of up to 0.01 inches. Philley [Phi03] determined that at least 60 images need to be taken over a one minute interval with an exposure time of 25 milliseconds. This gives a frame rate of one frame per second. In the earlier laboratory experiments, problems were encountered using the software for the chosen camera.

DiSebastian [DiS01] chose to use an ELECTRIM digital camera. This camera was chosen because of its PC/104 compatible interface card and software that is Windows 95/98 and DOS compatible. It has a spatial resolution of 1000x1000 pixels which meets the desired image resolution requirement. The chosen camera's focus and iris settings must be manually adjusted prior to use. Philley [Phi03] recommended the use of a camera with software-controlled focus capability. A manual focus camera would lose its settings due to vibrations caused during the shuttle launch.

*2.3.4 Electrical Power* The RIGEX system requires a DC power supply. This requirement is dictated by the autonomous nature of the experiment. The initial power supply design consisted of 30V battery cells [DiS01]. To develop these battery cells, 20 alkaline D-Cell batteries were stacked in series to create a 30V cell. The D-Cell batteries were chosen because they are dry cells (moist paste) and do not require liquid battery

acid. D-Cells have been approved for other missions flown on the shuttle [NAL85]. The D-Cell battery also has a lifetime of 17 ampere-hours (A-h). This means that the battery would have a lifetime of 1 hour with a current of 17A. By placing the batteries in series, the voltage can be increased to 30V, but the current capacity still remains at 17A-h. To increase the current capacity, the 30V cells can be placed in parallel. The current capacity can be increased to N · 17A-h for N 30V battery cells in parallel.

*2.3.4.1 Computer Power Requirements* The PC/104 requires at least the following supply voltages: ±5V DC and ±12V DC. DiSebastian [DiS01] proposed the use of DC-to-DC converters to drop the 30V supply voltage down to the required voltages.

*2.3.4.2 Heater Power Requirements* The experiment requires the use of electrically driven heaters for the tubes. There is also a requirement for heaters to maintain a stable operating temperature for the computer and cameras. The following table shows the experimental impedance values for the heaters used in the prototype inflation system [Phi03]. The heaters were connected in three different circuits. Heaters 1 and 2 were connected in series and then placed in parallel with the series circuit consisting of heaters 3 and 4. This gave a load impedance of 9.5Ω. Heaters 5 and 6 were connected in parallel on the second circuit, giving a load impedance of 13.65Ω. The final heater circuit consisted of heaters 7 and 8 in series with a load of 22.6Ω. These heaters

Table 2.3: Impedance values for heaters used in RIGEX prototype [Phi03]

| Heater Location on Tube | Heater Number | Impedance (Ω) |
|---|---|---|
| Top Left | 1 | 9.5 |
| Top Right | 2 | 9.5 |
| Bottom Left | 3 | 9.5 |
| Bottom Right | 4 | 9.5 |
| Left Side | 5 | 27.3 |
| Right Side | 6 | 27.3 |
| Front | 7 | 11.3 |
| Back | 8 | 11.3 |

were then powered using a 24V DC supply since the 30V battery cells were unavailable. The calculated current draw for the three heater circuits was 2.53A, 1.76A, and 1.06A respectively, for a total current draw of 5.35A. This yielded a lifetime measurement for one cell of 3.18h. The current draw was measured to be 3.33A as the experiment was started. The current draw later settled to 2.66A as the experiment settled into steady-state [Phi03]. With the maximum experimental current (3.33A), the lifetime of a single cell would be 5.1h. The current heaters used in the prototype system were MINCO Thermofoil® heaters. These heaters combine Kapton® insulation and polymide adhesive to create a flexible heater that can be attached to any surface. These heaters were also space flight rated, reaching temperatures of 260ºC [MI99].

*2.3.4.3 Miscellaneous Power Requirements* The next requirements for the power system are not for system power but for safety standards. NASA has requirements for the power system to be fused, with diode isolation, and the battery box design must meet required gas venting [NAG93, NAL89]. The diode isolation is for the prevention of circulating currents between battery cells in parallel. To do this, a Schottky barrier diode

(has required current capacity) is placed between the battery cell and the junction where other battery cells are connected ([NAL85]). The Schottky diode is chosen because of its low average voltage drops of 0.3V – 0.5V [Pi96]. The fusing is required to provide for short circuit protection. NASA [NAL85] has a table for peak short circuit current of alkaline-manganese batteries. It states a D-cell battery will peak at 8A to 12A. As the above work shows, the experimental current draw from the heaters peaked at 3.33A. If a 6A fuse is chosen (based on the rule of thumb of doubling the current), the fuse would blow before the peak current is reached by the batteries. Battery ventilation is required for use of any aqueous battery ([NAL85]). This ventilation is required to prevent buildup of hydrogen gas.

*2.3.5 Environment* The space environment is a harsh one. Temperatures within the shuttle cargo hold can get as low as –121º C and as high as 93º C [Bo98]. The temperature sensitive components of the experiment must be capable of handling this dynamic temperature range. The PC/104 computer has an operational temperature range of –20º C to 85º C [TriM01]. The lower portion of the temperature range is the critical one for the experiment. When the experiment is activated, the shuttle will be in its low earth orbit with its cargo bay facing the earth, sun, or deep space. When it is facing deep space, the temperature during this period should be at or below 0º C. This lower temperature requires that a heating system be in place to keep the computer in its operating temperature range. Additionally, heaters are required with each of the three

cameras as well as heaters to keep the battery pack warm. The batteries' service hours will decrease as temperature drops [Dur00].

*2.3.5.1 Environmental Control* Since the requirements for heaters has been established, controls must be put in place for the heaters to maintain a standard temperature. As mentioned in Section 2.3.2.3, temperature sensors are to be placed with the computer, with the cameras, and in the battery box. Theses sensors are there to help monitor the temperature at these locations. MINCO makes a temperature controller specially matched to the Thermofoil heaters that are being used in the RIGEX system. The matched controllers do not require a temperature sensor to maintain their set temperature.

### *2.4 Get-Away-Special System*

The GAS system is NASA's way of allowing companies, government agencies and educational institutions access to design relatively inexpensive experiments in space. The GAS can is a single sealed cylindrical container that is connected to the inside of the cargo-bay of the space shuttle. Several of these cans are in the shuttle, allowing multiple experiments during a single shuttle mission. The can, does require some physical restrictions on any experiment being conducted. Table 2.4 gives a list of the physical size and weight limitations of the GAS can cargo [DiS01].

Table 2.4: Physical and Weight Limitations for GAS can cargo [DiS01]

| Constraint | Limit |
|---|---|
| Weight | 200 lbs |
| Size | 19.75 inches (diameter)<br>28.25 inches (height) |
| Flight Time | 14 days |

*2.4.1 Inspection Process* In order for an experiment to get a launch date, the experiment must pass all required safety standards. After the safety requirements have been met, a launch date is assigned based on organizational type and the NASA priority list. Once a date is picked, the experiment is sent to NASA, where the representatives from the organization sponsoring the experiment and NASA inspectors inspect the experiment equipment. After the inspection is completed (provided no failures), the experiment is stored with NASA until it is launched. NASA requires the GAS experiments to be capable of being stored for up to four months. A detailed explanation of NASA's GAS procedure can be found in [DiS01].

*2.4.2 Experimental Process* All experiments requiring electrical power (self-powered or shuttle-powered) require the use of three relays. The first relay is connected to the main power bus of the experiment. This relay (relay C) is normally closed and is only opened only in the case of an emergency by the shuttle crew. Relay A is the environmental control relay. When the shuttle reaches an altitude of 50,000ft, a barometer turns on (close) relay A. This relay is designed to allow for operation of environmental controlling functions. In the case of RIGEX, it activates the heaters for the computer and cameras. Finally, when it is time to activate a GAS experiment, the

shuttle crew turns on relay B (close). This relay powers on the RIGEX computer, thus allowing the experiment to activate [DiS01].

### *2.5 PC/104 Computer*

The PC/104 computer is a small modular circuit board that contains all the necessary equipment to function as a computer. The Tri-M Engineering brand of PC/104 uses a ZFx86 "PC-on-a-chip" processor. The ZFx86 processor is a full computer on a single chip. It requires only external clocks, dynamic random access memory (DRAM) and BIOS read-only memory (ROM)/flash memory. The processor has the following properties [ZF01]:

1) 32-bit processor
2) 8 kilobyte (KB) of layer 1 cache
3) 4-256MB external memory bus capability
4) 12KB fail-safe boot ROM
5) Watchdog timer for timing events and processes
6) Pulse-width-modulator for controlling servos and motors

*2.5.1 Programming* The PC/104 computer board has the capability of running Linux, DOS, Windows 9X, and Windows NT [TriM01]. The current prototype setup uses Windows 98 as its operating system. This general operating system configuration can allow for any high-level programming language such as C, BASIC or JAVA to be used.

*2.5.2 Interfaces* The ZFx86 processor has several built-in interfaces [ZF01]. These interfaces include:

1) Universal serial bus (USB)
2) Extended IDE interface for connecting to hard drives and CD-ROMs
3) Standard computer interfaces for: keyboard, PS2 mouse, and floppy drive
4) Full PCI bus
5) Full ISA bus
6) IRQ pins to establish interrupts, allowing the computer to be notified of events by external devices
7) 1 parallel port for such uses as printing
8) 2 serial ports for RS-232 based communication with speeds up to 115.2kbaud
9) Ethernet 10/100BaseT

To access these connections with standard cables, it requires the utility board to be stacked on the primary computer board with the appropriate cable connections between the two.

*2.5.3 Memory* The PC/104 uses synchronous DRAM for its volatile memory. It has an onboard dual-inline-memory-module socket capable of handling up to 64MB of DRAM. For non-volatile memory requirements, the board has a 32-pin socket to house an M-Systems “DiskOnChip 2000” flash memory module which is capable of holding

from 8MB to 1GB of memory [TriM01]. Flash memory is solid state memory devices that can be read, written to, and erased when needed.

*2.5.4 External Boards* The PC/104 has a robust architecture allowing for many application-specific cards to be added to the computer. Some example types of cards found on the market for the PC/104 include:

1) Analog-to-Digital (A/D) converter boards
2) Counter/Timer boards
3) Digital camera interface boards
4) Relay boards
5) Global positioning system boards
6) Power supply boards

The RIGEX computer will use several of these boards.

Appendix A shows the current PC/104 configuration using the following application specific boards:

1) Diamond Systems DMM-32-AT – 32 Channel 16-bit A/D converter board
2) Diamond Systems Quartz-MM Counter/Timer board
3) Diamond Systems Pearl-MM 16 Relay switching board
4) Electrim EDU-1000U Digital Camera and interface board

The only programming information currently available for the boards is the manuals and some sample programs provided by the manufacturer.

### *2.6 Summary*

This chapter covered required background knowledge needed to proceed with designing the electrical system for RIGEX. The motivation, history, and current developments of inflatable structures are discussed. The previous work accomplished is analyzed. An understanding of how the experiment will be carried into space is developed in the GAS container section. Finally, an introduction to the PC/104 is covered.

# III. Methodology

## *3.1 Overview*

Section 2.3 gave brief descriptions of what is required of the electrical system in the RIGEX experiment. The goal of this chapter is to look at each of the requirements in more detail, analyze the options available, and come to a preliminary decision. The requirement areas to be studied are:

1) Power
2) Data acquisition
3) Height/Displacement of inflated tubes
4) Tube excitation
5) Environmental control
6) Computer configuration
7) Flow of operation
8) Required tests

All decisions and preliminary designs are made using logical design procedures and inputs/requirements from the user. Many of the requirements above can be solved in a variety of ways. The decisions made in this chapter are done so to minimize complexity, to fulfill the requirement, and to meet the safety requirements set forth by NASA.

## *3.2 Power System*

The entire experiment requires a DC power supply. The power supply must provide a supply voltage in the range of 24V to 30V. In order for the experiment to

perform its function, power must be supplied to two critical systems: tube heaters and computer.

### *3.2.1 Heater Power*

*3.2.1.1 Requirements* The heaters for the tube ovens are MINCO Thermofoil® heaters [MI99]. The experimental transition temperature for the tube inflation is 125ºC. According to [Phi03], the tubes must be heated approximately 47 minutes when starting from room temperature. The power supply is required to provide a maximum current of 3.33A for the duration of the heating.

*3.2.1.2 Options* There are two primary options for power. The first option is using power provided by the shuttle using the Hitchhiker GAS cans. The shuttle can provide 560W of power using two 28VDC power lines [NAG99]. This would provide all power needed to run the experiment. Choosing to use shore power for the experiment limits opportunities to get a flight date on the shuttle. This is due to the smaller number of GAS cans available with power versus the number of GAS cans with no power.

The other option is to self-power the experiment using some form of batteries. NASA [NAL85] provides all the safety guidelines for choosing batteries for manned space vehicle flights. The choice of using lithium-based batteries is ruled out because of the low safety approval rate and possible hazards. The battery choices are then reduced to choosing from liquid, gel, or solid batteries. If liquid or gel batteries are chosen, the container holding the batteries would be required to have a venting system for gases and

to be able to hold any spilled battery acid. If solid or paste batteries are chosen, there are no extra requirements made on the battery housing.

*3.2.1.3 Decision* It was decided in previous work [DiS01, Ph03] that D-cell batteries are the most suitable choice for the experiment. That decision was based on the high safety approval success. The D-cell batteries are sealed paste batteries. Each battery is supposed to provide 1.7A-hr of service.

To help determine if this choice of battery will work, a battery power system was designed and an experiment was conducted using one 30V battery cell consisting of D-cell batteries. This system was designed to power the three heater circuits [Phi03] for one hour. During the experiment, each heater circuit's current was measured and the battery voltage measured. Each of the measurements was taken using an HP 34401A Multimeter connected to a computer running LabView [NaI03]. Figure 3.1 shows the connection diagram for the battery cell test. Once the test was conducted, the consumed power and the total input impedance of the heaters were calculated using Ohm's Law. Figure 3.2 shows the results of the experiment.

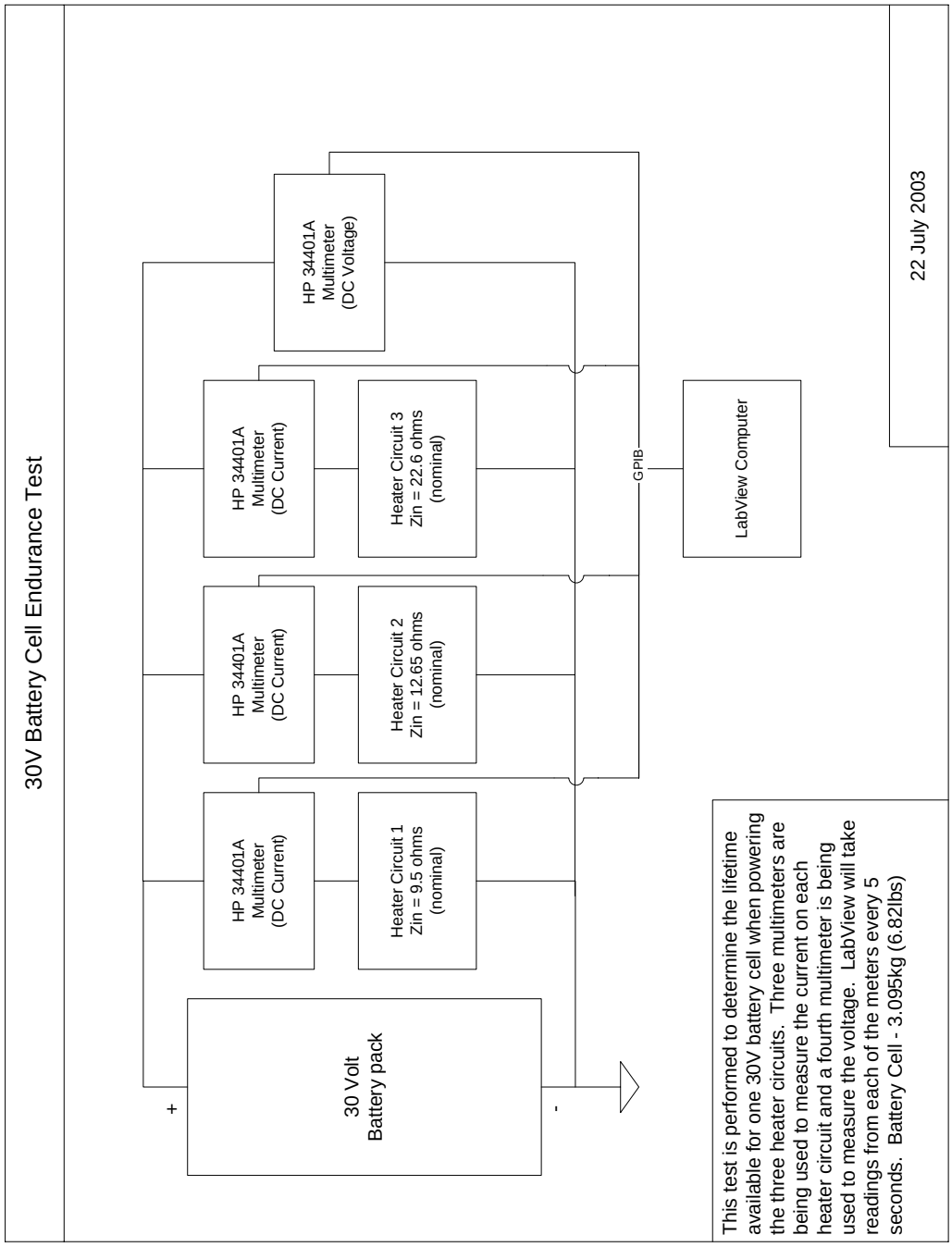



Figure 3.1: 30V battery cell endurance test

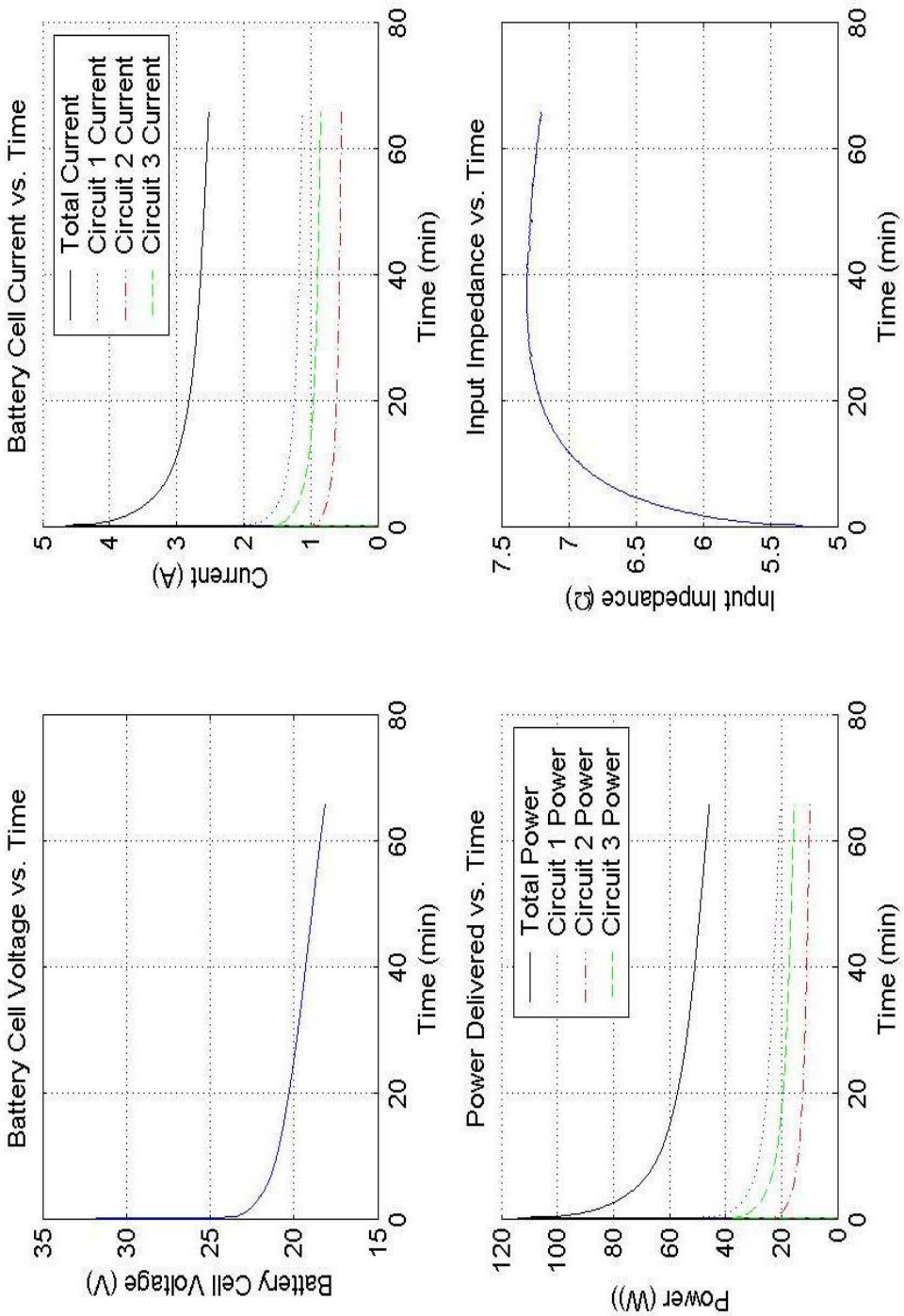



Figure 3.2: Battery cell endurance test results

From the plots shown, it is apparent that one 30V battery cell does not meet the current requirement set forth. The total current draw on the heater circuits drops below 3A after only ten minutes. After 60 minutes, the total current is about 2.5A. If another identical battery cell is placed in parallel, the current capability of the cells should double. This provides the required current over the duration of the tube heating. As a result, two 30V battery cells composed of D-cell batteries are placed in parallel for each tube oven. The battery cells for each of the three tubes are isolated from each other.

Since a bank of battery cells are placed in parallel, [NAL85] requires that each cell have a Schottky diode place between the positive terminal of the cell and the connection point shared between the cells. The diodes are placed there to prevent circulating currents that could possibly waste power and overheat the batteries.

### *3.2.2 Computer Power*

*3.2.2.1 Requirements* The processor board as well as all the other boards require ±12V and ±5V. The following list gives the power requirements for each of the boards:

1) MZ104+ Processor Board: 940mA from all four voltages
2) Pearl-MM Relay Board: 420mA from +5V
3) Quartz-MM Timer Board: 360mA from +5V
4) MSI-P440 Thermocouple A/D Board: 50mA from +5V
5) Diamond-32-AT A/D Board: 200mA from +5V

The total current required for the non-processor boards is 1.03A from the +5V power supply.

*3.2.2.2 Options* The first option is to develop a battery bank in combination with voltage regulators that produces the required voltages and current needs. The second option is to use a power supply board made for the PC/104 computer. In particular, two styles can be used. The first power supply board is the Jupiter-MM. It provides up to 10A at +5V, 2A at +12V, 1A at –12V, and 0.2A at –5V. The Jupiter-MM also has an auto-shutdown switch that the computer controls. The switch shuts off all power when the two connections are shorted to ground. The second power supply board is the HESC104. The HESC104, which provides the same power as the Jupiter-MM and also provides for battery backup.

*3.2.2.3 Decision* The decision was made to use the Jupiter-MM power supply. The reason for this selection is due to two factors. The first factor is the desire to reduce weight. The extra battery pack on the HESC104 would add extra weight. The second factor in choosing the Jupiter is lack of a need for backup power. If the experiment loses power for any reason, the computer has no need to continue running. The rest of the experiment will halt without power, and the computer would have no data to collect. The battery power would just be there to keep the computer alive until power is restored.

### *3.3 Data Acquisition*

The RIGEX experiment has three primary sets of sensors onboard: temperature, pressure, and vibration. Each set of sensors produces analog signals. This requires an

analog-to-digital (A/D) conversion to be performed by the computer. This means that each sensor has an A/D conversion requirement.

### *3.3.1 Temperature*

*3.3.1.1 Requirements* The experiment has seven places that require temperature monitoring. Each tube requires two temperature sensors, and the structure requires one. On the tubes, one sensor is placed at the base of the tube, and the other on the first fold of the tube before inflation. The last temperature sensor is placed on the structure. The exact location is not crucial because there is nothing aboard the experiment that is capable of heating the structure to a significant temperature.

The second requirement for temperature measurement is the A/D conversion. The data must be collected at a minimum of 1 sample/sec. This slow sampling rate is due to the very slow rate changes in temperature. The exact choice of hardware to accomplish this sampling is dependent on the choice of sensor.

*3.3.1.2 Options* There are a variety of options available for measuring temperature, but only two could be used in the RIGEX structure. The first choice of sensors is a thermistor. A thermistor is a temperature sensitive resistor that changes its resistance as it is heated or cooled. These variable resistors are often used in Wheatstone bridges. The Wheatstone bridge in this case consists of three additional resistors matched (equal resistance R ) to the thermistor's resistance (T) at a particular temperature connected in a standard bridge connection. Figure 3.3 shows the bridge connection. A

supply voltage is given to produce a voltage drop across the resistors. When the thermistor's resistance exactly matches the resistance of the other resistors (R), $V_{bridge}$ will be zero [Ir96]. If there is any change in T, a voltage will appear across $V_{bridge}$. The accuracy of the bridge is heavily dependent on $V_{supply}$. Any variation in the supply voltage could develop erroneous measurements across the bridge. Using this style of measurement would require a differential A/D conversion.

The second option to measure temperature is to use thermocouples. Thermocouple devices are composed of two metals connected together. When two metals are connected together in a loop (two junctions) and one junction is heated, the hot junction produces a voltage. This voltage is proportional to the temperature difference in the two metals at the hot junction. This effect is known as the Seebeck Affect [NPH00]. Thermocouples do not require any supply voltage to power them, but they do require special meters due to the nature of the devices. Because the thermocouples use special metals to produce the signals, they require signal conditioners to be placed at the leads of

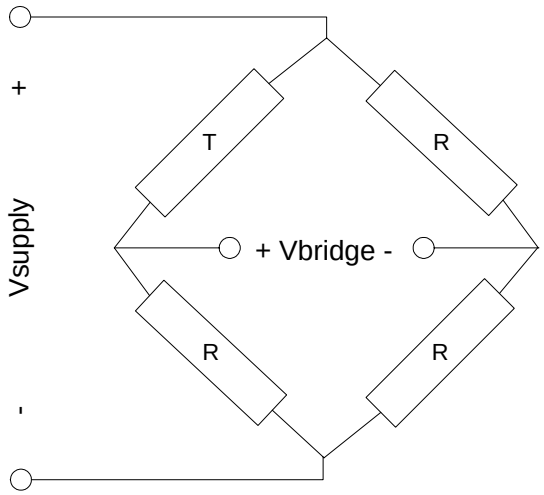


Figure 3.3: Thermister Wheatstone Bridge

the thermocouples. The signal conditioners must be present to allow for the implementation of the hot/cold junction. Placing the thermocouples into a standard A/D board would destroy the required hot/cold junction by introducing differing cold junctions.

*3.3.1.3 Decision* The decision was to use K-type thermocouples. These type thermocouples provide for very accurate readings at high temperatures and require no special source of voltage regulation. To solve the problem of special need of a thermocouple A/D board, the MSI-P440-K, an 8-channel thermocouple 12-bit A/D board for K-type thermocouples, is used. The MSI-P440-K uses AD596/597 devices that provide built-in ice point compensators [MiS03]. These signal conditioners allow the thermocouples to work and output the temperature proportional voltages.

*3.3.2 Pressure*

*3.3.2.1 Requirements* There are two requirements for the pressure system. The first requirement is for an electrically controlled valve. This valve is used to control the flow of nitrogen into the tube for inflation and to allow for nitrogen venting into the environment after inflation.

The second requirement is for a pressure transducer to convert the pressure into an electrical signal. The electrical signal is measured and recorded by the computer system's A/D board. DiSebastion [DiS01] made a pressure sensor sensitivity requirement of 0.001 atmospheres (atm). A pressure transducer with a rating of 15psi is

required to measure the pressure in the tubes. Also, pressure transducers with a rating of at least 500 psi are needed to measure the pressure in the inflation gas storage containers. This is needed to determine if any of the gas had leaked prior to inflation.

*3.3.2.2 Decision* The decision was made to use the same sensors as used in [Phi03]. The electrically controlled valve is a Series 9 valve made by Parker Hannifin Corporation [PaH03]. This valve is a three-way valve. At opening, the valve allows the gas to be entered into the tube. When the solenoid is closed, the valve allows the gas in the tube to be vented into the environment.

The ENDEVCO 8510 series of pressure transducers were chosen to be used in this experiment. The 8510 series of transducers measure pressure using a gage measurement. All pressure measurements are made with respect to the ambient pressure inside the GAS container. The transducers operate using a bridge system similar to that in Figure 3.3 [End03]. The supply voltage needed to excite the transducers is 12VDC and is provided by the computer power supply.

### *3.3.3 Vibration*

*3.3.3.1 Requirements* Vibrations experienced by the tubes during inflation and excitation must be measured. The structure itself experiences vibration due to the movements of the shuttle while in orbit. The vibrations are detected by measuring the acceleration in a particular direction. The accelerations must be measured in all three dimensions. This requirement dictates the use of a triaxial accelerometer. The

accelerometer must be small and lightweight, since it will be placed on the top of the tube.

*3.3.3.2 Options* Some of the smallest accelerometers are piezo style. These accelerometers output a charge signal versus a voltage signal. The charge signal then requires a charge amplifier to produce usable voltage signals for measurement. The primary problem with this type of accelerometer is to find a charge amplifier to condition the signal before measurement. The second choice of accelerometer is one that uses a supply voltage and produces an output voltage corresponding to a level of acceleration.

*3.3.3.3 Decision* The decision was made to find an accelerometer capable of producing its own voltage signal without the use of a signal conditioner. Summit Instruments has a triaxial accelerometer capable of measuring ±10g of acceleration. The 34200A has a built-in signal conditioner that outputs a voltage signal corresponding directly to the acceleration along the measured axis [SuI02]. Each output compensates for any temperature variations in the device. A 1KHz low pass filter is used on all outputs. The 34200A takes in an unregulated supply voltage ranging from 8VDC to 30VDC. Each axis has a sensitivity of 56 mV/g, where g is the acceleration of gravity, 9.8 m/sec$^2$. The 34200A has a small size (< 1in$^3$) with all outputs and connections made using a single DB-9 connector.

Each of the accelerometer's axis outputs is connected to the Diamond-MM A/D board. This means that each accelerometer uses three channels on the A/D board. The

34200A outputs a signal with a range of 0 VDC to 5 VDC. At this range, the 16-bit resolution of the A/D board gives a single bit level measurement accuracy of 76 μV. Using a sensitivity of 56mV/g, a 1 g acceleration has a resolution of approximately 10 bits. This will allow for very fine measurements of accelerations less than 1 g.

### *3.4 Height/Tilt Displacement*

*3.4.1 Requirement* After inflation, each tube's final height must be measured. Along with the height, any tilt angle due to imperfect inflation must also be measured. These measurements are important to the user for future use in designing inflatable trusses. Final inflation height is more important than the tilt angle. If the tubes in a truss were not to inflate fully, the final truss could have a severe bend to it. Tilt angle is not as critical because a truss of inflatable tubes would pull the bends out on their own as they inflate.

*3.4.2 Options* There are a couple of options on how to measure the data on height and tilt angle. The first option is to use a camera system to take pictures of the tube after inflation. The image could then be used to determine height and tilt angle. The second option is the possible use of a laser to measure distance and angle. This choice of measure would require placing optical sensors along the inside wall of the structure. The sensors would be needed to detect the reflected laser caused by any tilt in the tube.

*3.4.3 Decision* The decision was made to use an imaging system. The choice of using a camera versus a laser is the simplicity of use. The laser system would require monitoring multiple optical sensors, whereas the camera takes just one picture. This decision follows along the goal of minimizing complexity. The user and the sponsor of this experiment also would like to have a visual of how the tube inflation occurs. By having a camera onboard, pictures can be taken during the inflation process since no one observes the experiment as it is conducted.

*3.4.3.1 Distance Determination* To determine height from the camera based on the images alone, a white circular target with single peg at the center is placed on the top of the tube. To determine height, a relationship between the diameter of the tube's white circular target in pixels to its true diameter must be determined. This relationship is determined by setting the tube with its target between 1.66 in and 9.91in in 0.25 in intervals from the camera lens. At each of these distances, 10 pictures are taken. Each image is then preprocessed according to Ponziani [Pon03]. After each image has been preprocessed, the Matlab® program thresholds the image and fills in any holes, leaving only a circle or an ellipse. Once the circle/ellipse is determined, Matlab® uses *regionprops* script to determine the major axis length in pixels. The script uses the second moments of the image's pixels' distribution to determine the major and minor axes of the ellipse [Ma03]. The results of each of the 10 images are averaged and a ratio formed based on known diameter of the target and the averaged pixel diameter from the images. Each known distance from the camera lens has an averaged diameter/pixel data

point. The 36 data points have a least squares line fit in hopes that a linear relationship exists. Once the relationship of the diameters to height has been established, future images can have height measured by evaluating the ratio of the true diameter to pixel diameter using the least squares result.

*3.4.3.2 Tilt Angle Determination* For angle determination, the target peg is the primary tool used. After the image is thresholded, the algorithm subtracts the thresholded image with the peg from the filled in image (ellipse only). This results in leaving only the peg in the image. To get only the outline of the peg, the image is edge-detected. This edge-detection is accomplished using a Laplacian-based two-dimensional filter [Lim90]. Equation 3.1 gives an example of a Laplacian edge-detection filter [Lim90]. To use the edge detection filter, the image is two-dimensionally convolved with *h*. The result will be another image with lines showing all the 2nd derivative zero-crossings.

$$h = \begin{bmatrix} -1 & 2 & -1 \\ 2 & -4 & 2 \\ -1 & 2 & -1 \end{bmatrix} \qquad 3.1$$

Once the outline of the peg has been found, it is necessary to center the peg on an origin. This is accomplished by subtracting out the mean value of the edge pixels of the peg. To determine tilt angle, it is necessary to measure the length in pixels of the peg in the picture. This measure in pixels is then converted to physical distance, using the results from Section 3.4.3.1. Based on the tilt angle of the tube, the measured length of the peg gets larger as the angle increases up to 90º. At 90º, the measured length of the

peg is the length of the peg at that height. This gives a sine relationship between the tilt angle and the ratio of the peg lengths. From Figure 3.4, the tilt angle is determined by taking the inverse sine of measured peg length over the true peg length.

$$\alpha = \sin^{-1}\left(\frac{measured}{true}\right) \qquad 3.2$$

The shape of the peg in the image after edge-detection has a distorted, non-rectangular shape. This irregular shape makes measuring the length of the peg much more difficult. To determine the length of the peg in the image, it is necessary to find the major and minor axes of the peg. To do this, the covariance matrix (**K**) of the zero-meaned edge pixels is found using the following formula [ShB88]:

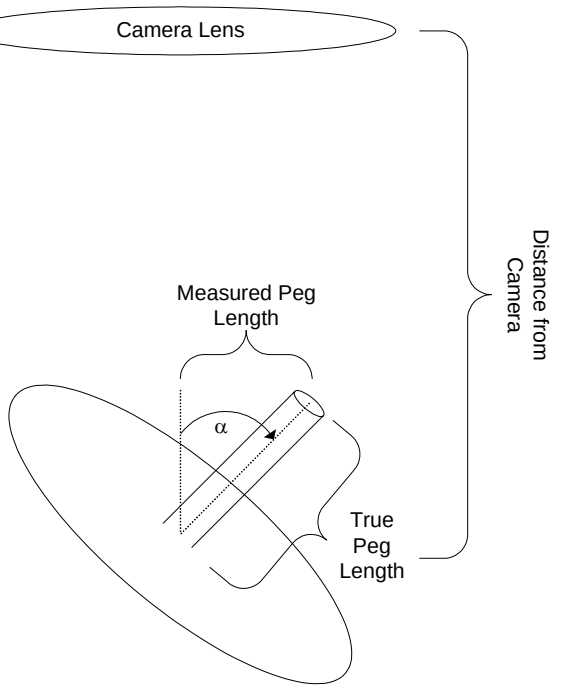



Figure 3.4: Peg Length determination of tilt angle

$$K = \frac{1}{N} \cdot A^T \cdot A \qquad 3.3$$

where **A** is a 2xN matrix containing the coordinates of each of the N edge pixels. The major axis of the peg is the eigenvector of **K** corresponding to smallest eigenvalue ([Str88]). The minor axis is the eigenvector of **K** corresponding to the largest eigenvalue. The edge-pixels of the peg can now be aligned onto their major and minor axes.

It is expected that the base of the peg will have a convex curve and the tip of the peg a concave edge. The round shape of the peg, and the fact that the camera is viewing the peg from above, causes this distortion. To get an accurate measure of the pixel length of the peg, the Euclidean distance between the center points of the tip and base must be found. The center of the base is found using the point with the largest vector projection onto the major axis of the peg. To find the center point of the tip, all the edge points are projected onto the major axis. A 1000 bin histogram of the projections is used to find the point on the major axis with the most edge-points in a single bin. This is assuming the concaved edge of the tip is smooth with many pixels having relatively the same projection value. To get the pixel closest to the major axis, the pixels in the bin are projected onto the minor axis. The pixel with the smallest projection is declared the center of the tip. The measured pixel length is found using the height relationship from Section 3.4.3.1, and the tilt angle is found using Equation 3.2.

It must also not be forgotten that this problem is a post-processing issue. After the experiment is conducted and all material recovered from the shuttle, the tubes can be removed and measured by hand with calibrated instruments as well as by the images.

### *3.5 Tube Excitation*

*3.5.1 Requirements* Part of the RIGEX experiment is to measure the modal characteristics of the tubes after they have been inflated. To determine the modes of the tubes, one must determine the transfer function of the tubes must be determined. To find the transfer function, the tubes must be excited by some vibrating source using some waveform that represents all frequencies between 0Hz and 1000Hz. The resulting data is then used to determine the transfer function of the tubes using Matlab®'s transfer function estimation program called TFE. The TFE program uses power spectral densities in its theory to estimate the transfer function (H(f)). This ratio consists of the cross power spectral density of the input and output and the power spectral density of the input signal. To perform this estimation, the input and output (x(t) and y(t) respectively) signals are assumed to be random. The power spectral density of the input signal is found by taking the Fourier transform of the autocorrelation of the input signal [ShB88].

$$R_X(\tau) = \int_{-\infty}^{\infty} x(t) \cdot x(t-\tau) dt \qquad 3.4$$

$$S_X(f) = \int_{-\infty}^{\infty} R_X(\tau) \cdot e^{j2\pi f} \cdot d\tau \qquad 3.5$$

The cross power spectral density is found by taking the Fourier transform of the cross-correlation of the input signal and the output signal [ShB88].

$$R_{XY}(\tau) = \int_{-\infty}^{\infty} x(t) \cdot y(t-\tau) dt \qquad 3.6$$

$$S_{XY}(f) = \int_{-\infty}^{\infty} R_{XY}(\tau) \cdot e^{j2\pi f} \cdot d\tau \qquad 3.7$$

It is shown in [ShB88], that the cross power spectral density is equal to the transfer function of the system multiplied by the power spectral density of the input signal.

$$S_{XY}(f) = H(f) \cdot S_X(f) \qquad 3.8$$

The transfer function can be estimated by forming the following ratio:

$$\hat{H}(f) = \frac{S_{XY}(f)}{S_X(f)} \qquad 3.9$$

The TFE program performs some additional signal conditioning before finding the power spectral densities.

Experimentally, it has been shown that the first and second modes of a fully inflated tube lie at 62Hz and 660Hz [SiT02]. Figure 3.5 shows the transfer function of a fully inflated tube that is excited using an up-chirp signal. The accelerometer used in this case had a sensitivity of 10mV/g.

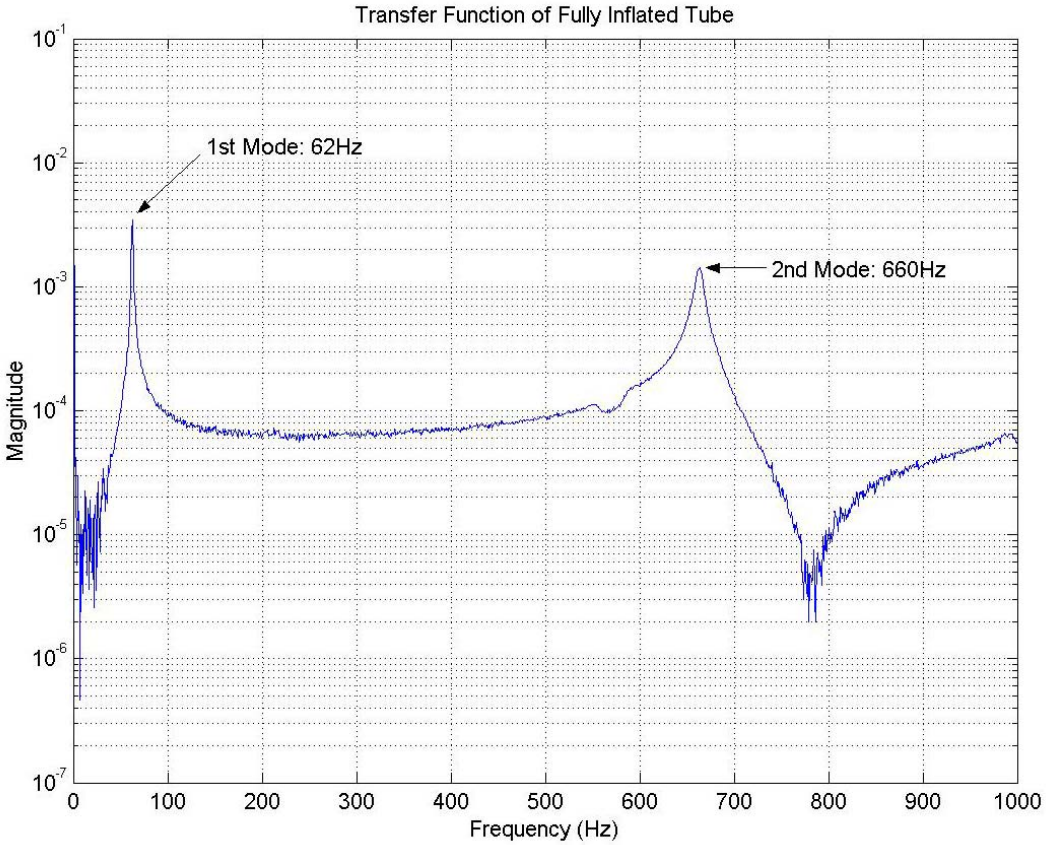



Figure 3.5: Transfer function of a fully inflated tube

*3.5.2 Options* There is no other option available for generating the excitation waveform than to use the 12-bit digital-to-analog converter (DAC) onboard the Diamond-MM A/D board. The use of this DAC requires a sampling rate of 5kHz or 5MHz [DMM00]. A 5MHz sampling rate would require 5 million data points for a 1-second duration waveform; therefore, the 5kHz rate is used. There are a couple of options available for the type of waveform. The first option is to use a white noise signal. An ideal realization of this signal offers a flat constant level power spectral density. This flat spectral density presents all frequencies to the tube at an even power level. An ideal realization of white noise is hard to achieve because noise data being converted to an

analog signal has a 5kHz sampling signal inserted into it. The other waveform option is to use an up-chirp. This up-chirp would have a starting and ending frequencies. The method of moving from start to finish can be linear or non-linear. The use of an up-chirp presents a relatively flat power spectral density between the start frequency and the stop frequency. The range of frequencies covered by the up-chirp must contain the mode frequencies of the tubes.

The next decision to be made is how the physical excitation is accomplished. The first option is to use some form of vibrating motor. This motor would have to be mounted to the base of the tube within the oven. The other option is to use a piezo-patch mounted at the base of the tube. The piezo-patch contracts or expands dependent on the polarity of the signal voltage. If a sinusoidal signal is applied to the patch, it contracts and expands with the frequency of the signal. The changes create vibrations in the tube allowing for vibration measurements to be obtained. The piezo-patches have a pure capacitive impedance and will not draw any significant current.

*3.5.3 Decision* It was decided by the user of the data to use an up-chirp signal with a start frequency of 5Hz, a stop frequency of 1000Hz and an amplitude of 10. The waveform data is generated using Matlab®. The signal has a duration of 1 second and is sampled at 5000 samples per second. This gives 5000 data points for the signal. The sampling rate is 5 times larger than the highest frequency in the signal, so aliasing will not present any problems. Figure 3.6 shows an example of a 5V up-chirp signal. The Diamond-MM A/D board has four D/A output channels. Each tube is assigned a channel,

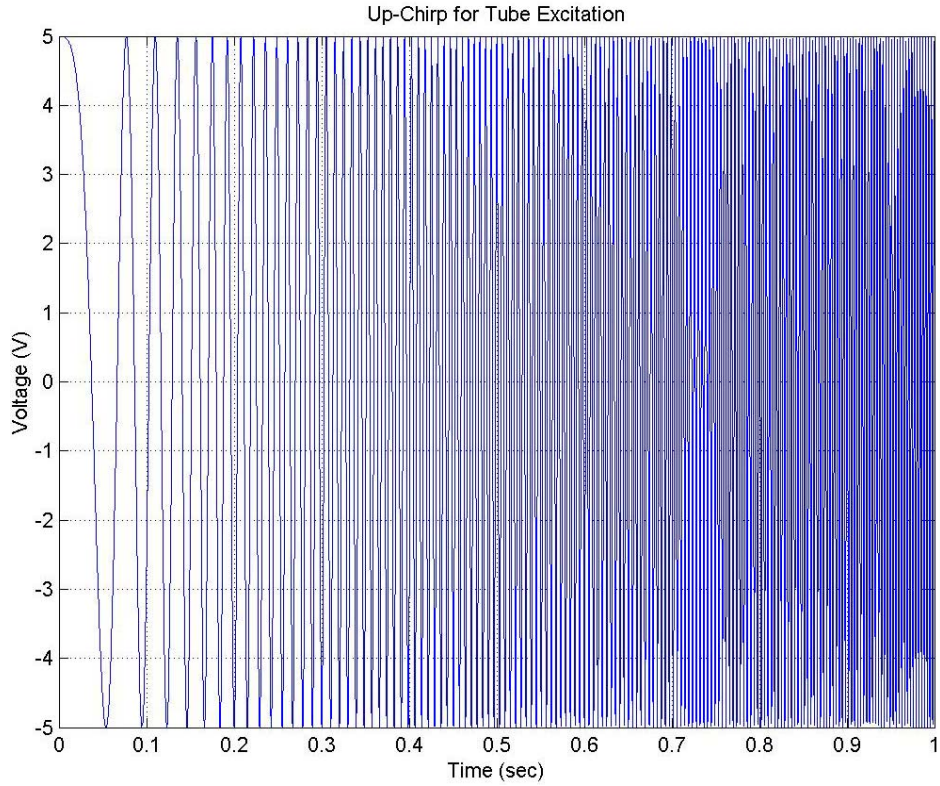



Figure 3.6: Excitation waveform up-chirp, $y(t) = 5\cos(2\pi(5 + 995t)t)$

channel 1 for tube1, channel 2 for tube 2, and channel 3 for tube 3. Channel 0 of the D/A can be used for testing. The piezo-patches were also chosen because a pair of the patches can be placed at the base of the tube opposite of each other. If the wire leads of the patches are connected with polarities reversed, the patches would contract/expand opposite of each other. This would create twice the "muscle" needed to induce the vibrations in the tubes.

### *3.6 Environmental Control*

Two environmental factors need to be addressed. The first is lighting. There must be light present for camera imaging. The second is the ambient temperature of the electronics in space. The majority of the electronics have a lower temperature operating range of –40ºC to –20ºC. The coldest temperature experienced in the shuttle's cargo bay is –121ºC.

#### *3.6.1 Lighting*

*3.6.1.1 Requirements* Each section of the structure must have a source of white light. This white light must be bright enough to provide for adequate imaging of the tubes. The lights must be mounted on the top plate of the structure next to the camera.

*3.6.1.2 Options* There are two types of white lights available for use. The first is a small incandescent light bulb. These bulbs need between 3VDC – 30VDC to operate. Like the bulb in a flashlight, these bulbs tend to use a large amount of power.

The next choice is to use bright white light emitting diodes (LEDs). An LED is a semiconductor diode that produces light of some wavelength with a forward bias voltage. Typical bright white LED bias voltage is around 3.6V with a forward current of 20mA.

*3.6.1.3 Decision* The decision was made to use bright white LEDs to light the interior of the structure. Figure 3.7 shows the circuit schematic on how the LEDs are connected. The voltage drop across the LEDs is 3.6 V, and the total current through the

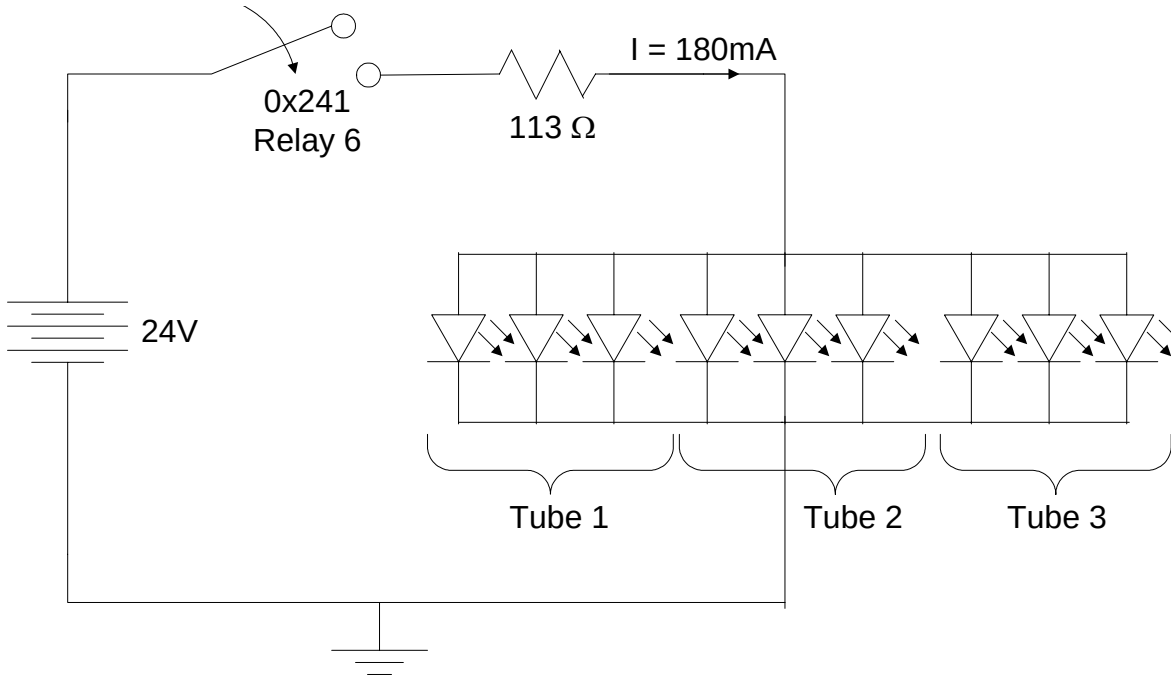



Figure 3.7: Lighting Circuit Diagram

circuit is 180mA. The resistor is used to regulate the current through each of the LEDs to 20mA each. Three LEDs are placed in each tube section of the structure. The switch to turn the lights on is controlled by one of the relays on the Pearl-MM relay board. The relay's control is performed by register 0x241 bit 6. When this bit is set high, the relay closes allowing the lights to turn on.

### *3.6.2 Heater Control*

*3.6.2.1 Requirements* The RIGEX experiment operates in a space environment where the ambient temperature can be well below -40ºC. The electronics in the experiment cannot operate much below –40ºC. Effort has been made to use extended

temperature range electronics for all the other systems, but there are none for the PC/104 that can get below –40ºC. This requires some form of heat to be supplied to the electronics, keeping them in their operational range. The choice of heaters for warming the electronics is the MINCO Thermofoil® heaters [MI99]. The next requirement is to find a way to control the temperature of the heaters. The heaters cannot be left on; if so, the electronics will overheat.

*3.6.2.2 Options* There are two options available. The first option is to use a temperature sensor and have the computer monitor the temperature of the heaters and control the amount of time that the heaters are turned on. This option presents more complicated programming and would require the computer to be activated before the shuttle reachs orbit.

The second option is to use MINCO's Heaterstat™ CT198 devices. These devices are matched to the Thermofoil® heaters to provide for thermostat control. The devices work by using a matched resistance in the controller that produces a signal proportional to the temperature of the matched heater. The electronics then threshold the temperature signal and cut on/off the heater as the signal varies below/above the threshold [MI01]. The threshold can be adjusted by using a potentiometer on the controller. The controllers are ordered preset to a specific temperature, and the potentiometer can adjust the set temperature by ±20% [MI01]. Figure 3.8 shows the operational configuration of the heater controller circuit.

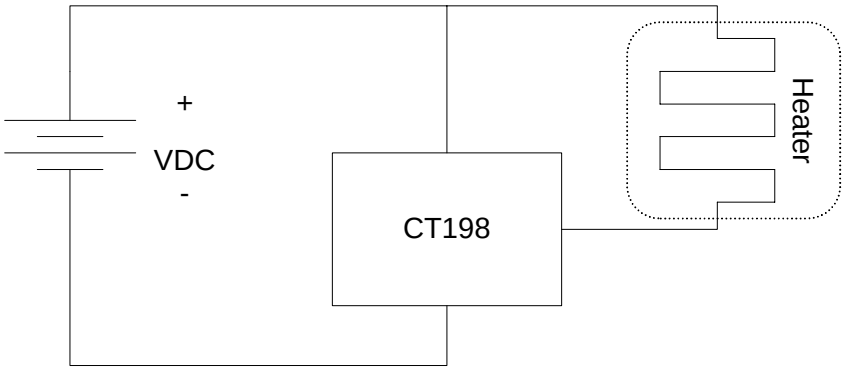



Figure 3.8: Heater Controller Circuit [MI01]

*3.6.2.3 Decision* The decision was made to use the HeaterstatTM [MI01] controllers. The reason behind the choice is the simplicity of operation. Once the controller has been calibrated to the desired temperature, no sensors are needed to monitor the temperature. To calibrate the controller, an ammeter is placed just after the power supply. The circuit then has power applied with the temperature of the heater being monitored by another temperature-sensing device. Once the desired temperature is achieved, the potentiometer is adjusted until the current begins to cut on and off regularly. This on/off behavior says the controller is right at the desired temperature.

### *3.7 Computer Configuration*

For this experiment, there are two computers running the experiment. The computer is separated into two systems based on function. The first (primary) computer conducts the flow of the experiment and collects all the analog data signals. The second computer performs all the imaging of the tubes. There are two reasons for this

separation. The first reason is the limitation of the PC/104 bus. The PC/104 can only have 10 external cards on the bus. Ten cards is an ideal case. Imperfections in the design of cards can cause the impedance of the card's bus to look like more than one. For data acquisition and flow control, the data acquisition computer will have four application cards as well as two sets of spacers. At worst, this could look like 10 cards. To perform the imaging of the three tubes, three cameras and three interface cards are needed. Adding three imaging cards to the computer stack could cause serious bus errors.

The second reason for the split is due to memory and processor requirements. Philley [Phi03] discussed problems with taking the pictures of the tubes as they were inflating. These problems were due to memory limitations and operating system. As multiple images are taken, the memory begins to fill quickly. Separating the computers allows for all the system memory to be dedicated to taking images. This allows the data collection and image collection to run independently with respect to system memory and code.

The only drawback about using a tandem computer system is the need to power the second computer. The power is available since a separate battery cell is needed for the data acquisition computer. For a two 24 VDC 17 A-hr battery cells in parallel, the difference between one computer and two is negligible.

### *3.7.1 Data Acquisition Computer*

As mentioned above, the data acquisition computer is the primary computer for the experiment. Its has the following responsibilities:

1) Control the flow of the experiment routine

2) Perform all analog data collection

3) Notify the imaging computer when to take pictures.

Each of the above functions is controlled by using a particular application board. Each application board has a base hardware address, which is assigned by setting physical jumpers on each of the boards. Each board's respective manual gives instructions on how to set the base address [DMM00, MiS03, PMM01, QMM01]. This base address points to a group of onboard registers. Appendix B contains a diagram showing how the data acquisition computer is configured as well as all the channel assignments.

*3.7.1.1 Pearl-MM Relay Board* The Pearl-MM board contains 16 relays. A single bit transmitted from the processor board controls each relay. The relays are grouped into two 8-bit registers where each register controls eight relays [PMM01]. To control the relays, the computer writes an 8-bit value to the register. If a 1 is written to a relay, the relay closes. If a 0 is written, the relay opens. The relay board for RIGEX has a base address of $240_{16}$. Relays 0-7 are controlled by register $240_{16}$. Relays 8-15 are controlled by register $241_{16}$.

The function of the relay board is to control the experiment's mechanical systems. The relay assignments for this board are shown in Appendix B. A group of relays are assigned to control the operation of each of the tubes. The first relay in the group turns on the heaters in the tube oven. This is accomplished by switching a voltage to activate a high-current solid-state relay. The second relay is used to apply voltage to the pin-puller.

This opens the heater box. The third relay is used to open the solenoid, allowing the inflation gas to enter the tube. Upon closing this relay after inflation, the gas is vented into the environment. Relays 6 and 7 (14 and 15) on register $241_{16}$ are used for special purposes. Relay 6 is used to turn on all the LEDs in the structure as shown in Figure 3.7. Relay 7 is used to shut the computer off by shorting the two pins on J3 of the Jupiter-MM power supply.

*3.7.1.2 Diamond-MM 32ch A/D Board* The Diamond-MM board performs the majority of the data collection. It provides for 16-bit analog-to-digital conversion. For a 0-5V range signal, the board provides a 1-bit resolution of 76μV. The board has 32 channels of A/D conversion. The board also has four 12-bit D/A channels. Timing and control of the conversions is performed by control registers and a built-in 10MHz oscillator. From this oscillator, sampling rates of 10MHz, 100kHz, and 5kHz or less can be achieved.

In order to read true voltage levels from the board or to write to it, the data must be converted from a digital code word to the appropriate voltage measurement (A/D operation) and vice versa (D/A operation). The digital code words are output as 16-bit signed integer values. The board uses the following conversions to convert A/D codes to real voltages:

$$Digital = MSB \cdot 256 + LSB \qquad 3.10$$

where MSB is most significant byte (bits 8-15) and LSB is least significant byte (bits 0-7). The digital code value is then scaled to equate to actual voltage levels. The conversion follows for a unipolar signal (i.e., 0-5V):

$$Voltage = \frac{Digital + 32768}{2^{16}} \cdot FullRange \tag{3.11}$$

Full range is the highest measurable voltage level minus the lowest measurable voltage level. The input voltage signals coming out of the 34200A accelerometer range from 0V to 5V. The conversion for the vibration signals uses a full range voltage of 5V. The D/A also has conversion requirements. The output voltage must be converted to a digital code that scales the data into 4096 ($2^{12}$) discrete levels. Once the data has been converted to digital code values, the codes are then split into a MSB and a LSB. The output voltage to digital code value for bipolar signals uses the following conversion:

$$Digital = \frac{OutputVoltage}{FullRange} \cdot 2048 + 2048 \tag{3.12}$$

$$LSB = Digital \cap 255 \tag{3.13}$$

$$MSB = floor\left(\frac{Digital}{256}\right) \tag{3.14}$$

The digital code is logically ANDed with 255 to produce the LSB. This isolates only the lower eight bits of the code value. Dividing the digital code by 256 and then truncating the remainder creates the MSB.

For RIGEX, the A/D board plays a vital role. On its 32 A/D channels, all vibration and pressure measurements are taken. The channels for the tubes are grouped into sets of four. Each set of four contains the channel for measuring pressure and three channels for measuring the three axes of vibration from the triaxial accelerometer. During inflation and excitation of the tubes, the channels associated with the tube in question are scanned and the data read. The minimum sampling rate for the vibration measurements has to be 2000 samples/sec to measure the 1000Hz frequency in the excitation process in order to satisfy Shannon's sampling thereom. Three additional channels are used to take measurements of the pressure in the inflation gas storage containers. These measurements are used to determine if any gas leaked between the final integration and the on-orbit experiment

Like the relay and counter/timer boards, the A/D board also requires a base address. An address of $380_{16}$ is assigned to the A/D board. The A/D board has a large number of control registers that are used. To access the board's registers, the computer reads/writes to an address equal to the base address plus the appropriate register number. Table 3.1 [DMM00] gives a list of the registers used on the A/D board. The 8254/8255 registers are registers selected and used to control the built-in counter/timer IC and the digital I/O controller IC. To access these registers, two page-bits are written to using register base + 8 lower two bits. To access the counter/timer (8254) IC registers, the last

two bits in base + 8 have a 00 written. To access the digital I/O (8255) IC registers, the last two bits in base + 8 have a 01 written. The other two pages of four registers are reserved and used for calibration. Appendix D provides descriptions on how these registers are used to perform A/D and D/A operations.

*3.7.1.3 MSI-P440-K Thermocouple A/D Board* The MSI-P440-K is a much simpler A/D board when compared to the Diamond-MM A/D board. The P440 has only eight channels for A/D. There are three registers used to control the operation of the eight A/D channels. The first register located at base address + 0 acts as the control register when it is written. For the RIGEX experiment, the control register always has the 01010XXX written. This sets channel XXX to use normal operation, internal

Table 3.1: Diamond-MM A/D board I/O register map

| **Base Address ($380_{16}$) +** | **Write Function** | **Read Function** |
|---|---|---|
| 0 | Start A/D conversion | A/D LSB (bits 7-0) |
| 1 | Auxiliary digital output | A/D MSB (bits 15-8) |
| 2 | A/D low channel register | A/D low channel register readback |
| 3 | A/D high channel register | A/D high channel register readback |
| 4 | D/A LSB register | Auxillary digital input port |
| 5 | D/A MSB + channel register | Update D/A channels |
| 6 | FIFO depth register | FIFO depth register |
| 7 | FIFO control register | FIFO status register |
| 8 | Miscellaneous control register | Status register |
| 9 | Operation control register | Operation status register |
| A | Counter/timer control register | Counter/timer control reg. Readback |
| B | Analog configuration register | Analog configuration. Reg. Readback |
| C | 8254/8255 register | 8254/8255 register |
| D | 8254/8255 register | 8254/8255 register |
| E | 8254/8255 register | 8254/8255 register |
| F | 8254/8255 register | 8254/8255 register |

clocking with internal acquisition, and an input signal of ±5V. Once this register is written, the A/D conversion begins. The program must then wait until an A/D converter has finished the conversion. This status check is done by checking bit 0 at the register located at base address + 8. When the status bit is a 0, the conversion is complete. Once the conversion is accomplished, the data can be read from two registers. The LSB is read from base address + 0, and the MSB is read from base address + 1. When the input signal is bipolar, the output digital value is a signed 12-bit integer. For example, if the input signal is ±5V, the digital data from the converter has a range of $000_{16}$-$7FF_{16}$ for 0-5V and $800_{16}$-$FFF_{16}$ for –5-(<0V).

The MSI-P440-K is used to monitor each of the thermocouples used in the RIGEX experiment. Each tube has two thermocouples located on it and the structure has a thermocouple located on the base plate of the structure.

The MSI-P440-K is assigned $300_{16}$ as its base address. This address does not conflict with any other board on the stack.

*3.7.1.4 Quartz-MM Timer/Counter Board* The Quartz-MM board is a multifunctional timer board that is capable of performing 16-bit up and down counting, timers, and 8-bit digital I/O. All the timing on the board is derived from a 4MHz oscillator. The timer chips on the board are able to divide the oscillator in increments of 10. Thus, timing signals at frequencies of 4MHz, 400kHz, 40kHz, 4kHz, and 400Hz are

Table 3.2: MSI-P440 Control Register (base address + 0)

| **Bit** | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| **Function** | PD1 | PD0 | ACQM | Range | Bipolar | A2 | A1 | A0 |
| PD1 and PD0 are used to select clock source and power down mode | | | | | | | | |
| ACQM is the acquisition mode, this bit sets internal acquisition timing (0) or external (1) | | | | | | | | |
| Range sets the range to be 5V (0) or 10V (1) | | | | | | | | |
| Bipolar sets the input voltages to be bipolar (1) or not (0) | | | | | | | | |
| A2, A1, and A0 are the binary addresses of the eight A/D channels | | | | | | | | |

available. The timer chips are also able to generate variable duty cycle square waves, one-shots, and timed pulses [QMM01].

The Quartz-MM board is used as a timing source to the imaging computer. The timing signal is a 400Hz clocking signal generated by the timers. This timing source is output on the $F_{out}$ pin of the board and connected to the interrupt input pin of the imaging computer's timer board (Appendix B). The Quartz-MM board for the data acquisition computer has a base address of $2C0_{16}$.

*3.7.2 Imaging Computer*

The imaging computer is responsible for taking all experimental pictures. The imaging computer is enabled when it receives an interrupt on the imaging computer's Quartz-MM interrupt pin. This interrupt is located at base address + 6 of the timer board. Upon seeing this interrupt, the imaging computer reads the two input bits (at base address + 2 of timer board), giving the tube address and then taking a picture of the tube with the appropriate camera. This interrupt process is repeated for each interrupt it receives. The imaging computer's Quartz-MM timer board has a base address of $240_{16}$.

*3.7.2.2 Electrim EDU-104 Camera Boards* The EDU-104 boards provides an interface between the PC/104 computer and the EDC-1000U cameras. The boards are capable of capturing 1134x486 pixel arrays. The boards are then able to interlace the data to produce an image of size 1134x972 pixels. The board exposure time can be set to between 1msec and 30msec [Elec97].

The PC/104 computer uses the manufacturer's sub-array capture routine to perform the required tasks of getting the images. The images are then saved as data files to be converted to images later using Matlab®.

The interface board for Camera 1 has a base address of $300_{16}$. Similarly, base addresses for Camera 2 and Camera 3 interface boards are $340_{16}$ and $380_{16}$ respectively.

## *3.8 Experiment Flow of Operations*

It is important to get a good feel for how the experiment will be conducted. To this point, the system configuration of the experiment has been discussed. This section covers the flow of the experiment and all the computer operations. The discussion starts with how system failsafe points are used, the initialization of system, and then the two major processes the tubes undergo.

### *3.8.1 Failsafe Points*

The RIGEX system will perform its function without supervision except for two relay switches controlled by the shuttle and its crew. One of the experimental requirements is for the design to allow the crew to shut down the experiment for any

needed purpose. Some reasons for this may be due to shuttle emergencies. In any event, safety of the shuttle and its crew is paramount. Because of this requirement, the computer needs to know the last executed instruction prior to system shutdown. The answer to this is failsafe points.

During initialization, the data acquisition computer opens the failsafe data file and reads the value stored there. If the value is 0, the computer starts from the very beginning. If the value is non-zero, the computer locates the address of the instruction last executed.

Another design question relates to the location of the failsafe points within the program execution cycle. There are certain instances in the experiment where the computer cannot rerun. One example might be after inflation of a tube. The computer cannot reheat the tube or reinflate it. Failsafe points in this case could be placed before the inflation point and after inflation. Table 3.3 shows the failsafe program markers.

### *3.8.2 Initialization*

The initialization of the experiment is shown in Figure 3.9 as the main event calendar. Initialization consists of all steps prior to entering into any of the tube inflation/excitation processes. Each step in the calendar is discussed below.

*3.8.2.1 Begin* This step is consists of the RIGEX structure and all its components being placed into the GAS can. The timing associated with the shuttle launch is also encompassed by this step.

Table 3.3: Failsafe data points

| Failsafe Marker | Failsafe Program Location |
|---|---|
| x0 | Tube x: Start of inflation routine- heaters have been activated |
| x1 | Tube x: Heating has been completed |
| x2 | Tube x: Tube has been inflated |
| x3 | Tube x: Excitation routine has been started |
| x4 | Tube x: Last set of images have been taken |
| x5 | Tube x: End of inflation and excitation |
| 40 | End of experiment |

*3.8.2.2 Baroswitch (Relay A)* As the shuttle reaches an altitude of 50,000ft, a barometrically controlled switch closes relay A onboard the shuttle. Every GAS can has a relay A connection. This relay is used to turn on environmental control systems for various experiments [DiS01]. For RIGEX, this relay connects power to the environmental heater controls.

*3.8.2.3 Environmental Heaters* Once relay A is closed by the baroswitch, the environmental heaters are turned on. These heaters are used to maintain the operating temperature for the computer and the cameras. The controllers are set to a temperature of 5ºC. This sets all the electronics to the center of their operating range.

*3.8.2.4 Relay B* Relay B is the switch that starts the experiment. Upon orbit, the shuttle crew will activate relay B. Once relay B is closed, power is applied to the PC/104 computer system. This relay will not activate power to the heater boxes.

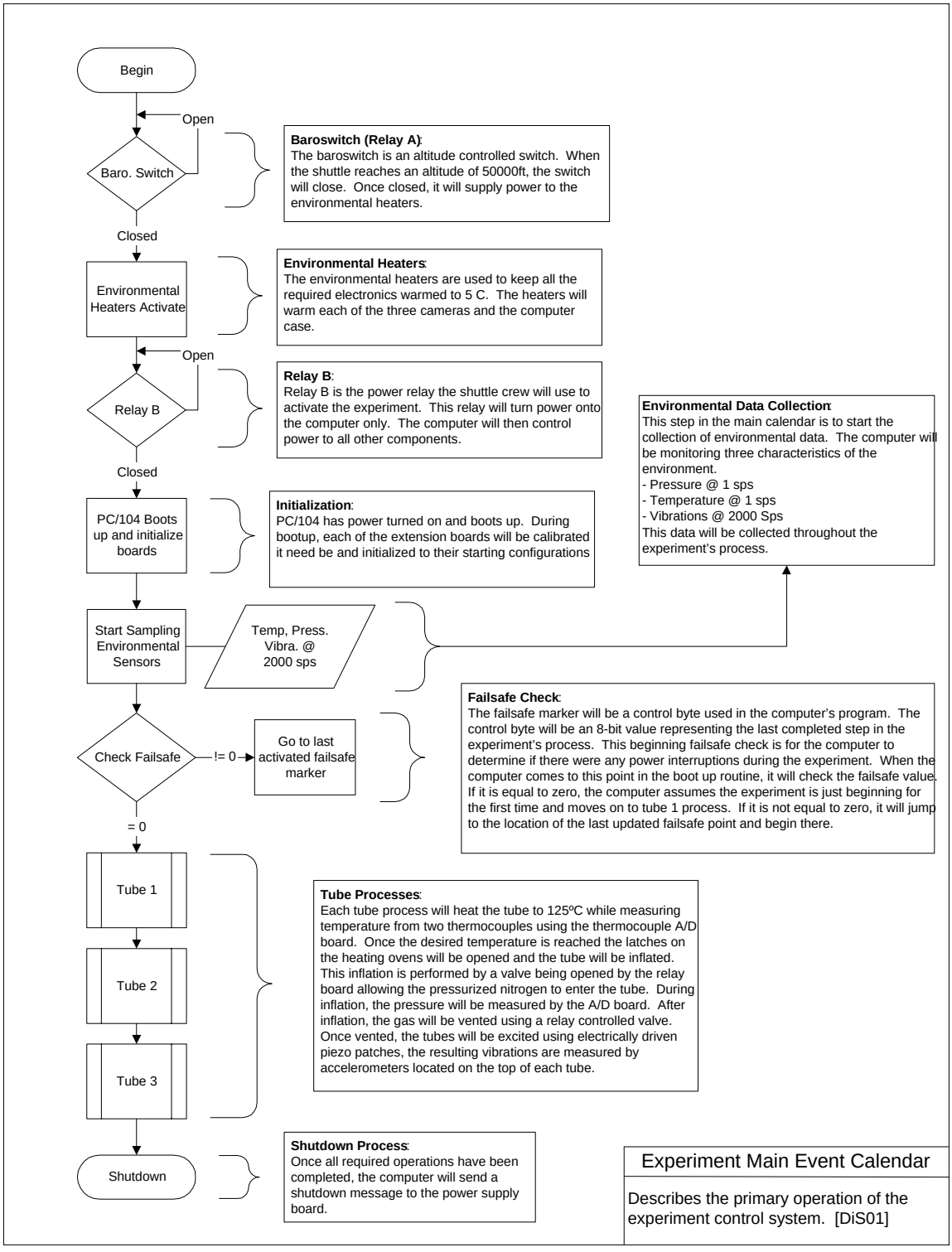



Figure 3.9: RIGEX main event calendar

*3.8.2.5 Initialization of the Computer* Once power is applied to the computer, the computer initializes all the application boards. This initialization includes setting the appropriate control registers onboard the A/D boards and timer board. This also includes initializing all needed variables to perform the needed functions.

*3.8.2.6 Environmental Data Collection* This stage in the experiment is used to monitor the operational environment. The required environmental measurements include pressure, temperature, and vibration. The environment's vibration measurements are critical to the excitation measurements. By sampling the shuttle vibrations, a true measurement of the tube vibration excitation is achieved. This is obtained by subtracting the shuttle vibrations from the vibration measurements taken during tube excitation. The temperature and pressure data are also gathered to help the user capture and understand operational fluctuations in the environment.

The temperature and pressure signals must have a minimum sampling rate of one sample/sec. The sampling rate is set low due to slow changes in the environment over the conduct of the experiment.

*3.8.2.7 Failsafe Point Check* This point in the initialization routine is used to determine if the computer was interrupted during a previous run of the experiment. This process is described in Section 3.8.1.

*3.8.3 Tube Processes*

Each tube undergoes two processes. The first process is the heating, inflation, and venting of the tube. The second process is the excitation process. As described in Section 3.5, the vibration response of the tube to the excitation waveform is measured. This section describes the details involved in each step of the processes. Appendix C shows the flow diagrams for each of the two processes.

*3.8.3.1 Inflation Process* The inflation process covers all the steps taken from heating the tubes to venting the gas from them. Also included in this process are the failsafe marking, data collection, and image collection.

*3.8.3.1.1 Mark Failsafe Point* The first step in the inflation process is to mark the failsafe file with the current routine location. This failsafe point is marked with x0, where x is equal to the tube number (tube 1, x = 1, failsafe point 10). This point is located here to mark that the computer last left off at the beginning of tube number x process.

*3.8.3.1.2 Activate Thermocouple A/D Channels* This step activates the data collection from each of the two thermocouples located on the tubes. Each thermocouple is sampled at one sample/sec. This gathers 60 samples/min. The room temperature experiments conducted by [Phi03] took approximately 40 minutes to get the tubes heated

to the inflation temperature. Forty minutes of time gives 2400 samples per thermocouple per tube. The rate of temperature change is not expected to exceed the sampling rate.

*3.8.3.1.3 Activate Heaters and Lights* At this step, the heaters are turned on by switching the appropriate relay for tube x (see Appendix B). The relay board activates a high-current solid-state relay to turn on current flow to the tube heaters. The relay board also turns on the lights in the structure. The lights, described in Section 3.6.1, are needed to illuminate the interior of the experiment so the images collected will have some valuable content.

*3.8.3.1.4 Test Temperature vs. Threshold* As temperature measurements are being taken by the thermocouple A/D board, each measurement is compared to a threshold temperature of 125ºC. If the threshold is not reached, the computer continues to wait and record temperature measurements. If the threshold is reached, the computer moves on to the inflation steps.

*3.8.3.1.5 Start Inflation Timer* An inflation timer is started prior to the inflation of the tube. This timer is required to allow the computer to know when the tube is finished inflating. Other methods are available for the computer to determine when the tube has stopped inflating. One method is to monitor the vibration data being recorded during inflation. When the accelerometer output begins to settle and nothing but noise is present, the tube has stopped moving and thus stopped inflating. This method may slow

the sampling down due to the processor operations needed to perform the calculations. This method will only work provided the shuttle does not make any changes in pitch, roll, or yaw. The timer is the simplest method and can just be set to just a few seconds.

*3.8.3.1.6 Activate A/D Board Channels* This step initializes all four A/D converter channels monitoring tube x's pressure and accelerometer's three axis outputs. The data is sampled at a minimum of 2000 samples/sec. This data is monitored throughout the inflation of the tube. Also during this step, the A/D board takes one pressure measurement from the gas reservoir to check the pressure before inflation. This measurement is taken in the event that the tube does not fully inflate; it helps to rule out any possible problems in the experiment when it is recovered from the shuttle.

*3.8.3.1.7 Start Signaling the Imaging Computer* This step notifies the imaging computer as to when images should be taken. The computer sends a timing signal using the counter/timer board to the imaging computer. The timing signal triggers an interrupt on the imaging computer. When the interrupt is triggered, the imaging computer reads the tube number using two bits on the digital inputs, takes a picture of the tube, and stores it. This process continues during each of the inflation routines. At this point, it is important to mark the failsafe file, x1. Once the heater box opens, the computer cannot reheat or inflate the tubes. If the computer sees this failsafe point, it knows that it must move on to the next failsafe point assuming the tube inflated properly.

*3.8.3.1.8 Open Heater Box and Inflation Valve* Since the tube has been heated to the required temperature, it is ready to be inflated. Before the inflation gas can be let into the tube, however, the heater box must be opened. If the box is not opened before the inflation valve is opened, the tube could jam the heater box shut by applying too much pressure on the box lever and the pin that opens the box. Applying a voltage across the pin-puller opens the heater box. The control voltage for the pin-puller is applied by the relay board. Appendix B gives the relay/tube assignments. The pin-puller for this experiment is a space-rated non-mechanical pin-pulling device. It uses shape memory alloys such that when an electrical charge is applied, the material contracts. For further information on these devices, see [TiA03]. After the heater box is opened, the relay board opens the inflation valve. The inflation valve is a voltage-controlled solenoid. The valve releases the $N_2$ gas into the tube, inflating it. The solenoid remains open during the inflation process.

*3.8.3.1.9 Check Inflation Timer* As data is being collected, the inflation timer is continually checked. If the timer has not expired, the A/D board continues recording data and the counter/timer board continues signaling the imaging computer. Once the inflation timer has expired, the computer begins the tube venting process.

*3.8.3.1.10 Venting* After inflation, the computer waits an experimentally determined number of seconds (based on ground testing) before opening the vent valve. This time is required to allow the tube to cool and become rigid. After the required wait

time has expired, the relay board closes the inflation solenoid valve. Once the solenoid is closed, it redirects the gas into the environment, thus venting the tubes. During the venting, the A/D board is still recording data, and the imaging computer is taking pictures.

*3.8.3.1.11 Halt Data Collection* This step marks the end of the inflation routine. At this point, the A/D board stops its data collection. The data is formatted according to Section 3.7.1.3 and written to a data file named, "tube-x_inflation.dat." The counter/timer board shuts off the timing signal going to the imaging computer.

*3.8.3.1.12 Mark Failsafe Point* The last step in the inflation process is to mark where the computer has reached. This failsafe point is marked as x2. The reason why the only failsafe point is at the beginning and end of the inflation routine is that the computer cannot start in the middle of the inflation process. Once the heater box is opened and the inflation valve is open, the tube cannot be heated up enough to fully inflate it from a half-inflated state. Once the failsafe point is marked, the computer moves onto the excitation process.

*3.8.3.2 Excitation Process* As described in Section 3.5, one of the goals of RIGEX is to estimate the transfer function of a space inflated tube. To do so, the tube must be excited and its induced vibrations must be measured. To meet this goal, the computer system follows the following routine.

*3.8.3.2.1 Signal Imaging Computer* This step is placed here to get a final inflated tube image after the $N_2$ gas has been vented. The image is needed to get a pre-excited picture of the tube.

*3.8.3.2.2 Mark Failsafe Point* At this step the failsafe file is marked with the following value: x3.

*3.8.3.2.3 Activate A/D Channels for Vibration* This step activates the three A/D channels of tube x's accelerometer. Each of the channels is sampled at 5000 samples/sec. This sampling rate correlates with the D/A sampling rate. This correlation prevents aliasing.

*3.8.3.2.4 Transmit Excitation Waveform* This step occurs when the tube is excited. The waveform from 3.5 is transmitted to a tube 25 times. This waveform is created using one of the four D/A converters on the A/D board. The waveform consists of 5000 samples and is transmitted at 5000 samples/sec. This means that the waveform lasts for exactly one second. The total excitation time per tube is 25 seconds.

*3.8.3.2.5 Halt Vibration Data Collection* After the computer transmits the excitation waveform the required number of times, the computer stops collecting

vibration data. After the data has been collected, the data is formatted as stated in Section 3.8.1.3 and saved to a file named, “Tube-x_excite.dat”.

*3.8.3.2.6 Mark Failsafe Point* At this failsafe point, the failsafe file is marked with x4.

*3.8.3.2.7 Signal Imaging Computer* The imaging computer is once again signaled to take a picture. This picture is recorded to monitor any changes in the tube structure during excitation.

*3.8.3.2.8 Mark Failsafe Point* This is the last step in the excitation process and the last step for tube x. This failsafe point is recorded with x5. After completion of this step, the computer moves on to the inflation process for the remaining tubes or shutdown if it has tested the last tube.

*3.8.4 Shutdown*

After the computer has completed all required tasks, the computer shuts itself off. This shutdown is performed so that the computer does not have to idle while the batteries slowly die. The shutdown procedure requires two steps. All the environmental data collected is saved to a file called, “environ.dat.” The second step in the procedure is performed by the last relay on the relay board register 0x241. This relay shorts pin 1 and pin 2 together on J3 on the Jupiter-MM power supply [JMM01].

### *3.9 Required Tests*

This section is provided to give an understanding of the required tests and experiments needed to make the RIGEX computer system flight ready. The list below contains the major experiments that are conducted in order:

1) Excitation waveform test
2) A/D conversion of vibration signals
3) A/D conversion of thermocouple signals
4) A/D conversion of pressure signals
5) Single tube experiment run through without imaging in vacuum
6) Test imaging computer
    a) Interrupt routine to take pictures on command
    b) Test accuracy of height/displacement measurements.
7) Single tube experiment run through with imaging in vacuum
8) Full run through with three tubes at room temperature in vacuum
9) Full run through with three tubes at simulated space environment

Appendix D contains routine flow diagrams for the above experiments.

*3.9.1 Excitation Waveform Test* The purpose of performing the excitation waveform experiment is to determine if the chosen waveform from Section 3.5 is sufficient. To perform the test, the PC/104 is programmed to output the sampled waveform through the D/A channels at a rate of 5000 samples/sec. The rate is

established by programming timer 0 on the A/D board to use a 10 kHz clock and to divide the clock by two giving 5000 Hz. The computer then uses the timer 0 interrupt pin to identify the high-to-low transitions. On these interrupts, the computer sends one sample out to the D/A converter. The waveform is repeated 10-25 times to attain an average for the tube transfer function. The measure of success is the quality of the transfer function as determined by the spectrum analyzer.

*3.9.2 A/D Conversion of Vibration Signals* This test is the next logical step following 3.8.1. From here, the computer is programmed to sample the output of the accelerometer attached to the top of an inflated tube. This sampling occurs at the same rate as the D/A sampling. Having both sampling rates the same reduces the chances of aliasing. The measure of success for this experiment is the quality of the transfer function found using the vibration data collected by the A/D board.

*3.9.3 A/D Conversion of Thermocouple Signals* This test is required to primarily to make sure that accurate readings are measured using K-type thermocouples and the MSI-P440 A/D board. Success for this experiment is measured by comparing temperature measurements made by the PC/104 with measurements made by precision laboratory equipment. The desired error is less than 1ºC.

*3.9.4 A/D Conversion of Pressure Signals* This test, like that of 3.8.3, is to make sure accurate pressure readings can be measured by the A/D board. Success is measured by comparing the A/D outputs with the laboratory equipment.

*3.9.5 Single Tube Experiment/No Imaging* This experiment is have the computer run through one iteration of the tube inflation and excitation processes. The experiment is conducted using the ¼ prototype setup used by [Phi03] with the exception of power. The heaters and computer are powered using the battery cells discussed in Section 3.2. The experiment is conducted in a vacuum chamber at room temperature. The experiment should produce data files containing the environmental data, temperature data, inflation data, and the excitation data. The excitation data on these files is then be analyzed to produce the transfer function of the inflated tube. The inflation data is compared to the data gathered by [Phi03]. The use of vibration data during inflation can be used to get a three dimensional picture of how the tube moved during inflation. The pressure and vibration data during this experiment is sampled at 5000 samples/sec. The temperature data is sampled at one sample/sec.

*3.9.6 Test Imaging Computer* This test consists of two parts. The first is a test to get the two computers working together. The test setup has the data acquisition computer begin signaling the imaging computer for a few seconds during which the tube number changes. The imaging computer monitors the tube number and takes pictures as it receives the interrupts. The primary purpose of this experiment is to ensure that the

camera system works and to determine how fast the imaging computer can take images and save them to secondary memory.

The second portion of this test is to determine the accuracy of the height/tilt measurement method defined in Section 3.4 and [Po03]. The test is conducted by taking pictures of a circular target at different known angles and distances from the camera. The program from Section 3.4 analyzes these "validation" images and compares the results to the true values. The desired accuracy for the height is 1mm and for angle about 1º.

*3.9.7 Single Tube Experiment with Imaging* This experiment is the same as described in Section 3.9.5 with the exception that the imaging computer is incorporated. The experiment produces the data as before and pictures as described in Section 3.8.

*3.9.8 Full Experiment in Vacuum at Room Temperature* This is the next step after completing a full test of one tube. This test is conducted using the full-size structure with three tubes. The experiment follows the operational flow as described in Section 3.8. The experiment is conducted in a vacuum chamber at room temperature. Because the system is at room temperature, the environmental heaters are not tested. The goal of this test is to have the computer fully deploy three tubes and collect all the required data.

*3.9.9 Full Experiment in Vacuum at Space Temperature* This final test is to perform the same test as described in Section 3.9.8 but to have the experiment in a cold environment. This cold environment will affect the battery power and the electronic

operation of the computer and camera systems. It will also allow for the testing of the environmental controllers. The goal of this experiment is to deploy the three tubes, collect all the required data, and to determine whether there are any differences in tube transition temperature and the amount of time required for heating.

### *3.10 Summary*

This chapter has presented all the RIGEX requirements for the electrical system being used to support the experiment. For each of the requirements, the preliminary solutions have been developed. The areas covered include the power system, the hardware, the software operation, and the required experiments needed to consider the electrical system flight worthy. The results of the experiments covered in this chapter will be the main focus of Chapter 4.

# IV. Analysis and Results

## *4.1 Introduction*

This chapter presents the results from the various experiments described in Section 3.9. Each experiment was setup to test a specific PC/104 function or to test the entire system in different environments. During each experiment, problems occurred. Not all problems will be described due to the cause being minor or lacking an important lesson to be learned. Any changes to the preliminary design are also described here.

## *4.2 Excitation Waveform Test*

The main purpose of this test is to determine if the upchirp signal described in Section 3.5 can adequately excite to the inflated tube in order to determine the tube's transfer function. Figure 3.5 shows the transfer function of the tube using lab equipment to excite and measure the vibrations in the tube.

*4.2.1 Description* The experiment was setup with an inflated tube bolted to a table with the piezo patches attached. The PC/104 was programmed to transmit the digitized chirp signal directly to the patches on the tube 25 times. Each transmission lasted for 1 sec with a sampling rate of 5 KHz. A small 10 mV/g accelerometer was attached to the top plate of the tube to measure the vibration signals as they traveled the length of the tube. To determine the transfer function, the digitized chirp and the accelerometer output were sent to a spectrum analyzer.

*4.2.2 Results* Two sets of results were collected. The data from the spectrum analyzer can be seen in Figure 4.1. The lab equipment used to develop Figure 3.5 was configured to also receive the same signals as the spectrum analyzer except it used a 100 mV/g accelerometer. As can be seen in each of the two figures, the first and second modes are apparent and can be measured to determine frequency and strength. This experiment showed that the digital chirp signal provided the necessary frequencies and voltage to excite the tube.

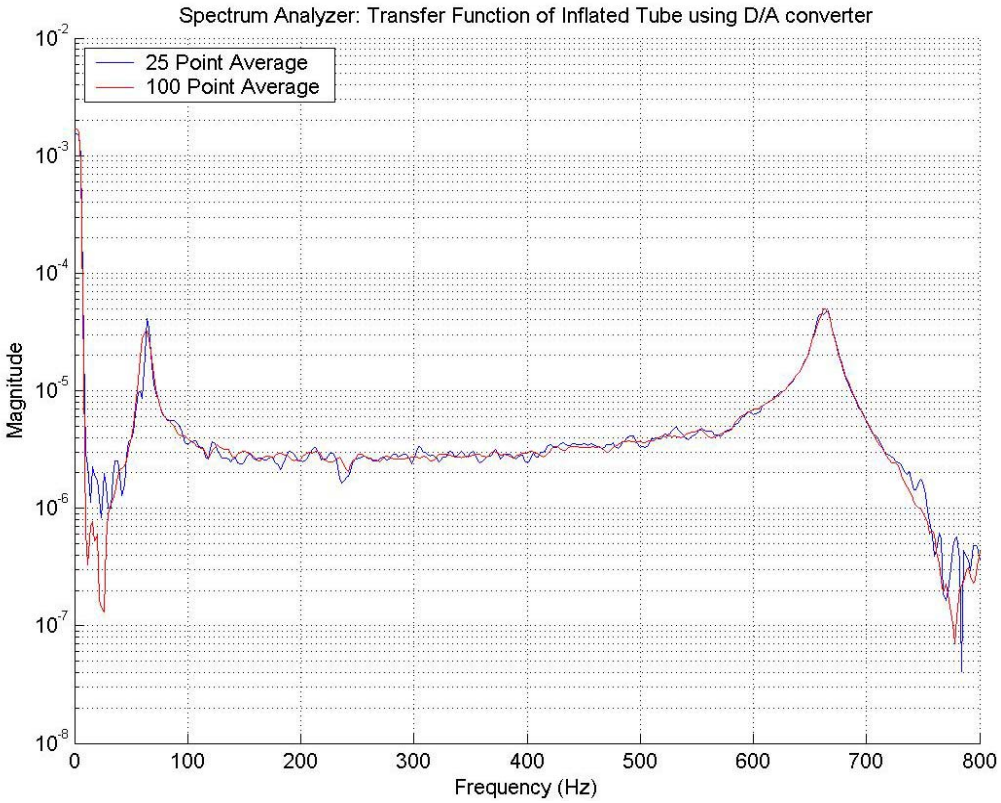



Figure 4.1: Transfer Function of inflated tube using chirp signal from PC/104

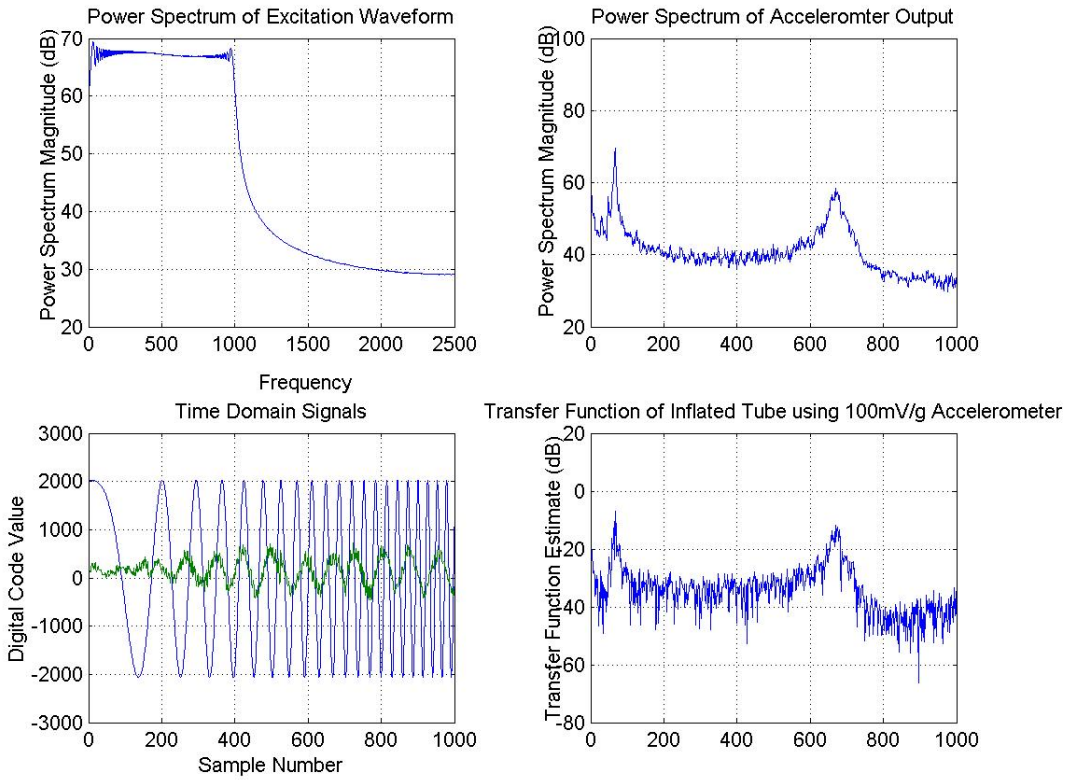



Figure 4.2: Lab equipment results for transfer function using chirp from PC/104

### *4.3 A/D Conversion of Vibration Signals*

This experiment consists of two parts. Each part is used to validate a portion of the A/D conversion process for the experiment's vibration signals.

*4.3.1 Experiment Part 1* This portion of the experiment excites the tube as before using the D/A converter and sample the accelerometer (10 mV/g) output using the A/D converters. The goal is to make sure the PC/104 can acquire enough valid data that has a high enough signal-to-noise ratio to estimate the transfer function of the inflated tube.

*4.3.1.1 Part 1 Description* This portion of the experiment has the D/A converter connected to the piezo patches to provide the excitation signal. The 10 mV/g accelerometer output is connected to one of the A/D converter channels. The excitation waveform is transmitted 25 times at a sampling rate of 5 kHz. The A/D converter samples the accelerometer output at the same rate. Once the data is collected, a Matlab® script loads the digital data and converts it to voltage values. The signals are then used to generate a transfer function for the inflated tube. All Matlab® scripts used for the experiment's data files are found in Appendix E.

*4.3.1.2 Part 1 Problems Encountered* The initial results of this portion of the experiment were not good. The data produced a transfer function where the 0 – 1000 Hz range was lower than the noise level from 1001 – 2500 Hz. The cause of this was determined to be from the digitized excitation waveform. Since the excitation waveform was being transmitted as a discrete amplitude signal, the original bandwidth (0 – 1000 Hz positive frequencies only) of the chirp signal was present at every harmonic of 5 kHz. This effect is known as aliasing [Skl00]. Because the A/D converter was sampling at the same rate as the D/A converter, the aliasing caused each of the chirp copies to fold over destructively on the 0 – 1000 Hz bandwidth [PrM96].

The solution to this problem was to remove the copies located at 5 kHz and up. Two methods existed to accomplish the solution. The first is to send the digitized chirp signal through a low-pass smoothing filter with a bandwidth of 1000 Hz. The second is to place an anti-aliasing 1000 Hz low-pass filter before the A/D converter input. It was

decided to place the filter between the accelerometer and the A/D converter. The chosen filter was a fourth order Butterworth low-pass filter with a bandwidth of 1000 Hz. This choice in filter should theoretically apply between 60 – 70 dB of attenuation [Dar76].

*4.3.1.3 Part 1 Results* After placing the filter before the A/D converter, the computer was able to collect data without suffering from any aliasing. Figure 4.3 shows the transfer function using a 10 mV/g accelerometer and the setup as described in Section 4.3.1.1. As can be seen in the figure, the first and second modes are readily present. This resulting figure satisfies the goal of this portion of the experiment.

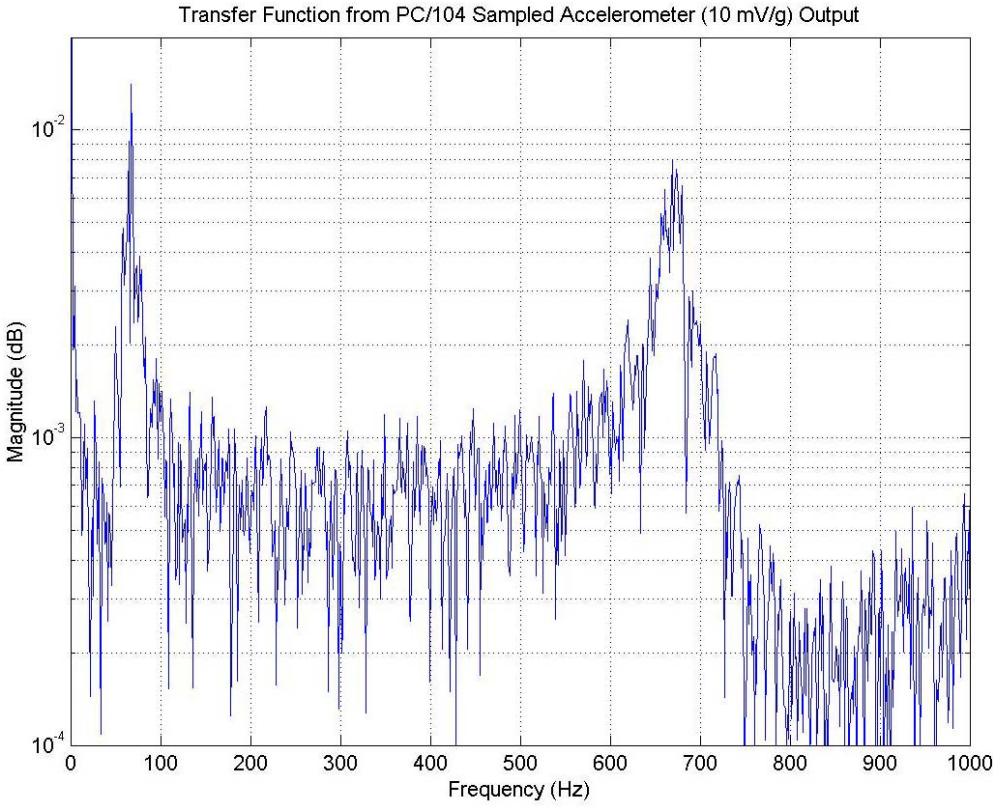



Figure 4.3: Transfer function of inflated tube using data collected by the PC/104

*4.3.2 Experiment Part 2* The second part of the experiment is to excite the tubes as before and use the triaxial accelerometers to provide the vibration signal to the computer. The goal of this portion of the experiment is to determine if the chosen accelerometers will do the job and provide data to estimate the tube's transfer function.

*4.3.2.1 Part 2 Description* As in Section 4.3.1.1, this part of the experiment is setup the same. The only change is the accelerometer being used. The accelerometer in this portion is the Summit Instruments 34200A [SuI02]. As described in Section 3.3.3.3, the accelerometer has three axes with voltage ranges of 0 – 5 V. The accelerometer outputs were connected to the A/D converter through the same filter described in Section 4.3.1.2.

*4.3.2.2 Part 2 Problems Encountered* During this portion of the experiment, two major problems came up. The first issue was the accelerometers. They were expected to have an estimated sensitivity of 56 mV/g, but the accelerometers arrived with a sensitivity of approximately 180 mV/g, three times better. This higher sensitivity meant the chirp copies at each harmonic of 5 kHz had more strength. This caused more aliasing than the $4^{th}$ order Butterworth filter was able to remedy. The solution to this was to smooth the digitized chirp before it reached the tube. This was accomplished by placing an $8^{th}$ order Butterworth low-pass 1000 Hz filter at the output of the D/A converter. This filter was implemented using a Maxim MAX274 4-stage $2^{nd}$ order continuous filter chip. The chip was connected up using the required resistors to produce the desired filter

[Max96]. Table 4.1 lists the required resistor values to implement the filter. Each section of the filter has its output tied to the input of the next section. Section 1's input is the input to the whole filter and Section 4's output is the output to the whole filter. Once the filter was built and tested, a current buffer chip was placed on the output to prevent any distortion that might be caused when current is drawn from the filter.

Since the digitized version of the excitation waveform was now being filtered, this required the filtered version of the excitation waveform for the transfer function estimation. To do this, the continuous time transfer function of the filter must be converted to a discrete time transfer function. To perform this change, the bilinear transformation was used [PrM96]. Equation 4.1 gives the s-domain substitution to convert the transfer function to the z-domain.

$$s = \frac{2}{T} \cdot \frac{1 - z^{-1}}{1 + z^{-1}} \qquad 4.1$$

The *T* parameter is a time step value determined from the sampling. The bilinear transformation was implemented using Matlab®'s *bilinear* function. The 8th order filter can be broken up into four 2nd order filters.

Table 4.1: Resistor Values for 8th Order Filter

| | Section 1 | Section 2 | Section 3 | Section 4 |
|---|---|---|---|---|
| $R_1$ | 400 kΩ | 400 kΩ | 400 kΩ | 400 kΩ |
| $R_2$ | 2 MΩ | 2 MΩ | 2 MΩ | 2 MΩ |
| $R_3$ | 203.9 kΩ | 240 kΩ | 360 kΩ | 1 MΩ |
| $R_4$ | 1.99 MΩ | 1.99 MΩ | 1.99 MΩ | 1.99 MΩ |

The four s-domain transfer functions are the following:

$$H_1(s) = \frac{3.94784176 \cdot 10^7}{s^2 + 1.23249113 \cdot 10^4 \cdot s + 3.94784176 \cdot 10^7} \quad 4.2$$

$$H_2(s) = \frac{3.94784176 \cdot 10^7}{s^2 + 1.04485553 \cdot 10^4 \cdot s + 3.94784176 \cdot 10^7} \quad 4.3$$

$$H_3(s) = \frac{3.94784176 \cdot 10^7}{s^2 + 6.98150145 \cdot 10^3 \cdot s + 3.94784176 \cdot 10^7} \quad 4.4$$

$$H_4(s) = \frac{3.94784176 \cdot 10^7}{s^2 + 2.45157729 \cdot 10^3 \cdot s + 3.94784176 \cdot 10^7} \quad 4.5$$

The full transfer function of the filter can be found by multiplying the four together. Once the filter's transfer functions were converted, the digitized excitation waveform was then filtered through the digital version of the filter in Matlab® using the *filter* function before being used for the transfer function estimation.

After the filter was implemented, the test was conducted and there were not any suitable results. It was determined that the filter's ±5 V swing was not enough to move the tube with the accelerometer attached. The accelerometer has a mass of 35 g [SuI02]. With the added mass at the top of the tube, a higher voltage on the piezo patches was needed to produce enough vibration in the tube. The higher vibration was needed in order for the signal to be detected above the thermal noise on the signal lines of the accelerometer. To test this idea, the output of the filter was connected to a power

amplifier with the amplifier's output connected to the piezo patches. This voltage amplification allowed the vibration signals to be seen above the noise on the accelerometer. Since only amplification of the voltage is necessary, it was decided to use a transformer to provide the necessary amplification. This amplification was done by attaching the filter output to the secondary side of the transformer (referencing to a step-down transformer) and connecting the primary side to the piezo patches. This connection allows for the transformer to work in the opposite direction as a step-up transformer [GuH95]. To test this theory, a 120 VAC/12 VAC power transformer was used. The secondary side also had a center tap allowing the transformer to operate as a 120 VAC/6 VAC transformer. The filter output was connected to the 6 VAC side and the patches to the 120 VAC side, providing a gain of approximately 20 to the voltage signal coming from the filter. Appendix G shows how the transformer is connected to the patches.

*4.3.2.3 Part 2 Results* After fixing the problems described in the previous section, the tube was excited and the data was collected. Using the above transformer produced good results on the spectrum analyzer so the data from the computer was processed. Figure 4.4 shows the transfer function for each of the three axes. The tube's transfer function is present along the Y axis of the tube which is the axis that the piezo patches are aligned. Figure 4.5 shows the same transfer function but they have been smoothed using a four point averaging filter. From the figures, it is clear that the first mode is present and its frequency can be determined but the second mode is noisy and does not have much

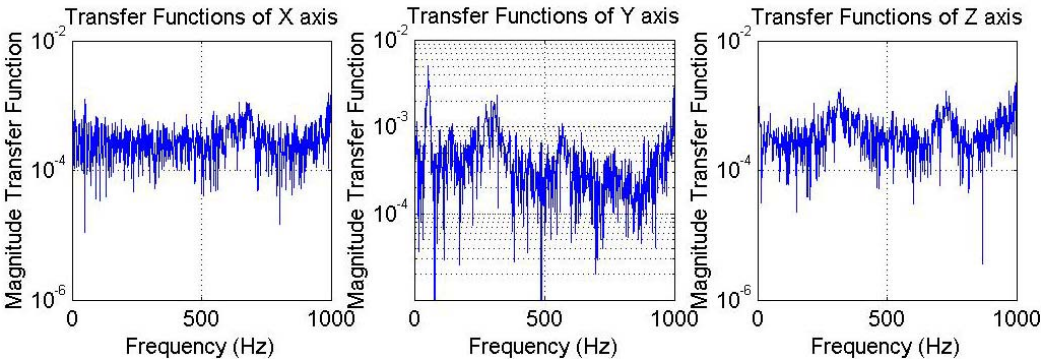


Figure 4.4: Transfer functions along each axis using a 120/12 transformer

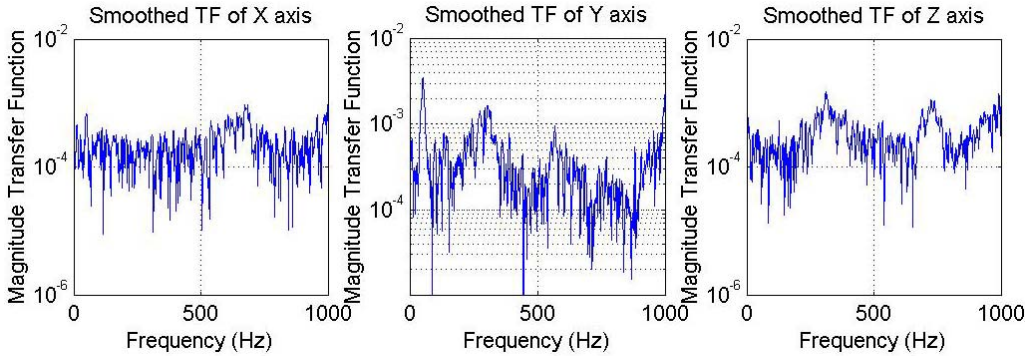


Figure 4.5: Averaged transfer functions along each axis using a 120/12 transformer

separation from the noise floor. The noisy second mode could be cleaned up if the excitation signal is boosted more. To help clear this up, a 230 VAC/6 VAC step-down transformer was used. This transformer also had a center tap on the secondary side. This allowed it to be operated as a 3 VAC/230 VAC step-up transformer, providing approximately a voltage gain of 80. The higher gain transformer performed as expected. The separation between the noise floor and the mode frequencies became more evident. Figure 4.6 shows the transfer functions of each accelerometer axis using the 3 VAC/230 VAC transformer. Figure 4.7 shows the smoothed version of the same transfer functions.

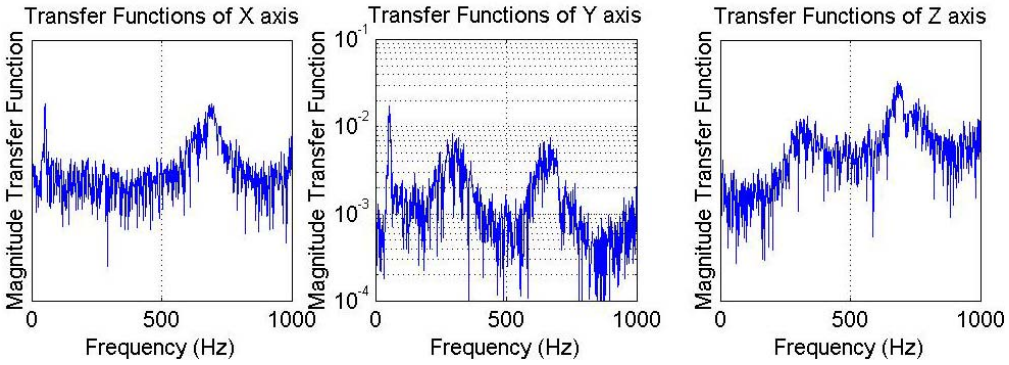



Figure 4.6: Transfer functions along each axis using a 230/6.3 transformer

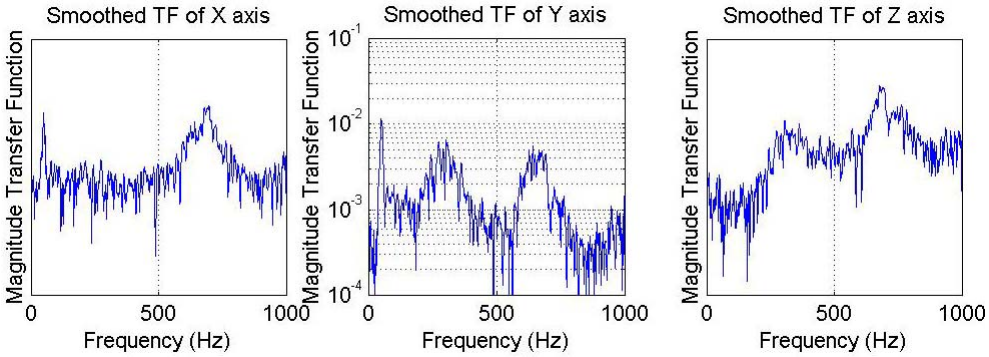



Figure 4.7: Averaged transfer functions along each axis using a 230/6.3 transformer

By raising the strength of the excitation signal, a third mode became noticeable. Also due to the weight of the accelerometer, the mode frequencies dropped down to approximately 50 Hz and 300 Hz for the first and second modes respectively.

In an effort to sharpen the second and third modes of the transfer function, a digital low-pass filter is used to remove some of the noise in the sampled vibration signal. Using the digital filter is valid because the excitation waveform is filtered before it gets to the tube. This means the only aliased by the A/D conversion is the additive noise on the accelerometer output. The low-pass filter was designed using the Remez Exchange

Algorithm [MiS01]. The Remez algorithm implements the Parks-McClellan equi-ripple filter design algorithm [MiS01]. The filter would be designed with a pass-band frequency of 1000 Hz with a 0.01 dB ripple. The cutoff frequency would be set to 1100 Hz with a stop-band ripple of 0.1 dB. Matlab®'s *remezord* script was used to estimate the required number of taps and the script *remez* was used to produce the FIR filter tap weights. The filter ended up being a 68 tap FIR filter. Figures 4.8 and 4.9 show the resulting transfer functions when the digital filter is used.

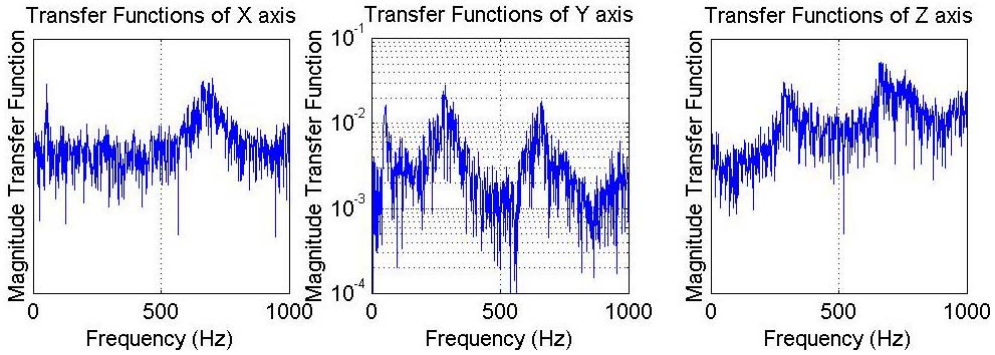


Figure 4.8: Transfer function after digitally filtering the accel. output

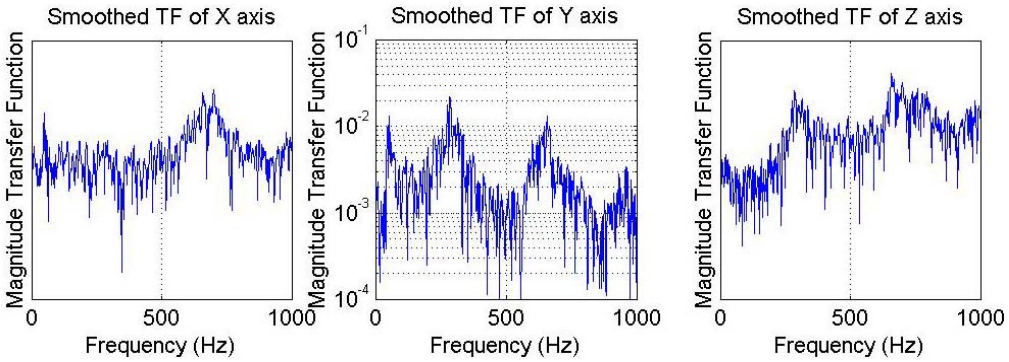


Figure 4.9: Smoothed transfer function after digitally filtering the accel. output

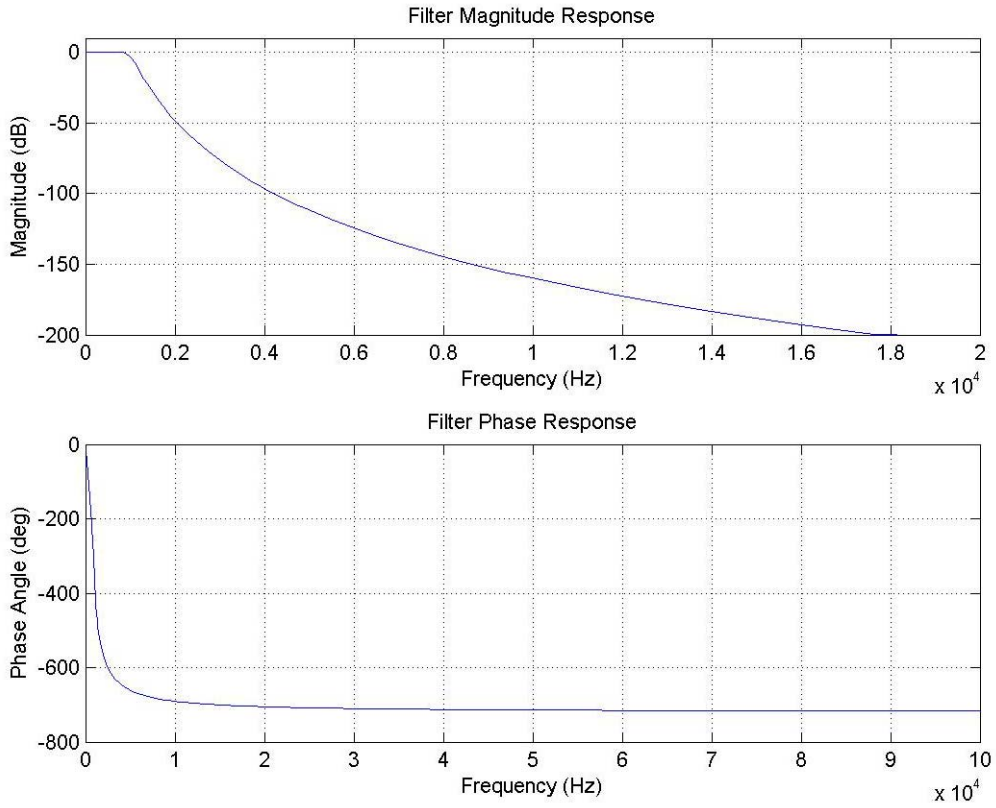



Figure 4.10: Filter board theoretical filter response

### *4.4 A/D Conversion of Thermocouple Signals*

The primary goal for this experiment was to determine how accurately the temperature signals could be measured from the thermocouples. The MSI-P440-K thermocouple A/D board was chosen to perform this job. The experiment was setup to measure the temperature once per second.

*4.4.1 Description* The experiment was setup by attaching an Omega Process Calibrator to the first input channel of the board. The calibrator is able to simulate a thermocouple and generate an artificial temperature signal for the board. The

temperature test program was setup to sample each channel 25 times at temperatures ranging from 0ºC to 200ºC in 10ºC increments. The data is then saved to a data file to be processed by a Matlab® script.

*4.4.2 Problems Encountered* When the experiment was first performed, all the temperatures on all the channels were averaging between 8º - 12°C above the true temperature. This can be seen in Figure 4.11. All connections were double checked to make sure everything was wired correctly and that the program was setup correctly. When the problem persisted, the temperature A/D board manual was studied. According to [MiS03], the board can generate temperature errors when the board's temperature rises to high. The manual also stated that if a channel is shorted and then sampled, the result is the temperature of the ice-point compensator (AD597) and thus the temperature of the board. According to the AD597 data sheet [AD98], the ice-point compensator can

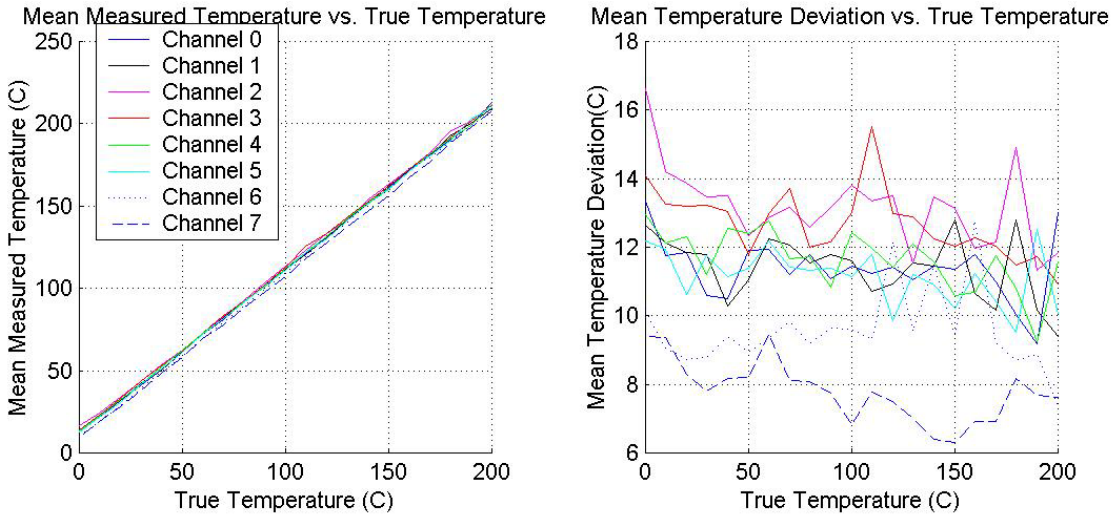



Figure 4.11: Uncompensated temperature mean test results

produce errors when its reference junction temperature is raised. Nowhere in the data sheet did it state what the reference temperature should be to provide proper temperature measurements. An assumption of 21ºC, room temperature, was made as being the reference temperature.

To provide the board temperature compensation, channel 7 on the board was shorted and the program was altered to sample channel 7 as the board temperature. Once the temperature of the board was determined, 21ºC was subtracted from it to give the amount of temperature compensation. Every sample taken after this point would have the compensation amount subtracted from the sample temperature. Figure 4.12 shows the mean results after performing the compensation. As can be seen in the figure, the compensation moved all the average temperatures closer to the true temperature line.

After the compensation solved the errors caused by the board temperature, the next step was to look at how well the final measured temperatures tracked with the true

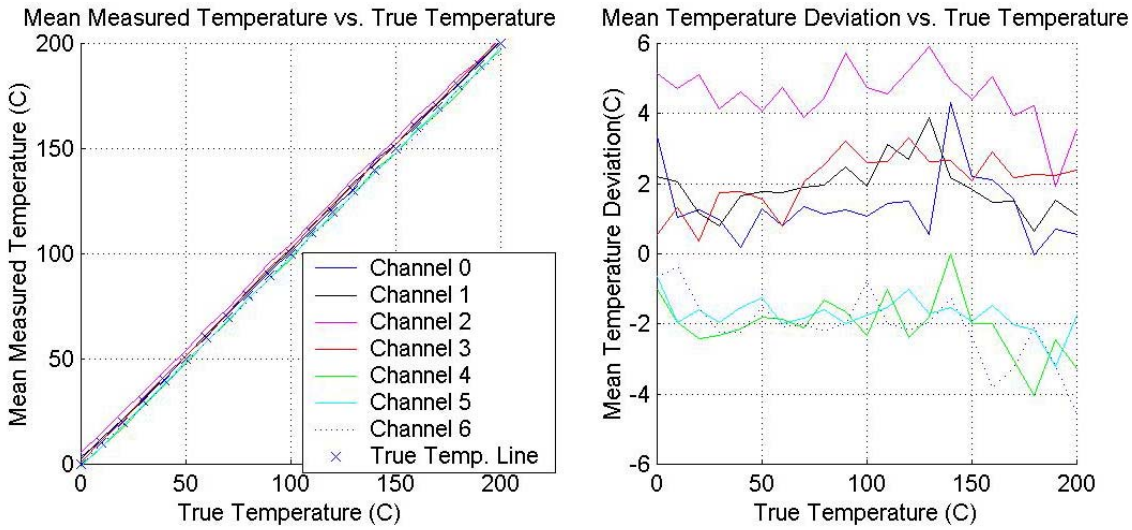



Figure 4.12: Compensated temperature mean test results

temperature. As can be seen in Figures 4.11 and 4.12, the measured temperatures were very noisy. The jumpiness of the sampled data posed problems when being compared to a particular threshold temperature (as would be required in the experiment). The solution to this problem was to implement a moving average filter while the data is collected by the computer. The averaging filter would then smooth out the noisiness of the samples. The next step was to determine how many samples would be used in the average. To determine this, the data collected from the compensation tests would be filtered and plotted against the true temperature line. The filter that tracks the true temperature line the best will be the chosen filter length. Figure 4.13 shows the results using a four tap averaging filter on channel 0's data. Figure 4.14 shows the results for 10-tap filter. The 10-tap averaging filter was chosen to be the filter length ($h_{temp}$). This choice was made because it smoothed most all of the large jumps but still tracked the true temperature line.

$$h_{temp} = \frac{1}{10} \cdot [1 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1] \tag{4.6}$$

*4.4.3 Results* To test all the fixes as described above, the computer was setup to sample the temperature at 1 sample per second and compare the 10 point average to the threshold temperature of 128ºC. The computer was programmed to continue sampling until the threshold was met. Once the threshold temperature is reached, the computer stopped sampling. Figure 4.15 shows the raw temperature measurements and also shows the averaged version. On each plot, the threshold temperature line is drawn to

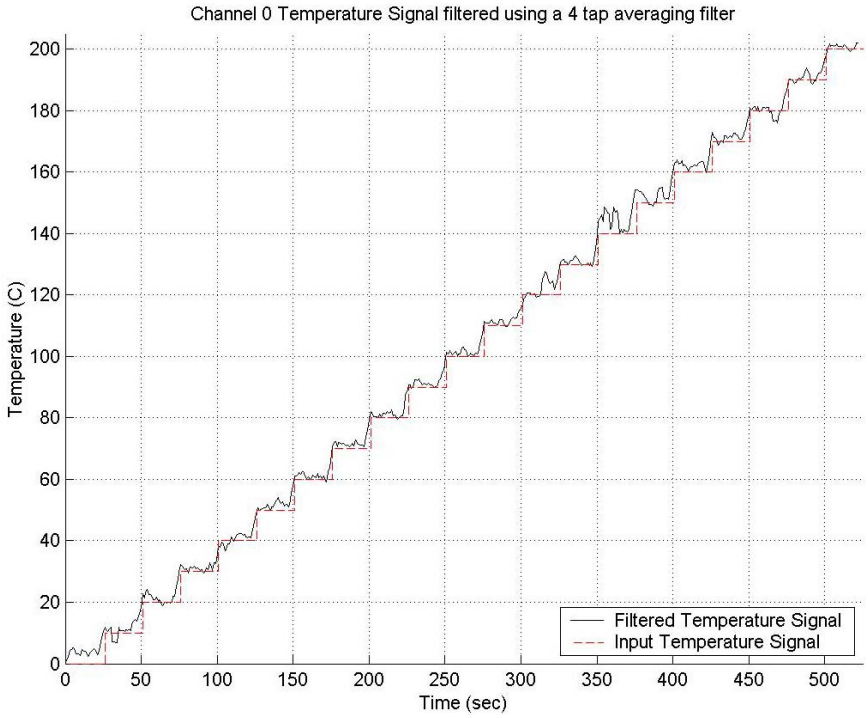



Figure 4.13: 4-tap averaging filter result versus true temperature line

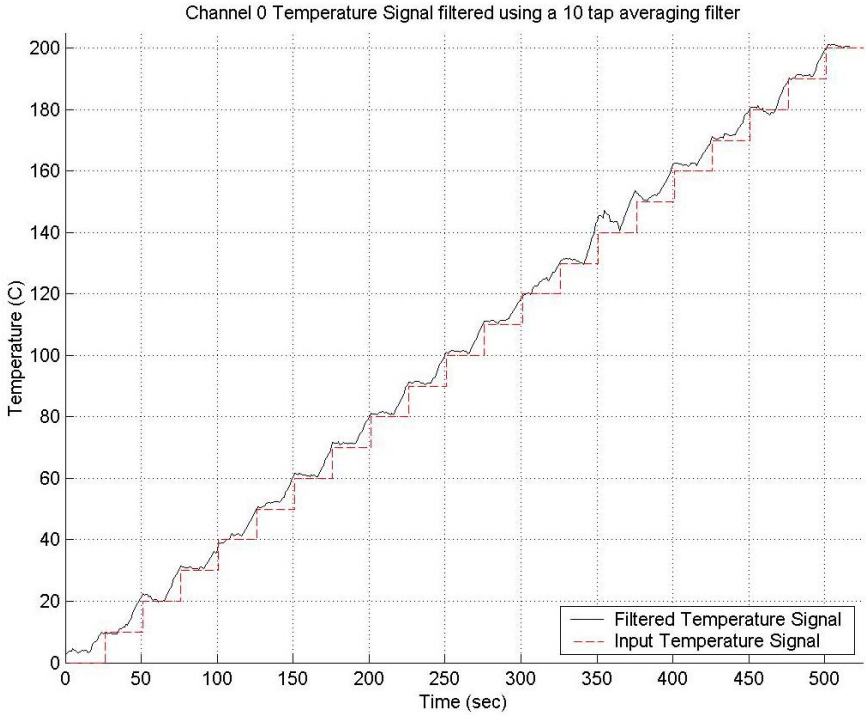



Figure 4.14: 10-tap averaging filter result versus true temperature line

demonstrate the need of the averaging filter. After compensation and filtering, the thermocouple A/D board worked and performed as desired.

### *4.5 A/D Conversion of Pressure Signals*

The goal of this portion of this experiment is to determine if pressure signals can be measured using the chosen pressure transducers. All pressure signals from the transducers are with respect to the outside pressure, if a positive pressure is measured, a

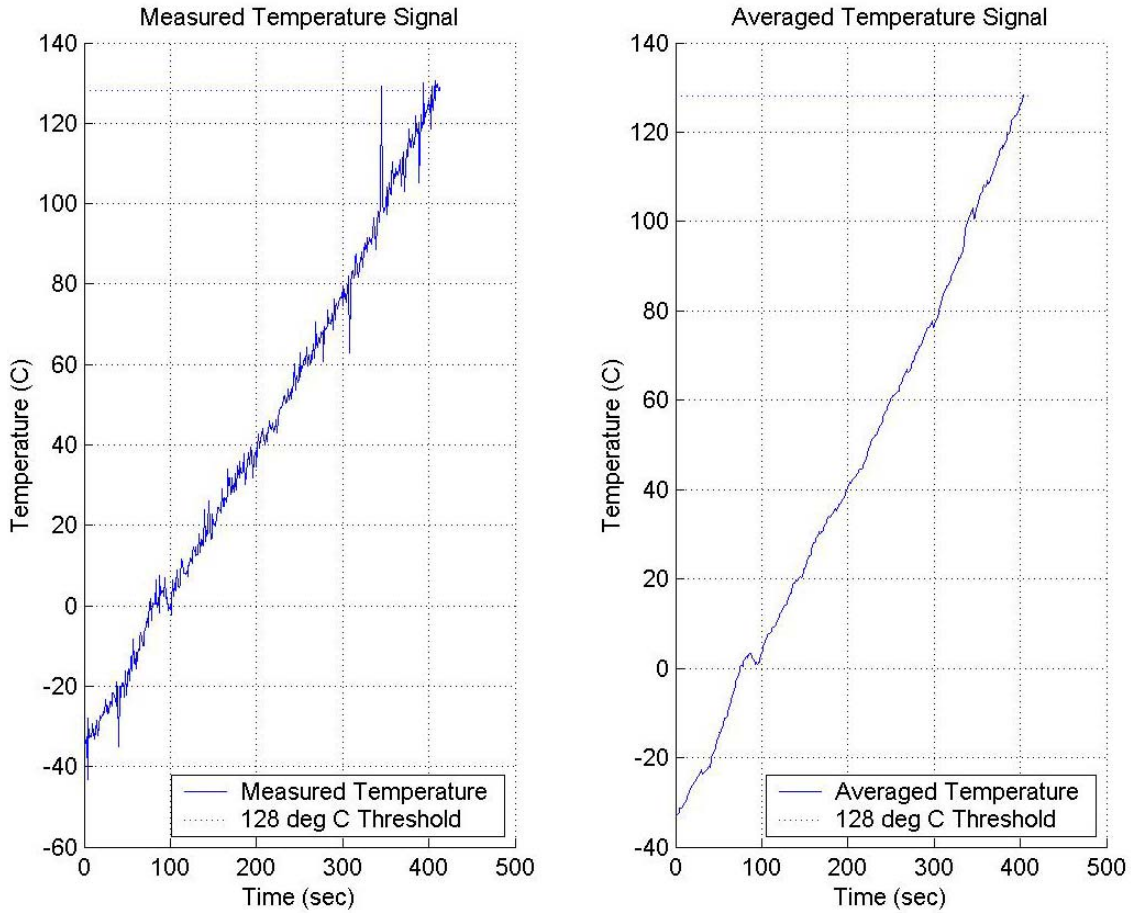



Figure 4.15: Threshold temperature test

positive signal is produced. The transducers produce differential voltage signal. This is because the transducers operate using a strain gauge system similar to that described in Section 3.3.1.2.

*4.5.1 Description* Due to the differential output of the transducers, the Diamond-MM A/D board was reconfigured so that the first eight channels were differential inputs. The output of the transducer was connected to channel 0 of the A/D board and the power connections were connected to +5 VDC and ground. To induce a positive pressure on the transducer, the experimenter blew on the sensor side of the transducer while data was being recorded. To induce a negative pressure, the experimenter would place a vacuum against the sensor side. The sampling rate of the pressure signal was 5 kHz. The sampling rate was chosen because the pressure will be measured the same time as the vibration signals.

*4.5.2 Results* No problems were encountered during this experiment. The only adjustments needed were to reduce the dynamic range of the A/D converter down from ±5 V to ±0.625 V. This provides much better accuracy. Figure 4.16 shows an example of a pressure signal placed on the transducer. Again, the pressures applied to the sensor were random and non-specific. The goal of validating the sampling of the pressure transducer was met.

### *4.6 Single Tube Test without Imaging*

For this portion of the RIGEX data acquisition testing, the desire for the experiment was to heat and inflate a rigidizable tube. Unfortunately, due to the limited number of rigidizable tubes available, this experiment had to be simulated by testing each part of the RIGEX programming. There are three main sections to the simulation: heating, inflation, and excitation. Each section gives the simulation setup followed by the resulting plots of the collected data.

*4.6.1.1 Heating Simulation Setup* The heating portion of the experiment was conducted using one of the rigidizable tubes. Since the heated tube would not be inflated,

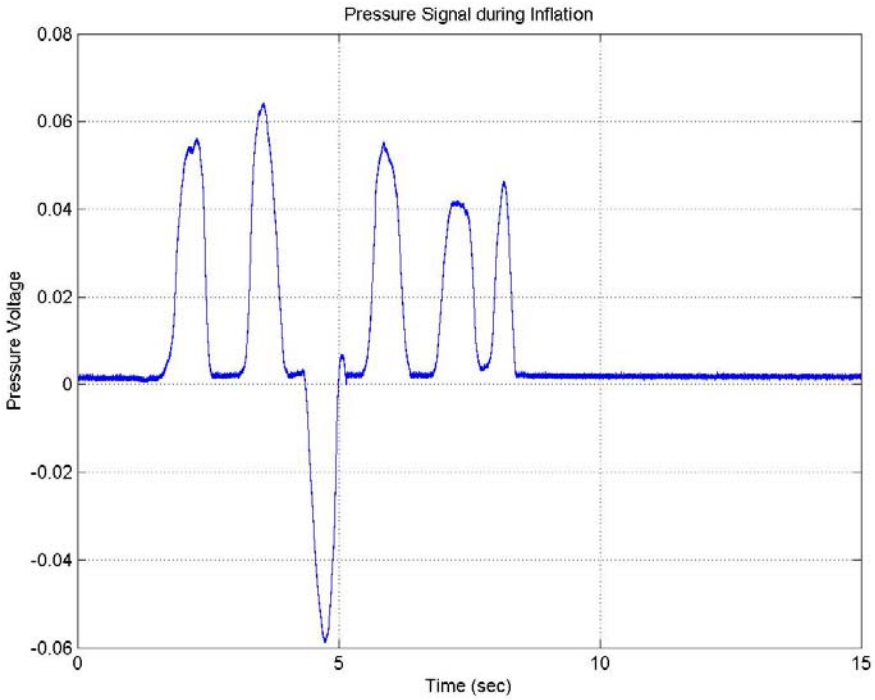



Figure 4.16: Example of a measured pressure signal

it could be heated multiple times without any wear. The experiment was setup with two thermocouples (T/C) connected to the tube and the PC/104's MSI-P440-K interface board. Another T/C was placed on the structure approximately 8 inches above the oven and connected to the interface board. A separate thermocouple-meter was then connected to the tube as another visual temperature measurement.

The program is written to measure the two T/C channels for the tube, the T/C channel for the structure and a fourth T/C channel that has been shorted once per second. As described above in Section 4.4, the shorted channel provides for the board/measurement temperature differential. The program then performs a 10 point moving average on the high T/C channel for threshold testing. The temperature threshold for the test was set to 130ºC in order to guarantee that all points on the tube would reach the transition temperature of 125ºC. The higher threshold also allows for any transition temperature tolerance.

*4.6.1.2 Heating Simulation Results* The computer performed its job as expected. When the average temperature reached the threshold, the computer shutoff the current to the heater box and ceased excitation (transition to the inflation routine for the real RIGEX operation). Figure 4.17 shows a tube heating result, the high and low T/C signals show the raw data as well as the 10 point averaged version. It also shows the 10 point averaged

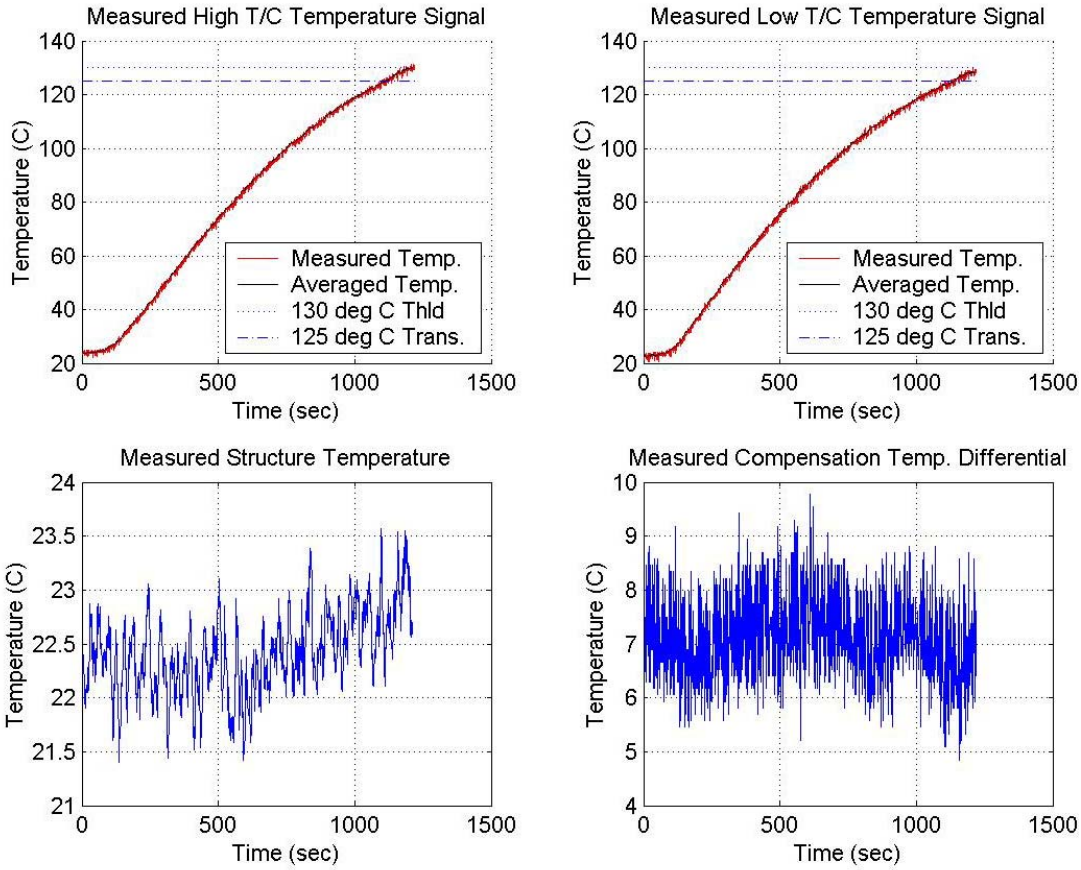



Figure 4.17: Tube heating curves.

structure temperature and the temperature compensation differential. It can be seen that the low T/C signal reached transition before the heating process was concluded. It can be seen that the tube heating only required about 200 seconds longer to go from 125ºC to 130ºC. This experiment demonstrates that the computer is capable of controlling the tube heating process using real T/C signals in the real heating environment.

*4.6.2.1 Inflation Simulation Setup* This portion of the experiment suffered from the lack of a rigidizable tube. To test this portion of the RIGEX experiment, a rubber

tube was used instead of a rigidizable tube. This allowed the test to be repeated as needed. The rubber tube had the same inflated dimensions but does not have the same folded dimensions due to the non-rigid structure of the rubber. This portion of the experiment is to measure the pressure signal of the inflation process as well as the vibration signals during inflation.

The goal of the vibration signals is to attempt to map the movement of the top plate of the tube during inflation. Unfortunately, the top plate has six degrees of freedom with respect to its movement. It has the three dimensions of translation, which the accelerometer can measure, but it also has three axes of rotation that cannot be measured by the accelerometer. This lack of rotation data leads to errors in the mapping of the inflation. In order to measure the rotation, it would require adding a second accelerometer to the top of the tube. Adding a second accelerometer would add too much weight.

*4.6.2.2 Inflation Simulation Results* After the data was collected, it was noticed that there was a lot of noise on the signal. To help mitigate the noise, the power spectral density (psd) of the vibration signals were analyzed. The noise floor of the psd tended to start around 50 – 100 Hz. A digital low-pass filter was designed to push the noise floor down to increase the separation between the vibration signal and the noise. The filter was designed using the Remez Exchange Algorithm as described in section 4.3.2.3. The pass-band frequency will be set to 50 Hz and the stop-band frequency set at 60 Hz. The resulting filter order was 674 tap weights.

Once the signals were filtered for noise, a double integration of the signals is needed to convert from acceleration to displacement. Two running sums were performed on each of the three vibration signals. To properly perform the discrete integration, each running sum was multiplied by the sampling period. Figures 4.18 through 4.20 show the three axes' vibration signals. Each figure shows the unfiltered and filtered acceleration signals, the psd of the unfiltered and filtered acceleration signals and the displacement signal after the two discrete integrations. Figure 4.21 shows the movement of the top plate of the tube. The movement curves are acquired by combining the X, Y, and Z dimensions together in time. The diamond marks the starting point. In this particular inflation test, the rubber tube began to inflate and then fell far to the side (due to the weight of the accelerometer) causing a large amount of rotation in the top plate of the tube. Once enough pressure was reached the tube straightened up but the straightening is not recorded properly due to the rotation error. Figure 4.22 shows the pressure signal as the rubber tube inflated. In the figure, the pressure drops and rises as a new fold in the tube comes unfolded.

This portion of the experiment demonstrates the tube motion during inflation can be measured but will have error due to any rotation of the accelerometer. This example uses a rubber tube. Until a rigidizable tube is inflated, the amount of rotation error is unknown.

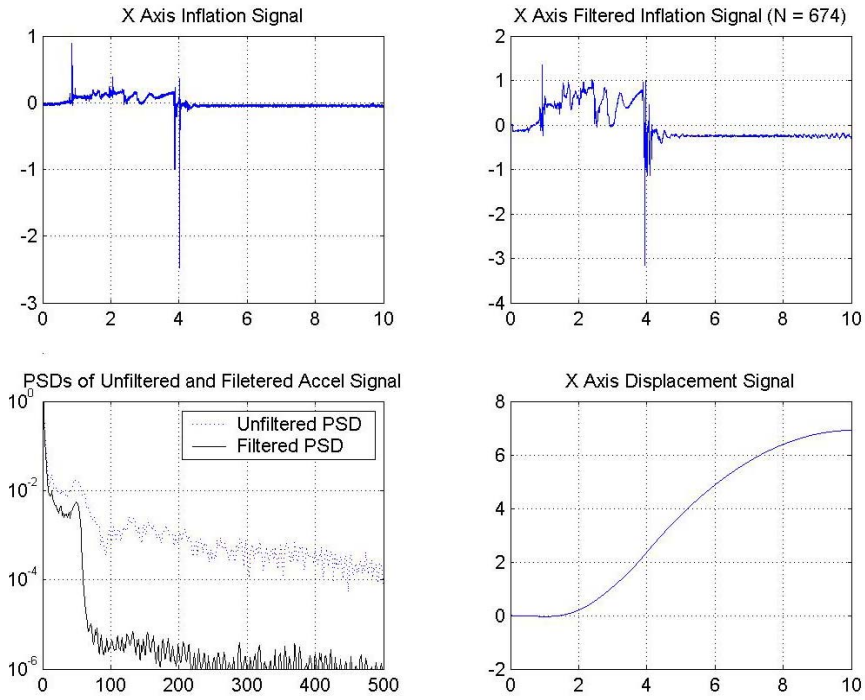



Figure 4.18: X axis acceleration and displacement signals

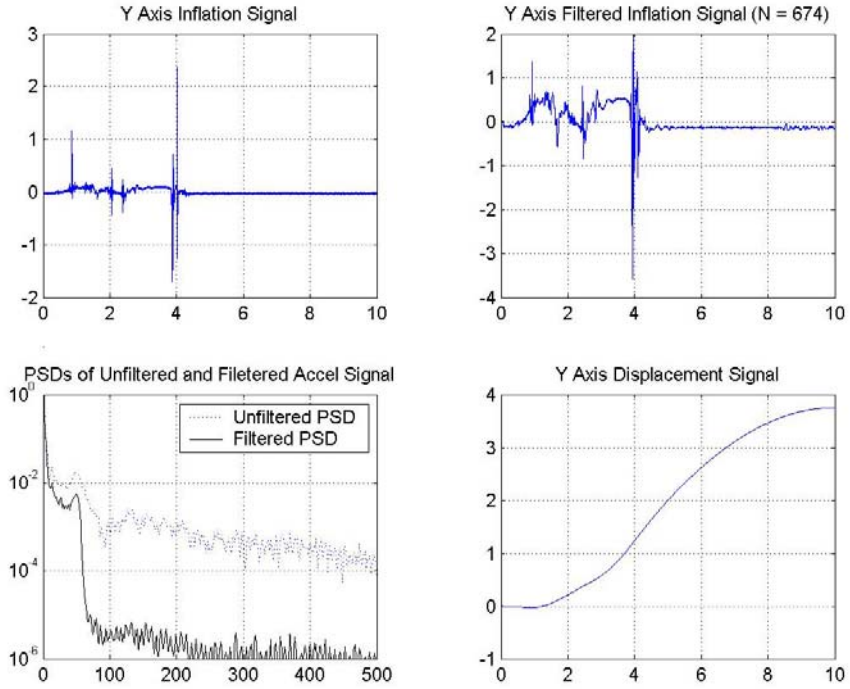



Figure 4.19: Y axis acceleration and displacement signals

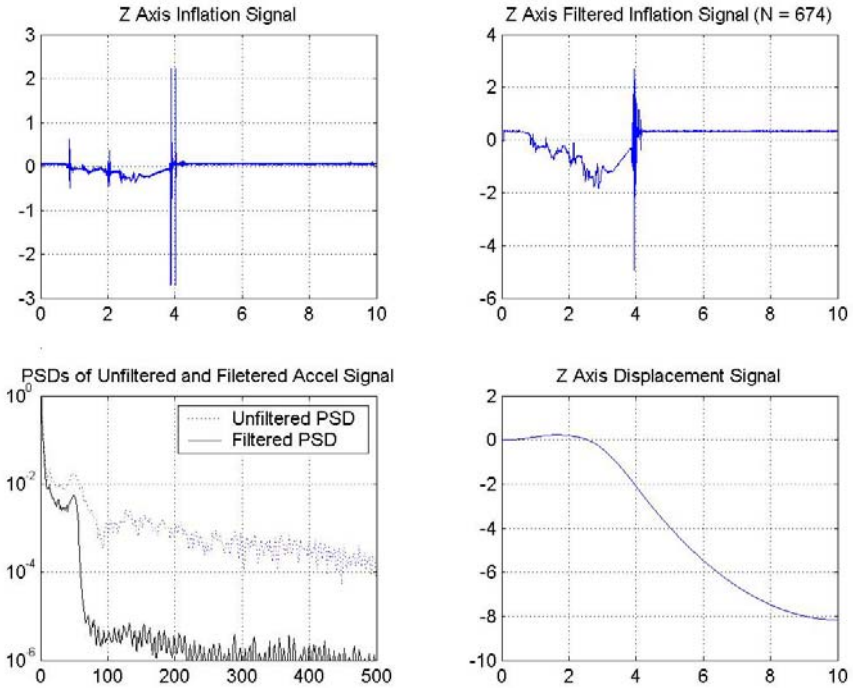



Figure 4.20: Z axis acceleration and displacement signals

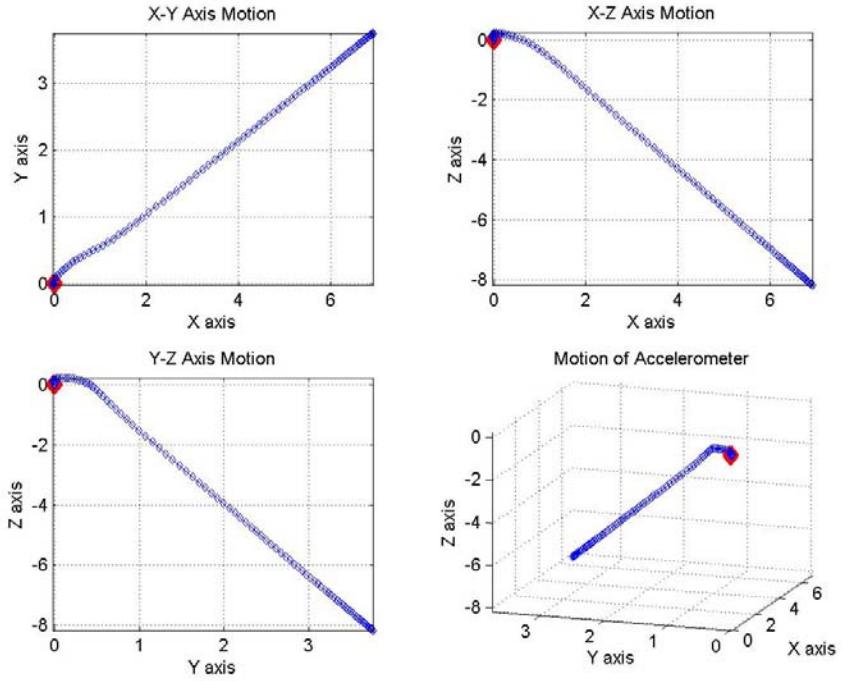



Figure 4.21: 2-D and 3-D movement plots

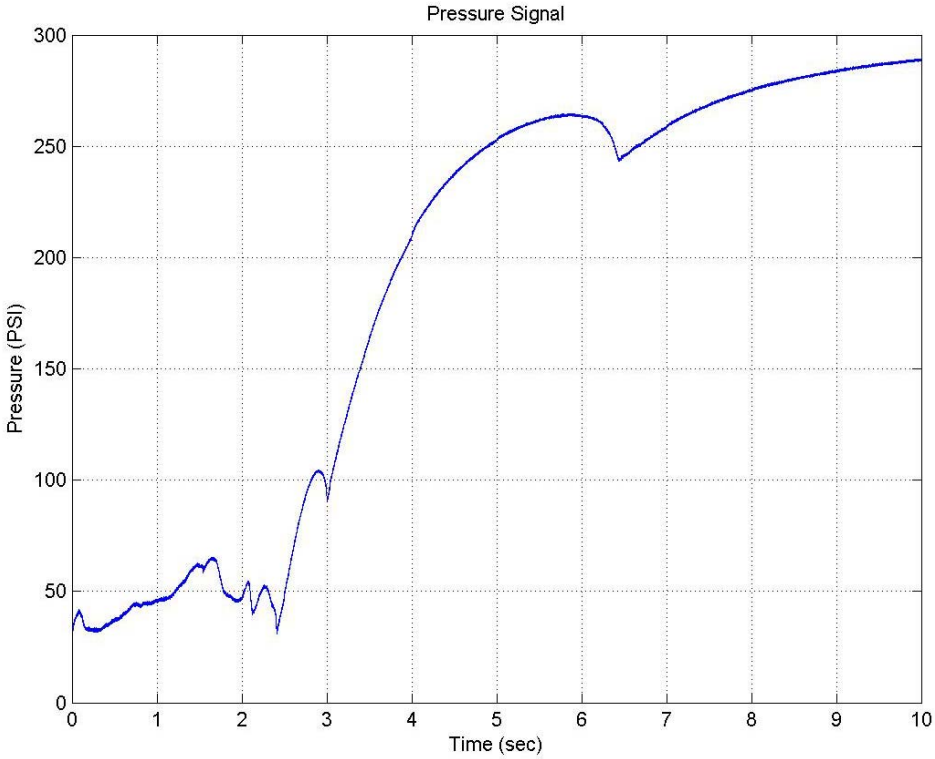



Figure 4.22: Pressure signal during inflation

*4.6.3.1 Excitation Simulation Setup* This portion of the experiment is setup the same as the final setup in Section 4.3. The only difference is the relay board is placed between the filter and the transformer. The relay is placed there to control to which tube the filtered signal will be transmitted. The excitation was conducted ten times. The ten sets of data were used to find the average transfer function and one standard deviation contours.

*4.6.3.2 Excitation Simulation Results* Figure 4.23 shows the Y axis averaged transfer function and the one standard deviation contours (dotted). The contours in the figure show that the modes will be present above the noise floor 50% of the time.

Also calculated was the accuracy of the vibration measurements. The average variance of the measured accelerations was $3.61 \cdot 10^{-4}$ volts (peak-to-peak voltage of the accelerations 200 mV).

### *4.7 Testing Imaging Computer*

This experiment is has two parts. The first part is the functionality testing of the PC/104 to take the required pictures. This test involves the validation of the image collection and determining how much time it takes for the computer to take the image from the camera into RAM and then to write the image to the secondary memory. The second portion of the image testing is to determine how accurate the height and displacement measurements are using the algorithm described in Section 3.4.

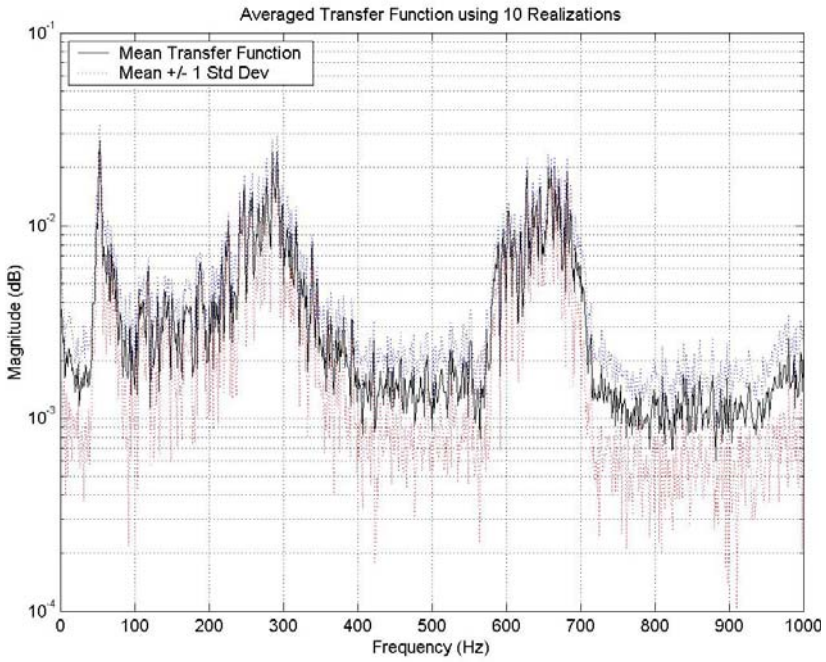



Figure 4.23: Average transfer function with 1 std. dev. contours

*4.7.1.1 Part 1 Description* The imaging program used to test the computer's ability to take images performed the following steps:

1) Declare the global variables and pixel data types.
2) Initialize the DAC onboard the image capture board
3) Set the bias and gain levels on the image capture board
4) Allocate memory buffers for a 486 x 1134 pixel image (1 byte/pixel)
5) Check for the interrupt (bit 7 on input digital port) to take a picture
6) Check tube number (bits 1 and 0) and assign address for respective camera
7) Take the image
8) Write the image to data file in ASCII

This portion of the experiment has the input digital port connected to ground and +5 V signals. Bit 7 was connected to +5 V to provide for the logic 1 to signal the imaging computer to take an image. Bit 1 was connected to ground and bit 0 was connected to +5 V. These two bits (01) notified the computer to image tube 1. The resulting image data is then imported into Matlab® to be converted into an image and interlaced.

*4.7.1.2 Part 1 Problems Encountered* The first problem encountered was getting the program to link in the DAC initialization (InitDAC, SetBiasValue, SetGainValue) and image capture (USUBCAM) subroutines provided by the manufacturer of the camera and image capture board. The subroutines were provided in *.obj* files. Sample programs were provided to show how the subroutines were to be used. The testing program was modeled after the sample programs. The initial attempt to compile and link the program

with the object files was performed in Microsoft® Visual C++ 6.0. Linking errors occurred stating the subroutines were not defined. Upon this, the manufacturer was contacted, they replied back stating that subroutines were written in 16-bit assembly code and that anything higher than Visual C++ 1.5 would not support the 16-bit coding [Email03]. It was recommended to use Watcom C/C++ Compiler.

The recommended compiler was downloaded off the Internet. The test program was compiled and linked as a 16-bit DOS program. When the program was executed using Microsoft® Windows 98, an error appeared when the image capture routine was called. To determine if the image capture subroutine (USUBCAM) was the problem, the subroutine call was commented out. The expected result should be an image of all zeros. At the completion of this test, the computer output a data file with all zeros. This confirmed that the USUBCAM routine was producing the errors. The manufacturer was contacted about the error. The reply was that the subroutine will only work in a 16-bit environment. Windows 98 and MS-DOS are all 32-bit environments. The solution to the problem was to find a 16-bit system or a windows C compiler to produce a 16-bit Windows program. Borland® Turbo C++ 4.5 is a C++ compiler made for 16-bit windows programs. The test program was compiled and linked with the object files using Turbo C++ with success. The program was then executed on the PC/104 under Windows 98 with success. The resultant data file contained all the pixel values in ASCII format. It was intuitively determined that Windows 98 was able to accommodate the 16-bit windows program because it was made for Windows whereas Windows 98 was unable to accommodate the 16-bit DOS version.

*4.7.1.3 Part 1 Results* Once the programming issues were resolved, the image data file was imported into Matlab® to be converted into an image. Copying each row and placing the copy beneath the original performed the interlacing needed to produce the required 972 x 1134 image. This interlacing is required because the 486 x 1134 image is very distorted. Any circular objects in the image would appear elliptical unless the interlacing is performed. Figure 4.24 shows the original image and Figure 4.25 shows the interlaced version. From the figures, it is apparent the interlacing is required before any image analysis can be performed. This can be seen by the circular flow valve at the top of the structure.

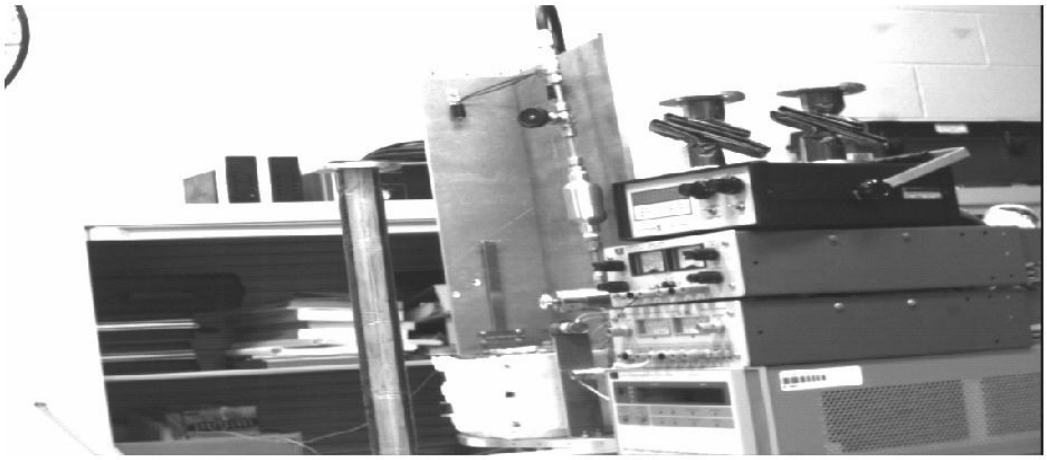


Figure 4.24: Non-interlaced test image

Figure 4.25: Interlaced test image

To determine the image capture speed and write speed of the system, the program was altered to have five image buffers. The program takes five images, places them into RAM and then reports how many seconds elapsed. The program then reports how many seconds elapsed after writing the images to file. The resulting time to take five images was 6.97 seconds, which is approximately 1.4 seconds per image. It then took 198 seconds to write the five images to file, approximately 40 seconds each. The write process may run faster when the computer is using the solid-state flash memory. From the above information, the computer could take about four images for a 5-second

inflation time. This portion of experiment validated that the computer would be able to properly take pictures during the course of the experiment.

*4.7.2.1 Part 2 Distance Test Description* To determine the height of the inflated tube, ten images were taken at thirty-six different evenly spaced distances from the camera lens. The 360 images were then imported into Matlab® where they were analyzed according to Section 3.4. The major axis of the target in each image was measured using Matlab®'s *regionprops* function giving the major axis length in number of pixels. For each of the thirty-six distances, the ten images' major axes were averaged. From the average, a ratio was formed using the known diameter of the target to the average measured pixel length. Each of the thirty-six data points were then plotted against their true distance from the lens. Once the data points were plotted, a least squares line fit was performed. Figure 4.26 shows the relationship between the distance from the lens to measured meter (target's diameter) per pixel ratio. Equation 4.7 gives the least squares line fit equation to the thirty-six data points [Pon03].

$$d = 2156.4 \cdot \frac{meter}{pixel} - 0.0241 \qquad 4.7$$

*4.7.2.2 Part 2 Distance Test Results* Once the distance equation was found, the same 360 images were analyzed individually and their distances averaged, the statistics on their deviation from the true distance was also determined. Figure 4.27 shows the results from the 360 test images. The maximum deviation was approximately 1.5 mm

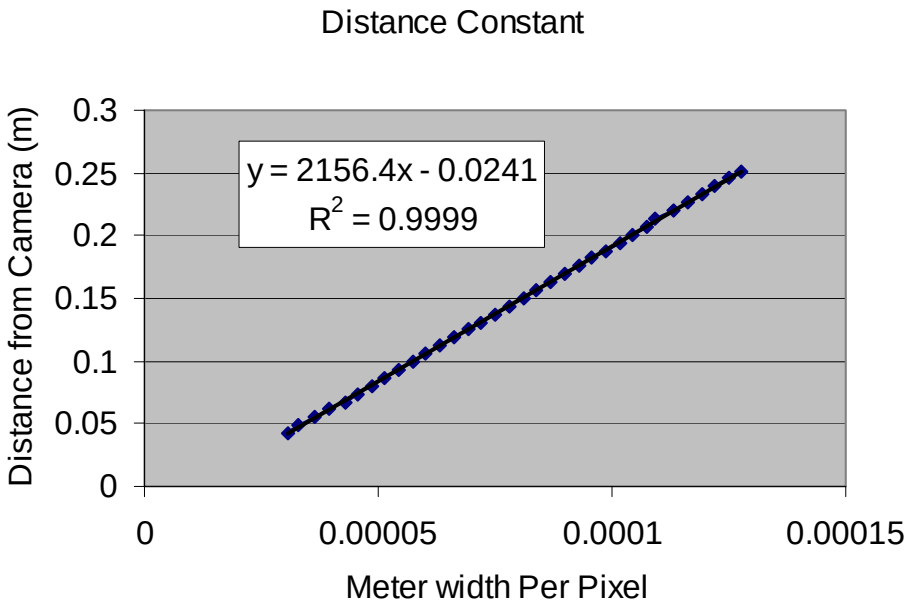



Figure 4.26: Least squares fit to relate distance from camera to meter length/pixel

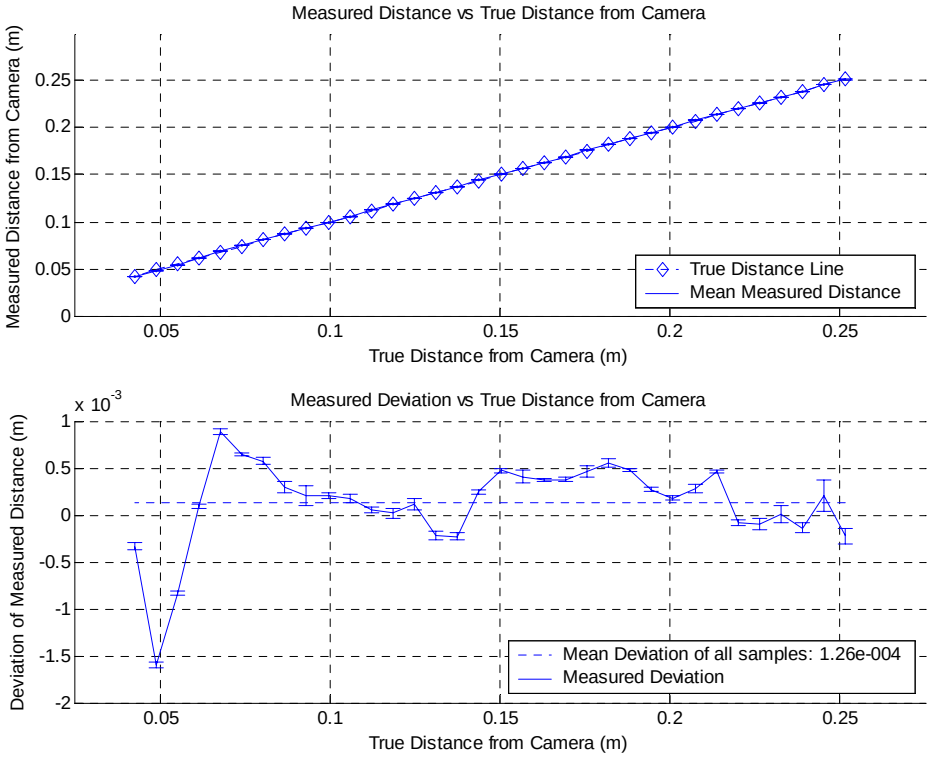



Figure 4.27: Results from test images for determining distance from camera

and an average deviation of about 0.1 mm [Pon03]. To also validate the algorithm, nine images were taken at different distances other than the thirty-six test distances and analyzed using the algorithm. Figure 4.28 displays the results from the validation set of images compared to the true distance from the camera lens. The average deviation of the nine images is 0.922 mm [Pon03]. This meets the requirement for distance measure of the imaged tubes.

*4.7.2.3 Part 2 Angle Test Description* This portion of the experiment uses the procedure described in Section 3.4.3.2. Figure 4.29 shows analysis of the peg described in the angle measurement procedure. The lines on the bottom portion of the figure are

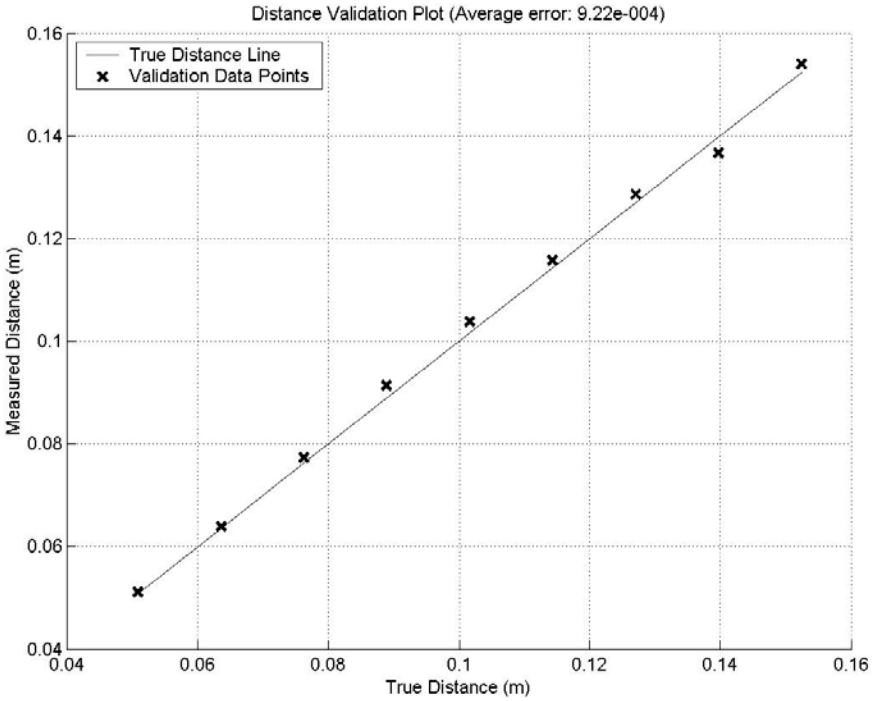



Figure 4.28: Distance measurement validation test results

representative of the eigenvectors of the edge pixel distribution. The x's are the estimated tip and base of the peg for length determination. Ten images were taken at fifteen different angles for a total of 150 test images. The tilt angles being tested ranged between 0.14º to 15.4º at a distance of 9.5 cm. The resulting angle measurements were averaged and plotted versus the true angle with the single standard deviation error bars.

*4.7.2.4 Part 2 Angle Test Results* The results from the 150 test images are shown in Figure 4.30. The average error from the true angle was determined to be 0.916º [Pon03]. To validate the process, seven images were taken at angles not measured before in the first test. The seven images were processed according to the defined algorithm and resulted in an average error of 0.09º [Pon03]. Figure 4.31 shows the validation results.

There is a problem with the error results. The error numbers appear to meet the requirement set in Chapter 3. From the figures, it is apparent the angle measurements have very large variance. Averaging several measurements together from multiple images can reduce the variance. Another way to reduce the variance is to average the images together and perform one measurement on the averaged image. Averaging the images together can help to reduce the amount of noise and possibly sharpen the images [Lim90]. Multiple images of the inflated tube can be taken during the different stages of the excitation process.

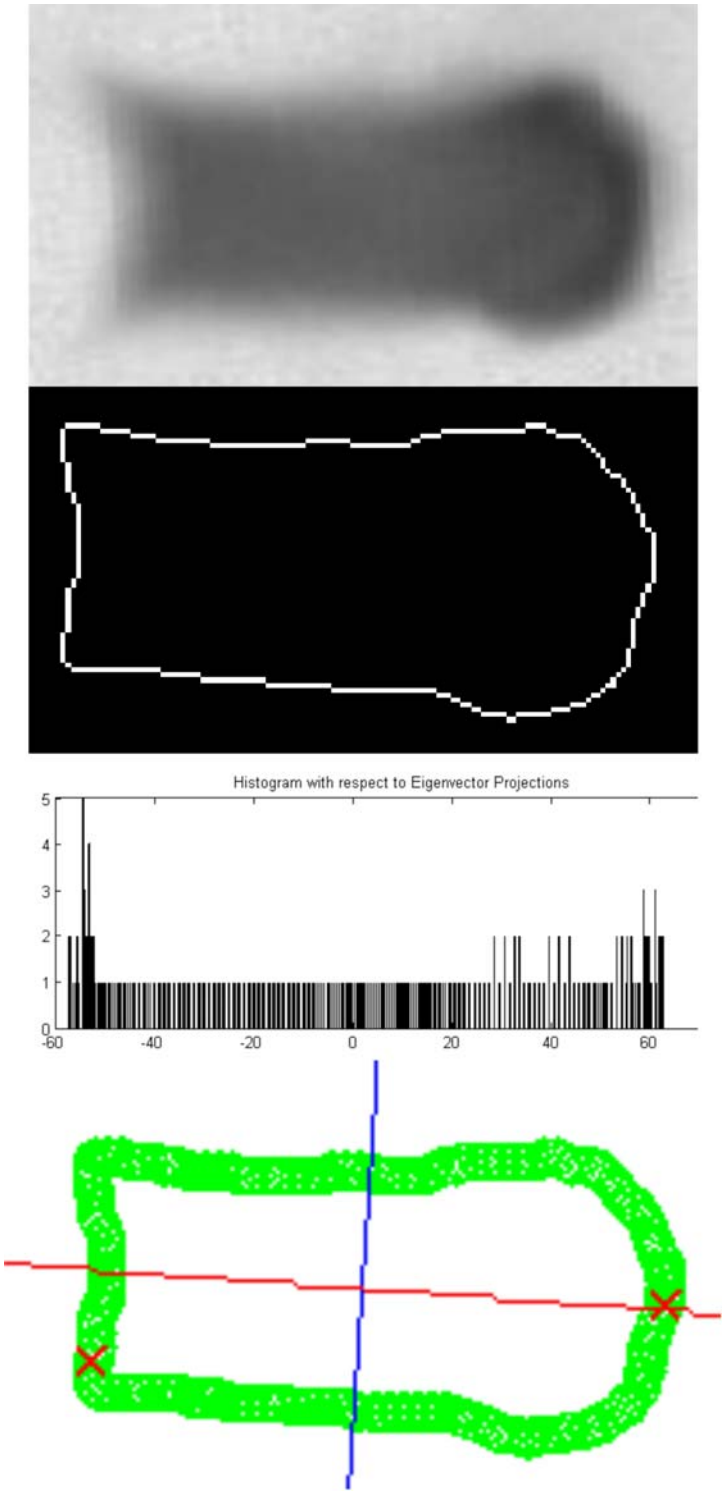



Figure 4.29: Peg analysis progression

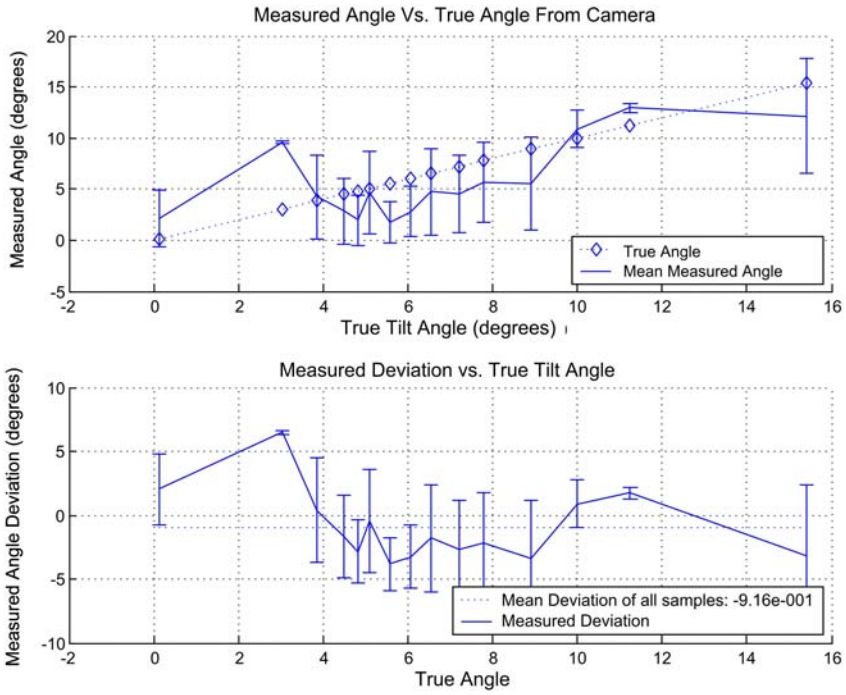



Figure 4.30: Averaged angle measurement test with error bars

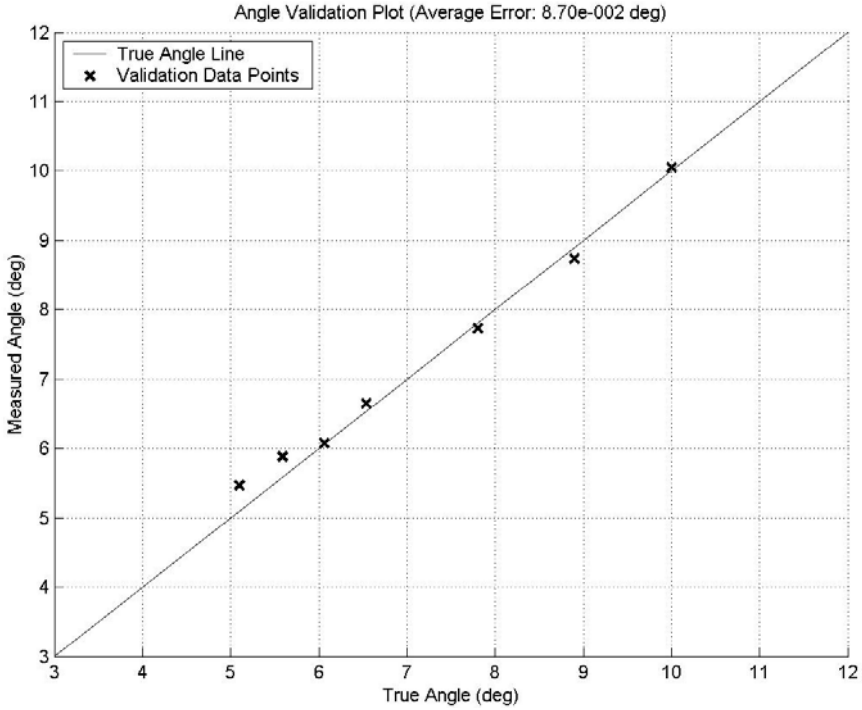



Figure 4.31: Angle measurement validation results

The second problem that occurred during the testing process dealt with shallow angles. The algorithm assumes that the peg will be long enough such that in the image the peg length will be longer than its width. For shallow angles, the length of the peg is smaller than the width of the peg. This problem leads to erroneous measurements [Pon03]. To fix this problem, the image will have to be modified by the user such that the width of the peg in the image must be reduced so that the length is longer. This processing step can be seen in Figure 4.32. Upon making this change, the angle can be more accurately measured.

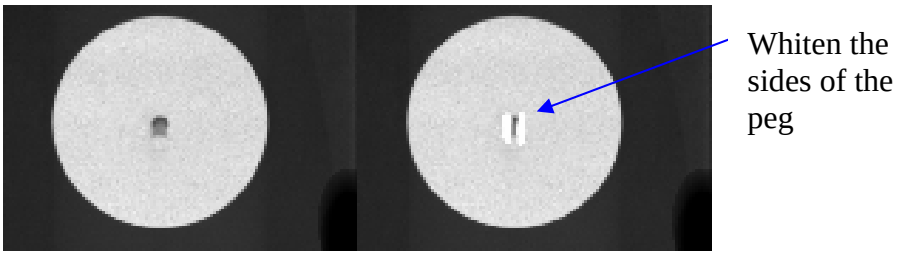



Figure 4.32: Peg width adjustment processing

### *4.8 Single Tube Test with Imaging*

This portion of the testing had to be simulated. The quarter-structure was unavailable to inflate a tube. The setup of the simulation involves many different steps to get the experiment setup. The first step involved is standing-up the imaging computer as a separate computer from the data acquisition computer. The next step is to transfer the DAC's smoothing filter from a breadboard to a PC/104 size protoboard. Once the first two steps are completed, each of the test programs are executed to make sure the

computers operate properly. Once initial testing is completed, the computers are connected together as shown in Appendix B and G and the single tube experiment programs are executed.

*4.8.1 Standing-up the Imaging and Data Computers* Two steps are involved in this setup. Each computer must have their flash disks formatted to allow an operating system to be loaded. To format the flash disks, the computers are booted using a DOS boot disk, the format command: `A:\dformat /S:doc514.exb /D:WIN[D000]` The format command will configure the flash disk to run as a FAT file system C drive when no other hard drive is present.

Once the flash disks are formatted, an operating system must be loaded. For all testing done, Windows 98® (Win98) was the chosen operating system due to availability and the capabilities of the MZ104+ processor board. Once the operating system was installed using the Win98 boot disk and installation CD, the computer's *config.sys* and *autoexec.bat* files must be modified to load the CD *.sys* driver files and *mscdex.exe* program respectively in order for the CD-ROM drive to be recognized by the computer. The CD-ROM drive is required because Win98 continually recognized a monitor attached to the computer and wanted to load the drivers during the boot process. The Win98 installation disk supposedly had the drivers for the monitor being used. When the CD-ROM was running it was discovered that the Win98 CD did not have to correct drivers so the drivers were later found on the Internet and loaded. After the video monitor drivers were resolved, Win98 discovered two ethernet controllers built into the

processor board. These drivers were located on the *MZ104+ Getting Started* CD. After all the drivers had been loaded, Win98 continued to find an "Early non-VGA device" during its boot process and wanted to load the drivers. This human-interface requirement during boot up is unwanted and cannot occur during the experiment on the shuttle. The manufacturers for the processor board and processor IC were contacted about this and no response has been received. To avoid development delays, the remainder of the experiment testing continued.

*4.8.2 Filter Board Construction* The DAC's smoothing filter was built onto a PC/104 prototype board. The filter board was constructed on a PC/104 prototype board. The resistors were placed into sockets with each socket containing half the needed resistors. The resistor sockets, filter socket and the sockets for the current buffers were connected by wire wrapping the terminals together. The connections to the board were made using three pairs of twisted wire. The first pair contained the ±5V. The second contained the input signal and ground. The third pair held the output signal and ground.

Four Intersil HA5002 current buffers were used to push the signal through the transformer to the piezo patches. The transformer drew approximately 0.6 A of current when the filter board's power was applied. Each current buffer has a maximum current rating of ±200 mA [In03].

The use of the current buffers is required to pass the filter circuit's voltage to the transformer without drawing current from the filter. Other options to replace the current

buffers would be to possibly use audio amplifiers. This portion of the filter board could use more design work even though the current buffers get the job done.

*4.8.3 Computer Handshaking* Upon initial testing of the two computers' programs separately, the programs executed as to design. The handshaking operation between the two computers was conducted using five wire connections between the computers' digital I/O ports on the counter boards and the data computer's A/D board. The handshaking process follows the pattern (pin numbers are for the respective board's 50 pin connector):

- Data computer addresses the tube to be imaged on its auxiliary digital output pins in binary (pins 43 and 44). These two pins connect to bits 1 and 0 of the imaging computer's counter board digital input port (pins 46 and 48).
- Data computer checks to see if the imaging computer is ready by reading its counter board's digital input bit 7 (pin 34) to see if it is logical 1 (high). This signal line is connected to the imaging computer's counter board digital output bit 7 (pin 33).
- If the imaging computer is ready, the data computer then signals the imaging computer to take a picture by placing a logical 1 on its bit 7 pin of its counter board digital output port (pin 33). This signal line is connected to the imaging computer's counter board digital input port bit 7 (pin 34).

- The imaging computer begins its imaging routine by dropping its output bit 7 low.

- Once the imaging routine is completed, the imaging computer checks to see if the experiment is completed by looking for logic 1 on bit 6 of its digital input port (pin 36). This signal connection is connected to the data computer's counter board digital output bit 6 (pin 35). If the signal is high, the imaging computer exits the imaging program. If the signal is low, the imaging computer will signal that it is ready and the process repeats as needed.

Appendix D shows where the handshaking process takes place in the experiment routine.

*4.8.4 Computer Integration Results* The single tube inflation routine was executed along with the imaging program. The single tube inflation routine implemented the routines described in Appendix C for only one tube. The heating was simulated using an Omega Process Calibration device set to source Type K T/C. The two computers were connected together as described in Section 4.8.3.

The experiment was simulated ten times with both computers running their respective programs. The handshaking worked according to design. Figure 4.33 shows an example heating curve as collected from the T/C A/D board for one of the simulations. Figures 4.34 and 4.35 show two example transfer functions estimated from two of the trial runs of the experiment. The strength of the transfer functions is lower than previous results of the tube excitation (Figure 4.6 and 4.8). The previous results had the filter

buffers going directly into the transformer. The input to the transformer is approximately 1 Ω.

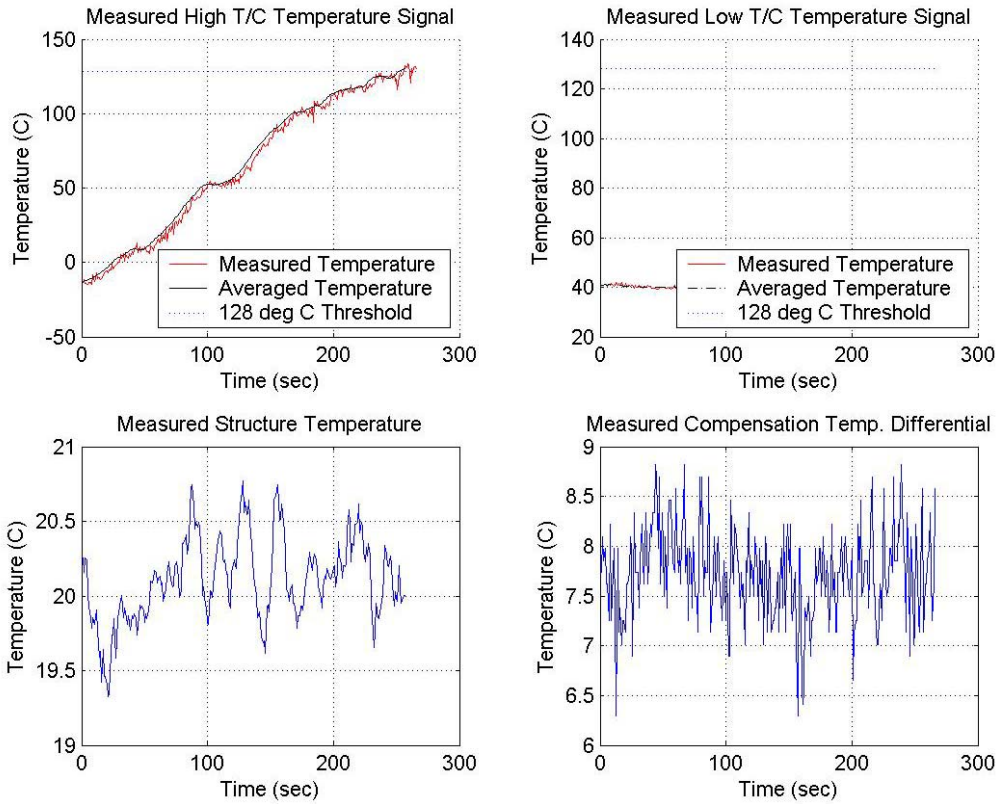



Figure 4.33: Temperature curve from simulated heating using process calibrator

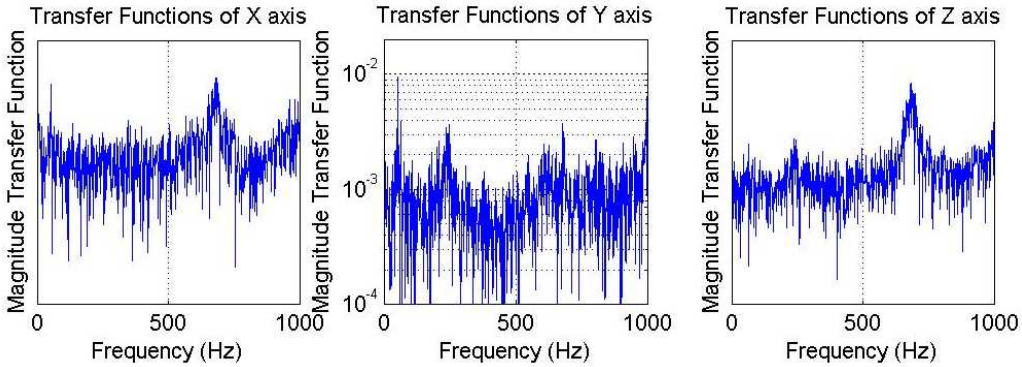



Figure 4.34: Example transfer function estimation from tube excitation

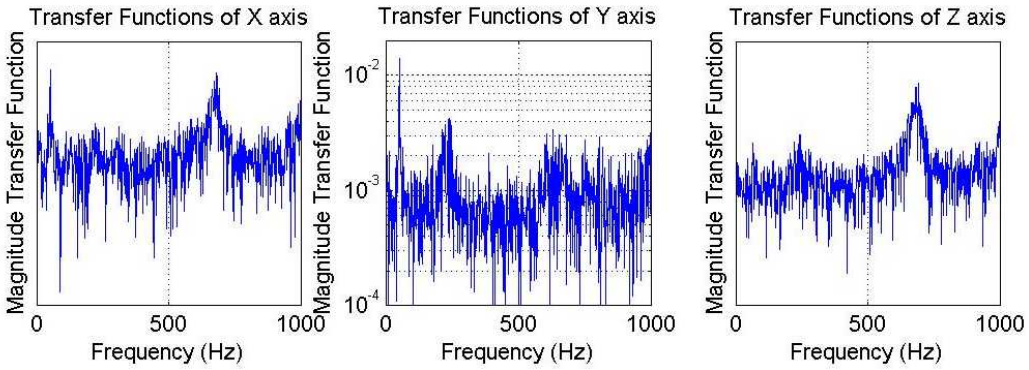



Figure 4.35: Example transfer function estimation from tube excitation

During the experiment, the filtered excitation waveform will be routed through relays to switch the signal to the appropriate transformer/tube, as can be seen in Figure 4.36. The resistance of the relay is near 0.1 Ω. Because the relay is close to the resistance of the transformer, a partial amount of the power formally being delivered to the transformer is now being wasted in the relay. This reduces the strength of the waveform when it finally gets to the tube. This reduced strength is unavoidable because relays provide the lowest resistive switching. The only alternative is to have a separate filter board per tube but space may limit this option. The averaged transfer function from the ten trials can be seen in Figure 4.37.

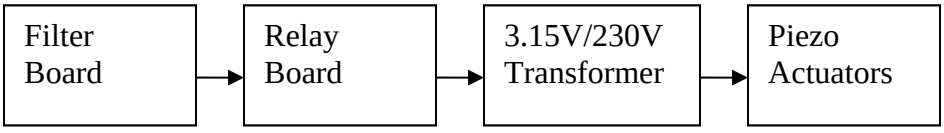



Figure 4.36: Excitation block diagram

The imaging computer took pictures as commanded by the data computer. It was instructed three times to take pictures and produced fifteen images as programmed (five images per notification). The program captured five images in an average of seven seconds and wrote individual sets of image data to file in an average of sixty seconds. The above heating curve and transfer functions validate that the two computer systems are working properly.

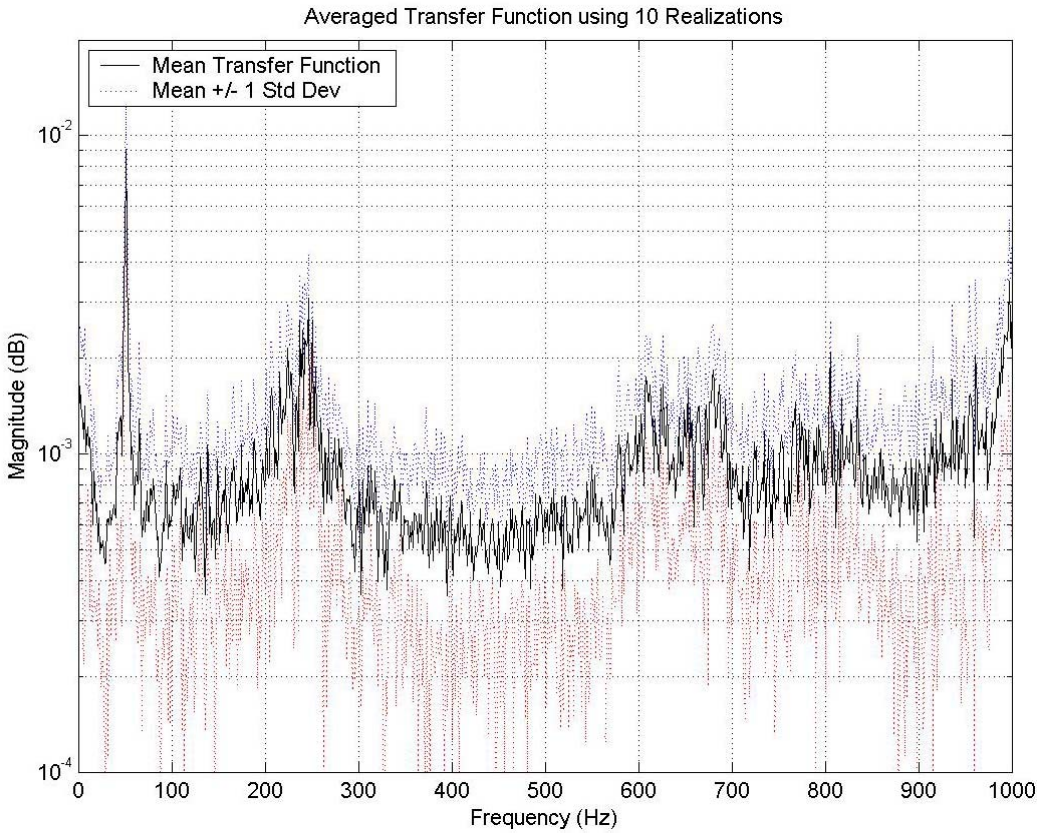



Figure 4.37: Average transfer function plot from 10 single tube integration trials

### *4.9 Three Tube Experiment*

Due to the time constraints and delays associated with the RIGEX support structure, the fully assembled system could not be tested. The three tube data collection routine was tested in the same manner as the single tube test routine. The tube heating was simulated with the Omega Process Calibrator as in previous experiments. The purpose of this test is to fully simulate the entire experiment and make sure the computers switch between tubes appropriately and operate correctly. Since no tube was inflated, there was no valid inflation data collected.

*4.9.1 Heating Simulation* For each of the three tubes, the Omega Process Calibrator was connected to the tube's "high" channel input. The temperature was raised from 0ºC to 130ºC in about eighty seconds. Each time the temperature reached the threshold of 130ºC, the computer properly switched over to the next step in the experiment. The data results in Figure 4.38 shows the heating curves for each tube's high channel.

*4.9.2 Transfer Function Results* The experiment was able to properly determine the transfer function of the tube being excited. Each time the excitation process was performed, the computer was able to collect real vibration data. Figure 4.39 shows the transfer functions collected during each tube's respective excitation process. The X axis transfer function shows the 1$^{st}$ and 3$^{rd}$ mode. The Y axis transfer function shows the 1$^{st}$ and 2$^{nd}$ modes and is also the primary axis being excited. The Z axis transfer function

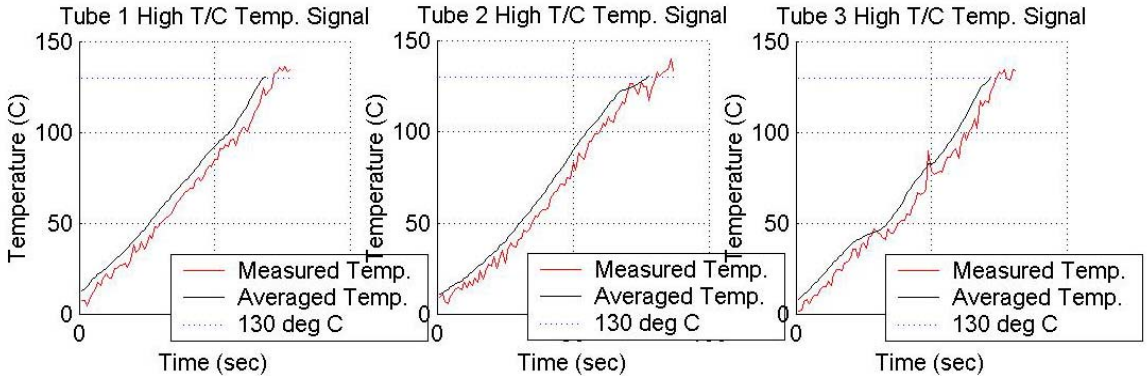



Figure 4.38: Three tube experiment temperature simulations

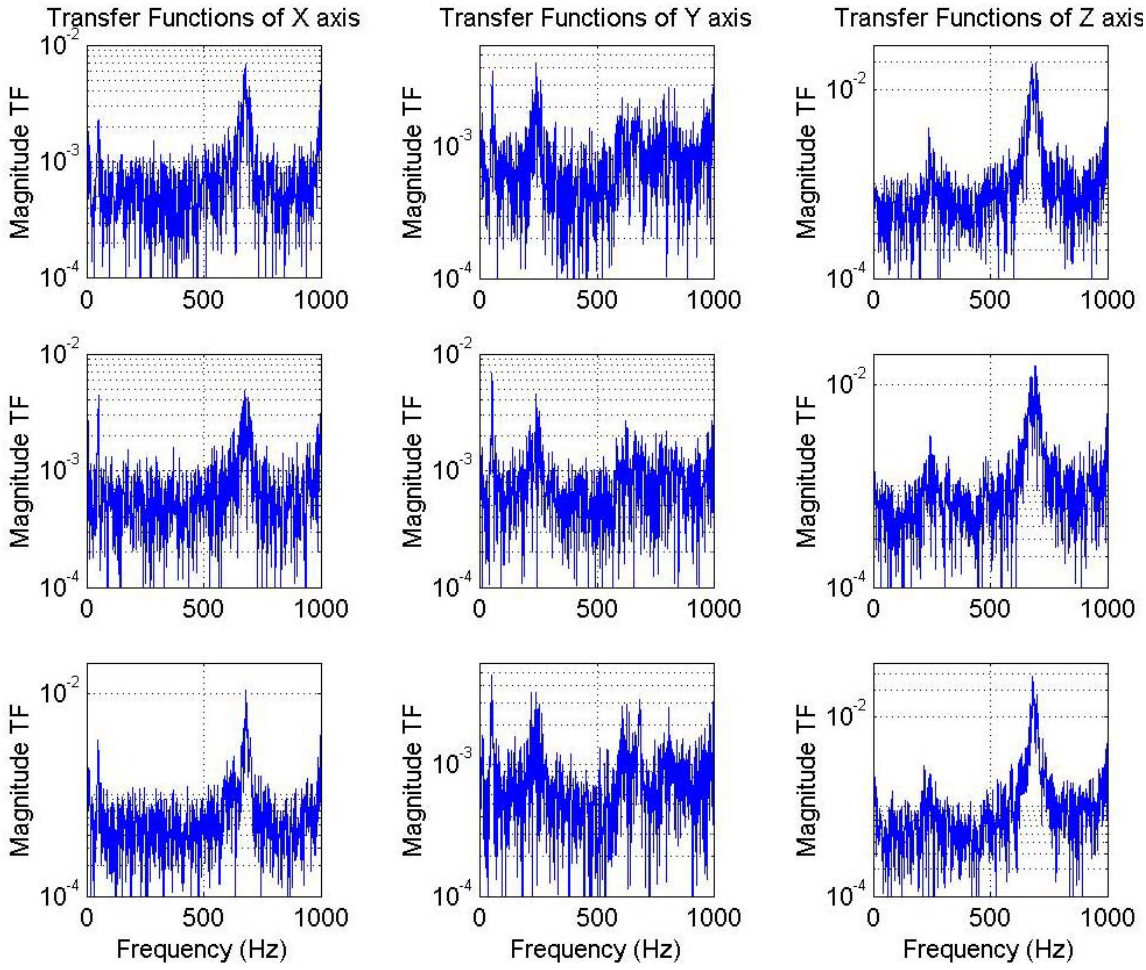



Figure 4.39: Three tube simulated experiment transfer function results

shows the 2nd and 3rd modes. It can be seen, as before in Section 4.8.4, the transfer functions are not as strong when the signal has to travel through the relays.

*4.9.3 Camera System* The other important aspect of this simulated experiment is to verify if the imaging computer would properly select which tube to image based on the commands of the data computer. The imaging computer worked as expected. Figures 4.40 through 4.42 show a test image taken by each tube's respective camera system. The white markings on the side of the image are due to a three foot ribbon cable connecting the camera to the camera board on the imaging computer. The markings will not show when the extension cable is not present as in the case shown in Figure 4.25.

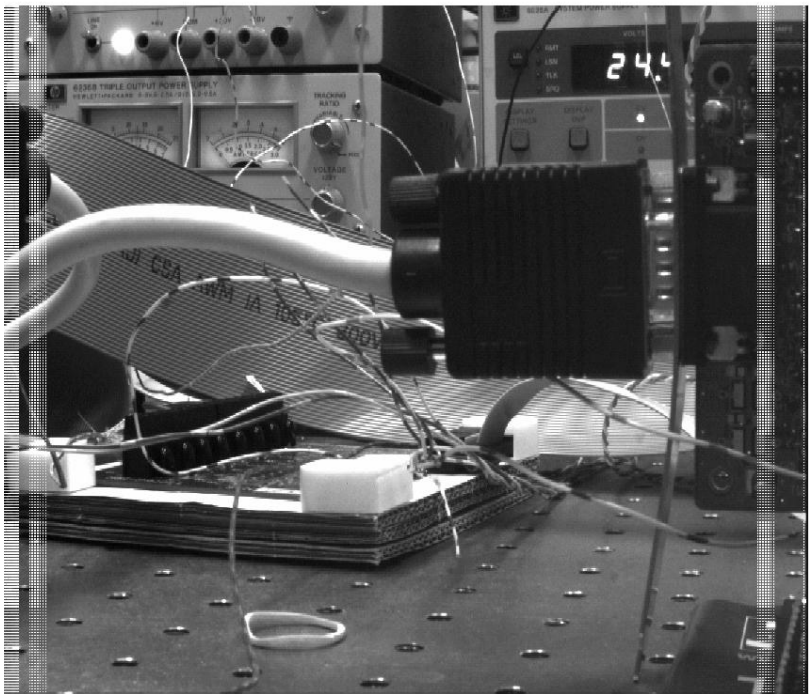


Figure 4.40: Camera 1 test image

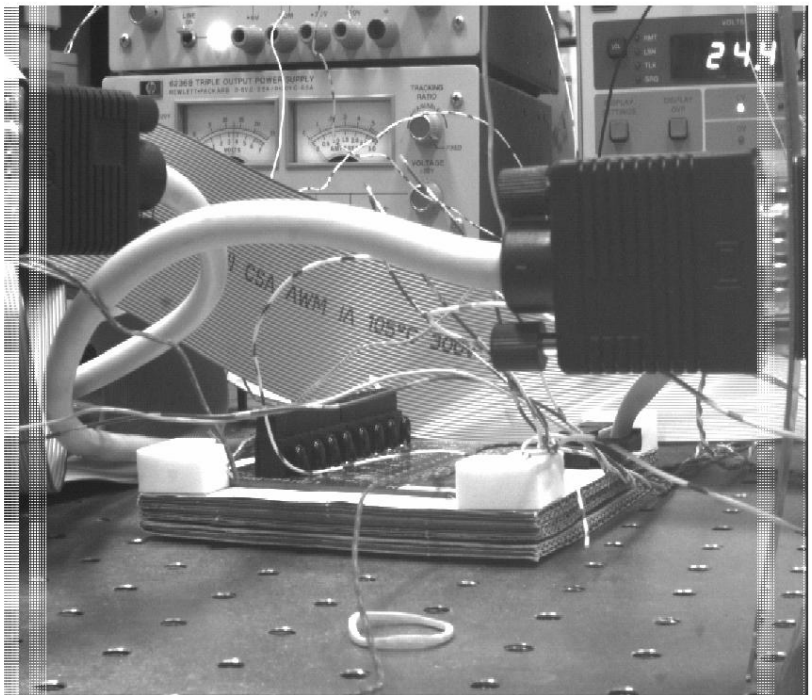


Figure 4.41: Camera 2 test image

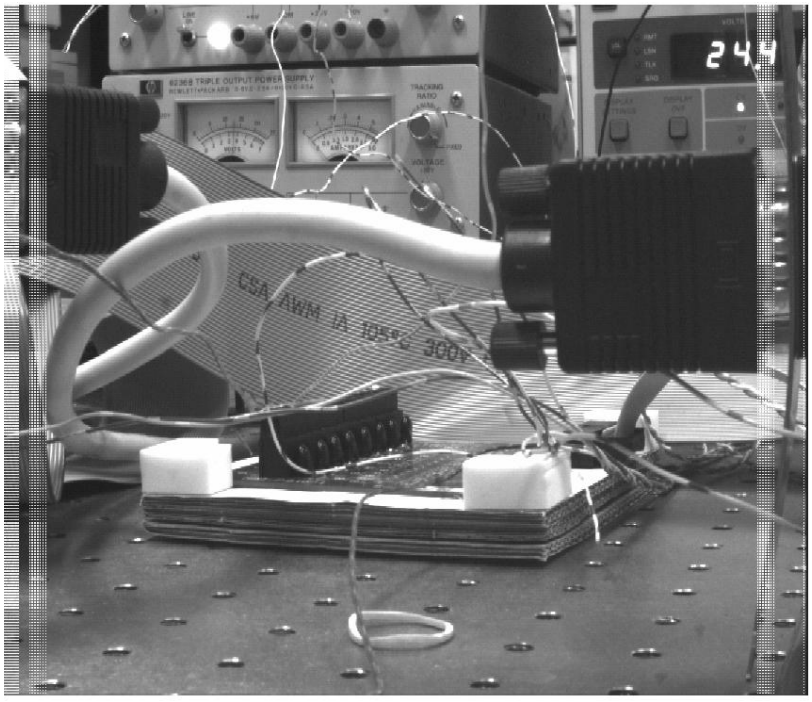


Figure 4.42: Camera 3 test image

*4.9.4 Failsafe System* The failsafe system built into the data acquisition computer was also tested. The program was cut off just before program label Tube22. The failsafe data file had twenty-two recorded as its last point achieved. When the program was restarted the computer immediately began where it left off (imaging before excitation). This validated the failsafe operation in the three tube experiment process.

During the testing of the data computer's failsafe system, it was observed that if the imaging computer was restarted, the previous images would be overwritten. This required a failsafe system to be developed for the imaging computer. To do this, a data file called *im_fail.dat* was created that had three numbers in it (0 0 0 for start of experiment). When the program starts, it will load the three numbers in the data file as its image counters. The image counters are incremented each time the computer is notified to image a particular tube. When the program finishes writing a set of images to file, it will update the *im_fail.dat* file with the current image counter values. This process facilitates a failsafe operation for the imaging computer. The above process was tested and proved to work correctly. It also provides a method for determining how well the experiment performed when it is recovered after the space flight.

## *4.10 Summary*

This chapter has shown the results of the implementation and testing of the RIGEX computer systems. The data collected during the experiments validate that the computers are performing their mission. The Matlab® scripts used to analyze the ASCII

data files are located in Appendix E. The source code for the data computer and imaging computer is located in Appendix F.

# V. Conclusions and Recommendations

## *5.1 Summary of Results*

This thesis has presented a two computer system design to support the RIGEX effort. The computer system consists of a data acquisition computer which is responsible for the control and execution of the experiment as well as the collection of temperature, pressure, and vibration data. The second computer system is responsible for the collection of image data.

*5.1.1 Experiment Control* Through the use of the relay board and the T/C A/D board, the data acquisition computer has demonstrated that the experiment can be controlled successfully. The failsafe system designed will also allow the computer system to restart where the experiment left off in case the experiment is shutdown and restarted.

*5.1.2 Temperature Data Collection* The results in Chapter 4 shows the temperature data collected by the MSI-P440-K T/C A/D board tends to have high variance. To mitigate the variance, a 10-point moving average filter was implemented to smooth the temperature signal for thresholding purposes. The averaging filter performed very well and adequately represented the true temperature. It was also required to have channel 7 of the board shorted. This was required to compensate for the heating of the T/C A/D board. The difference between the board's temperature and 21ºC was used as a differential to unbias the temperature measurements.

*5.1.3 Tube Inflation* The data computer was able to sample the pressure sensors and the accelerometers during inflation. The data collected was then used to provide plots showing how the pressure in the tube changed during inflation. The acceleration signals were then used to attempt to model the movement of the accelerometer during inflation. Unfortunately, the accelerometer was only able to measure three of the six degrees of freedom. The rotational movements of the tube's top plate could not be measured. This introduced errors in the 3-D movement plots. The inflation acceleration signals might provide information/data in future tube modeling efforts.

*5.1.4 Tube Excitation* One of the critical requirements of the experiment is to measure the modal characteristics of the tubes after inflation. To do this, a chirp signal ranging in frequency from 5 - 1000 Hz was used to excite the tubes with piezo patches. The waveform was digitally produced using a DAC and then smoothed. The piezo patches required higher voltage than what could be produced by the filter board. To raise the voltage of the signal, a step-up transformer (coil ratio approximately 1:73) was used. The transformer was able to excite the tubes well enough to collect vibration data. The data was then digitally filtered to reduce noise and then used to estimate the transfer function of the inflated tube.

*5.1.5 Imaging System* The imaging computer was programmed to take five images each time it was notified by the data acquisition computer. The image data then

took an average of 60 seconds per image to be written to file. The handshaking process successfully worked between the computers, preventing one computer from moving ahead of the other.

A second failsafe system was designed for the imaging computer. The system prevented the computer from overwriting previous images upon restarting.

*5.1.6 Data files* All data files produced by the experiment are ASCII text files. The Matlab® scripts in Appendix E can be used to analyze all the data files.

### *5.2 Recommendations*

The following recommendations help further the development of the RIGEX computer systems and show some areas for improvement.

*5.2.1 Filter Board Improvement* The filter board currently uses four HA5002 current buffers. The current buffers are used to pass the voltage signal of the output of the filter to the transformer without drawing current from the filter. The function that the current buffers provide could be redesigned. The possible use of audio amplifiers instead of current buffers is an option. The signals being sent are in the audio frequency range and the audio amplifiers are used to power low resistive loads. The only limitation for the area of voltage amplification is the limited power supplies available (±5V and ±12V).

*5.2.2 Computer Operating Systems* Currently the computers use Windows 98®. When the computers boot up, Win98 will "find" a device on the processor board that it calls a "PCI Early Non-VGA device" and want to load drivers for it. This requires someone at the keyboard to hit the escape key or click cancel. The manufacturers for the board (Tri-M) and the processor (ZF Micro-Devices) were contacted and neither knew why Win98 was finding the device. It is recommended that a new operating system be found to run the computers. Possible choices include Windows 95® or Windows CE®.

*5.2.3 Environmental Heating* During the development of the computer systems, the environmental heating was not worked on. It is recommended to test the Thermostat® heater controllers before use. The controllers are made to resistively match the heater that is attached to it. There is a high probability that the heaters on ordered for environmental heating do not match the present RIGEX heaters. The controllers need to be tested in a refrigerated environment to determine the temperature range that each controller can handle for each heater use for the environment control. Each controller can only go 20% above and below the rated operational temperature.

*5.2.4 Lighting System* It is recommended the lighting system be tested and to make sure the designed circuit in Figure 3.7 provides adequate lighting and does not draw too much current.

*5.2.5 Tests* It is recommended the RIGEX system be tested executing the entire experiment in a climate controlled environment. This testing would allow a little more realistic scenario of the experiment. It will also point out any problems that may occur with the batteries or electronics due to the cold.

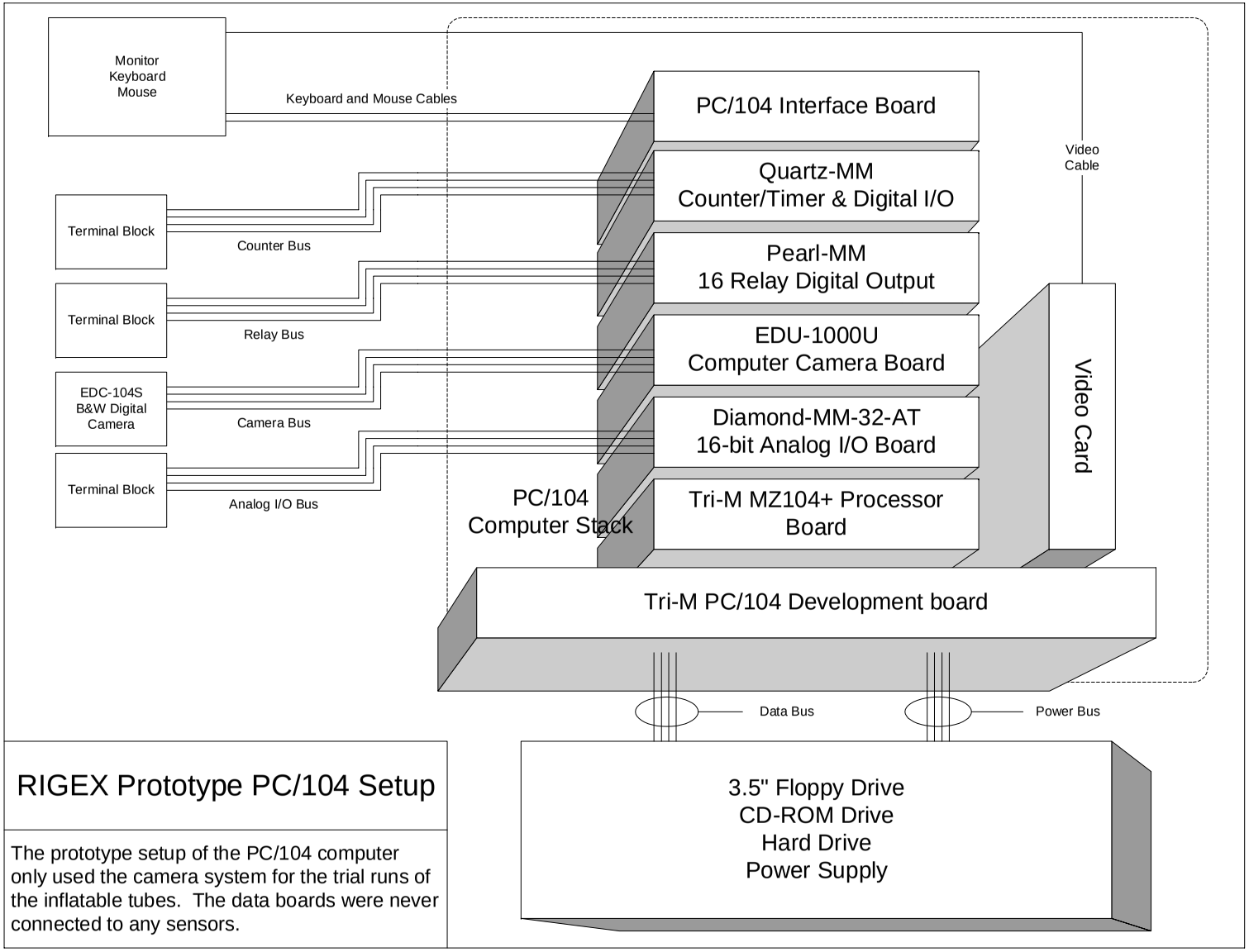
*5.2.6 Optics* Another area that could be improved upon is the imaging system. The cameras currently being used are slow to taking pictures. An option that could possibly be investigated is the use of a commercial digital camera made with some sort of computer interface. The commercial digital cameras have the capability of taking quick pictures and video. They also have their own flash memory storage. This option would reduce the amount of computing required (removal altogether of the imaging computer).

If the current cameras cannot be replaced, it might be feasible to find some shorter lenses. Presently, a fully inflated tube with accelerometer and target comes to less than an inch from the lens of the camera. The short spacing does not allow for the camera image to be focused.

### *5.3 Conclusion*

The RIGEX computer systems are an integral part of the research effort. Upon redesign of some parts and final testing, the computer systems will be ready for launch. The temperature, pressure, vibration and image data collected by the computers will provide insight into the development of future rigidized inflatable structures. The future of lighter/larger space structures starts here.

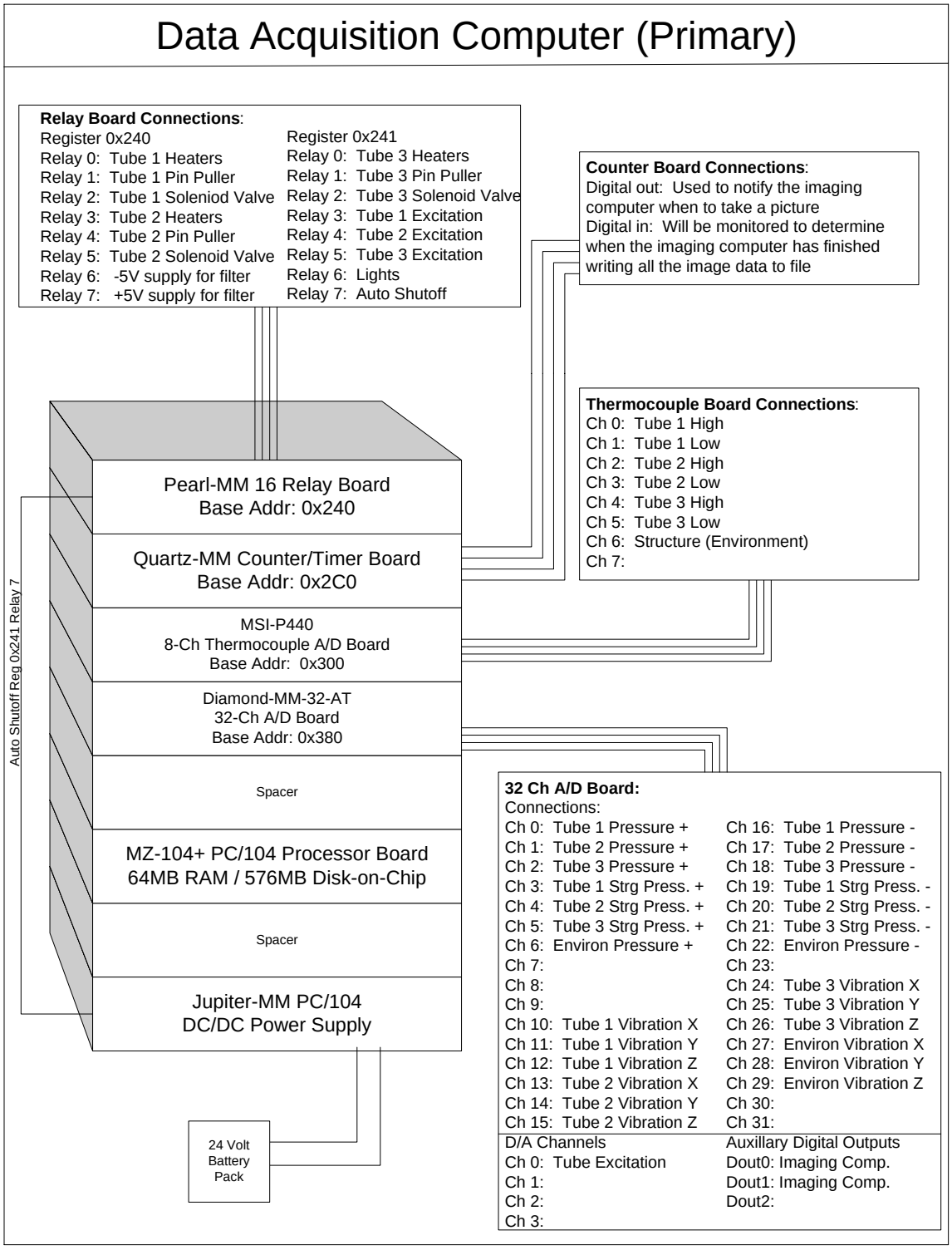
# Appendix A:  Philley's Prototype Computer

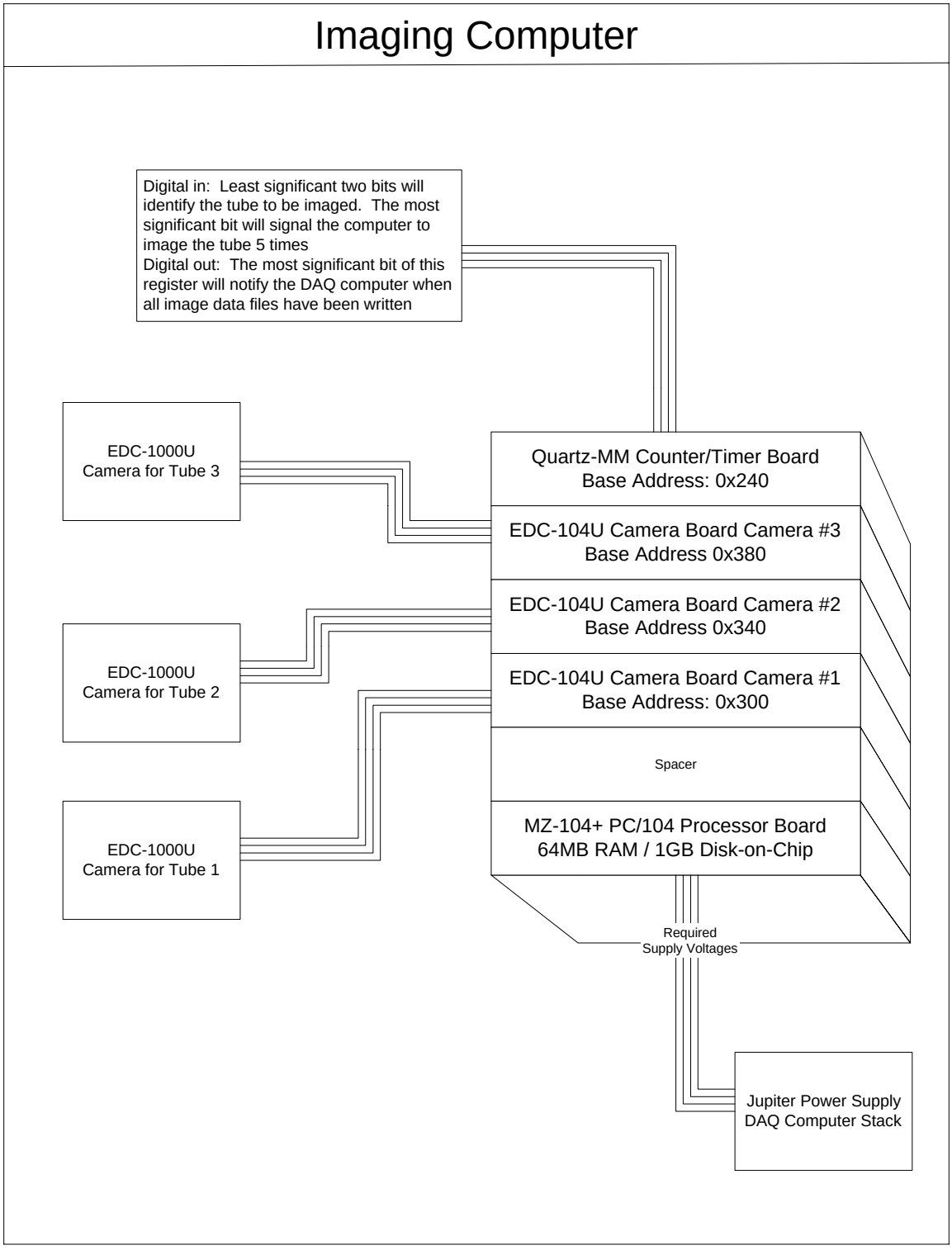



## RIGEX Prototype PC/104 Setup

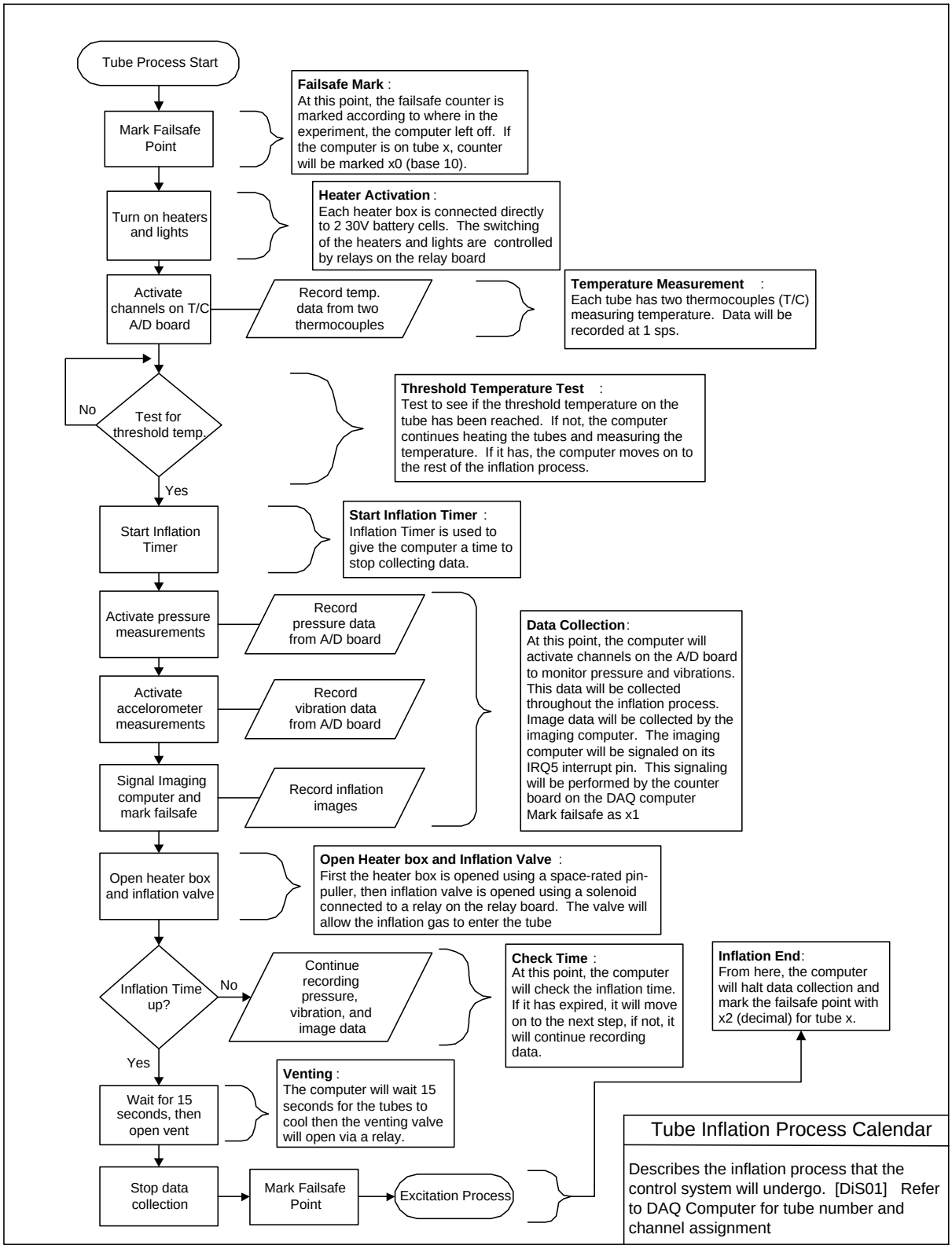
The prototype setup of the PC/104 computer only used the camera system for the trial runs of the inflatable tubes. The data boards were never connected to any sensors.

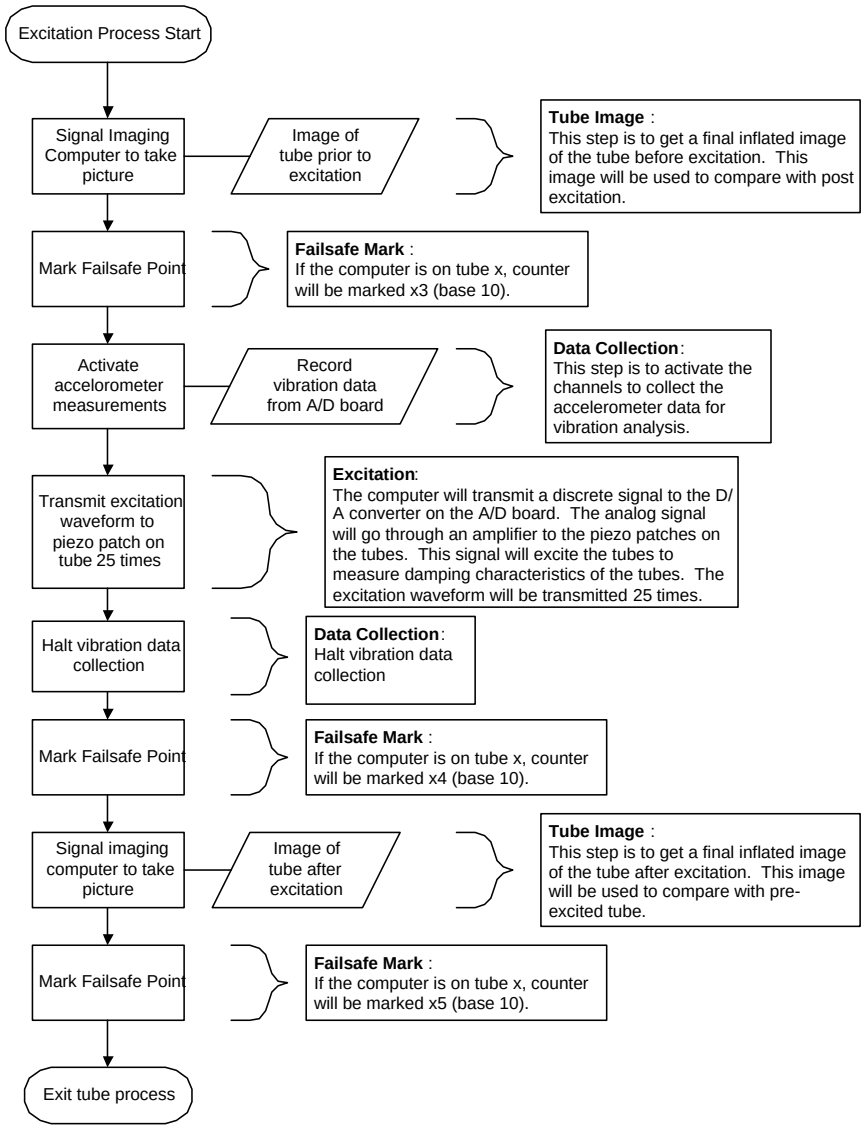
# Appendix B: Computer Channel Assignment


Data Acquisition Computer (Primary)
Relay Board Connections:
Register 0x240
Relay 0: Tube 1 Heaters
Relay 1: Tube 1 Pin Puller
Relay 2: Tube 1 Soleniod Valve
Relay 3: Tube 2 Heaters
Relay 4: Tube 2 Pin Puller
Relay 5: Tube 2 Solenoid Valve
Relay 6: -5V supply for filter
Relay 7: +5V supply for filter
Register 0x241
Relay 0: Tube 3 Heaters
Relay 1: Tube 3 Pin Puller
Relay 2: Tube 3 Solenoid Valve
Relay 3: Tube 1 Excitation
Relay 4: Tube 2 Excitation
Relay 5: Tube 3 Excitation
Relay 6: Lights
Relay 7: Auto Shutoff
Counter Board Connections:
Digital out: Used to notify the imaging computer when to take a picture
Digital in: Will be monitored to determine when the imaging computer has finished writing all the image data to file
Thermocouple Board Connections:
Ch 0: Tube 1 High
Ch 1: Tube 1 Low
Ch 2: Tube 2 High
Ch 3: Tube 2 Low
Ch 4: Tube 3 High
Ch 5: Tube 3 Low
Ch 6: Structure (Environment)
Ch 7:
Pearl-MM 16 Relay Board
Base Addr: 0x240
Quartz-MM Counter/Timer Board
Base Addr: 0x2C0
MSI-P440
8-Ch Thermocouple A/D Board
Base Addr: 0x300
Diamond-MM-32-AT
32-Ch A/D Board
Base Addr: 0x380
Spacer
MZ-104+ PC/104 Processor Board
64MB RAM / 576MB Disk-on-Chip
Spacer
Jupiter-MM PC/104
DC/DC Power Supply
Auto Shutoff Reg 0x241 Relay 7
24 Volt Battery Pack
32 Ch A/D Board:
Connections:
Ch 0: Tube 1 Pressure +
Ch 1: Tube 2 Pressure +
Ch 2: Tube 3 Pressure +
Ch 3: Tube 1 Strg Press. +
Ch 4: Tube 2 Strg Press. +
Ch 5: Tube 3 Strg Press. +
Ch 6: Environ Pressure +
Ch 7:
Ch 8:
Ch 9:
Ch 10: Tube 1 Vibration X
Ch 11: Tube 1 Vibration Y
Ch 12: Tube 1 Vibration Z
Ch 13: Tube 2 Vibration X
Ch 14: Tube 2 Vibration Y
Ch 15: Tube 2 Vibration Z
Ch 16: Tube 1 Pressure -
Ch 17: Tube 2 Pressure -
Ch 18: Tube 3 Pressure -
Ch 19: Tube 1 Strg Press. -
Ch 20: Tube 2 Strg Press. -
Ch 21: Tube 3 Strg Press. -
Ch 22: Environ Pressure -
Ch 23:
Ch 24: Tube 3 Vibration X
Ch 25: Tube 3 Vibration Y
Ch 26: Tube 3 Vibration Z
Ch 27: Environ Vibration X
Ch 28: Environ Vibration Y
Ch 29: Environ Vibration Z
Ch 30:
Ch 31:
D/A Channels
Ch 0: Tube Excitation
Ch 1:
Ch 2:
Ch 3:
Auxillary Digital Outputs
Dout0: Imaging Comp.
Dout1: Imaging Comp.
Dout2:


Imaging Computer
Digital in: Least significant two bits will identify the tube to be imaged. The most significant bit will signal the computer to image the tube 5 times
Digital out: The most significant bit of this register will notify the DAQ computer when all image data files have been written
EDC-1000U Camera for Tube 3
EDC-1000U Camera for Tube 2
EDC-1000U Camera for Tube 1
Quartz-MM Counter/Timer Board Base Address: 0x240
EDC-104U Camera Board Camera #3 Base Address 0x380
EDC-104U Camera Board Camera #2 Base Address 0x340
EDC-104U Camera Board Camera #1 Base Address: 0x300
Spacer
MZ-104+ PC/104 Processor Board 64MB RAM / 1GB Disk-on-Chip
Required Supply Voltages
Jupiter Power Supply DAQ Computer Stack

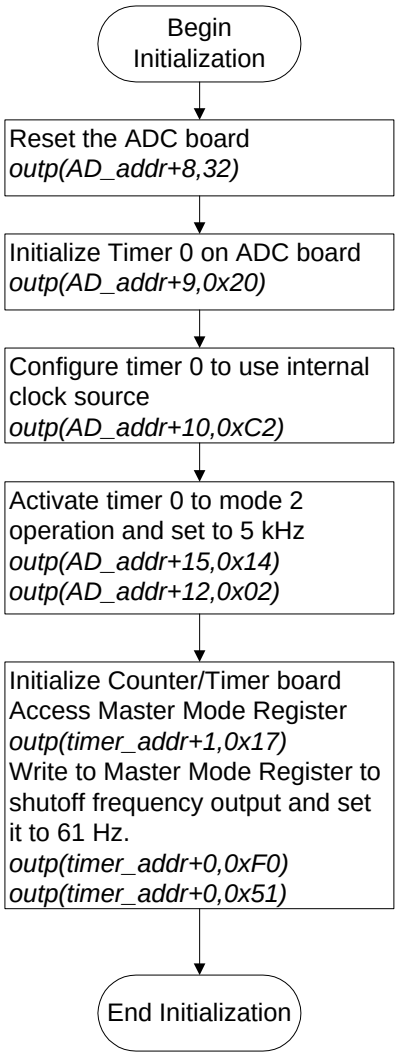
# Appendix C: Inflation/Excitation Routines


Tube Process Start
Mark Failsafe Point
Failsafe Mark :
At this point, the failsafe counter is marked according to where in the experiment, the computer left off. If the computer is on tube x, counter will be marked x0 (base 10).
Turn on heaters and lights
Heater Activation :
Each heater box is connected directly to 2 30V battery cells. The switching of the heaters and lights are controlled by relays on the relay board
Activate channels on T/C A/D board
Record temp. data from two thermocouples
Temperature Measurement :
Each tube has two thermocouples (T/C) measuring temperature. Data will be recorded at 1 sps.
No
Test for threshold temp.
Threshold Temperature Test :
Test to see if the threshold temperature on the tube has been reached. If not, the computer continues heating the tubes and measuring the temperature. If it has, the computer moves on to the rest of the inflation process.
Yes
Start Inflation Timer
Start Inflation Timer :
Inflation Timer is used to give the computer a time to stop collecting data.
Activate pressure measurements
Record pressure data from A/D board
Activate accelorometer measurements
Record vibration data from A/D board
Signal Imaging computer and mark failsafe
Record inflation images
Data Collection:
At this point, the computer will activate channels on the A/D board to monitor pressure and vibrations. This data will be collected throughout the inflation process. Image data will be collected by the imaging computer. The imaging computer will be signaled on its IRQ5 interrupt pin. This signaling will be performed by the counter board on the DAQ computer
Mark failsafe as x1
Open heater box and inflation valve
Open Heater box and Inflation Valve :
First the heater box is opened using a space-rated pin-puller, then inflation valve is opened using a solenoid connected to a relay on the relay board. The valve will allow the inflation gas to enter the tube
Inflation Time up?
No
Continue recording pressure, vibration, and image data
Check Time :
At this point, the computer will check the inflation time. If it has expired, it will move on to the next step, if not, it will continue recording data.
Inflation End:
From here, the computer will halt data collection and mark the failsafe point with x2 (decimal) for tube x.
Yes
Wait for 15 seconds, then open vent
Venting :
The computer will wait 15 seconds for the tubes to cool then the venting valve will open via a relay.
Stop data collection
Mark Failsafe Point
Excitation Process
Tube Inflation Process Calendar
Describes the inflation process that the control system will undergo. [DiS01] Refer to DAQ Computer for tube number and channel assignment

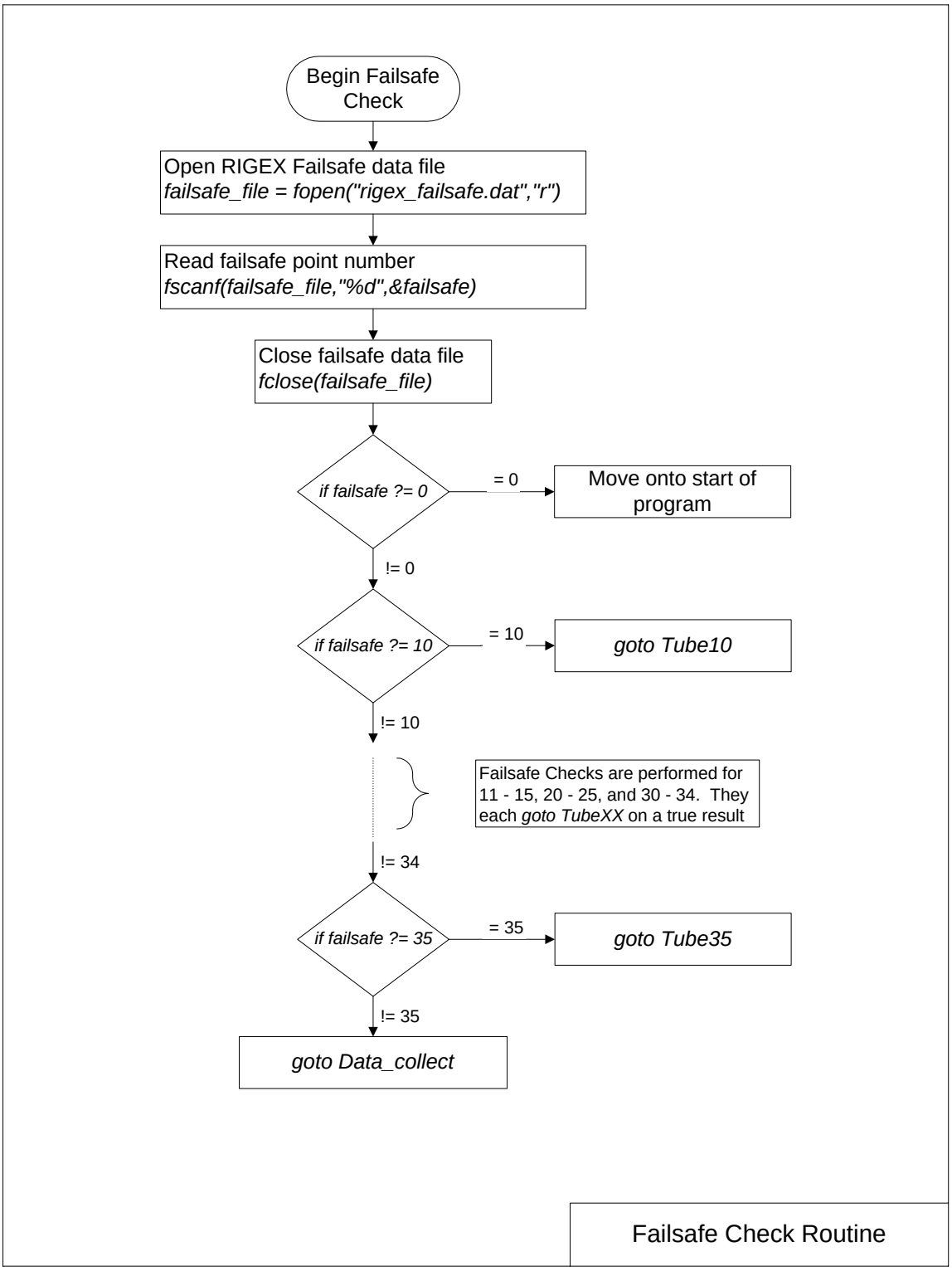


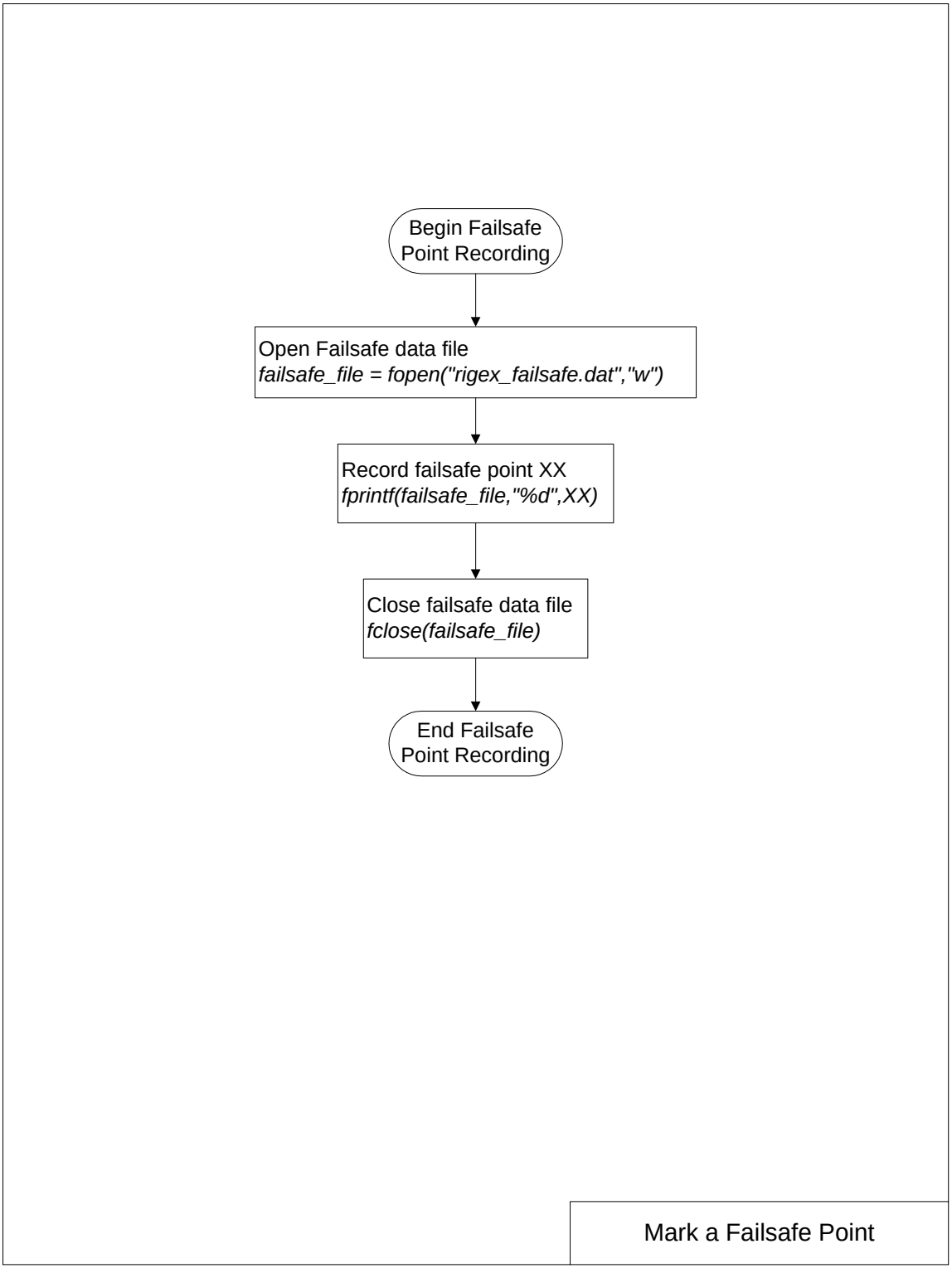

| Tube Excitation Process Calendar |
| --- |
| Describes the excitation process that the control system will undergo. [DiS01] Refer to DAQ Computer for tube number and channel assignment |

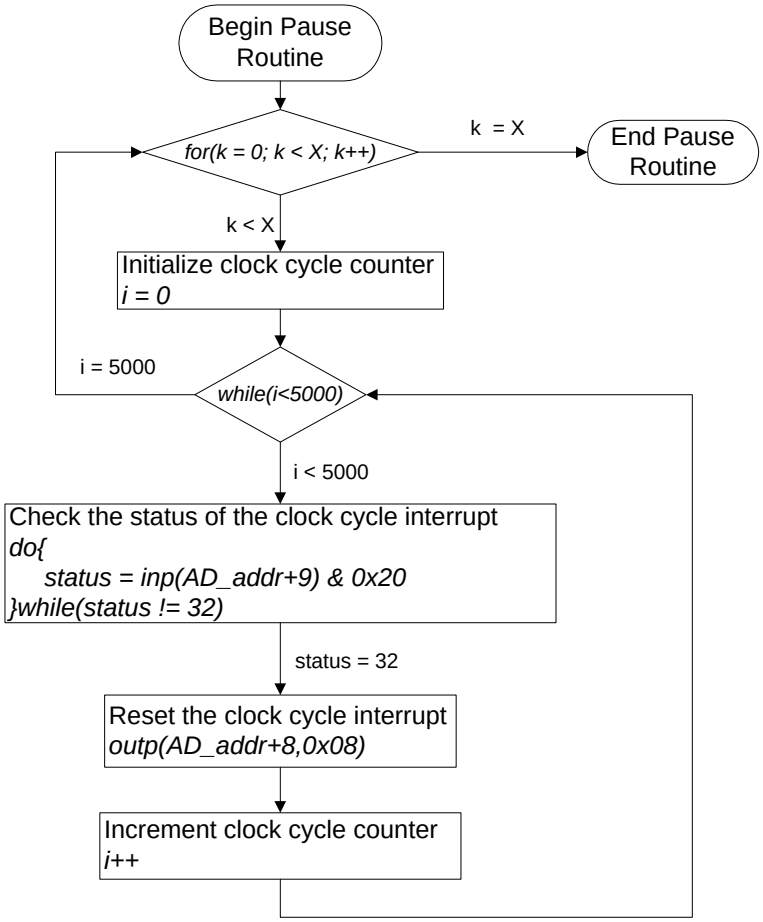
# Appendix D: Program Flowcharts



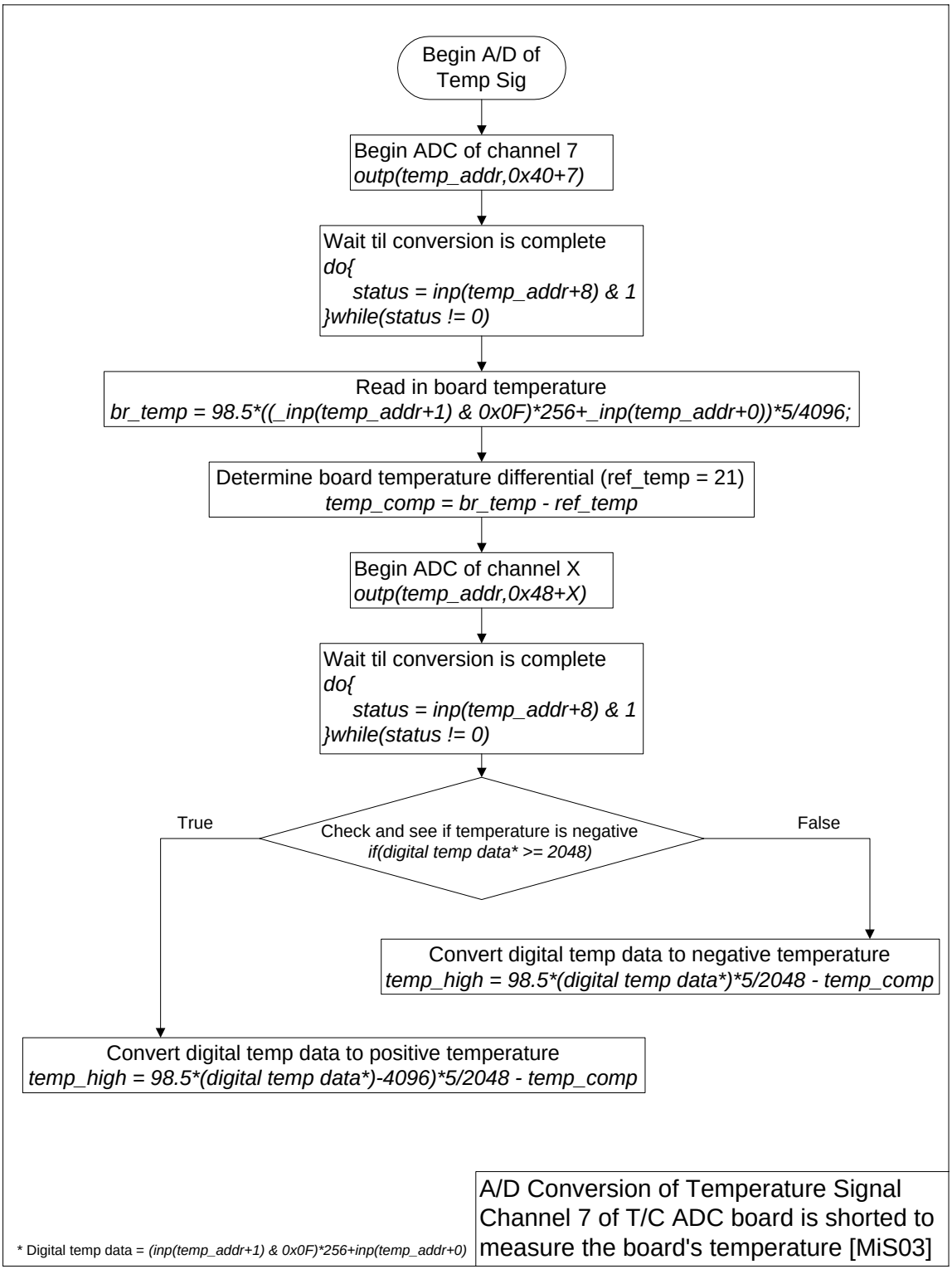

Program routine to initialize the DAQ Computer [DMM00], [QMM01]

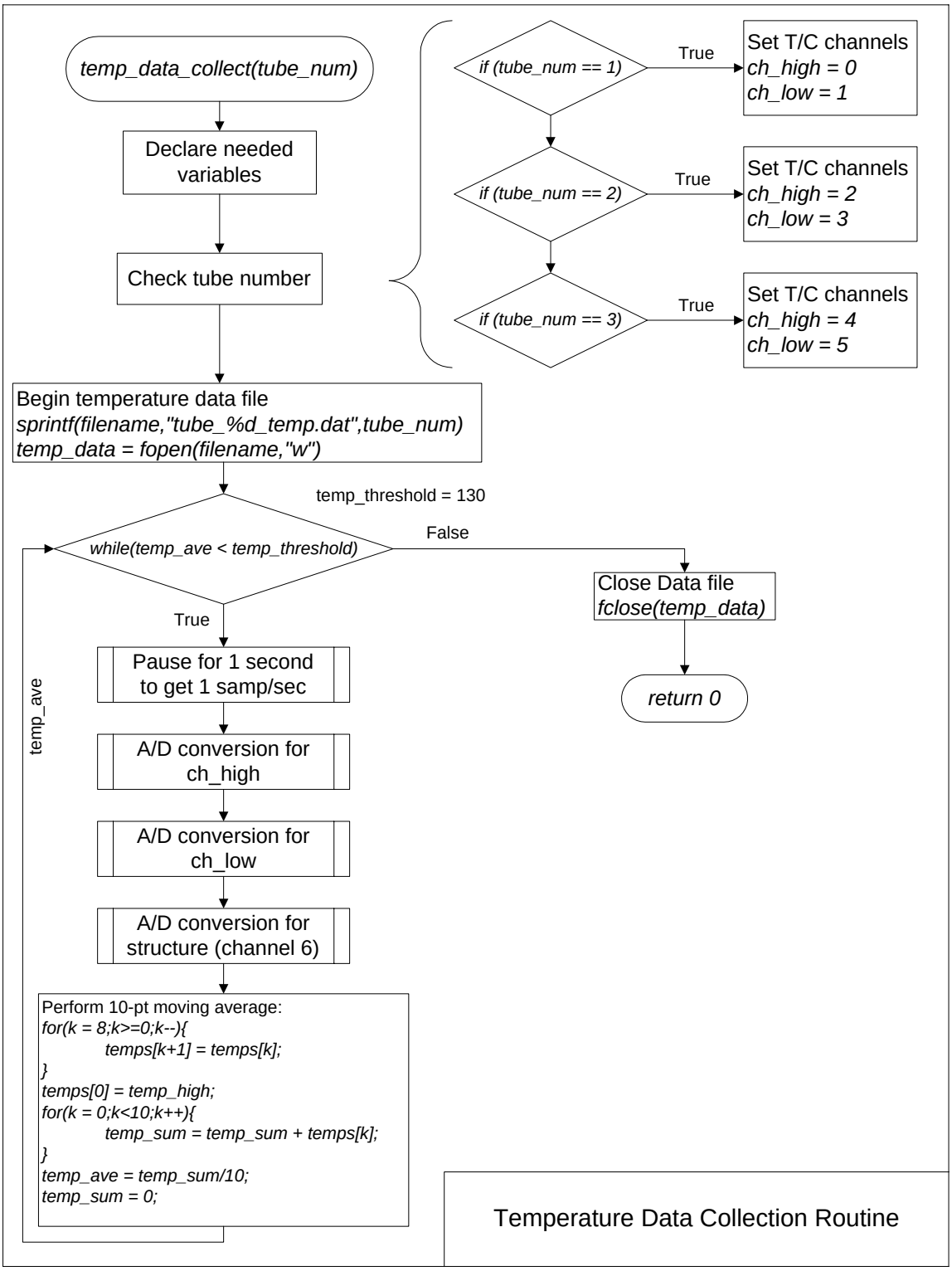

Begin Failsafe Check
Open RIGEX Failsafe data file
failsafe_file = fopen("rigex_failsafe.dat","r")
Read failsafe point number
fscanf(failsafe_file,"%d",&failsafe)
Close failsafe data file
fclose(failsafe_file)
if failsafe ?= 0
= 0
Move onto start of program
!= 0
if failsafe ?= 10
= 10
goto Tube10
!= 10
Failsafe Checks are performed for 11 - 15, 20 - 25, and 30 - 34. They each goto TubeXX on a true result
!= 34
if failsafe ?= 35
= 35
goto Tube35
!= 35
goto Data_collect
Failsafe Check Routine

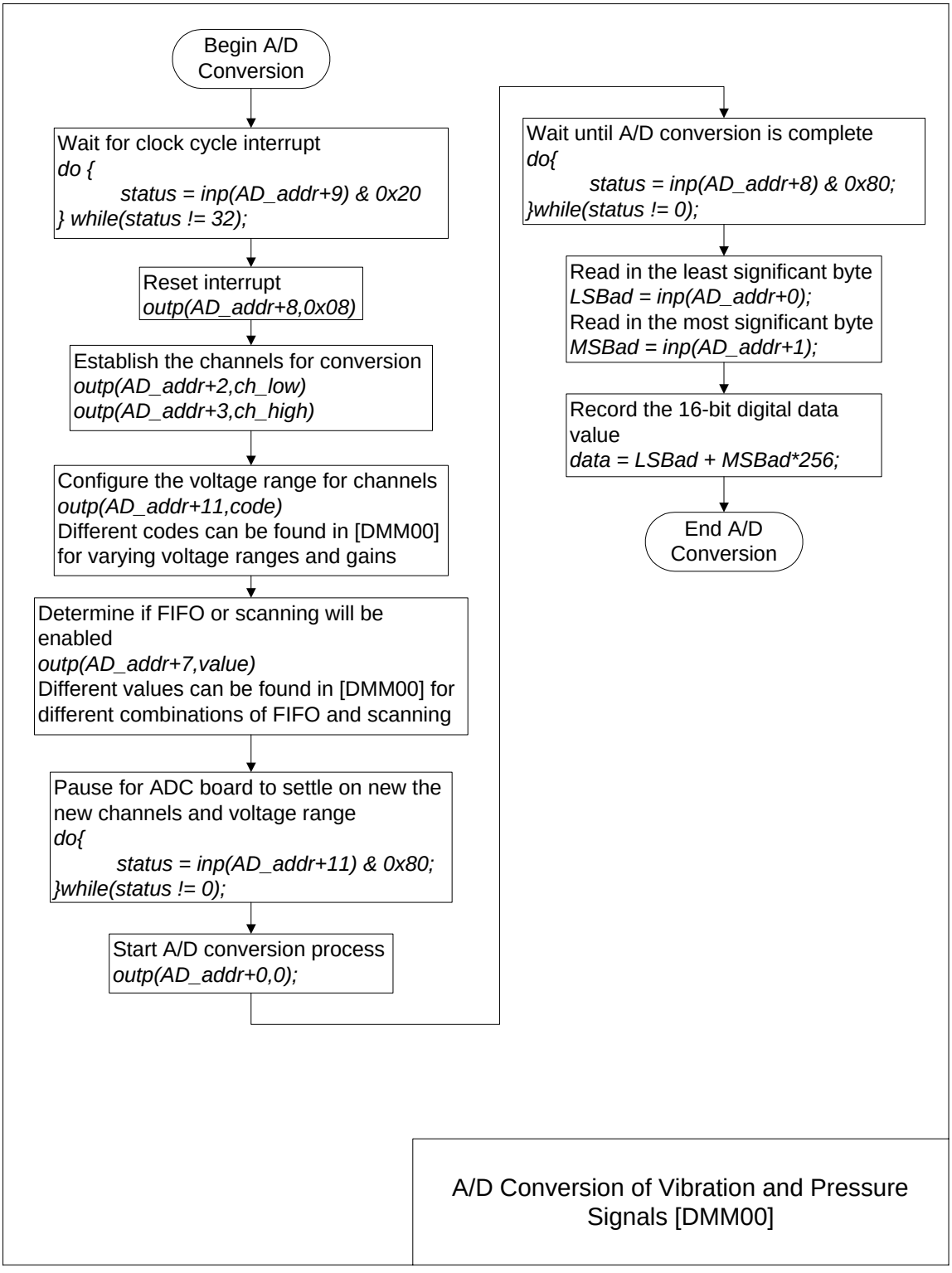

Begin Failsafe Point Recording
Open Failsafe data file
failsafe_file = fopen("rigex_failsafe.dat","w")
Record failsafe point XX
fprintf(failsafe_file,"%d",XX)
Close failsafe data file
fclose(failsafe_file)
End Failsafe Point Recording
Mark a Failsafe Point



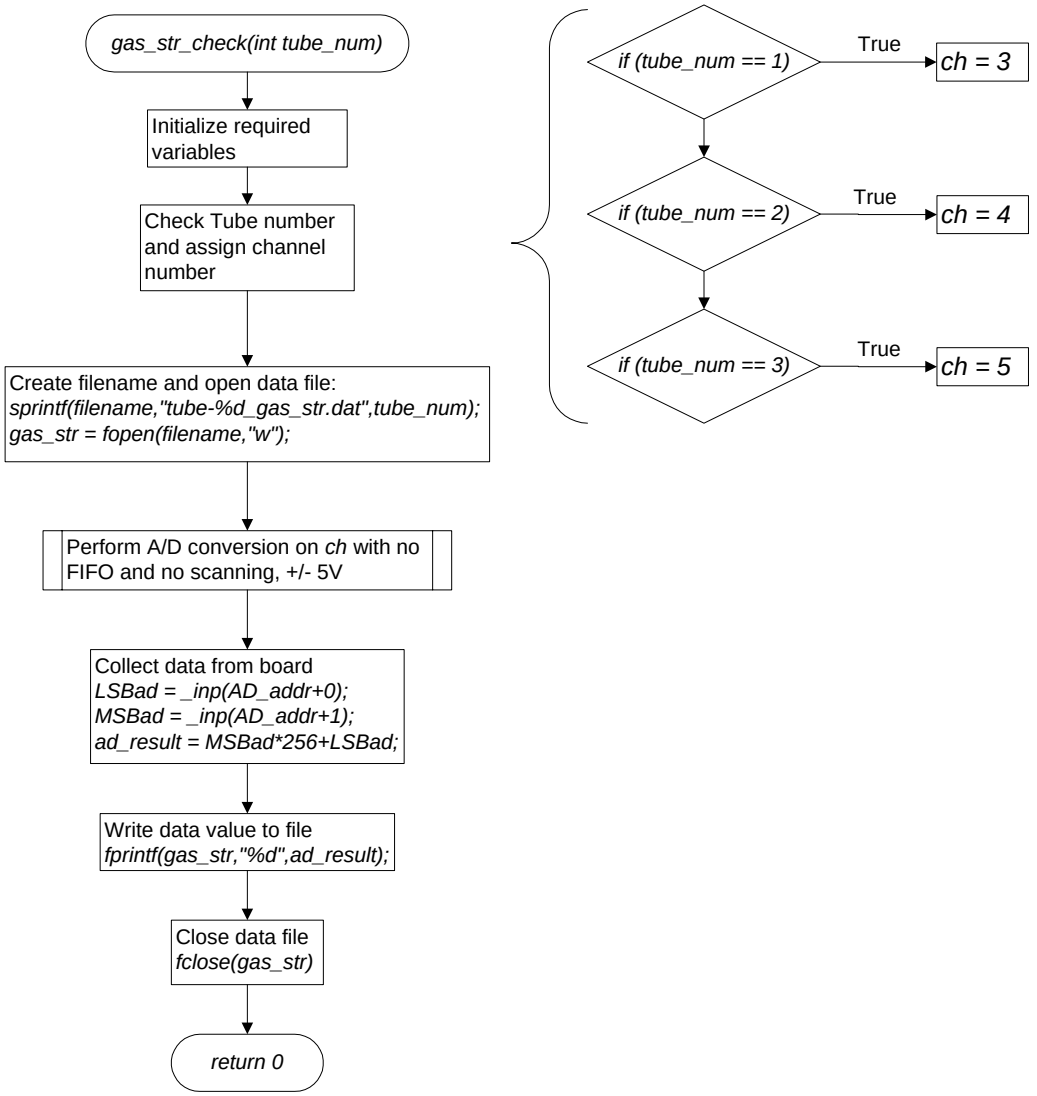

Flowchart showing how to pause for X seconds. This routine uses the 5 kHz timer onboard the ADC board. Each clock cycle will set an interrupt which is then counted where 5000 counts equals 1 second.


Begin A/D of Temp Sig
Begin ADC of channel 7
outp(temp_addr,0x40+7)
Wait til conversion is complete
do{
status = inp(temp_addr+8) & 1
}while(status != 0)
Read in board temperature
br_temp = 98.5*((_inp(temp_addr+1) & 0x0F)*256+_inp(temp_addr+0))*5/4096;
Determine board temperature differential (ref_temp = 21)
temp_comp = br_temp - ref_temp
Begin ADC of channel X
outp(temp_addr,0x48+X)
Wait til conversion is complete
do{
status = inp(temp_addr+8) & 1
}while(status != 0)
Check and see if temperature is negative
if(digital temp data* >= 2048)
True
False
Convert digital temp data to negative temperature
temp_high = 98.5*(digital temp data*)*5/2048 - temp_comp
Convert digital temp data to positive temperature
temp_high = 98.5*(digital temp data*)-4096)*5/2048 - temp_comp
* Digital temp data = (inp(temp_addr+1) & 0x0F)*256+inp(temp_addr+0)
A/D Conversion of Temperature Signal
Channel 7 of T/C ADC board is shorted to measure the board's temperature [MiS03]

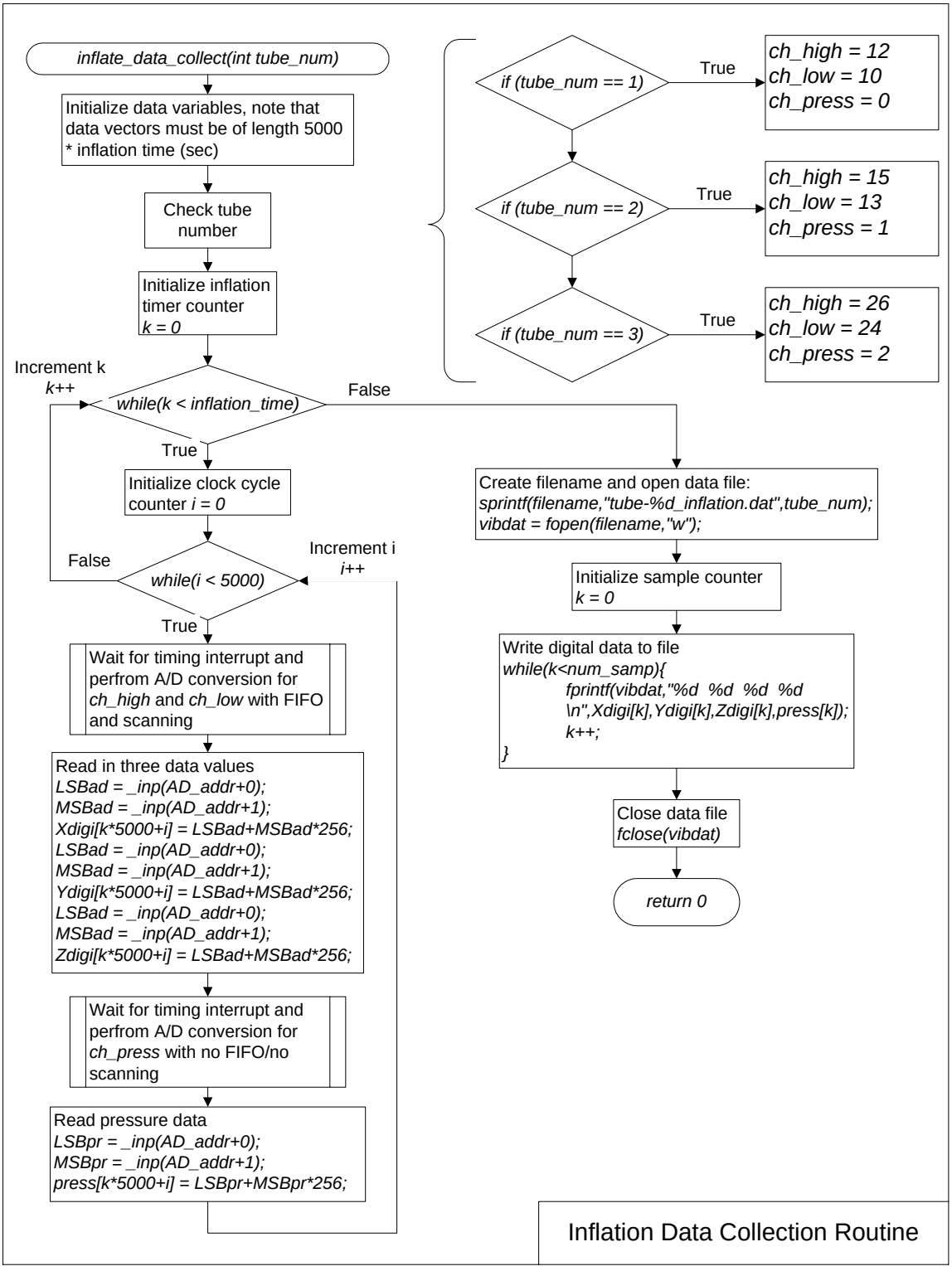

temp_data_collect(tube_num)
Declare needed variables
Check tube number
if (tube_num == 1)
True
Set T/C channels
ch_high = 0
ch_low = 1
if (tube_num == 2)
True
Set T/C channels
ch_high = 2
ch_low = 3
if (tube_num == 3)
True
Set T/C channels
ch_high = 4
ch_low = 5
Begin temperature data file
sprintf(filename,"tube_%d_temp.dat",tube_num)
temp_data = fopen(filename,"w")
temp_threshold = 130
while(temp_ave < temp_threshold)
False
Close Data file
fclose(temp_data)
return 0
True
temp_ave
Pause for 1 second to get 1 samp/sec
A/D conversion for ch_high
A/D conversion for ch_low
A/D conversion for structure (channel 6)
Perform 10-pt moving average:
for(k = 8;k>=0;k--){
temps[k+1] = temps[k];
}
temps[0] = temp_high;
for(k = 0;k<10;k++){
temp_sum = temp_sum + temps[k];
}
temp_ave = temp_sum/10;
temp_sum = 0;
Temperature Data Collection Routine

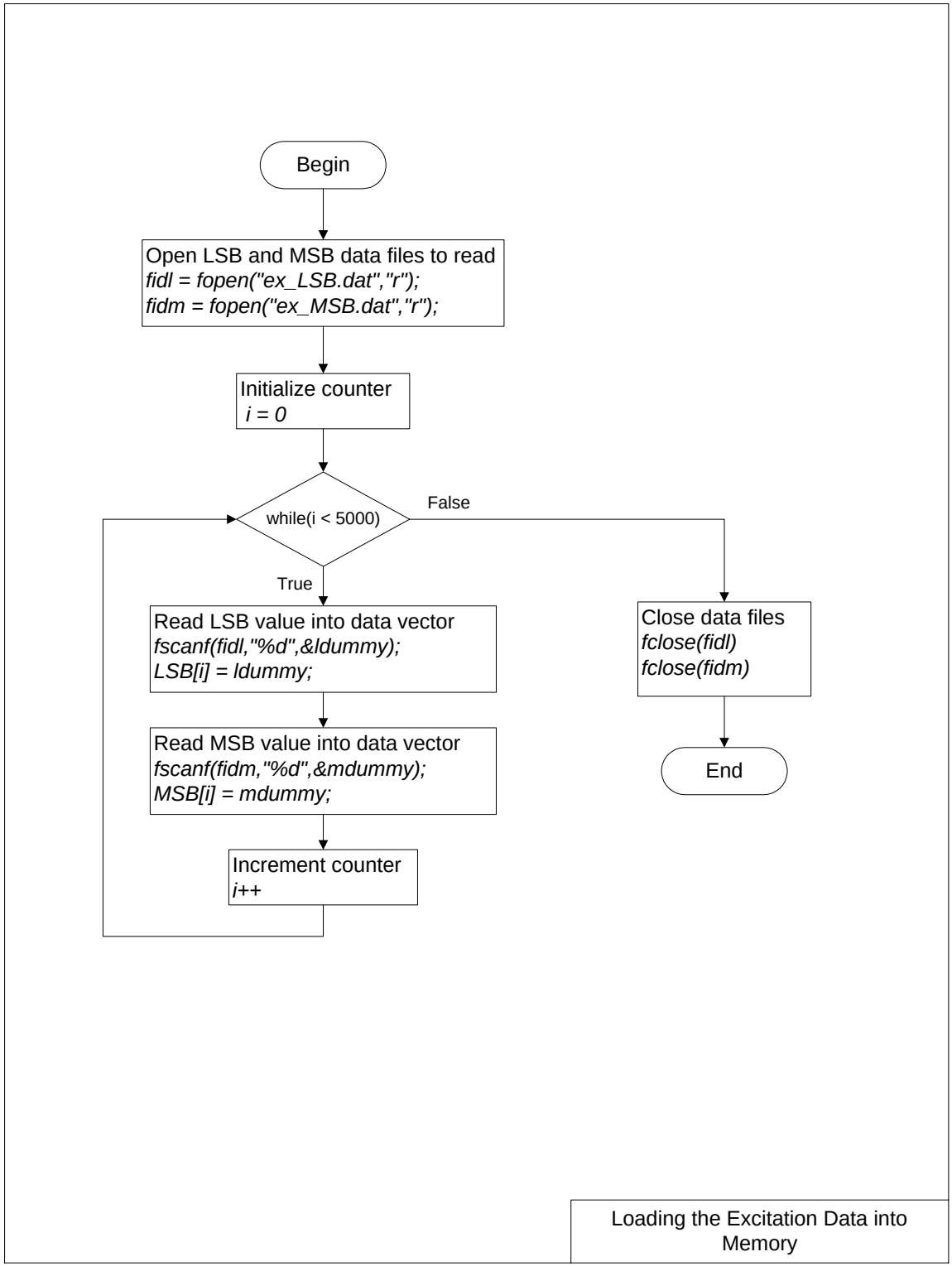


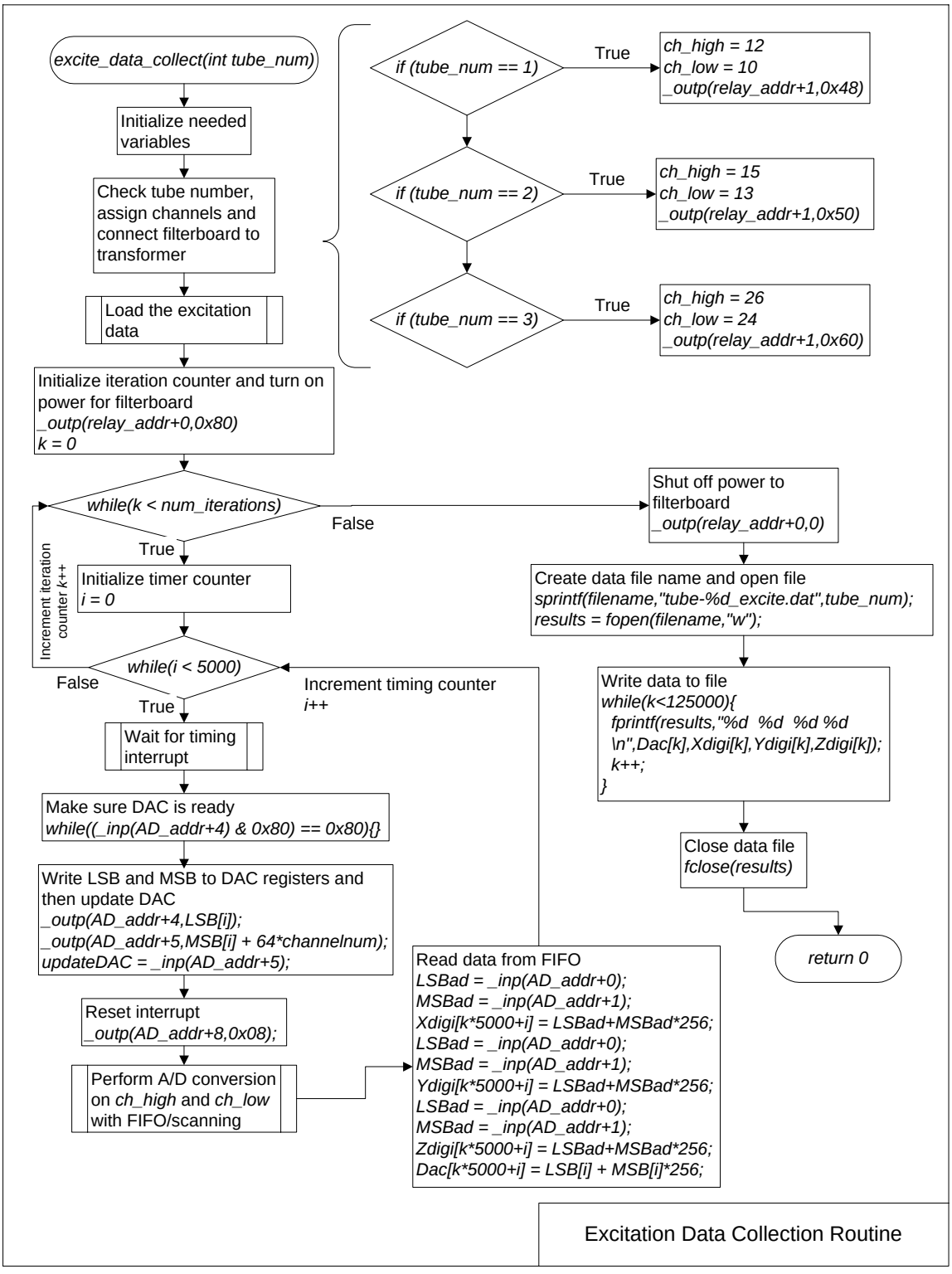

A/D Conversion of Vibration and Pressure Signals [DMM00]

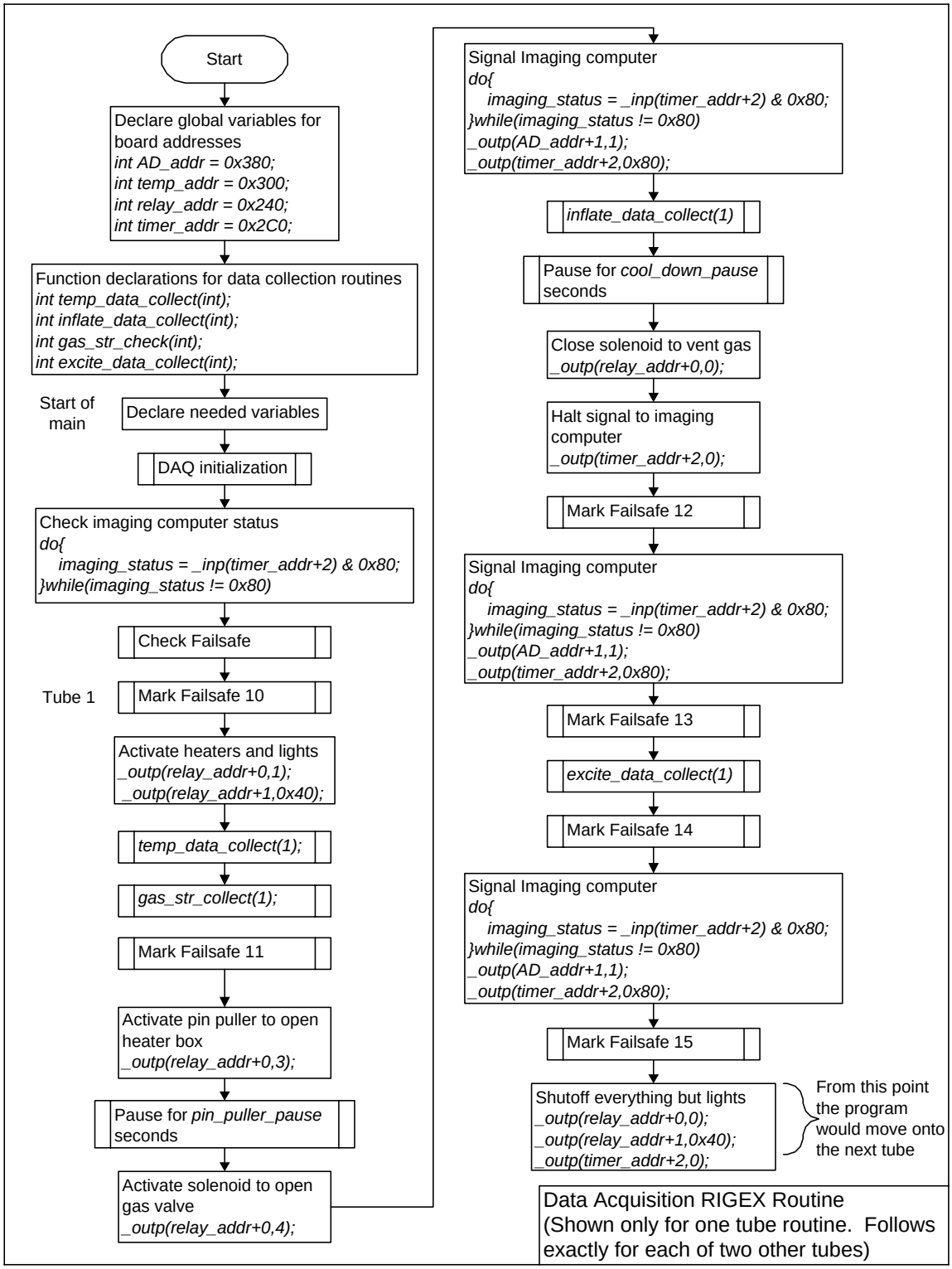


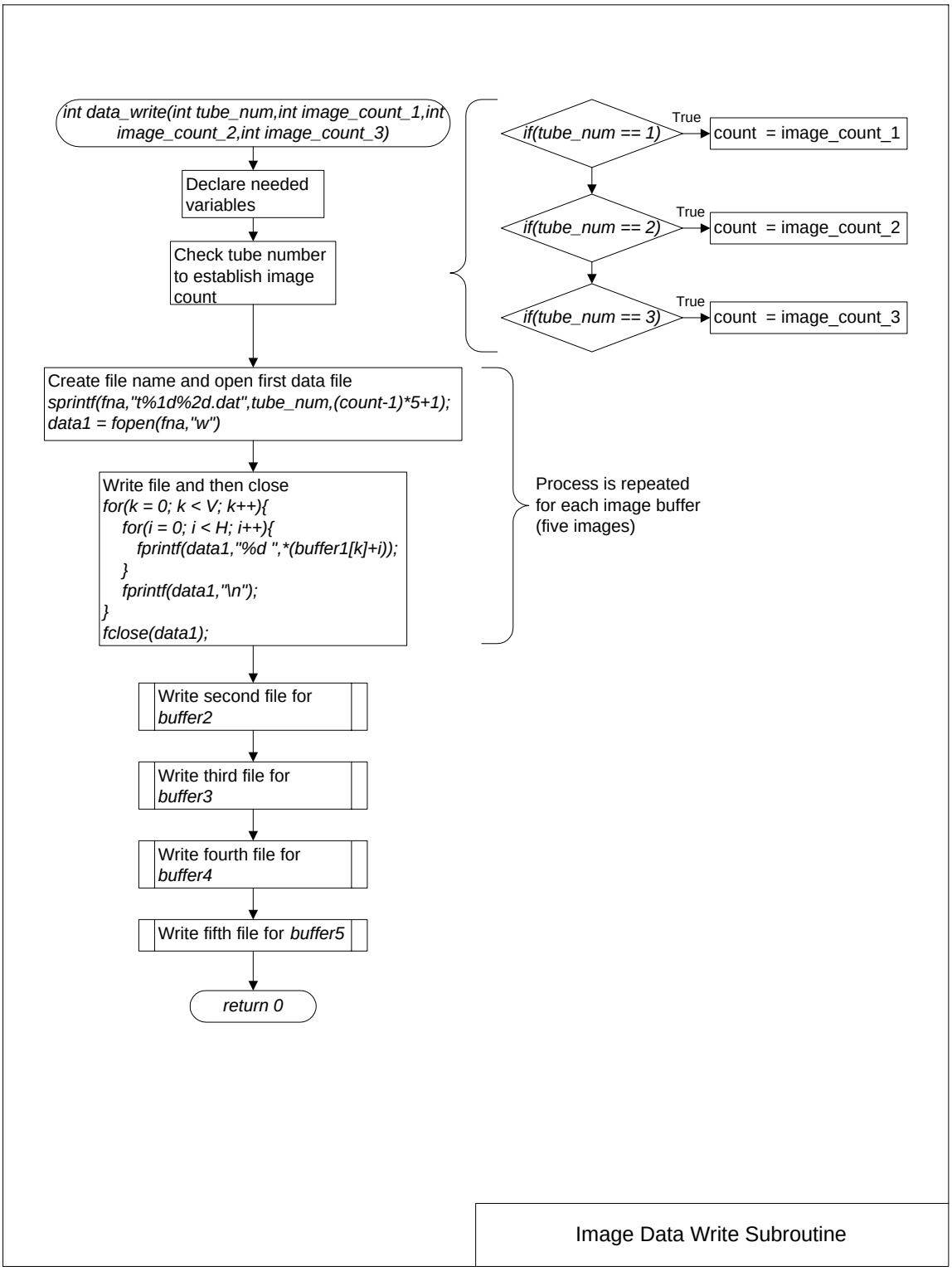

Gas Storage Press Measurement

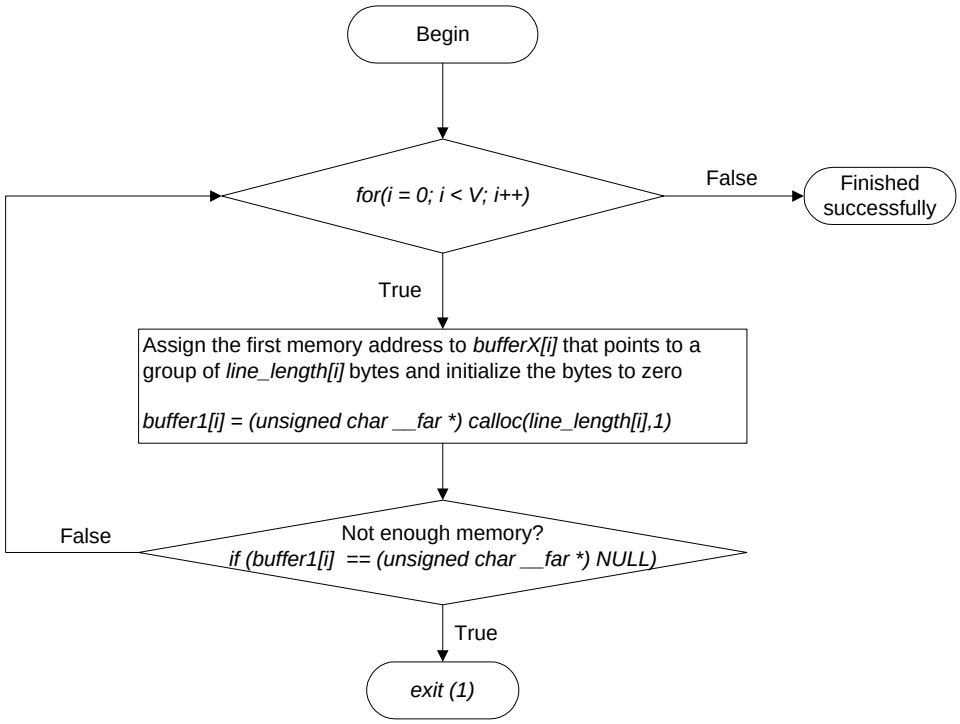

inflate_data_collect(int tube_num)
Initialize data variables, note that data vectors must be of length 5000 * inflation time (sec)
Check tube number
Initialize inflation timer counter k = 0
Increment k k++
while(k < inflation_time)
False
True
Initialize clock cycle counter i = 0
Increment i i++
while(i < 5000)
False
True
Wait for timing interrupt and perfrom A/D conversion for ch_high and ch_low with FIFO and scanning
Read in three data values
LSBad = _inp(AD_addr+0);
MSBad = _inp(AD_addr+1);
Xdigi[k*5000+i] = LSBad+MSBad*256;
LSBad = _inp(AD_addr+0);
MSBad = _inp(AD_addr+1);
Ydigi[k*5000+i] = LSBad+MSBad*256;
LSBad = _inp(AD_addr+0);
MSBad = _inp(AD_addr+1);
Zdigi[k*5000+i] = LSBad+MSBad*256;
Wait for timing interrupt and perfrom A/D conversion for ch_press with no FIFO/no scanning
Read pressure data
LSBpr = _inp(AD_addr+0);
MSBpr = _inp(AD_addr+1);
press[k*5000+i] = LSBpr+MSBpr*256;
if (tube_num == 1)
True
ch_high = 12
ch_low = 10
ch_press = 0
if (tube_num == 2)
True
ch_high = 15
ch_low = 13
ch_press = 1
if (tube_num == 3)
True
ch_high = 26
ch_low = 24
ch_press = 2
Create filename and open data file:
sprintf(filename,"tube-%d_inflation.dat",tube_num);
vibdat = fopen(filename,"w");
Initialize sample counter
k = 0
Write digital data to file
while(k<num_samp){
fprintf(vibdat,"%d %d %d %d \n",Xdigi[k],Ydigi[k],Zdigi[k],press[k]);
k++;
}
Close data file
fclose(vibdat)
return 0
Inflation Data Collection Routine


Begin
Open LSB and MSB data files to read
fidl = fopen("ex_LSB.dat","r");
fidm = fopen("ex_MSB.dat","r");
Initialize counter
i = 0
while(i < 5000)
False
True
Read LSB value into data vector
fscanf(fidl,"%d",&ldummy);
LSB[i] = ldummy;
Read MSB value into data vector
fscanf(fidm,"%d",&mdummy);
MSB[i] = mdummy;
Increment counter
i++
Close data files
fclose(fidl)
fclose(fidm)
End
Loading the Excitation Data into Memory

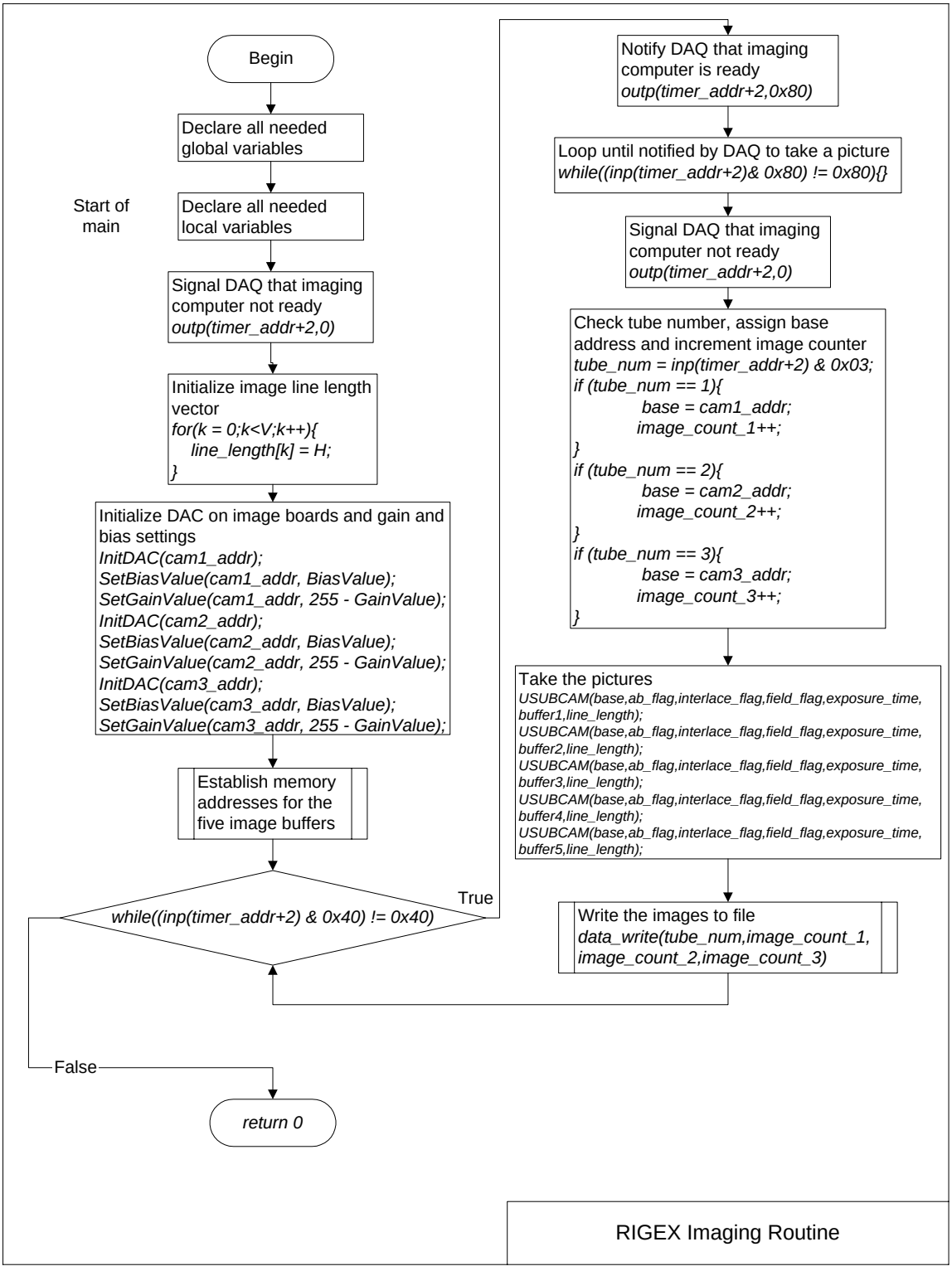

excite_data_collect(int tube_num)
Initialize needed variables
Check tube number, assign channels and connect filterboard to transformer
Load the excitation data
Initialize iteration counter and turn on power for filterboard
_outp(relay_addr+0,0x80)
k = 0
if (tube_num == 1)
True
ch_high = 12
ch_low = 10
_outp(relay_addr+1,0x48)
if (tube_num == 2)
True
ch_high = 15
ch_low = 13
_outp(relay_addr+1,0x50)
if (tube_num == 3)
True
ch_high = 26
ch_low = 24
_outp(relay_addr+1,0x60)
while(k < num_iterations)
False
True
Shut off power to filterboard
_outp(relay_addr+0,0)
Increment iteration counter k++
Initialize timer counter
i = 0
while(i < 5000)
False
True
Increment timing counter
i++
Wait for timing interrupt
Make sure DAC is ready
while((_inp(AD_addr+4) & 0x80) == 0x80){}
Write LSB and MSB to DAC registers and then update DAC
_outp(AD_addr+4,LSB[i]);
_outp(AD_addr+5,MSB[i] + 64*channelnum);
updateDAC = _inp(AD_addr+5);
Reset interrupt
_outp(AD_addr+8,0x08);
Perform A/D conversion on ch_high and ch_low with FIFO/scanning
Read data from FIFO
LSBad = _inp(AD_addr+0);
MSBad = _inp(AD_addr+1);
Xdigi[k*5000+i] = LSBad+MSBad*256;
LSBad = _inp(AD_addr+0);
MSBad = _inp(AD_addr+1);
Ydigi[k*5000+i] = LSBad+MSBad*256;
LSBad = _inp(AD_addr+0);
MSBad = _inp(AD_addr+1);
Zdigi[k*5000+i] = LSBad+MSBad*256;
Dac[k*5000+i] = LSB[i] + MSB[i]*256;
Create data file name and open file
sprintf(filename,"tube-%d_excite.dat",tube_num);
results = fopen(filename,"w");
Write data to file
while(k<125000){
fprintf(results,"%d %d %d %d \n",Dac[k],Xdigi[k],Ydigi[k],Zdigi[k]);
k++;
}
Close data file
fclose(results)
return 0
Excitation Data Collection Routine


Start
Declare global variables for board addresses
int AD_addr = 0x380;
int temp_addr = 0x300;
int relay_addr = 0x240;
int timer_addr = 0x2C0;
Function declarations for data collection routines
int temp_data_collect(int);
int inflate_data_collect(int);
int gas_str_check(int);
int excite_data_collect(int);
Start of main
Declare needed variables
DAQ initialization
Check imaging computer status
do{
imaging_status = _inp(timer_addr+2) & 0x80;
}while(imaging_status != 0x80)
Check Failsafe
Tube 1
Mark Failsafe 10
Activate heaters and lights
_outp(relay_addr+0,1);
_outp(relay_addr+1,0x40);
temp_data_collect(1);
gas_str_collect(1);
Mark Failsafe 11
Activate pin puller to open heater box
_outp(relay_addr+0,3);
Pause for pin_puller_pause seconds
Activate solenoid to open gas valve
_outp(relay_addr+0,4);
Signal Imaging computer
do{
imaging_status = _inp(timer_addr+2) & 0x80;
}while(imaging_status != 0x80)
_outp(AD_addr+1,1);
_outp(timer_addr+2,0x80);
inflate_data_collect(1)
Pause for cool_down_pause seconds
Close solenoid to vent gas
_outp(relay_addr+0,0);
Halt signal to imaging computer
_outp(timer_addr+2,0);
Mark Failsafe 12
Signal Imaging computer
do{
imaging_status = _inp(timer_addr+2) & 0x80;
}while(imaging_status != 0x80)
_outp(AD_addr+1,1);
_outp(timer_addr+2,0x80);
Mark Failsafe 13
excite_data_collect(1)
Mark Failsafe 14
Signal Imaging computer
do{
imaging_status = _inp(timer_addr+2) & 0x80;
}while(imaging_status != 0x80)
_outp(AD_addr+1,1);
_outp(timer_addr+2,0x80);
Mark Failsafe 15
Shutoff everything but lights
_outp(relay_addr+0,0);
_outp(relay_addr+1,0x40);
_outp(timer_addr+2,0);
From this point the program would move onto the next tube
Data Acquisition RIGEX Routine
(Shown only for one tube routine. Follows exactly for each of two other tubes)


int data_write(int tube_num,int image_count_1,int image_count_2,int image_count_3)
Declare needed variables
Check tube number to establish image count
if(tube_num == 1)
True
count = image_count_1
if(tube_num == 2)
True
count = image_count_2
if(tube_num == 3)
True
count = image_count_3
Create file name and open first data file
sprintf(fna,"t%1d%2d.dat",tube_num,(count-1)*5+1);
data1 = fopen(fna,"w")
Write file and then close
for(k = 0; k < V; k++){
for(i = 0; i < H; i++){
fprintf(data1,"%d ",*(buffer1[k]+i));
}
fprintf(data1,"\n");
}
fclose(data1);
Process is repeated for each image buffer (five images)
Write second file for buffer2
Write third file for buffer3
Write fourth file for buffer4
Write fifth file for buffer5
return 0
Image Data Write Subroutine




Memory Allocation for the one image buffer


Begin
Declare all needed global variables
Start of main
Declare all needed local variables
Signal DAQ that imaging computer not ready
outp(timer_addr+2,0)
Initialize image line length vector
for(k = 0;k<V;k++){
line_length[k] = H;
}
Initialize DAC on image boards and gain and bias settings
InitDAC(cam1_addr);
SetBiasValue(cam1_addr, BiasValue);
SetGainValue(cam1_addr, 255 - GainValue);
InitDAC(cam2_addr);
SetBiasValue(cam2_addr, BiasValue);
SetGainValue(cam2_addr, 255 - GainValue);
InitDAC(cam3_addr);
SetBiasValue(cam3_addr, BiasValue);
SetGainValue(cam3_addr, 255 - GainValue);
Establish memory addresses for the five image buffers
while((inp(timer_addr+2) & 0x40) != 0x40)
True
False
return 0
Notify DAQ that imaging computer is ready
outp(timer_addr+2,0x80)
Loop until notified by DAQ to take a picture
while((inp(timer_addr+2)& 0x80) != 0x80){}
Signal DAQ that imaging computer not ready
outp(timer_addr+2,0)
Check tube number, assign base address and increment image counter
tube_num = inp(timer_addr+2) & 0x03;
if (tube_num == 1){
base = cam1_addr;
image_count_1++;
}
if (tube_num == 2){
base = cam2_addr;
image_count_2++;
}
if (tube_num == 3){
base = cam3_addr;
image_count_3++;
}
Take the pictures
USUBCAM(base,ab_flag,interlace_flag,field_flag,exposure_time, buffer1,line_length);
USUBCAM(base,ab_flag,interlace_flag,field_flag,exposure_time, buffer2,line_length);
USUBCAM(base,ab_flag,interlace_flag,field_flag,exposure_time, buffer3,line_length);
USUBCAM(base,ab_flag,interlace_flag,field_flag,exposure_time, buffer4,line_length);
USUBCAM(base,ab_flag,interlace_flag,field_flag,exposure_time, buffer5,line_length);
Write the images to file
data_write(tube_num,image_count_1, image_count_2,image_count_3)
RIGEX Imaging Routine

# Appendix E: Matlab® Scripts

```
% RIGEX file to generate excitation signal for DAC
% This script creates an upchirp 3.4V cosine wave
% that has a linear frequency sweep from 5 Hz to
% 1000 Hz.  The script will convert the voltages
% to 12-bit digital data values and then split
% the data into a least significant byte (LSB) and
% a most significant byte (MSB).  The corresponding
% data files are then written.

% Establish the start and stop frequencies
f0 = 5;
f1 = 1000;

% Create the 5000 sample length time signal
t = 0:1/5000:1;

% Create the upchirp signal
y = 3.4*chirp(t,f0,1,f1);

% Plot the chirp signal and its FFT
figure(10)
clf
subplot(1,2,1)
hold on
plot(t,y,'b')
grid on
title('Up-Chirp for Tube Excitation')
xlabel('Time (t)')
axis square
subplot(1,2,2)
semilogy(-2500:2500,abs(fftshift(fft(y))))
title('FFT of Up-Chirp Waveform')
xlabel('Frequency (Hz)')
ylabel('Magnitude')
grid on;
axis square

% Convert the voltages to digital data
data = floor((y/5)*2048) + 2048;

% Split the data into MSB and LSB
LSB = bitand(data,255);
MSB = floor(data/256);

% Write the data files
fidl = fopen('ex_LSB.dat','w');
fidm = fopen('ex_MSB.dat','w');
h = waitbar(0,'Writing data...');
for k = 1:length(y)
    fprintf(fidl,'%d\n',LSB(k));
    fprintf(fidm,'%d\n',MSB(k));
    waitbar(k/5000,h);
end

fclose(fidl);
```

```
fclose(fidm);
close(h);
% RIGEX file to analyze the ADC data for the
% ADC testing using the 10mV/g accelerometer
% This script will produce a sample plot of the
% data in the time domain and will also perform
% a transfer function estimation from the data.


load vib_data.dat

% Extract and convert the digital data into voltages
% The first column contains the input digital signal
% The second column contains the ADC data
vin = (10/2048)*(vib_data(:,1)-2048*ones(length(vib_data(:,1)),1));
vout1 = vib_data(:,2)/32768*5;

% Release some memory
clear vib_data

% Plot 200 samples of the time domain for the input
% signal and the accelerometer signal
figure(1)
clf
subplot(2,1,1)
plot(vin(4900:5100),'b');
subplot(2,1,2)
hold on;
plot(vout1(4900:5100),'b');

% Perform the transfer function estimation using
% a 4096 point FFT and a sampling frequency of 5kHz
[H1,F] = tfe(vin,vout1,4096,5000);

% Plot the magnitude of the transfer function
figure(2)
clf
semilogy(F,abs(H1),'b')
axis([0 1000 0.0001 0.02])
xlabel('Frequency (Hz)');
ylabel('Magnitude (dB)');
title('Transfer Function from PC/104 Sampled Accelerometer (10 mV/g)
Output');
grid on;
```

```
% RIGEX file to load the 10 excitation experiment
% data files.  This script will analyze each file and
% produce the average transfer function with its 1 std
% dev contours.  It will also produce the transfer
% function from concatenating all the accelerometer
% signals together.  The Y-axis is used because the
% piezo patches are primarily aligned along the Y-axis

% Load the data files
load tube-1_excite.dat;
load tube-1_excite1.dat;
load tube-1_excite2.dat;
load tube-1_excite3.dat;
load tube-1_excite4.dat;
load tube-1_excite5.dat;
load tube-1_excite6.dat;
load tube-1_excite7.dat;
load tube-1_excite8.dat;
load tube-1_excite9.dat;

% Extract the vibration signals.  X-axis is column 2
% Y-axis is column 3 and Z-axis is column 4
vibx_excite =
(tube_1_excite(:,2)+32768*ones(size(tube_1_excite(:,2))))*5/65536;
viby_excite =
(tube_1_excite(:,3)+32768*ones(size(tube_1_excite(:,3))))*5/65536;
vibz_excite =
(tube_1_excite(:,4)+32768*ones(size(tube_1_excite(:,4))))*5/65536;
vibx_excite1 =
(tube_1_excite1(:,2)+32768*ones(size(tube_1_excite1(:,2))))*5/65536;
viby_excite1 =
(tube_1_excite1(:,3)+32768*ones(size(tube_1_excite1(:,3))))*5/65536;
vibz_excite1 =
(tube_1_excite1(:,4)+32768*ones(size(tube_1_excite1(:,4))))*5/65536;
vibx_excite2 =
(tube_1_excite2(:,2)+32768*ones(size(tube_1_excite2(:,2))))*5/65536;
viby_excite2 =
(tube_1_excite2(:,3)+32768*ones(size(tube_1_excite2(:,3))))*5/65536;
vibz_excite2 =
(tube_1_excite2(:,4)+32768*ones(size(tube_1_excite2(:,4))))*5/65536;
vibx_excite3 =
(tube_1_excite3(:,2)+32768*ones(size(tube_1_excite3(:,2))))*5/65536;
viby_excite3 =
(tube_1_excite3(:,3)+32768*ones(size(tube_1_excite3(:,3))))*5/65536;
vibz_excite3 =
(tube_1_excite3(:,4)+32768*ones(size(tube_1_excite3(:,4))))*5/65536;
vibx_excite4 =
(tube_1_excite4(:,2)+32768*ones(size(tube_1_excite4(:,2))))*5/65536;
viby_excite4 =
(tube_1_excite4(:,3)+32768*ones(size(tube_1_excite4(:,3))))*5/65536;
vibz_excite4 =
(tube_1_excite4(:,4)+32768*ones(size(tube_1_excite4(:,4))))*5/65536;
vibx_excite5 =
(tube_1_excite5(:,2)+32768*ones(size(tube_1_excite5(:,2))))*5/65536;
```

```
viby_excite5 =
(tube_1_excite5(:,3)+32768*ones(size(tube_1_excite5(:,3))))*5/65536;
vibz_excite5 =
(tube_1_excite5(:,4)+32768*ones(size(tube_1_excite5(:,4))))*5/65536;
vibx_excite6 =
(tube_1_excite6(:,2)+32768*ones(size(tube_1_excite6(:,2))))*5/65536;
viby_excite6 =
(tube_1_excite6(:,3)+32768*ones(size(tube_1_excite6(:,3))))*5/65536;
vibz_excite6 =
(tube_1_excite6(:,4)+32768*ones(size(tube_1_excite6(:,4))))*5/65536;
vibx_excite7 =
(tube_1_excite7(:,2)+32768*ones(size(tube_1_excite7(:,2))))*5/65536;
viby_excite7 =
(tube_1_excite7(:,3)+32768*ones(size(tube_1_excite7(:,3))))*5/65536;
vibz_excite7 =
(tube_1_excite7(:,4)+32768*ones(size(tube_1_excite7(:,4))))*5/65536;
vibx_excite8 =
(tube_1_excite8(:,2)+32768*ones(size(tube_1_excite8(:,2))))*5/65536;
viby_excite8 =
(tube_1_excite8(:,3)+32768*ones(size(tube_1_excite8(:,3))))*5/65536;
vibz_excite8 =
(tube_1_excite8(:,4)+32768*ones(size(tube_1_excite8(:,4))))*5/65536;
vibx_excite9 =
(tube_1_excite9(:,2)+32768*ones(size(tube_1_excite9(:,2))))*5/65536;
viby_excite9 =
(tube_1_excite9(:,3)+32768*ones(size(tube_1_excite9(:,3))))*5/65536;
vibz_excite9 =
(tube_1_excite9(:,4)+32768*ones(size(tube_1_excite9(:,4))))*5/65536;

% DAC signal is located in column 1
% and is the same for each data file
DAC_out = (tube_1_excite(:,1)-
2048*ones(size(tube_1_excite(:,1))))*5/2048;

% Relieve some memory
clear tube_1_excite;
clear tube_1_excite1;
clear tube_1_excite2;
clear tube_1_excite3;
clear tube_1_excite4;
clear tube_1_excite5;
clear tube_1_excite6;
clear tube_1_excite7;
clear tube_1_excite8;
clear tube_1_excite9;


% Remove the DC bias from the excitation
% vibration signals
vibx_excite = vibx_excite - ones(size(vibx_excite))*mean(vibx_excite);
viby_excite = viby_excite - ones(size(viby_excite))*mean(viby_excite);
vibz_excite = vibz_excite - ones(size(vibz_excite))*mean(vibz_excite);
vibx_excite1 = vibx_excite1 -
ones(size(vibx_excite1))*mean(vibx_excite1);
```

```
viby_excite1 = viby_excite1 -
ones(size(viby_excite1))*mean(viby_excite1);
vibz_excite1 = vibz_excite1 -
ones(size(vibz_excite1))*mean(vibz_excite1);
vibx_excite2 = vibx_excite2 -
ones(size(vibx_excite2))*mean(vibx_excite2);
viby_excite2 = viby_excite2 -
ones(size(viby_excite2))*mean(viby_excite2);
vibz_excite2 = vibz_excite2 -
ones(size(vibz_excite2))*mean(vibz_excite2);
vibx_excite3 = vibx_excite3 -
ones(size(vibx_excite3))*mean(vibx_excite3);
viby_excite3 = viby_excite3 -
ones(size(viby_excite3))*mean(viby_excite3);
vibz_excite3 = vibz_excite3 -
ones(size(vibz_excite3))*mean(vibz_excite3);
vibx_excite4 = vibx_excite4 -
ones(size(vibx_excite4))*mean(vibx_excite4);
viby_excite4 = viby_excite4 -
ones(size(viby_excite4))*mean(viby_excite4);
vibz_excite4 = vibz_excite4 -
ones(size(vibz_excite4))*mean(vibz_excite4);
vibx_excite5 = vibx_excite5 -
ones(size(vibx_excite5))*mean(vibx_excite5);
viby_excite5 = viby_excite5 -
ones(size(viby_excite5))*mean(viby_excite5);
vibz_excite5 = vibz_excite5 -
ones(size(vibz_excite5))*mean(vibz_excite5);
vibx_excite6 = vibx_excite6 -
ones(size(vibx_excite6))*mean(vibx_excite6);
viby_excite6 = viby_excite6 -
ones(size(viby_excite6))*mean(viby_excite6);
vibz_excite6 = vibz_excite6 -
ones(size(vibz_excite6))*mean(vibz_excite6);
vibx_excite7 = vibx_excite7 -
ones(size(vibx_excite7))*mean(vibx_excite7);
viby_excite7 = viby_excite7 -
ones(size(viby_excite7))*mean(viby_excite7);
vibz_excite7 = vibz_excite7 -
ones(size(vibz_excite7))*mean(vibz_excite7);
vibx_excite8 = vibx_excite8 -
ones(size(vibx_excite8))*mean(vibx_excite8);
viby_excite8 = viby_excite8 -
ones(size(viby_excite8))*mean(viby_excite8);
vibz_excite8 = vibz_excite8 -
ones(size(vibz_excite8))*mean(vibz_excite8);
vibx_excite9 = vibx_excite9 -
ones(size(vibx_excite9))*mean(vibx_excite9);
viby_excite9 = viby_excite9 -
ones(size(viby_excite9))*mean(viby_excite9);
vibz_excite9 = vibz_excite9 -
ones(size(vibz_excite9))*mean(vibz_excite9);

% FIR Lowpas filter to filter noise off excitation vibrations
[N,Fo,Ao,W] = REMEZORD([1000 1020],[1 0],[0.01 0.1],5000);
```

```
filt_coeff = remez(N,Fo,Ao,W);

% Filter the excitation signals
viby_excite = filter(filt_coeff,1,viby_excite);
viby_excite1 = filter(filt_coeff,1,viby_excite1);
viby_excite2 = filter(filt_coeff,1,viby_excite2);
viby_excite3 = filter(filt_coeff,1,viby_excite3);
viby_excite4 = filter(filt_coeff,1,viby_excite4);
viby_excite5 = filter(filt_coeff,1,viby_excite5);
viby_excite6 = filter(filt_coeff,1,viby_excite6);
viby_excite7 = filter(filt_coeff,1,viby_excite7);
viby_excite8 = filter(filt_coeff,1,viby_excite8);
viby_excite9 = filter(filt_coeff,1,viby_excite9);

% Using Bilinear transformation to convert
% DAC smoothing filter equations to z
% domain equations
[num1d,den1d] = bilinear([3.94784176e7],[1 1.23249113e4
3.9478176e7],5000);
[num2d,den2d] = bilinear([3.94784176e7],[1 1.04485553e4
3.9478176e7],5000);
[num3d,den3d] = bilinear([3.94784176e7],[1 6.98150145e3
3.9478176e7],5000);
[num4d,den4d] = bilinear([3.94784176e7],[1 2.45157729e3
3.9478176e7],5000);

% Smoothing digitized excitation waveform
% with filter
y_dacout =
filter(num4d,den4d,filter(num3d,den3d,filter(num2d,den2d,filter(num1d,d
en1d,DAC_out))));


% Estimate the transfer function for each
% iteration of the experiment
[Ty,F] = tfe(y_dacout,viby_excite,4096,5000);
[Ty1,F] = tfe(y_dacout,viby_excite1,4096,5000);
[Ty2,F] = tfe(y_dacout,viby_excite2,4096,5000);
[Ty3,F] = tfe(y_dacout,viby_excite3,4096,5000);
[Ty4,F] = tfe(y_dacout,viby_excite4,4096,5000);
[Ty5,F] = tfe(y_dacout,viby_excite5,4096,5000);
[Ty6,F] = tfe(y_dacout,viby_excite6,4096,5000);
[Ty7,F] = tfe(y_dacout,viby_excite7,4096,5000);
[Ty8,F] = tfe(y_dacout,viby_excite8,4096,5000);
[Ty9,F] = tfe(y_dacout,viby_excite9,4096,5000);

% Copy the DAC signal to be the same
% length as the vibration signals
y_out = repmat(y_dacout,10,1);

% Concatenate the accelerometer signals
% together into one column
viby = [viby_excite;
        viby_excite1;
        viby_excite2;
```

```
        viby_excite3;
        viby_excite4;
        viby_excite5;
        viby_excite6;
        viby_excite7;
        viby_excite8;
        viby_excite9];

% Get the estimated transfer function
% of the concatenated vibration signal
% (averaging in the frequency domain)
[T,Fa] = tfe(y_out,viby,4096,5000);

% Create a matrix of all the vibration
% signals to determine the mean and
% variance of the vibration signals
viby_mat = [viby_excite viby_excite1 viby_excite2 viby_excite3
viby_excite4 viby_excite5 viby_excite6 viby_excite7 viby_excite8
viby_excite9];
viby_mu = mean(viby_mat,2);
viby_mat_var = mean(var(viby_mat'))

% Get the estimated transfer function
% of the time domain averaged vibration
% signals
[T_mu,Fm] = tfe(y_dacout,viby_mu,4096,5000);

% Get the power spectral densities of the
% 10 vibration signals
[Py1,F] = psd(viby_excite1,4096,5000);
[Py2,F] = psd(viby_excite2,4096,5000);
[Py3,F] = psd(viby_excite3,4096,5000);
[Py4,F] = psd(viby_excite4,4096,5000);
[Py5,F] = psd(viby_excite5,4096,5000);
[Py6,F] = psd(viby_excite6,4096,5000);
[Py7,F] = psd(viby_excite7,4096,5000);
[Py8,F] = psd(viby_excite8,4096,5000);
[Py9,F] = psd(viby_excite9,4096,5000);
[Py,F] = psd(viby_excite,4096,5000);

% Find the average transfer function and
% the variance at each point along the
% transfer function
Ty_mat = [abs(Ty) abs(Ty1) abs(Ty2) abs(Ty3) abs(Ty4) abs(Ty5) abs(Ty6)
abs(Ty7) abs(Ty8) abs(Ty9)];
Tya = mean(Ty_mat,2);
Ty_var = var(Ty_mat')';

% 4-pt averaging filter
h = ones(4,1)/4;

% Smooth the transfer function a little
Tyf = conv2(abs(Ty),h);
Ty1f = conv2(abs(Ty1),h);
Ty2f = conv2(abs(Ty2),h);
```

```
Ty3f = conv2(abs(Ty3),h);
Ty4f = conv2(abs(Ty4),h);
Ty5f = conv2(abs(Ty5),h);
Ty6f = conv2(abs(Ty6),h);
Ty7f = conv2(abs(Ty7),h);
Ty8f = conv2(abs(Ty8),h);
Ty9f = conv2(abs(Ty9),h);

% Find the average of smoothed transfer
% function and the variance at each point
% along the transfer function
Tyf_mat = [abs(Tyf) abs(Ty1f) abs(Ty2f) abs(Ty3f) abs(Ty4f) abs(Ty5f)
abs(Ty6f) abs(Ty7f) abs(Ty8f) abs(Ty9f)];
Tyaf = mean(Tyf_mat,2);
Tyf_var = var(Tyf_mat')';

% Plot the average transfer function and
% its contours using the average of the
% 10 transfer functions
figure(1)
clf
semilogy(F,Tya,'k',F,Tya+sqrt(Ty_var),'b:',F,Tya-sqrt(Ty_var),'r:');
xlabel('Frequency (Hz)');
ylabel('Magnitude (dB)');
title('Averaged Transfer Function using 10 Realizations');
legend('Mean Transfer Function','Mean +/- 1 Std Dev',2);
axis([0 1000 0.0001 0.1]);
grid on;

% Plot the average smooth transfer function and
% its contours using the average of the
% 10 smoothed transfer functions
figure(2)
clf
semilogy(F,Tyaf,'k',F,Tyaf+sqrt(Tyf_var),'b:',F,Tyaf-
sqrt(Tyf_var),'r:');
xlabel('Frequency (Hz)');
ylabel('Magnitude (dB)');
title('Average of Smoothed Transfer Function using 10 Realizations');
legend('Mean Transfer Function','Mean +/- 1 Std Dev',2);
grid on;
axis([0 1000 0.0001 0.1]);

% Plot the frequency averaged transfer function
% and compare it to the time averaged transfer
% function
figure(3)
clf
subplot(211)
semilogy(Fa,abs(T),'k');
xlabel('Frequency (Hz)');
ylabel('Magnitude (dB)');
title('Transfer Function with 250 periodigram averages');
axis([0 1000 0.0001 0.1])
grid on;
```

```
subplot(212)
semilogy(Fm,abs(T_mu),'k');
xlabel('Frequency (Hz)');
ylabel('Magnitude (dB)');
title('Transfer Function with 10 averaged vibration signals');
axis([0 1000 0.0001 0.1])
grid on;
```

```
% RIGEX file to load the 6 temperature channel
% data to determine thermocouple (T/C) A/D board
% accuracy when compensating during data
% collection.  This script outputs the
% mean temperatures of each channel and
% the mean deviation from true temperature.
% It will also ouput each channel's smoothed
% temperature signal superimposed on the true
% temperature.  The averaging filter tap size
% is changed for each run of the script.

% Load the data files
load Ch-0-temp-comp.dat
load Ch-1-temp-comp.dat
load Ch-2-temp-comp.dat
load Ch-3-temp-comp.dat
load Ch-4-temp-comp.dat
load Ch-5-temp-comp.dat
load Ch-6-temp-comp.dat

% Extract the T/C board temperature
ch0_board = Ch_0_temp_comp(:,22);
ch1_board = Ch_1_temp_comp(:,22);
ch2_board = Ch_2_temp_comp(:,22);
ch3_board = Ch_3_temp_comp(:,22);
ch4_board = Ch_4_temp_comp(:,22);
ch5_board = Ch_5_temp_comp(:,22);
ch6_board = Ch_6_temp_comp(:,22);

% Extract the 25 temperature samples
% at each of the 21 temperatures from
% 0 deg C - 200 deg C and find the mean
% temperature signal
ch0_temp = Ch_0_temp_comp(:,1:21); ch0_temp_mu = mean(ch0_temp,1);
ch1_temp = Ch_1_temp_comp(:,1:21); ch1_temp_mu = mean(ch1_temp,1);
ch2_temp = Ch_2_temp_comp(:,1:21); ch2_temp_mu = mean(ch2_temp,1);
ch3_temp = Ch_3_temp_comp(:,1:21); ch3_temp_mu = mean(ch3_temp,1);
ch4_temp = Ch_4_temp_comp(:,1:21); ch4_temp_mu = mean(ch4_temp,1);
ch5_temp = Ch_5_temp_comp(:,1:21); ch5_temp_mu = mean(ch5_temp,1);
ch6_temp = Ch_6_temp_comp(:,1:21); ch6_temp_mu = mean(ch6_temp,1);

% Subtract the true temperature from the
% 25 x 21 samples
ch0_dev = ch0_temp - ones(25,1)*[0:10:200];
ch1_dev = ch1_temp - ones(25,1)*[0:10:200];
ch2_dev = ch2_temp - ones(25,1)*[0:10:200];
ch3_dev = ch3_temp - ones(25,1)*[0:10:200];
ch4_dev = ch4_temp - ones(25,1)*[0:10:200];
ch5_dev = ch5_temp - ones(25,1)*[0:10:200];
ch6_dev = ch6_temp - ones(25,1)*[0:10:200];

% Determine the mean and variance of the
% measured temperature deviation
ch0_dev_mu = mean(ch0_dev,1); ch0_dev_var = var(ch0_dev,1);
ch1_dev_mu = mean(ch1_dev,1); ch1_dev_var = var(ch1_dev,1);
```

```
ch2_dev_mu = mean(ch2_dev,1); ch2_dev_var = var(ch2_dev,1);
ch3_dev_mu = mean(ch3_dev,1); ch3_dev_var = var(ch3_dev,1);
ch4_dev_mu = mean(ch4_dev,1); ch4_dev_var = var(ch4_dev,1);
ch5_dev_mu = mean(ch5_dev,1); ch5_dev_var = var(ch5_dev,1);
ch6_dev_mu = mean(ch6_dev,1); ch6_dev_var = var(ch6_dev,1);

% Plot the mean temperature signal for
% each ADC channel and plot the mean
% deviation for each channel
figure(1)
clf
subplot(121)
hold on
plot([0:10:200],ch0_temp_mu,'b');
plot([0:10:200],ch1_temp_mu,'k');
plot([0:10:200],ch2_temp_mu,'m');
plot([0:10:200],ch3_temp_mu,'r');
plot([0:10:200],ch4_temp_mu,'g');
plot([0:10:200],ch5_temp_mu,'c');
plot([0:10:200],ch6_temp_mu,'b:');
plot([0:10:200],[0:10:200],'bx')
xlabel('True Temperature (C)');
ylabel('Mean Measured Temperature (C)');
title('Mean Measured Temperature vs. True Temperature')
axis([0 200 0 200])
axis square
grid on
legend('Channel 0','Channel 1','Channel 2','Channel 3','Channel
4','Channel 5','Channel 6','True Temp. Line',2)
subplot(122)
hold on
plot([0:10:200],ch0_dev_mu,'b');
plot([0:10:200],ch1_dev_mu,'k');
plot([0:10:200],ch2_dev_mu,'m');
plot([0:10:200],ch3_dev_mu,'r');
plot([0:10:200],ch4_dev_mu,'g');
plot([0:10:200],ch5_dev_mu,'c');
plot([0:10:200],ch6_dev_mu,'b:');
xlabel('True Temperature (C)');
ylabel('Mean Temperature Deviation(C)');
title('Mean Temperature Deviation vs. True Temperature')
axis square
grid on
legend('Channel 0','Channel 1','Channel 2','Channel 3','Channel
4','Channel 5','Channel 6',2)

% Plot each channel's mean deviation
% with 1 std deviation error bars
figure(2)
clf
subplot(421)
errorbar([0:10:200],ch0_dev_mu,sqrt(ch0_dev_var)); grid on
ylabel('Channel 0 Deviation')
subplot(422)
errorbar([0:10:200],ch1_dev_mu,sqrt(ch1_dev_var)); grid on
```

```
ylabel('Channel 1 Deviation')
subplot(423)
errorbar([0:10:200],ch2_dev_mu,sqrt(ch2_dev_var)); grid on
ylabel('Channel 2 Deviation')
subplot(424)
errorbar([0:10:200],ch3_dev_mu,sqrt(ch3_dev_var)); grid on
ylabel('Channel 3 Deviation')
subplot(425)
errorbar([0:10:200],ch4_dev_mu,sqrt(ch4_dev_var)); grid on
ylabel('Channel 4 Deviation')
subplot(426)
errorbar([0:10:200],ch5_dev_mu,sqrt(ch5_dev_var)); grid on
ylabel('Channel 5 Deviation')
xlabel('True Temperature (C)')
subplot(427)
errorbar([0:10:200],ch6_dev_mu,sqrt(ch6_dev_var)); grid on
ylabel('Channel 6 Deviation')
xlabel('True Temperature (C)')

% Create the smoothing filter
tap = 10;
h = ones(1,tap)/tap;

% Vectorize each channel's temp data
ch0 = ch0_temp(:);
ch1 = ch1_temp(:);
ch2 = ch2_temp(:);
ch3 = ch3_temp(:);
ch4 = ch4_temp(:);
ch5 = ch5_temp(:);
ch6 = ch6_temp(:);

% Smooth each channel
ch0out = conv2(ch0,h','valid');
ch1out = conv2(ch1,h','valid');
ch2out = conv2(ch2,h','valid');
ch3out = conv2(ch3,h','valid');
ch4out = conv2(ch4,h','valid');
ch5out = conv2(ch5,h','valid');
ch6out = conv2(ch6,h','valid');

% Create the true temperature signal
true = ones(25,1)*[0:10:200]; true = true(:);

% Figures 3 - 9 show each channel's
% smoothed response plotted against the
% true temperature signal
figure(3)
clf
hold on
plot(1:length(ch0out),ch0out,'k');
stairs(1:length(true),true,'r--');
xlabel('Time (sec)');
ylabel('Temperature (C)');
```

```
title(sprintf('Channel 0 Temperature Signal filtered using a %d tap
averaging filter',tap))
legend('Filtered Temperature Signal','Input Temperature Signal',4);
axis([0 length(true) 0 205])
grid on

figure(4)
clf
hold on
plot(1:length(ch1out),ch1out,'b:');
stairs(1:length(true),true,'k');
xlabel('Time (sec)');
ylabel('Temperature (C)');
title(sprintf('Channel 1 Temperature Signal filtered using a %d tap
averaging filter',tap))
legend('Filtered Temperature Signal','Input Temperature Signal',4);
axis([0 length(true) 0 205])
grid on

figure(5)
clf
hold on
plot(1:length(ch2out),ch2out,'b:');
stairs(1:length(true),true,'k');
xlabel('Time (sec)');
ylabel('Temperature (C)');
title(sprintf('Channel 2 Temperature Signal filtered using a %d tap
averaging filter',tap))
legend('Filtered Temperature Signal','Input Temperature Signal',4);
axis([0 length(true) 0 205])
grid on

figure(6)
clf
hold on
plot(1:length(ch3out),ch3out,'b:');
stairs(1:length(true),true,'k');
xlabel('Time (sec)');
ylabel('Temperature (C)');
title(sprintf('Channel 3 Temperature Signal filtered using a %d tap
averaging filter',tap))
legend('Filtered Temperature Signal','Input Temperature Signal',4);
axis([0 length(true) 0 205])
grid on

figure(7)
clf
hold on
plot(1:length(ch4out),ch4out,'b:');
stairs(1:length(true),true,'k');
xlabel('Time (sec)');
ylabel('Temperature (C)');
title(sprintf('Channel 4 Temperature Signal filtered using a %d tap
averaging filter',tap))
legend('Filtered Temperature Signal','Input Temperature Signal',4);
```

```
axis([0 length(true) 0 205])
grid on

figure(8)
clf
hold on
plot(1:length(ch5out),ch5out,'b:');
stairs(1:length(true),true,'k');
xlabel('Time (sec)');
ylabel('Temperature (C)');
title(sprintf('Channel 5 Temperature Signal filtered using a %d tap
averaging filter',tap))
legend('Filtered Temperature Signal','Input Temperature Signal',4);
axis([0 length(true) 0 205])
grid on

figure(9)
clf
hold on
plot(1:length(ch6out),ch6out,'b:');
stairs(1:length(true),true,'k');
xlabel('Time (sec)');
ylabel('Temperature (C)');
title(sprintf('Channel 6 Temperature Signal filtered using a %d tap
averaging filter',tap))
legend('Filtered Temperature Signal','Input Temperature Signal',4);
axis([0 length(true) 0 205])
grid on
```

```
% RIGEX file to determine accuracy of T/C
% ADC board when measuring the heating process
% of a rigidizable tube.  The script will
% output the temperature plots for each of the
% heating processes loaded.  The threshold
% temperature is set at 130 C and transition
% temperature is set to 125 C.

% Load in temperature data
load tube_1_temp7.dat
load tube_1_temp8.dat
load tube_1_temp9.dat

% Extracting temperature data
comp_temp7 = tube_1_temp7(:,1);
comp_temp8 = tube_1_temp8(:,1);
comp_temp9 = tube_1_temp9(:,1);
high_ch_temp7 = tube_1_temp7(:,2);
high_ch_temp8 = tube_1_temp8(:,2);
high_ch_temp9 = tube_1_temp9(:,2);
low_ch_temp7 = tube_1_temp7(:,3);
low_ch_temp8 = tube_1_temp8(:,3);
low_ch_temp9 = tube_1_temp9(:,3);
environ_temp7 = tube_1_temp7(:,4);
environ_temp8 = tube_1_temp8(:,4);
environ_temp9 = tube_1_temp9(:,4);


% Smoothing temperature signals with
% 10 point averaging filter
h = ones(10,1)/10;
high_ch_temp_ave9 = conv2(high_ch_temp9,h,'valid');
low_ch_temp_ave9 = conv2(low_ch_temp9,h,'valid');
environ_temp_ave9 = conv2(environ_temp9,h,'valid');
high_ch_temp_ave8 = conv2(high_ch_temp8,h,'valid');
low_ch_temp_ave8 = conv2(low_ch_temp8,h,'valid');
environ_temp_ave8 = conv2(environ_temp8,h,'valid');
high_ch_temp_ave7 = conv2(high_ch_temp7,h,'valid');
low_ch_temp_ave7 = conv2(low_ch_temp7,h,'valid');
environ_temp_ave7 = conv2(environ_temp7,h,'valid');



% Temperature Signals
figure(1)
clf
subplot(221)
hold on;
plot(high_ch_temp9,'r');
plot(high_ch_temp_ave9,'k')
plot(130*ones(length(high_ch_temp9),1),':b');
plot(125*ones(length(high_ch_temp9),1),'b-.');
grid on;
title('Measured High T/C Temperature Signal');
ylabel('Temperature (C)')
```

```
xlabel('Time (sec)')
legend('Measured Temp.','Averaged Temp.','130 deg C Thshld','125 deg C
Trans.',4)
subplot(222)
hold on;
plot(low_ch_temp9,'r');
plot(low_ch_temp_ave9,'k')
plot(130*ones(length(low_ch_temp9),1),':b');
plot(125*ones(length(low_ch_temp9),1),'b-.');
grid on;
title('Measured Low T/C Temperature Signal');
ylabel('Temperature (C)')
xlabel('Time (sec)')
legend('Measured Temp.','Averaged Temp.','130 deg C Thshld','125 deg C
Trans.',4)
subplot(223)
plot(environ_temp_ave9);
grid on;
title('Measured Structure Temperature')
ylabel('Temperature (C)');
xlabel('Time (sec)');
subplot(224)
plot(comp_temp9);
grid on;
title('Measured Compensation Temp. Differential')
ylabel('Temperature (C)');
xlabel('Time (sec)');

figure(2)
clf
subplot(221)
hold on;
plot(high_ch_temp8,'r');
plot(high_ch_temp_ave8,'k')
plot(130*ones(length(high_ch_temp8),1),':b');
plot(125*ones(length(high_ch_temp8),1),'b-.');
grid on;
title('Measured High T/C Temperature Signal');
ylabel('Temperature (C)')
xlabel('Time (sec)')
legend('Measured Temp.','Averaged Temp.','130 deg C Thshld','125 deg C
Trans.',4)
subplot(222)
hold on;
plot(low_ch_temp8,'r');
plot(low_ch_temp_ave8,'k')
plot(130*ones(length(low_ch_temp8),1),':b');
plot(125*ones(length(low_ch_temp8),1),'b-.');
grid on;
title('Measured Low T/C Temperature Signal');
ylabel('Temperature (C)')
xlabel('Time (sec)')
legend('Measured Temp.','Averaged Temp.','130 deg C Thshld','125 deg C
Trans.',4)
subplot(223)
```

```
plot(environ_temp_ave8);
grid on;
title('Measured Structure Temperature')
ylabel('Temperature (C)');
xlabel('Time (sec)');
subplot(224)
plot(comp_temp8);
grid on;
title('Measured Compensation Temp. Differential')
ylabel('Temperature (C)');
xlabel('Time (sec)');

figure(3)
clf
subplot(221)
hold on;
plot(high_ch_temp7,'r');
plot(high_ch_temp_ave7,'k')
plot(130*ones(length(high_ch_temp7),1),':b');
plot(125*ones(length(high_ch_temp7),1),'b-.');
grid on;
title('Measured High T/C Temperature Signal');
ylabel('Temperature (C)')
xlabel('Time (sec)')
legend('Measured Temp.','Averaged Temp.','130 deg C Thshld','125 deg C
Trans.',4)
subplot(222)
hold on;
plot(low_ch_temp7,'r');
plot(low_ch_temp_ave7,'k')
plot(130*ones(length(low_ch_temp7),1),':b');
plot(125*ones(length(low_ch_temp7),1),'b-.');
grid on;
title('Measured Low T/C Temperature Signal');
ylabel('Temperature (C)')
xlabel('Time (sec)')
legend('Measured Temp.','Averaged Temp.','130 deg C Thshld','125 deg C
Trans.',4)
subplot(223)
plot(environ_temp_ave7);
grid on;
title('Measured Structure Temperature')
ylabel('Temperature (C)');
xlabel('Time (sec)');
subplot(224)
plot(comp_temp7);
grid on;
title('Measured Compensation Temp. Differential')
ylabel('Temperature (C)');
xlabel('Time (sec)');
```

```
% RIGEX file analyzes the data file produced
% during the pressure/inflation test program.
% The file will output the pressure signal and
% will produce each axes' unfiltered/filtered
% vibration signal, unfiltered/filtered PSD and
% displacement signal.  It will also produce 3D
% visual of the path and create an avi file
% showing the movement of the accelerometer every
% 500 samples

% Load data file
load press_test_strup.dat

% Separate the vibration signals from pressure signal
A =
(press_test_strup(:,1:3)+32768*ones(size(press_test_strup(:,1:3))))*5/6
5536;

% Convert pressure data to voltage
pressure = press_test3(:,4)*5/32768/0.00349;

% Plot the pressure signal vs time
figure(1)
clf
plot(0:1/5000:(length(pressure)-1)/5000,pressure);
xlabel('Time (sec)');
ylabel('Pressure (PSI)');
title('Pressure Signal'); grid on;


% Detrend each acceleration signal and find
% its PSD
vibxi = A(:,1) - mean(A(:,1))*ones(size(A(:,1)));
[P1,F] = psd(vibxi,4096,5000);
vibyi = A(:,2) - mean(A(:,2))*ones(size(A(:,2)));
[P2,F] = psd(vibyi,4096,5000);
vibzi = A(:,3) - mean(A(:,3))*ones(size(A(:,3)));
[P3,F] = psd(vibzi,4096,5000);

% Develop lowpass filter to remove noise from acceleration signals
[N,Fo,Ao,W] = REMEZORD([50 60],[1 0],[0.01 0.1],5000);
filt_coeff = remez(N,Fo,Ao,W);

% Filter the acceleration signals using above filter
vibxif = filter(filt_coeff,1,vibxi);
vibyif = filter(filt_coeff,1,vibyi);
vibzif = filter(filt_coeff,1,vibzi);

% Get the filtered accleration signal PSDs
[P1f,F] = psd(vibxif,4096,5000);
[P2f,F] = psd(vibyif,4096,5000);
[P3f,F] = psd(vibzif,4096,5000);

% Convert acceleration signal units from voltage to g's
vibxif = vibxif/186e-3;
```

```
vibyif = vibyif/186e-3;
vibzif = vibzif/186e-3;

% Perform 2 running sums to perform the two
% required discrete integrations
ddx = zeros(size(vibxif)); dx = ddx;
ddy = zeros(size(vibyif)); dy = ddy;
ddz = zeros(size(vibzif)); dz = ddz;
h = waitbar(0,'Performing first running sum...');
for k = 1:(length(vibxif))
    ddx(k) = sum(vibxif(1:k))/5000;
    ddy(k) = sum(vibyif(1:k))/5000;
    ddz(k) = sum(vibzif(1:k))/5000;
    waitbar(k/length(vibxif),h);
end
close(h);
h = waitbar(0,'Performing second running sum...');
for k = 1:(length(vibxif))
    dx(k) = sum(ddx(1:k))/5000;
    dy(k) = sum(ddy(1:k))/5000;
    dz(k) = sum(ddz(1:k))/5000;
    waitbar(k/length(vibxif),h);
end
close(h);

% Figure 2: X Axis Inflation/Displacement Signal
figure(2)
clf
subplot(221)
plot(0:1/5000:(length(vibxi)-1)/5000,vibxi)
title('X Axis Inflation Signal')
grid on
subplot(222)
plot(0:1/5000:(length(vibxif)-1)/5000,vibxif)
grid on
title(sprintf('X Axis Filtered Inflation Signal (N = %d)',N));
subplot(223)
semilogy(F,P1,'b:',F,P1f,'k');
axis([0 500 1e-6 1])
title('PSDs of Unfiltered and Filtered Accel Signal');
grid on;
legend('Unfiltered PSD','Filtered PSD',1);
subplot(224)
plot(0:1/5000:(length(dx)-1)/5000,(dx))
grid on;
title('X Axis Displacement Signal');

% Figure 3: Y Axis Inflation/Displacement Signal
figure(3)
subplot(221)
plot(0:1/5000:(length(vibyi)-1)/5000,vibyi)
title('Y Axis Inflation Signal')
grid on
subplot(222)
plot(0:1/5000:(length(vibyif)-1)/5000,vibyif)
```

```
grid on
title(sprintf('Y Axis Filtered Inflation Signal (N = %d)',N));
subplot(223)
semilogy(F,P1,'b:',F,P1f,'k');
axis([0 500 1e-6 1])
title('PSDs of Unfiltered and Filtered Accel Signal');
grid on;
legend('Unfiltered PSD','Filtered PSD',1);
subplot(224)
plot(0:1/5000:(length(dy)-1)/5000,(dy))
grid on;
title('Y Axis Displacement Signal');

% Figure 4: Z Axis Inflation/Displacement Signal
figure(4)
subplot(221)
plot(0:1/5000:(length(vibzi)-1)/5000,vibzi)
title('Z Axis Inflation Signal')
grid on
subplot(222)
plot(0:1/5000:(length(vibzif)-1)/5000,vibzif)
grid on
title(sprintf('Z Axis Filtered Inflation Signal (N = %d)',N));
subplot(223)
semilogy(F,P1,'b:',F,P1f,'k');
axis([0 500 1e-6 1])
title('PSDs of Unfiltered and Filtered Accel Signal');
grid on;
legend('Unfiltered PSD','Filtered PSD',1);
subplot(224)
plot(0:1/5000:(length(dz)-1)/5000,(dz))
grid on;
title('Z Axis Displacement Signal');

% 3D view of accelerometer path
figure(5)
clf
subplot(121)
plot3((dx),(dy),(dz));
grid on;
title('Movement of Accelerometer');
xlabel('X axis');
ylabel('Y axis');
zlabel('Z axis');
axis square
subplot(122)
plot3((dx),(dy),(dz));
title('Movement of Accelerometer');
xlabel('X axis');
ylabel('Y axis');
zlabel('Z axis');
axis square
grid on;

% Figure 11: Frame Grab figure for motion video of inflation
```

```
figure(11)
clf
subplot(221)
plot((dx(1)),(dy(1)),'rd','markersize',8,'linewidth',3);
title('X-Y Axis Motion');
xlabel('X axis');
ylabel('Y axis');
axis([min((dx)) max((dx)) min((dy)) max((dy))])
hold on;
grid on;
subplot(222)
plot((dx(1)),(dz(1)),'rd','markersize',8,'linewidth',3);
title('X-Z Axis Motion');
xlabel('X axis');
ylabel('Z axis');
axis([min((dx)) max((dx)) min((dz)) max((dz))])
hold on;
grid on;
subplot(223)
plot((dy(1)),(dz(1)),'rd','markersize',8,'linewidth',3);
title('Y-Z Axis Motion');
xlabel('Y axis');
ylabel('Z axis');
grid on;
hold on;
axis([min((dy)) max((dy)) min((dz)) max((dz))])
subplot(224)
plot3((dx(1)),(dy(1)),(dz(2)),'rd','markersize',8,'linewidth',3);
grid on;
title('Motion of Accelerometer');
xlabel('X axis');
ylabel('Y axis');
zlabel('Z axis');
axis([min((dx)) max((dx)) min((dy)) max((dy)) min((dz)) max((dz))])
hold on;
M1(1) = getframe(gcf);
i = 0;
for k = 2:length(dx)
    if(mod(k,500) == 0)
        subplot(221)
        plot((dx(((i*500)+1):(((i+1)*500)-
1))),(dy(((i*500)+1):(((i+1)*500)-1))),'b');
        plot((dx(k)),(dy(k)),'bd','markersize',4);
        subplot(222)
        plot((dx(((i*500)+1):(((i+1)*500)-
1))),(dz(((i*500)+1):(((i+1)*500)-1))),'b');
        plot((dx(k)),(dz(k)),'bd','markersize',4);
        subplot(223)
        plot((dy(((i*500)+1):(((i+1)*500)-
1))),(dz(((i*500)+1):(((i+1)*500)-1))),'b');
        plot((dy(k)),(dz(k)),'bd','markersize',4);
        subplot(224)
        plot3((dx(((i*500)+1):(((i+1)*500)-
1))),(dy(((i*500)+1):(((i+1)*500)-1))),(dz(((i*500)+1):(((i+1)*500)-
1))),'b');
```

```
        plot3((dx(k)),(dy(k)),(dz(k)),'bd','markersize',4);
        i = i + 1;
        M1(k/500+1) = getframe(gcf);
    end
end

% Convert Matlab Movie file to avi
movie2avi(M1,'Accel_Motion','quality',100,'fps',5);
% RIGEX file to load image data file and
% output the non-interlaced and interlaced
% versions of the image

% Load image data file
load tube-1_1.dat

% Image the data using grayscale colormap
figure(1)
image(tube_1_1)
colormap(gray(256))
axis image
axis off

% Interlace the image by copying each row
tube1_1 = zeros(2*486,1134);
for i = 1:(2*486)
    tube1_1(i,:) = tube_1_1(round(i/2),:);
end

% Image the interlaced version of the image
figure(2)
image(tube1_1)
colormap(gray(256))
axis image
axis off
```

The following Matlab® script is a function call to analyze the images from the RIGEX experiment. The image data file is passed as an argument to the function call. The additional arguments are: threshold (170 used in testing), radius of the target circle in meters, peg length in meters (0 if not used), 1 or 0 if the peg is used in the target.

```
function [distance, tiltangle] = tube_image_interp(file,t,r,l,pegtest)

%-----Example Variable Settings-----------------------------
% file = 'U.dat'     % Image File (string)
% t = 170;          % Threshold (0 to 255)
% r = 0.01905;      % Target Radius (meters)
% l = 0.01898142;   % Peg Length (meters)
% pegtest = 1 yes 0 no
cx = 000;          % Horizontal Center before inflation (pixel value)
cy = 000;          % Vertical Center before inflation (pixel value)

% Load the image data from the file
e = waitbar(0,'Reading Image');
fid = fopen(file,'r');
im_vec = fscanf(fid,'%d');
fclose(fid);
test1 = zeros(486,1134);

% Convert the data from a vector to a matrix
for k = 1:486
    test1(k,:) = im_vec(((k-1)*1134+1):(k*1134))';
end

% Interlace the image by copying the rows
waitbar(1/10,e,'Resizing Image');
temp = zeros(2*486,1134);
for i = 1:(2*486)
    temp(i,:) = test1(round(i/2),:);
end
test1 = temp;

waitbar(2/10,e,'Thresholding Image');

% Set points below threshold to zero
test1th = test1;
j = find(test1 <= t);

% Set points above threshold to one
test1th(j) = 0;
j = find(test1 > t);
test1th(j) = 1;

waitbar(3/10,e,'Filtering Image');

% Remove peg from image, leaving only the ellipse
test1circ = imfill(test1th,'holes');

% Peg analysis if a peg is used in the target
```

```
if (pegtest == 1)
    waitbar(4/10,e,'Locating Peg');

    % Invert the Ellipse Picture
    test1peg = abs(test1circ-1);

    % Find the points where both the threshold image
    % and  inverted ellipse are equal (leaving only
    % the peg) and make double array
    test1peg = abs(test1peg-1) & abs(test1th-1);
    test1peg = double(test1peg);

    % Use a median filter to remove any noise present
    test1peg = medfilt2(test1peg,[3 3]);

    % Find the area that the peg is in
    STATS = regionprops(test1peg,'BoundingBox');

    % Put the Boundary into an array
    Box(1) = STATS.BoundingBox(1)-10;
    Box(2) = STATS.BoundingBox(3)+Box(1)+20;
    Box(3) = STATS.BoundingBox(2)-10;
    Box(4) = STATS.BoundingBox(4)+Box(3)+20;

    % Threshold peg image
    test1peg = edge(test1peg,'zerocross');

    % turn peg image back into a double array
    test1peg = double(test1peg);

    waitbar(5/10,e,'Procesing Peg');

    % Index the points on the edge-detected peg image
    [m,n] = find(test1peg ~= 0);
    A1 = [m n];

    % Determine the Center of the peg Image
    center = mean(A1,1);

    % Detrend data points
    o = A1(:,1)-center(1);
    p = A1(:,2)-center(2);
    A1 = [m,n,o,p];
    A2 = A1(:,3:4);
    clear m; clear n; clear o; clear p;

    % Determine the covariance of the edge-detected pixel distribution
    A2_var = cov(A2);

    % Find the eigenvectors and eigenvalues of the distribution
    % (find the axis) Minor axis is the eigenvector corresponding to
    % the largest eigenvalue Major axis is the eigenvector
    % corresponding to the smallest eigenvalue
    [U,L] = eig(A2_var);
```

```
    waitbar(6/10,e);

    % Determine vector projection of each pixel with respect to major
    % axis (eigenvector)
    pr_eigvec = zeros(length(A2(:,1)),1);
    for k = 1:length(A2(:,1))
        pr_eigvec(k) = A2(k,:)*U(:,2);
    end

    % Take histogram of all of the projections on the major axis
    [N,bin_cen] = hist(pr_eigvec,1000);

    % Flip histogram from left to right
    M = fliplr(N);

    % Find the bin with the most number of pixels
    [m,j] = max(M);

    % Determine the index of the pixels that were located in the bin
    pt_need_b = find(pr_eigvec <= bin_cen(1000-j)+.1 & pr_eigvec >=
bin_cen(1000-j)-.1);

    % See if there is more than one pixel in the bin
    if length(pt_need_b)>1
        test_ordist = zeros(length(pt_need_b),1);

        % Choosing the pixel that has the smallest projection on the
        % minor axis
        for k = 1:length(pt_need_b)
            test_ordist(k) = [A2(pt_need_b(k),1)
A2(pt_need_b(k),2)]*U(:,1);
        end
        [m,v] = min(test_ordist);
        Pt_need_b = pt_need_b(v);
    else
        % Choose the single pixel if only one to choose from
        Pt_need_b = pt_need_b;
    end
    waitbar(7/10,e);

    % Record base pixel
    pt_base = [A2(Pt_need_b,1) A2(Pt_need_b,2)];

    % Find the pixel with the largest projection magnitude (opposite
    % pixel)
    [m,pt_need_t] = max(abs(pr_eigvec));

    % Record tip pixel
    pt_tip = [A2(pt_need_t,1) A2(pt_need_t,2)];

    % Determine the euclidian distance of the peg length
    peglength = sqrt((pt_base-pt_tip)*(pt_base-pt_tip)');
end

% Determine properties of ellipse
```

```
circle =
regionprops(test1circ,'MajorAxisLength','Centroid','MinorAxisLength');

% Determine (meters/pixel) constant
c = (2*r)/(circle.MajorAxisLength);

% Find distance from camera
distance = 2156.4*c - 0.0241;

% Calculate the tilt angle from peglength or axis lengths
% and calculate the orientation if a peg is used as a target
if (pegtest == 1)
    if asin((peglength*c)/l)*(180/pi) < 30
        tiltangle = asin((peglength*c)/l)*(180/pi);
        tiltmethod = 'Peg Length';
    else
        tiltangle =
acos(circle.MinorAxisLength/circle.MajorAxisLength)*(180/pi);
        tiltmethod = 'Elliptical Analyzation';
    end
    tiltorientation = 90- atan((-1*U(2,2))/U(2,1))*(180/pi);
    if pt_tip(1,2) > 0
        tiltorientation = tiltorientation + 180;
    end
else
    tiltangle =
acos(circle.MinorAxisLength/circle.MajorAxisLength)*(180/pi);
    tiltmethod = 'Elliptical Analyzation';
end

% Find the center offset of the tube
centeroffset = (sqrt((circle.Centroid(:,1)-cx)^2+(circle.Centroid(:,2)-
cy)^2))*c;
centerorientation = atan((circle.Centroid(:,2)-cy)/(cx-
circle.Centroid(:,1)))*(180/pi);
if (cx-circle.Centroid(:,2)) > 0
    centerorientation = centerorientation +180;
end
if centerorientation < 0
    centerorientation = centerorientation +360;
end
waitbar(9/10,e,'Writing Data');

% Write the analysis data to file
filename = strcat('ImageData_',file);
fidl = fopen(filename,'w');
fprintf(fidl,'Image \t\t\t%s\n',file);
fprintf(fidl,'Major Axis Length
%4.4f\tpixels\n',circle.MajorAxisLength);
fprintf(fidl,'Distance From Camera\t%4.4f\tmeters\n',distance);
fprintf(fidl,'Tilt Angle \t\t\t%4.4f\tdegrees\n',tiltangle);
fprintf(fidl,'Tilt Angle Method\t\t%s\n',tiltmethod);
if (pegtest == 1)
    fprintf(fidl,'Tilt Orientation
\t\t%4.4f\tdegrees\n',tiltorientation);
```

```
end
fprintf(fidl,'Center Offset \t\t%4.4f\tmeters\n',centeroffset);
fprintf(fidl,'Center Orientation
\t%4.4f\tdegrees\n',centerorientation);
fclose(fidl);

% Load the watermark key for signature
load wmark_key;
gain = 2;

% Encoded the signature
info_data1 = (dec2bin(double(sprintf('AFIT RIGEX GE04M!!'))))';
data1 = (str2num(info_data1(:)))';
data1(find(data1 == 0)) = -1;
data1_mat = zeros(18,7);
for k = 1:18
    data1_mat(k,:) = data1(((k-1)*7+1):(k*7));
end

% Embedding the signature
wmark = kron(data1_mat,key);

% Writing output image file
out_image = test1 + wmark*gain;
imwrite(uint8(out_image),strcat(file(1:(length(file)-4),'.bmp'),'bmp');

waitbar(10/10,e);
close(e);

return
```

```
% RIGEX Tube Experiment Data Interpretation Script
% David C. Moody 1st Lt. GE-04M
% This script will evaluate the results of running the RIGEX experiment
% using tube 1, other tubes can be analyzed by changing the tube number
% in the loading data section
% The script will output 11 figures:
% Figure 1: Pressure Signal
% Figure 2: X Axis Inflation/Displacement Signal
% Figure 3: Y Axis Inflation/Displacement Signal
% Figure 4: Z Axis Inflation/Displacement Signal
% Figure 5: 3D Plot of Accelerometer Motion
% Figure 6: Temperature Signal
% Figure 7: Excitation Vibration Signals
% Figure 8: Transfer Functions from excitation signals
% Figure 9: PSDs of the signals during excitation
% Figure 10: Smoothed Transfer Functions (4pt average) from excitation
% Figure 11: Frame Grab figure for motion video of inflation

% Loading data
load tube-1_excite.dat
load tube-1_inflation.dat
load tube_1_temp.dat
load tube-1_gas_str.dat

% Converting pressure data to voltage
press_inflate = tube_1_inflation(:,4)*0.625/32768;

% Converting inflation vibration data to voltage signals
vibx_inflate =
(tube_1_inflation(:,1)+32768*ones(size(tube_1_inflation(:,1))))*5/65536
;
viby_inflate =
(tube_1_inflation(:,2)+32768*ones(size(tube_1_inflation(:,2))))*5/65536
;
vibz_inflate =
(tube_1_inflation(:,3)+32768*ones(size(tube_1_inflation(:,3))))*5/65536
;

% Remove means from inflation vibration data
vibxi = vibx_inflate - mean(vibx_inflate)*ones(size(vibx_inflate));
vibyi = viby_inflate - mean(viby_inflate)*ones(size(viby_inflate));
vibzi = vibz_inflate - mean(vibz_inflate)*ones(size(vibz_inflate));

% FIR Lowpas filter to filter noise off inflation vibrations
[N,Fo,Ao,W] = REMEZORD([50 60],[1 0],[0.01 0.1],5000);
filt_coeff = remez(N,Fo,Ao,W);

% Get the unfiltered inflation acceleration signal PSDs
[P1,Finf] = psd(vibxi,4096,5000);
[P2,Finf] = psd(vibyi,4096,5000);
[P3,Finf] = psd(vibzi,4096,5000);

% Filter noise off inflation vibration signals
vibxif = filter(filt_coeff,1,vibxi);
vibyif = filter(filt_coeff,1,vibyi);
```

```
vibzif = filter(filt_coeff,1,vibzi);

% Get the filtered inflation accleration signal PSDs
[P1f,Finf] = psd(vibxif,4096,5000);
[P2f,Finf] = psd(vibyif,4096,5000);
[P3f,Finf] = psd(vibzif,4096,5000);

% Convert inflation voltage signal to g's
vibxif = vibxif/186e-3;
vibyif = vibyif/186e-3;
vibzif = vibzif/186e-3;

% Perform running sums to get displacement signals
ddx = zeros(size(vibxif)); dx = ddx;
ddy = zeros(size(vibyif)); dy = ddy;
ddz = zeros(size(vibzif)); dz = ddz;
h = waitbar(0,'Performing first running sum on inflation
acceleration...');
for k = 1:(length(vibxif))
    ddx(k) = sum(vibxif(1:k))/5000;
    ddy(k) = sum(vibyif(1:k))/5000;
    ddz(k) = sum(vibzif(1:k))/5000;
    waitbar(k/length(vibxif),h);
end
close(h);
h = waitbar(0,'Performing second running sum on inflation
velocities...');
for k = 1:(length(vibxif))
    dx(k) = sum(ddx(1:k))/5000;
    dy(k) = sum(ddy(1:k))/5000;
    dz(k) = sum(ddz(1:k))/5000;
    waitbar(k/length(vibxif),h);
end
close(h);

% Converting storage pressure data to voltage
gas_storage = tube_1_gas_str*0.625/32768;
fprintf('Storage Gas Pressure Voltage: %2.2e\n',gas_storage);

% Loading temperature data
comp_temp = tube_1_temp(:,1);
high_ch_temp = tube_1_temp(:,2);
low_ch_temp = tube_1_temp(:,3);
environ_temp = tube_1_temp(:,4);

% Converting excitation vibration data to voltage
vibx_excite =
(tube_1_excite(:,2)+32768*ones(size(tube_1_excite(:,2))))*5/65536;
viby_excite =
(tube_1_excite(:,3)+32768*ones(size(tube_1_excite(:,3))))*5/65536;
vibz_excite =
(tube_1_excite(:,4)+32768*ones(size(tube_1_excite(:,4))))*5/65536;

% Converting excitation waveform data to voltage
```

```
DAC_out = (tube_1_excite(:,1)-
2048*ones(size(tube_1_excite(:,1))))*5/2048;

% Remove the DC bias from the excitation vibration signals
vibx_excite = vibx_excite - ones(size(vibx_excite))*mean(vibx_excite);
viby_excite = viby_excite - ones(size(viby_excite))*mean(viby_excite);
vibz_excite = vibz_excite - ones(size(vibz_excite))*mean(vibz_excite);

% FIR Lowpas filter to filter noise off excitation vibrations
[N,Fo,Ao,W] = REMEZORD([1000 1020],[1 0],[0.01 0.1],5000);
filt_coeff = remez(N,Fo,Ao,W);

% Filter the excitation signals
vibx_excite = filter(filt_coeff,1,vibx_excite);
viby_excite = filter(filt_coeff,1,viby_excite);
vibz_excite = filter(filt_coeff,1,vibz_excite);

% Using Bilinear transformation to convert DAC smoothing filter
% equations to z domain equations
[num1d,den1d] = bilinear([3.94784176e7],[1 1.23249113e4
3.9478176e7],5000);
[num2d,den2d] = bilinear([3.94784176e7],[1 1.04485553e4
3.9478176e7],5000);
[num3d,den3d] = bilinear([3.94784176e7],[1 6.98150145e3
3.9478176e7],5000);
[num4d,den4d] = bilinear([3.94784176e7],[1 2.45157729e3
3.9478176e7],5000);

% Smoothing digitized excitation waveform with filter
y_dacout =
filter(num4d,den4d,filter(num3d,den3d,filter(num2d,den2d,filter(num1d,d
en1d,DAC_out))));

% Smoothing temperature signals
h = ones(10,1)/10;
high_ch_temp_ave = conv2(high_ch_temp,h,'valid');
low_ch_temp_ave = conv2(low_ch_temp,h,'valid');
environ_temp_ave = conv2(environ_temp,h,'valid');

% Transfer function estimation for each of the three accelerometer axes
[Tx,F] = tfe(y_dacout,vibx_excite,4096,5000);
[Ty,F] = tfe(y_dacout,viby_excite,4096,5000);
[Tz,F] = tfe(y_dacout,vibz_excite,4096,5000);

% PSD estimation for the smoothed DAC signal and vibration signals
[Pdac,Fp] = psd(y_dacout,4096,5000,boxcar(4096));
[Pvibx,Fp] = psd(vibx_excite,4096,5000);
[Pviby,Fp] = psd(viby_excite,4096,5000);
[Pvibz,Fp] = psd(vibz_excite,4096,5000);

% 4 point average smoothing of transfer functions
h = ones(4,1)/4;
Txf = conv2(Tx,h);
Tyf = conv2(Ty,h);
Tzf = conv2(Tz,h);
```

```
% Figure 1: Pressure Signal
figure(1)
clf
plot((0:length(press_inflate)-1)/5000,press_inflate);
grid on;
title('Pressure Signal during Inflation')
xlabel('Time (sec)')
ylabel('Pressure Voltage');

% Figure 2: X Axis Inflation/Displacement Signal
figure(2)
clf
subplot(221)
plot(0:1/5000:(length(vibxi)-1)/5000,vibxi)
title('X Axis Inflation Signal')
grid on
subplot(222)
plot(0:1/5000:(length(vibxif)-1)/5000,vibxif)
grid on
title(sprintf('X Axis Filtered Inflation Signal (N = %d)',N));
subplot(223)
semilogy(Finf,P1,'b:',Finf,P1f,'k');
axis([0 500 1e-6 1])
title('PSDs of Unfiltered and Filtered Accel Signal');
grid on;
legend('Unfiltered PSD','Filtered PSD',1);
subplot(224)
plot(0:1/5000:(length(dx)-1)/5000,(dx))
grid on;
title('X Axis Displacement Signal');

% Figure 3: Y Axis Inflation/Displacement Signal
figure(3)
subplot(221)
plot(0:1/5000:(length(vibyi)-1)/5000,vibyi)
title('Y Axis Inflation Signal')
grid on
subplot(222)
plot(0:1/5000:(length(vibyif)-1)/5000,vibyif)
grid on
title(sprintf('Y Axis Filtered Inflation Signal (N = %d)',N));
subplot(223)
semilogy(Finf,P2,'b:',Finf,P2f,'k');
axis([0 500 1e-6 1])
title('PSDs of Unfiltered and Filtered Accel Signal');
grid on;
legend('Unfiltered PSD','Filtered PSD',1);
subplot(224)
plot(0:1/5000:(length(dy)-1)/5000,(dy))
grid on;
title('Y Axis Displacement Signal');

% Figure 4: Z Axis Inflation/Displacement Signal
figure(4)
```

```
subplot(221)
plot(0:1/5000:(length(vibzi)-1)/5000,vibzi)
title('Z Axis Inflation Signal')
grid on
subplot(222)
plot(0:1/5000:(length(vibzif)-1)/5000,vibzif)
grid on
title(sprintf('Z Axis Filtered Inflation Signal (N = %d)',N));
subplot(223)
semilogy(Finf,P3,'b:',Finf,P3f,'k');
axis([0 500 1e-6 1])
title('PSDs of Unfiltered and Filtered Accel Signal');
grid on;
legend('Unfiltered PSD','Filtered PSD',1);
subplot(224)
plot(0:1/5000:(length(dz)-1)/5000,(dz))
grid on;
title('Z Axis Displacement Signal');

% Figure 5: 3D Plot of Accelerometer Motion
figure(5)
clf
subplot(121)
plot3((dx),(dy),(dz));
grid on;
title('Movement of Accelerometer');
xlabel('X axis');
ylabel('Y axis');
zlabel('Z axis');
axis square
subplot(122)
plot3((dx),(dy),(dz));
title('Movement of Accelerometer');
xlabel('X axis');
ylabel('Y axis');
zlabel('Z axis');
axis square
grid on;

% Figure 6: Temperature Signal
figure(6)
clf
subplot(221)
hold on;
plot(high_ch_temp,'r');
plot(high_ch_temp_ave,'k')
plot(128*ones(length(high_ch_temp),1),':b');
grid on;
title('Measured High T/C Temperature Signal');
ylabel('Temperature (C)')
xlabel('Time (sec)')
legend('Measured Temperature','Averaged Temperature','130 deg C
Threshold',4)
subplot(222)
hold on;
```

```
plot(low_ch_temp,'r');
plot(low_ch_temp_ave,'k-.')
plot(128*ones(length(low_ch_temp),1),':b');
grid on;
title('Measured Low T/C Temperature Signal');
ylabel('Temperature (C)')
xlabel('Time (sec)')
legend('Measured Temp.','Averaged Temp.','130 deg C Threshold',4)
subplot(223)
plot(environ_temp_ave);
grid on;
title('Measured Structure Temperature')
ylabel('Temperature (C)');
xlabel('Time (sec)');
subplot(224)
plot(comp_temp);
grid on;
title('Measured Compensation Temp. Differential')
ylabel('Temperature (C)');
xlabel('Time (sec)');

% Figure 7: Excitation Vibration Signals
figure(7)
clf
subplot(311)
plot((0:length(vibx_excite)-1)/5000,vibx_excite);
grid on;
ylabel('X Axis Excitation Signal');
title('Excitation Vibration Signals')
subplot(312)
plot((0:length(viby_excite)-1)/5000,viby_excite);
grid on;
ylabel('Y Axis Excitation Signal');
subplot(313)
plot((0:length(vibz_excite)-1)/5000,vibz_excite);
grid on;
ylabel('Z Axis excitation Signal');
xlabel('Time (sec)');

% Figure 8: Transfer Functions from excitation signals
figure(8)
clf
subplot(131)
semilogy(F,abs(Tx))
grid on;
title('Transfer Functions of X axis')
ylabel('Magnitude Transfer Function')
xlabel('Frequency (Hz)');
axis([0 1000 0.00001 0.1])
axis square
subplot(132)
semilogy(Fp,abs(Ty))
grid on;
title('Transfer Functions of Y axis')
ylabel('Magnitude Transfer Function')
```

```
xlabel('Frequency (Hz)');
axis([0 1000 0.0001 0.1])
axis square
subplot(133)
semilogy(F,abs(Tz))
grid on;
title('Transfer Functions of Z axis')
ylabel('Magnitude Transfer Function')
xlabel('Frequency (Hz)');
axis([0 1000 0.00001 0.1])
axis square

% Figure 9: PSDs of the signals during excitation
figure(9)
clf
subplot(221)
semilogy(Fp,abs(Pdac));
xlabel('Frequency (Hz)');
ylabel('Magnitude (dB)');
title('PSD of Excitation Waveform');
grid on;
subplot(222)
semilogy(Fp,abs(Pvibx));
grid on;
xlabel('Frequency (Hz)');
ylabel('Magnitude (dB)');
title('PSD of Accelerometer X Output');
subplot(223)
semilogy(Fp,abs(Pviby));
grid on;
xlabel('Frequency (Hz)');
ylabel('Magnitude (dB)');
title('PSD of Accelerometer Y Output');
subplot(224)
semilogy(Fp,abs(Pvibz));
grid on;
xlabel('Frequency (Hz)');
ylabel('Magnitude (dB)');
title('PSD of Accelerometer Z Output');

% Figure 10: Smoothed Transfer Functions (4pt average) from excitation
figure(10)
clf
subplot(131)
semilogy(F,abs(Txf(4:length(Txf))))
grid on;
title('Smoothed TF of X axis')
ylabel('Magnitude Transfer Function')
xlabel('Frequency (Hz)');
axis([0 1000 0.00001 0.1])
axis square
subplot(132)
semilogy(F,abs(Tyf(4:length(Tyf))))
grid on;
title('Smoothed TF of Y axis')
```

```
ylabel('Magnitude Transfer Function')
xlabel('Frequency (Hz)');
axis([0 1000 0.0001 0.1])
axis square
subplot(133)
semilogy(F,abs(Tzf(4:length(Tzf))))
grid on;
title('Smoothed TF of Z axis')
ylabel('Magnitude Transfer Function')
xlabel('Frequency (Hz)');
axis([0 1000 0.00001 0.1])
axis square

% Figure 11: Frame Grab figure for motion video of inflation
figure(11)
clf
subplot(221)
plot((dx(1)),(dy(1)),'rd','markersize',8,'linewidth',3);
title('X-Y Axis Motion');
xlabel('X axis');
ylabel('Y axis');
axis([min((dx)) max((dx)) min((dy)) max((dy))])
hold on;
grid on;
subplot(222)
plot((dx(1)),(dz(1)),'rd','markersize',8,'linewidth',3);
title('X-Z Axis Motion');
xlabel('X axis');
ylabel('Z axis');
axis([min((dx)) max((dx)) min((dz)) max((dz))])
hold on;
grid on;
subplot(223)
plot((dy(1)),(dz(1)),'rd','markersize',8,'linewidth',3);
title('Y-Z Axis Motion');
xlabel('Y axis');
ylabel('Z axis');
grid on;
hold on;
axis([min((dy)) max((dy)) min((dz)) max((dz))])
subplot(224)
plot3((dx(1)),(dy(1)),(dz(2)),'rd','markersize',8,'linewidth',3);
grid on;
title('Motion of Accelerometer');
xlabel('X axis');
ylabel('Y axis');
zlabel('Z axis');
axis([min((dx)) max((dx)) min((dy)) max((dy)) min((dz)) max((dz))])
hold on;
M1(1) = getframe(gcf);
i = 0;
for k = 2:length(dx)
    if(mod(k,500) == 0)
        subplot(221)
```

```
        plot((dx(((i*500)+1):(((i+1)*500)-
1))),(dy(((i*500)+1):(((i+1)*500)-1))),'b');
        plot((dx(k)),(dy(k)),'bd','markersize',4);
        subplot(222)
        plot((dx(((i*500)+1):(((i+1)*500)-
1))),(dz(((i*500)+1):(((i+1)*500)-1))),'b');
        plot((dx(k)),(dz(k)),'bd','markersize',4);
        subplot(223)
        plot((dy(((i*500)+1):(((i+1)*500)-
1))),(dz(((i*500)+1):(((i+1)*500)-1))),'b');
        plot((dy(k)),(dz(k)),'bd','markersize',4);
        subplot(224)
        plot3((dx(((i*500)+1):(((i+1)*500)-
1))),(dy(((i*500)+1):(((i+1)*500)-1))),(dz(((i*500)+1):(((i+1)*500)-
1))),'b');
        plot3((dx(k)),(dy(k)),(dz(k)),'bd','markersize',4);
        i = i + 1;
        M1(k/500+1) = getframe(gcf);
    end
end

% Convert Matlab movie to avi
movie2avi(M1,'Accel_Motion','quality',100,'fps',5);
```

# Appendix F:  Experiment C Code

```
// RIGEX:  Data Acquisition Program Routine
// Routine executes the RIGEX experiment while collecting and
// temperature, pressure, and vibration data.  The program will produce
// 12 data files, 4 data files for each tube tested.  The program will
// also update a required file called rigex_failsafe.dat.  This file is
// used to monitor how far in the experiment the computer has
// progressed in the case of power failure.

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

// Global variables containing the addresses for the different boards
int AD_addr = 0x380;
int temp_addr = 0x300;
int relay_addr = 0x240;
int timer_addr = 0x2C0;

// Function declarations for the data collection subroutines
int temp_data_collect(int);
int inflate_data_collect(int);
int gas_str_check(int);
int excite_data_collect(int);

int main(void){

      // Needed variables
      FILE *failsafe_file;
      int failsafe;
      int pin_puller_pause = 2;
      int i = 0;
      int k = 0;
      int status;
      int cool_down_pause = 15;


      _outp(timer_addr+2,0);

    // Initialize A/D Board
      printf("Reseting the A/D board\n");
      _outp(AD_addr+8,32);

      // Enable AD (internal timer controlled) and Timer 0
      // interrupt interrupts occur on base+9 read
      printf("Initializing Timer 0\n");
      _outp(AD_addr+9,0x20);

      // Configure timer 0 to use internal clock source
      printf("Configure timer 0 to use internal clock source\n");
      _outp(AD_addr+10,0xC2);

      // set counter 0 to mode 2 operation (clk source)
      printf("%d\n",_inp(AD_addr+10));
      _outp(AD_addr+15,0x14);
      _outp(AD_addr+12,0x02);
```

```
printf("Initializing Timer Board...\n");
_outp(timer_addr+1,0x17);     // Access Master Mode Register
_outp(timer_addr+0,0xF0);     // Write LSB to MM Register
_outp(timer_addr+0,0x51);     // Write MSB to MM Register


// Check status of imaging computer
printf("Checking if imaging computer is ready to proceed...\n");
do{
      imaging_status = _inp(timer_addr+2) & 0x80;
}while(imaging_status != 0x80);

// Check Failsafe File
failsafe_file = fopen("rigex_failsafe.dat","r");
fscanf(failsafe_file,"%d",&failsafe);
fclose(failsafe_file);
printf("Checking Failsafe value:  %d\n",failsafe);
if (failsafe !=0){
      if (failsafe == 10)
            goto Tube10;
      if (failsafe == 11)
            goto Tube11;
      if (failsafe == 12)
            goto Tube12;
      if (failsafe == 13)
            goto Tube13;
      if (failsafe == 14)
            goto Tube14;
      if (failsafe == 15)
            goto Tube15;
      if (failsafe == 20)
            goto Tube20;
      if (failsafe == 21)
            goto Tube21;
      if (failsafe == 22)
            goto Tube22;
      if (failsafe == 23)
            goto Tube23;
      if (failsafe == 24)
            goto Tube24;
      if (failsafe == 25)
            goto Tube25;
      if (failsafe == 30)
            goto Tube30;
      if (failsafe == 31)
            goto Tube31;
      if (failsafe == 32)
            goto Tube32;
      if (failsafe == 33)
            goto Tube33;
      if (failsafe == 34)
            goto Tube34;
      if (failsafe == 35)
            goto Tube35;
      else
```

```
                goto Data_collect;
        }


        /* Tube 1 Process*/
        // Mark failsafe point
        failsafe_file = fopen("rigex_failsafe.dat","w");
        fprintf(failsafe_file,"%d",10);
        fclose(failsafe_file);

        // Activate Tube 1 Heaters and lights
            printf("Activating Heaters and Lights...\n");
Tube10:     _outp(relay_addr+0,1);
            _outp(relay_addr+1,0x40);

        // Collect temperature data and check versus threshold
        printf("Collecting Temperature Data...\n");
        temp_data_collect(1);
        printf("Threshold Temperature Achieved...\n");

        // Sample Gas Storage Container
        printf("Checking Gas Storage Pressure...\n");
        gas_str_check(1);


        // Mark failsafe point
        failsafe_file = fopen("rigex_failsafe.dat","w");
        fprintf(failsafe_file,"%d",11);
        fclose(failsafe_file);

        // Open Heater Box and Inflation Valve
        printf("Opening Heater Box...\n");
Tube11: _outp(relay_addr+0,3);
        for(k = 0; k < pin_puller_pause; k++){
              i = 0;
              while(i<5000){
              // Loop to count 5000 cycles of clk
                    do {
                          // load status register
                          status = _inp(AD_addr+9) & 0x20;
                    } while(status != 32);  // check for timing interrupt
                    _outp(AD_addr+8,0x08);  // Activate interrupts
                    i++;
              }
        }
        printf("Opening Gas Valve...\n");
        _outp(relay_addr+0,4);

        // Signal Imaging Computer
        printf("Signaling Imaging Computer...\n");
        do{
              imaging_status = _inp(timer_addr+2) & 0x80;
        }while(imaging_status != 0x80);
        _outp(AD_addr+1,1);
        _outp(timer_addr+2,0x80);
```

```
	// Sample Pressure and Vibration Upon Inflation
	printf("Inflation Data being collected...\n");
	inflate_data_collect(1);

	// Pause to cool the tube
	printf("Cooling down...\n");
	for(k = 0; k < cool_down_pause; k++){
		i = 0;
		while(i<5000){
			// Loop to count 5000 cycles of clk
			do {
				// load status register
				status = _inp(AD_addr+9) & 0x20;
			} while(status != 32);// check for timing interrupt
			_outp(AD_addr+8,0x08);// Activate interrupts
			i++;
		}
	}

	// Vent the gas from the tube
	printf("Venting Gas...\n");
	_outp(relay_addr+0,0);

	// Halt Imaging
	printf("Halt Imaging...\n");
	_outp(timer_addr+2,0);

	// Mark failsafe point
	failsafe_file = fopen("rigex_failsafe.dat","w");
	fprintf(failsafe_file,"%d",12);
	fclose(failsafe_file);

	// Signal Imaging Computer
	printf("Signal Imaging Computer for one image...\n");
	do{
		imaging_status = _inp(timer_addr+2) & 0x80;
	}while(imaging_status != 0x80);

Tube12:	_outp(AD_addr+1,1);
	_outp(timer_addr+2,0x80);


	// Mark failsafe point
	failsafe_file = fopen("rigex_failsafe.dat","w");
	fprintf(failsafe_file,"%d",13);
	fclose(failsafe_file);

	// Excite Tube and measure vibrations
	printf("Exciting the tube and collecting data...\n");
Tube13:	excite_data_collect(1);

	// Mark failsafe point
	failsafe_file = fopen("rigex_failsafe.dat","w");
```

```
      fprintf(failsafe_file,"%d",14);
      fclose(failsafe_file);

      // Signal Imaging Computer
      printf("Signal imaging computer for one picture...\n");
Tube14:     do{
                  imaging_status = _inp(timer_addr+2) & 0x80;
            }while(imaging_status != 0x80);
            _outp(AD_addr+1,1);
            _outp(timer_addr+2,0x80);

      // Mark failsafe point
      failsafe_file = fopen("rigex_failsafe.dat","w");
      fprintf(failsafe_file,"%d",15);
      fclose(failsafe_file);



Tube15:     _outp(relay_addr+0,0);
            // Shut everything off except lights
            _outp(relay_addr+1,0x40);
            _outp(timer_addr+2,0);

      /* Tube 2 Process*/
      // Mark failsafe point
      failsafe_file = fopen("rigex_failsafe.dat","w");
      fprintf(failsafe_file,"%d",20);
      fclose(failsafe_file);

      // Activate Tube 1 Heaters and lights
          printf("Activating Heaters and Lights...\n");
Tube20:     _outp(relay_addr+0,0x08);


      // Collect temperature data and check versus threshold
      printf("Collecting Temperature Data...\n");
      temp_data_collect(2);
      printf("Threshold Temperature Achieved...\n");

      // Sample Gas Storage Container
      printf("Checking Gas Storage Pressure...\n");
      gas_str_check(2);


      // Mark failsafe point
      failsafe_file = fopen("rigex_failsafe.dat","w");
      fprintf(failsafe_file,"%d",21);
      fclose(failsafe_file);

      // Open Heater Box and Inflation Valve
      printf("Opening Heater Box...\n");
Tube21: _outp(relay_addr+0,0x18);
      for(k = 0; k < pin_puller_pause; k++){
            i = 0;
```

```
      while(i<5000){
            do {
                  // load status register
                  status = _inp(AD_addr+9) & 0x20;
            } while(status != 32);// check for timing interrupt
            _outp(AD_addr+8,0x08);// Activate interrupts
            i++;
      }
}
printf("Opening Gas Valve...\n");
_outp(relay_addr+0,0x20);

// Signal Imaging Computer
printf("Signaling Imaging Computer...\n");
do{
      imaging_status = _inp(timer_addr+2) & 0x80;
}while(imaging_status != 0x80);
_outp(AD_addr+1,2);
_outp(timer_addr+2,0x80);


// Sample Pressure and Vibration Upon Inflation
printf("Inflation Data being collected...\n");
inflate_data_collect(2);

// Pause to cool the tube
printf("Cooling down...\n");
for(k = 0; k < cool_down_pause; k++){
      i = 0;
      while(i<5000){
            do {
                  // load status register
                  status = _inp(AD_addr+9) & 0x20;
            } while(status != 32);// check for timing interrupt
            _outp(AD_addr+8,0x08);// Activate interrupts
            i++;
      }
}

// Vent the gas from the tube
printf("Venting Gas...\n");
_outp(relay_addr+0,0);

// Halt Imaging
printf("Halt Imaging...\n");
_outp(timer_addr+2,0);

// Mark failsafe point
failsafe_file = fopen("rigex_failsafe.dat","w");
fprintf(failsafe_file,"%d",22);
fclose(failsafe_file);

// Signal Imaging Computer
printf("Signal Imaging Computer for one image...\n");
do{
```

```
            imaging_status = _inp(timer_addr+2) & 0x80;
      }while(imaging_status != 0x80);

Tube22:     _outp(AD_addr+1,2);
      _outp(timer_addr+2,0x80);


      // Mark failsafe point
      failsafe_file = fopen("rigex_failsafe.dat","w");
      fprintf(failsafe_file,"%d",23);
      fclose(failsafe_file);

      // Excite Tube and measure vibrations
          printf("Exciting the tube and collecting data...\n");
Tube23:     excite_data_collect(2);

      // Mark failsafe point
      failsafe_file = fopen("rigex_failsafe.dat","w");
      fprintf(failsafe_file,"%d",24);
      fclose(failsafe_file);

      // Signal Imaging Computer
          printf("Signal imaging computer for one picture...\n");
Tube24:     do{
                  imaging_status = _inp(timer_addr+2) & 0x80;
            }while(imaging_status != 0x80);
            _outp(AD_addr+1,2);
            _outp(timer_addr+2,0x80);

      // Mark failsafe point
      failsafe_file = fopen("rigex_failsafe.dat","w");
      fprintf(failsafe_file,"%d",25);
      fclose(failsafe_file);



Tube25:     _outp(relay_addr+0,0);
            _outp(relay_addr+1,0x40);
            _outp(timer_addr+2,0);


      /* Tube 3 Process*/
      // Mark failsafe point
      failsafe_file = fopen("rigex_failsafe.dat","w");
      fprintf(failsafe_file,"%d",30);
      fclose(failsafe_file);

      // Activate Tube 1 Heaters and lights
          printf("Activating Heaters and Lights...\n");
Tube30:     _outp(relay_addr+1,0x41);

      // Collect temperature data and check versus threshold
      printf("Collecting Temperature Data...\n");
      temp_data_collect(3);
      printf("Threshold Temperature Achieved...\n");
```

```
	// Sample Gas Storage Container
	printf("Checking Gas Storage Pressure...\n");
	gas_str_check(3);



	// Mark failsafe point
	failsafe_file = fopen("rigex_failsafe.dat","w");
	fprintf(failsafe_file,"%d",31);
	fclose(failsafe_file);

	// Open Heater Box and Inflation Valve
	 printf("Opening Heater Box...\n");
Tube31: _outp(relay_addr+1,0x43);
	for(k = 0; k < pin_puller_pause; k++){
		i = 0;
		while(i<5000){
			do {
				// load status register
				status = _inp(AD_addr+9) & 0x20;
			} while(status != 32);// check for timing interrupt
			_outp(AD_addr+8,0x08);// Activate interrupts
			i++;
		}
	}
	printf("Opening Gas Valve...\n");
	_outp(relay_addr+1,0x44);

	// Signal Imaging Computer
	printf("Signaling Imaging Computer...\n");
	do{
		imaging_status = _inp(timer_addr+2) & 0x80;
	}while(imaging_status != 0x80);
	_outp(AD_addr+1,3);
	_outp(timer_addr+2,0x80);


	// Sample Pressure and Vibration Upon Inflation
	printf("Inflation Data being collected...\n");
	inflate_data_collect(3);

	// Pause to cool the tube
	printf("Cooling down...\n");
	for(k = 0; k < cool_down_pause; k++){
		i = 0;
		while(i<5000){
			do {
				// load status register
				status = _inp(AD_addr+9) & 0x20;
			} while(status != 32); // check for timing interrupt
			_outp(AD_addr+8,0x08); // Activate interrupts
			i++;
		}
	}
```

```
      // Vent the gas from the tube
      printf("Venting Gas...\n");
      _outp(relay_addr+1,0x40);

      // Halt Imaging
      printf("Halt Imaging...\n");
      _outp(timer_addr+2,0);

      // Mark failsafe point
      failsafe_file = fopen("rigex_failsafe.dat","w");
      fprintf(failsafe_file,"%d",32);
      fclose(failsafe_file);

      // Signal Imaging Computer
      printf("Signal Imaging Computer for one image...\n");
      do{
            imaging_status = _inp(timer_addr+2) & 0x80;
      }while(imaging_status != 0x80);

Tube32:     _outp(AD_addr+1,3);
      _outp(timer_addr+2,0x80);


      // Mark failsafe point
      failsafe_file = fopen("rigex_failsafe.dat","w");
      fprintf(failsafe_file,"%d",33);
      fclose(failsafe_file);

      // Excite Tube and measure vibrations
          printf("Exciting the tube and collecting data...\n");
Tube33:     excite_data_collect(3);

      // Mark failsafe point
      failsafe_file = fopen("rigex_failsafe.dat","w");
      fprintf(failsafe_file,"%d",34);
      fclose(failsafe_file);

      // Signal Imaging Computer
          printf("Signal imaging computer for one picture...\n");
Tube34:     do{
                  imaging_status = _inp(timer_addr+2) & 0x80;
            }while(imaging_status != 0x80);
            _outp(AD_addr+1,3);
            _outp(timer_addr+2,0x80);

      // Mark failsafe point
      failsafe_file = fopen("rigex_failsafe.dat","w");
      fprintf(failsafe_file,"%d",35);
      fclose(failsafe_file);



Tube35:     _outp(relay_addr+0,0);
            _outp(relay_addr+1,0);
```

```
      // Mark failsafe point
      failsafe_file = fopen("rigex_failsafe.dat","w");
      fprintf(failsafe_file,"%d",40);
      fclose(failsafe_file);

      _outp(relay_addr+1,0x80);

Data_collect: _outp(timer_addr+2,0x40); //Notify image computer to end
      return 0;

}


int temp_data_collect(int tube_num){
      float temps[10] = {0};
      float temp_comp;
      FILE *temp_data;
      float temp_threshold = 130;
      float temp_high,temp_low,temp_struct;
      int i,k;
      const int ref_temp = 21;
      int status;
      float temp_ave =0;
      float temp_sum = 0;
      int ch_high,ch_low;
      char filename[15];
      float br_temp;

      if (tube_num == 1)
      {
            ch_high = 0;
            ch_low = 1;
      }
      if (tube_num == 2)
      {
            ch_high = 2;
            ch_low = 3;
      }
      if (tube_num == 3)
      {
            ch_high = 4;
            ch_low = 5;
      }

      sprintf(filename,"tube_%d_temp.dat",tube_num);
      temp_data = fopen(filename,"w");

      while (temp_ave<temp_threshold){
            i = 0;
            while(i<5000){      // Loop to go through the data array
                  do {          // Loop to wait for timing interrupt
                        // load status register
                        status = _inp(AD_addr+9) & 0x20;
                  }while(status != 32);  // check for timing interrupt
```

```
                  _outp(AD_addr+8,0x08);  // Activate interrupts
                  i++;
            }

            _outp(temp_addr,0x40+7);
            do{
                  status = _inp(temp_addr+8) & 1;
            }while(status != 0);

            br_temp = 98.5*((_inp(temp_addr+1) &
0x0F)*256+_inp(temp_addr+0))*5/4096;

            temp_comp = br_temp - ref_temp;

            _outp(temp_addr+0,0x48+ch_high);
            do{
                  status = _inp(temp_addr+8) & 1;
            }while(status != 0);

            if (((_inp(temp_addr+1) & 0x0F)*256+_inp(temp_addr+0)) >=
2048)

                  temp_high = 98.5*(((_inp(temp_addr+1) &
0x0F)*256+_inp(temp_addr+0))-4096)*5/2048 - temp_comp;

            else
                  temp_high = 98.5*((_inp(temp_addr+1) &
0x0F)*256+_inp(temp_addr+0))*5/2048 - temp_comp;

            _outp(temp_addr+0,0x48+ch_low);
            do{
                  status = _inp(temp_addr+8) & 1;
            }while(status != 0);
            if (((_inp(temp_addr+1) & 0x0F)*256+_inp(temp_addr+0)) >=
2048)

                  temp_low = 98.5*(((_inp(temp_addr+1) &
0x0F)*256+_inp(temp_addr+0))-4096)*5/2048 - temp_comp;

            else
                  temp_low = 98.5*((_inp(temp_addr+1) &
0x0F)*256+_inp(temp_addr+0))*5/2048 - temp_comp;

            _outp(temp_addr+0,0x48+6);
            do{
                  status = _inp(temp_addr+8) & 1;
            }while(status != 0);
            if (((_inp(temp_addr+1) & 0x0F)*256+_inp(temp_addr+0)) >=
2048)

                  temp_struct = 98.5*(((_inp(temp_addr+1) &
0x0F)*256+_inp(temp_addr+0))-4096)*5/2048 - temp_comp;

            else
```

```
                  temp_struct = 98.5*((_inp(temp_addr+1) &
0x0F)*256+_inp(temp_addr+0))*5/2048 - temp_comp;

            fprintf(temp_data,"%f  %f  %f
%f\n",temp_comp,temp_high,temp_low,temp_struct);

            // Shift the 10 last samples down the filter
            for(k = 8;k>=0;k--){
                  temps[k+1] = temps[k];
            }
            // assign most recent sample
            temps[0] = temp_high;

            // add up 10 samples
            for(k = 0;k<10;k++){
                  temp_sum = temp_sum + temps[k];
            }
            // average
            temp_ave = temp_sum/10;
            // reset sum
            temp_sum = 0;
      }
      fclose(temp_data);
      return 0;
}

int inflate_data_collect(int tube_num){
      int inflation_time = 5;
      int i,k;
      int status;
      int ch_high,ch_low,ch_press;
      short signed int
      Xdigi[25000],Ydigi[25000],Zdigi[25000],press[25000];
      FILE *vibdat;
      int MSBad,LSBad,MSBpr,LSBpr;
      char filename[20];
      int num_samp =25000;

      if (tube_num ==1)
      {
            ch_high = 12;
            ch_low = 10;
            ch_press = 0;
      }
      if (tube_num == 2)
      {
            ch_high = 15;
            ch_low = 13;
            ch_press = 1;
      }
      if (tube_num == 3)
      {
            ch_high = 26;
            ch_low = 24;
            ch_press = 2;
```

```
}

k = 0;
while(k<inflation_time){
      i = 0;
      while(i<5000){
            do {
                  // load status register
                  status = _inp(AD_addr+9) & 0x20;
            } while(status != 32);

            _outp(AD_addr+8,0x08);

            // Configure to use only selected channels
            _outp(AD_addr+2,ch_low);
            _outp(AD_addr+3,ch_high);

            // Configure channels to 0-5V
            _outp(AD_addr+11,13);

            // Enable FIFO and scanning
            _outp(AD_addr+7,0x0C);

            // Wait for A/D to settle
            do{
                  status = _inp(AD_addr+11) & 0x80;
            }while(status != 0);

            // Activate A/D conversion
            _outp(AD_addr+0,0);

            // loop to wait till A/D conversion complete
            do{
                  status = _inp(AD_addr+8) & 0x80;
            }while(status != 0);

            // Collect data from FIFO
            LSBad = _inp(AD_addr+0);
            MSBad = _inp(AD_addr+1);
            Xdigi[k*5000+i] = LSBad+MSBad*256;
            LSBad = _inp(AD_addr+0);
            MSBad = _inp(AD_addr+1);
            Ydigi[k*5000+i] = LSBad+MSBad*256;
            LSBad = _inp(AD_addr+0);
            MSBad = _inp(AD_addr+1);
            Zdigi[k*5000+i] = LSBad+MSBad*256;

            // Reset FIFO
            _outp(AD_addr+7,0x0F);

            // Configure to use only selected pressure channel
            _outp(AD_addr+2,ch_press);
            _outp(AD_addr+3,ch_press);

            // Configure channels to +/- 5V
```

```
                  _outp(AD_addr+11,0);

                  // Disable FIFO and scanning
                  _outp(AD_addr+7,15);

                  // Wait for A/D to settle
                  do{
                        status = _inp(AD_addr+11) & 0x80;
                  }while(status != 0);

                  // Activate A/D conversion
                  _outp(AD_addr+0,0);

                  // loop to wait till A/D conversion complete
                  do{
                        status = _inp(AD_addr+8) & 0x80;
                  }while(status != 0);
                  LSBpr = _inp(AD_addr+0);
                  MSBpr = _inp(AD_addr+1);
                  press[k*5000+i] = LSBpr+MSBpr*256;

                  i++;
            }
            k++;
      }

      sprintf(filename,"tube-%d_inflation.dat",tube_num);
      vibdat = fopen(filename,"w");

      k = 0;
      while(k<num_samp){
            fprintf(vibdat,"%d  %d  %d %d\n", Xdigi[k], Ydigi[k],
Zdigi[k], press[k]);
            k++;
      }
      fclose(vibdat);
      return 0;
}

int gas_str_check(int tube_num){
      FILE *gas_str;
      int status;
      int MSBad,LSBad;
      int ad_result;
      int ch;
      char filename[18];


      if (tube_num == 1)
            ch = 3;
      if (tube_num == 2)
            ch = 4;
      if (tube_num == 3)
            ch = 5;
```

```
	sprintf(filename,"tube-%d_gas_str.dat",tube_num);
	gas_str = fopen(filename,"w");

	// Configure to use only selected channel
	_outp(AD_addr+2,ch);
	_outp(AD_addr+3,ch);

	// Configure channels to +/-5V  BiP 0-5V range
	_outp(AD_addr+11,0);

	// Disable FIFO and scanning
	_outp(AD_addr+7,0);

	// Wait for A/D to settle
	do{
		status = _inp(AD_addr+11) & 0x80;
	}while(status != 0);

	// Activate A/D conversion
	_outp(AD_addr+0,0);

	// loop to wait till A/D conversion complete
	do{
		status = _inp(AD_addr+8) & 0x80;
	}while(status != 0);
	LSBad = _inp(AD_addr+0);
	MSBad = _inp(AD_addr+1);
	ad_result = MSBad*256+LSBad;
	fprintf(gas_str,"%d",ad_result);
	fclose(gas_str);
	return 0;
}

int excite_data_collect(int tube_num){
	short signed int LSB[5000] = {0};
	short signed int MSB[5000] = {0};
	short signed int Xdigi[125000] = {0};
	short signed int Ydigi[125000] = {0};
	short signed int Zdigi[125000] = {0};
	short signed int Dac[125000] = {0};
	char filename[17];
	long int num_samples = 125000;
	short int environ_high = 29;
	short int environ_low = 27;
	short int ch_high = 0;
	short int ch_low = 0;
	short int status = 0;
	short int LSBad = 0;
	short int MSBad = 0;
	short int i = 0;
	int k = 0;
	short int ldummy = 0;
	short int mdummy = 0;
	short int num_iterations = 25;
	short int channelnum = 0;
```

```
short int updateDAC = 0;
FILE *fidl;
FILE *fidm;
FILE *results;



if (tube_num == 1)
{
      ch_high = 12;
      ch_low = 10;
      _outp(relay_addr+1,0x48);
}
if (tube_num == 2)
{
      ch_high = 15;
      ch_low = 13;
      _outp(relay_addr+1,0x50);
}
if (tube_num == 3)
{
      ch_high = 26;
      ch_low = 24;
      _outp(relay_addr+1,0x60);
}

fidl = fopen("ex_LSB.dat","r");
fidm = fopen("ex_MSB.dat","r");

i = 0;
while(i<5000){
      fscanf(fidl,"%d",&ldummy);
      LSB[i] = ldummy;
      fscanf(fidm,"%d",&mdummy);
      MSB[i] = mdummy;
      i++;
}
fclose(fidl);
fclose(fidm);

_outp(relay_addr+0,0xC0);

// Loop to perform iterations
k = 0;
while(k<num_iterations){

      // Loop to go through the data array
      i = 0;
      while(i<5000){
            // Loop to wait for timing interrupt
            do {
                  // load status register
                  status = _inp(AD_addr+9) & 0x20;
            // check for timing interrupt
```

```
        } while(status != 32);

        // wait if DAC not ready
        while((_inp(AD_addr+4) & 0x80) == 0x80){}

        // load LSB to register
        _outp(AD_addr+4,LSB[i]);

        // load MSB to register
        _outp(AD_addr+5,MSB[i] + 64*channelnum);

        // Activate DAC
        updateDAC = _inp(AD_addr+5);

        // reset interrupts
        _outp(AD_addr+8,0x08);

        // Configure channels to sample
        _outp(AD_addr+2,ch_low);
        _outp(AD_addr+3,ch_high);

        // Configure channels to 0-5V
        _outp(AD_addr+11,13);

        // Enable FIFO and scanning
        _outp(AD_addr+7,0x0C);

        // Wait for A/D to settle
        do{
              status = _inp(AD_addr+11) & 0x80;
        }while(status != 0);

        // Activate A/D conversion
        _outp(AD_addr+0,0);

        // loop to wait till A/D conversion complete
        do{
              status = _inp(AD_addr+8) & 0x80;
        }while(status != 0);

        // Collect sampled data from A/D FIFO
        LSBad = _inp(AD_addr+0);
        MSBad = _inp(AD_addr+1);
        Xdigi[k*5000+i] = LSBad+MSBad*256;
        LSBad = _inp(AD_addr+0);
        MSBad = _inp(AD_addr+1);
        Ydigi[k*5000+i] = LSBad+MSBad*256;
        LSBad = _inp(AD_addr+0);
        MSBad = _inp(AD_addr+1);
        Zdigi[k*5000+i] = LSBad+MSBad*256;

        // Create a excitation data vector
        Dac[k*5000+i] = LSB[i] + MSB[i]*256;
        i++;
}
```

```
            k++;
      }

      _outp(relay_addr+0,0);

      sprintf(filename,"tube-%d_excite.dat",tube_num);
      results = fopen(filename,"w");

      k = 0;
      while(k<num_samples){
            fprintf(results,"%d  %d  %d  %d\n", Dac[k], Xdigi[k],
Ydigi[k], Zdigi[k]);
            k++;
      }
      fclose(results);
      return 0;
}
```

```
#include <stdio.h>
#include <malloc.h>
#include <conio.h>
#include <stdlib.h>
#include <time.h>

#define V                   (486)             // Lines per field
#define H                   (1134)            // Bytes returned per
line

#define DEFAULT_EXPOSURE    (100)             // 100 millisecond
exposure


#define BiasValue  (127)                      // mid scale of bias
range
#define GainValue  (200)                      // typical gain value


typedef unsigned char pixel;                  // one byte per pixel


typedef pixel __far* __far* field;            // 2 dimensional array
of
                                              // pixels

extern "C" {
       void __pascal __far USUBCAM(unsigned int, int, int, int,
unsigned int, field, int __far *);
       // initialize D/A converter
       void __cdecl __far InitDAC(unsigned int);
       // set bias D/A converter
       void __cdecl __far SetBiasValue(unsigned int, unsigned int);
       // set gain D/A converter
       void __cdecl __far SetGainValue(unsigned int, unsigned int);
}


//  Flags:

int ab_flag = !0;                             // 0 disables anti-
blooming
                                              // !0 enables anti-
blooming

int interlace_flag = 0;                       // 0 for non-interlace
mode
                                              // !0 for interlace
mode

int field_flag = 0;                           // if interlace mode,
                                              // 0 for first frame,

                        // !0 for second frame
```

```
// exposure time value in milliseconds for exposure control
unsigned exposure_time = DEFAULT_EXPOSURE;


pixel __far* buffer1[V];                          // image buffer
pixel __far* buffer2[V];                          // image buffer
pixel __far* buffer3[V];                          // image buffer
pixel __far* buffer4[V];                          // image buffer
pixel __far* buffer5[V];                          // image buffer

int line_length[V] = {H} ;                       // line length array


int cam1_addr = 0x300;
int cam2_addr = 0x340;
int cam3_addr = 0x380;
int timer_addr = 0x240;

int data_write(int,int,int,int);


int __cdecl main (void){

      int i,k;
      int tube_num;
      int base;
      int image_count_1 = 0;
      int image_count_2 = 0;
      int image_count_3 = 0;


      // Notify DAQ computer that imaging computer is not ready
      outp(timer_addr+2,0);

      for(k = 0;k<V;k++){
            line_length[k] = H;
      }


      //  Initialize the DAC which controls bias and gain settings
      //  and set the desired bias and gain values:
      //  (Note gain is inversely proportional to gain voltage.)


      InitDAC(cam1_addr);                // call only once at start
      SetBiasValue(cam1_addr, BiasValue); // call to set bias voltage
      // call to set gain voltage
      SetGainValue(cam1_addr, 255 - GainValue);
      InitDAC(cam2_addr);                // call only once at start
      SetBiasValue(cam2_addr, BiasValue); // call to set bias voltage
      // call to set gain voltage
      SetGainValue(cam2_addr, 255 – GainValue
      InitDAC(cam3_addr);                // call only once at start
      SetBiasValue(cam3_addr, BiasValue); // call to set bias voltage
      // call to set gain voltage
```

```
      SetGainValue(cam3_addr, 255 - GainValue);

      // Allocate memory for five images using the 5 image buffers
      // pointers
      printf("Establishing Image buffer 1 for %d x %d image\n", V,
line_length[0]);

      for(i = 0; i<V; i++){
            buffer1[i] = (unsigned char __far *)
calloc(line_length[i],1);

            //printf ("Allocated memory for buffer %d!\n\r", i);
            if (buffer1[i]  == (unsigned char __far *) NULL) {
                  printf ("\n\n\rCannot allocate memory for buffer
%d!\n\r", i);
                  exit (1);
            }
      }

      printf("Establishing Image buffer 2 for %d x %d
image\n",V,line_length[0]);

      for(i = 0; i<V; i++){
            buffer2[i] = (unsigned char __far *)
calloc(line_length[i],1);

            //printf ("Allocated memory for buffer %d!\n\r", i);
            if (buffer2[i]  == (unsigned char __far *) NULL) {
                  printf ("\n\n\rCannot allocate memory for buffer
%d!\n\r", i);
                  exit (1);
            }
      }

      printf("Establishing Image buffer 3 for %d x %d
image\n",V,line_length[0]);

      for(i = 0; i<V; i++){
            buffer3[i] = (unsigned char __far *)
calloc(line_length[i],1);

            //printf ("Allocated memory for buffer %d!\n\r", i);
            if (buffer3[i]  == (unsigned char __far *) NULL) {
                  printf ("\n\n\rCannot allocate memory for buffer
%d!\n\r", i);
                  exit (1);
            }
      }
      printf("Establishing Image buffer 4 for %d x %d
image\n",V,line_length[0]);

      for(i = 0; i<V; i++){
            buffer4[i] = (unsigned char __far *)
calloc(line_length[i],1);
```

```
            //printf ("Allocated memory for buffer %d!\n\r", i);
            if (buffer4[i]  == (unsigned char __far *) NULL) {
                  printf ("\n\n\rCannot allocate memory for buffer
%d!\n\r", i);
                  exit (1);
            }
      }

      printf("Establishing Image buffer 5 for %d x %d
image\n",V,line_length[0]);

      for(i = 0; i<V; i++){
            buffer5[i] = (unsigned char __far *)
calloc(line_length[i],1);

            //printf ("Allocated memory for buffer %d!\n\r", i);
            if (buffer5[i]  == (unsigned char __far *) NULL) {
                  printf ("\n\n\rCannot allocate memory for buffer
%d!\n\r", i);
                  exit (1);
            }
      }

      // Loop waiting for signal to end program
      while((inp(timer_addr+2) & 0x40) != 0x40){

            // Notify DAQ computer that imaging computer is ready
            outp(timer_addr+2,0x80);

            // Loop until notified to take pictures
            while((inp(timer_addr+2)& 0x80) != 0x80){
                  printf("Checking for image notification...\n");
            }

            // Notify DAQ computer images are being written
            outp(timer_addr+2,0);

            // Check tube number to be imaged
            printf("Checking tube number...\n");
            tube_num = inp(timer_addr+2) & 0x03;
            printf("Tube number %d will be imaged...\n",tube_num);

            // Establish camera address to be imaged
            printf("Setting base address...\n");
            if (tube_num == 1){
                   base = cam1_addr;
                  image_count_1++;
            }
            if (tube_num == 2){
                   base = cam2_addr;
                  image_count_2++;
            }
            if (tube_num == 3){
                   base = cam3_addr;
                  image_count_3++;
```

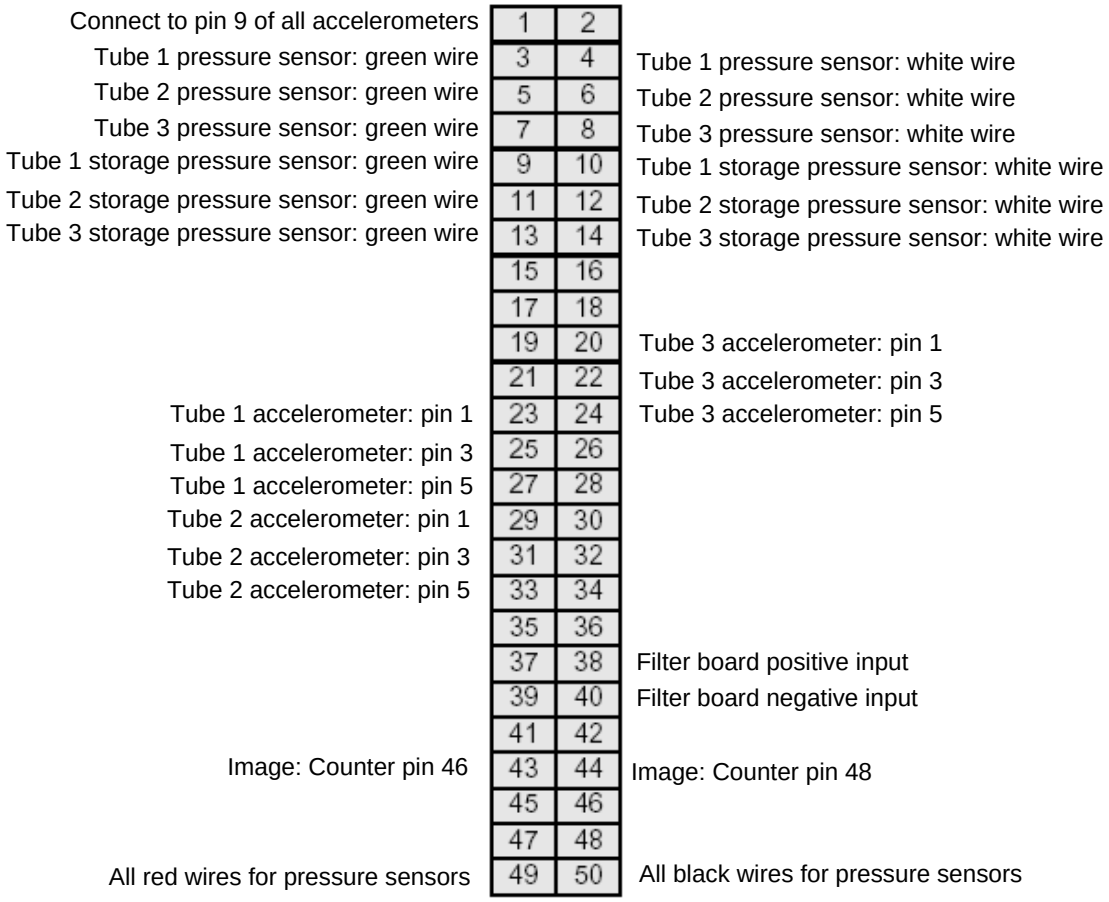
```
            }

            printf("Taking pictures...\n");

            // Take the five images
USUBCAM(base,ab_flag,interlace_flag,field_flag,exposure_time,buffer1,li
      ne_length);
USUBCAM(base,ab_flag,interlace_flag,field_flag,exposure_time,buffer2,li
      ne_length);

USUBCAM(base,ab_flag,interlace_flag,field_flag,exposure_time,buffer3,li
      ne_length);

            USUBCAM(base,ab_flag,interlace_flag,field_flag,exposure_tim
            e,buffer4,line_length);

            USUBCAM(base,ab_flag,interlace_flag,field_flag,exposure_tim
            e,buffer5,line_length);

            // Write the five images to data files
data_write(tube_num,image_count_1,image_count_2,image_count_3);

            }
            return 0;
}


int data_write(int tube_num,int image_count_1,int image_count_2,int
image_count_3){


      int i,k;
      FILE *data1,*data2,*data3,*data4,*data5;
      char fna[8];
      char fnb[8];
      char fnc[8];
      char fnd[8];
      char fne[8];
      int count = 0;
      clock_t start,end;

      // establish which image counter to use
      if (tube_num == 1){
            count  = image_count_1;
      }

      if (tube_num == 2){
            count  = image_count_2;
      }

      if (tube_num == 3){
            count  = image_count_3;
      }

      // Create first file name
```

```
sprintf(fna,"t%1d%2d.dat",tube_num,(count-1)*5+1);
data1 = fopen(fna,"w");

// Write first file and measure time to write
printf("Writing image 1 file...\n");
start = clock();
for(k = 0; k < V; k++){
       for(i = 0; i < H; i++){
                  fprintf(data1,"%d ",*(buffer1[k]+i));
       }
       fprintf(data1,"\n");
}
fclose(data1);
end = clock();
printf("%2.2f sec to write file\n",(end-start)/CLK_TCK);

// Create second file name
sprintf(fnb,"t%1d%2d.dat",tube_num,(count-1)*5+2);
data2 = fopen(fnb,"w");

// Write second file and measure time to write
printf("Writing image 2 file...\n");
start = clock();
for(k = 0; k < V; k++){
       for(i = 0; i < H; i++){
                        fprintf(data2,"%d ",*(buffer2[k]+i));
       }
       fprintf(data2,"\n");
}
fclose(data2);
end = clock();
printf("%2.2f sec to write file\n",(end-start)/CLK_TCK);

// Create third file name
sprintf(fnc,"t%1d%2d.dat",tube_num,(count-1)*5+3);
data3 = fopen(fnc,"w");

// Write third file and measure time to write
printf("Writing image 3 file...\n");
start = clock();
for(k = 0; k < V; k++){
       for(i = 0; i < H; i++){
                        fprintf(data3,"%d ",*(buffer3[k]+i));
       }
       fprintf(data3,"\n");
}
fclose(data3);
end = clock();
printf("%2.2f sec to write file\n",(end-start)/CLK_TCK);

// Create fourth file name
sprintf(fnd,"t%1d%2d.dat",tube_num,(count-1)*5+4);
data4 = fopen(fnd,"w");

// Write fourth file and measure time to write
```

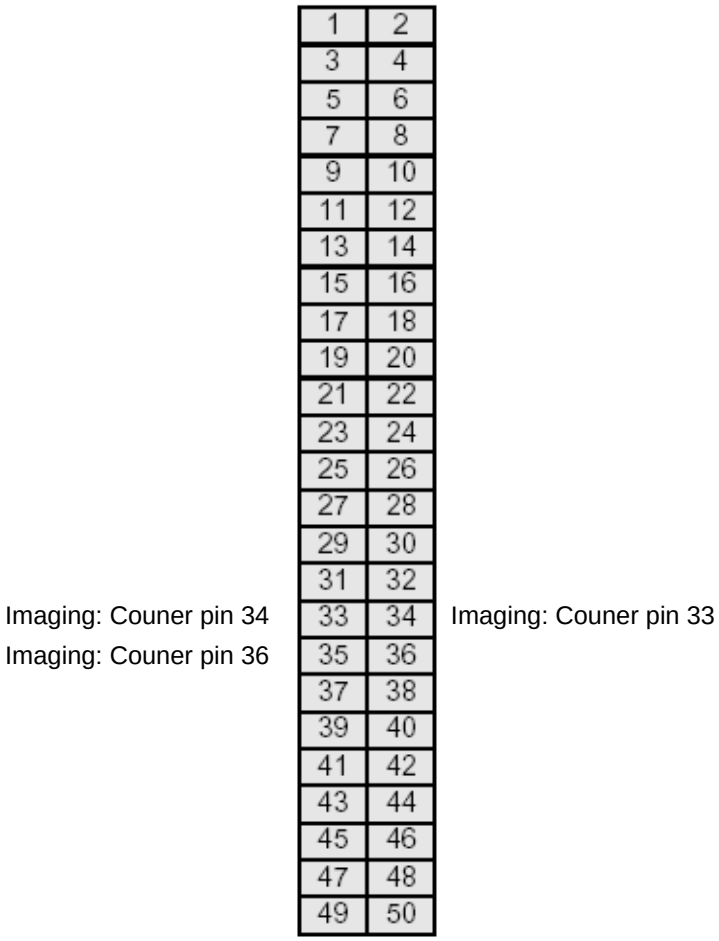
```
	printf("Writing image 4 file...\n");
	start = clock();
	for(k = 0; k < V; k++){
		for(i = 0; i < H; i++){
				fprintf(data4,"%d ",*(buffer4[k]+i));
		}
		fprintf(data4,"\n");
	}
	fclose(data4);
	end = clock();
	printf("%2.2f sec to write file\n",(end-start)/CLK_TCK);

	// Create fifth file name
	sprintf(fne,"t%1d%2d.dat",tube_num,(count-1)*5+5);
	data5 = fopen(fne,"w");
	// Write fifth file and measure time to write
	printf("Writing image 5 file...\n");
	start = clock();
	for(k = 0; k < V; k++){
		for(i = 0; i < H; i++){
				fprintf(data5,"%d ",*(buffer5[k]+i));
		}
		fprintf(data5,"\n");
	}
	fclose(data5);
	end = clock();
	printf("%2.2f sec to write file\n",(end-start)/CLK_TCK);

	return 0;
}
```

# Appendix G:  Wiring Connections

# Data Acquisition Computer ADC Board

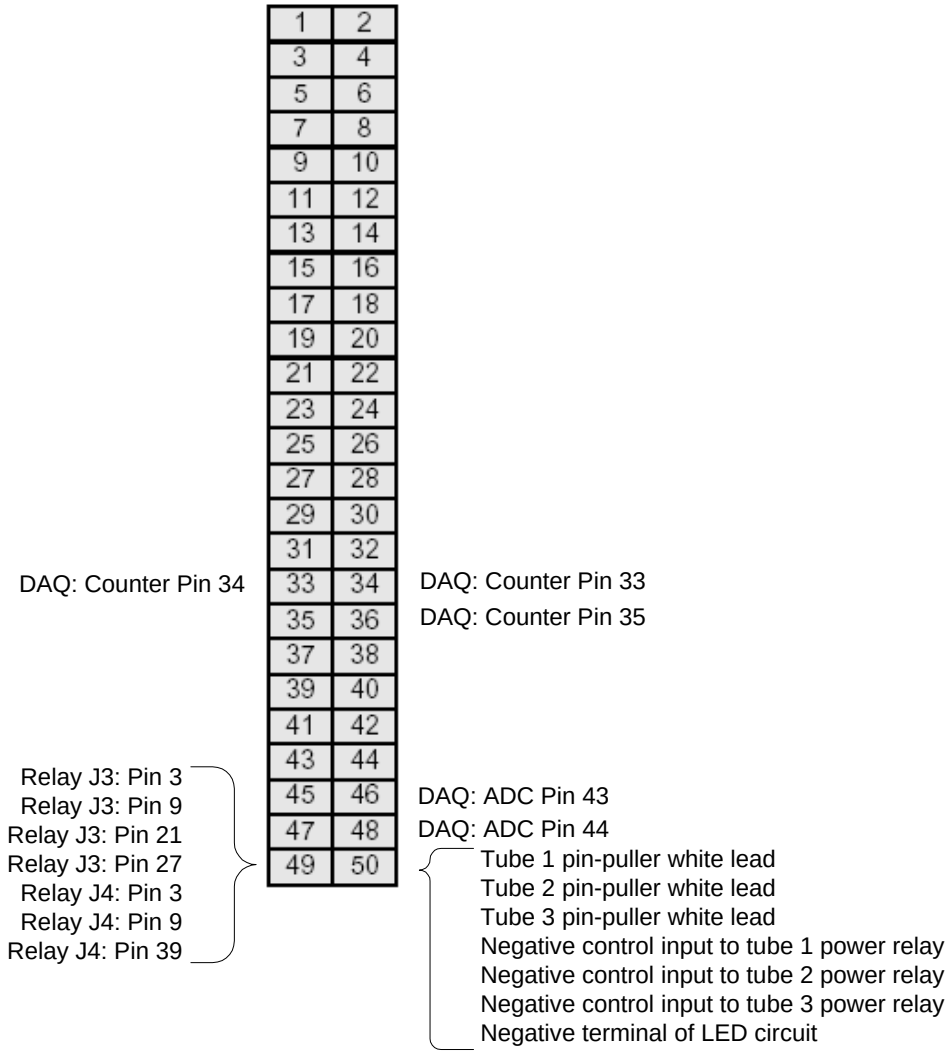


# Data Acquisition Counter Board

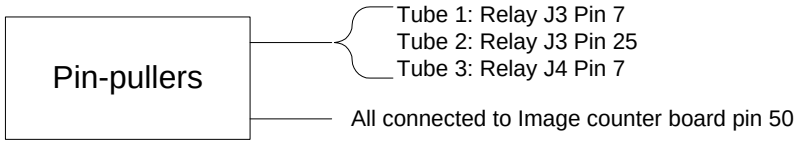



| | |
|---|---|
| 1 | 2 |
| 3 | 4 |
| 5 | 6 |
| 7 | 8 |
| 9 | 10 |
| 11 | 12 |
| 13 | 14 |
| 15 | 16 |
| 17 | 18 |
| 19 | 20 |
| 21 | 22 |
| 23 | 24 |
| 25 | 26 |
| 27 | 28 |
| 29 | 30 |
| 31 | 32 |
| 33 | 34 |
| 35 | 36 |
| 37 | 38 |
| 39 | 40 |
| 41 | 42 |
| 43 | 44 |
| 45 | 46 |
| 47 | 48 |
| 49 | 50 |

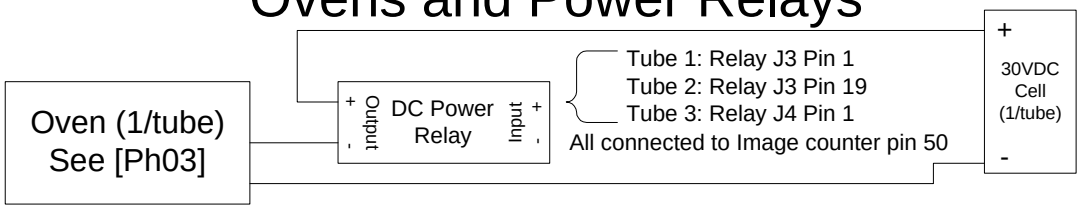
# Imaging Computer Counter Board



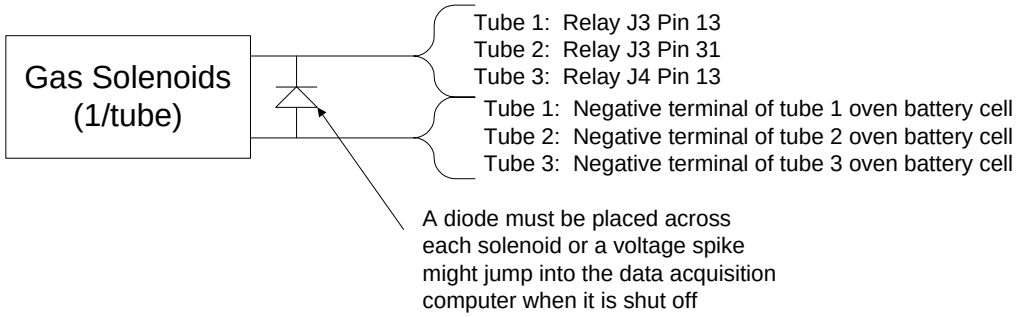
# Pin-Pullers




# Ovens and Power Relays

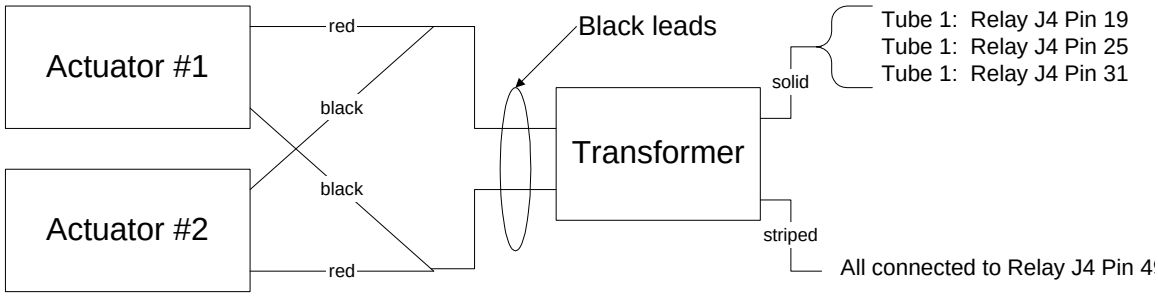



# Gas Solenoids



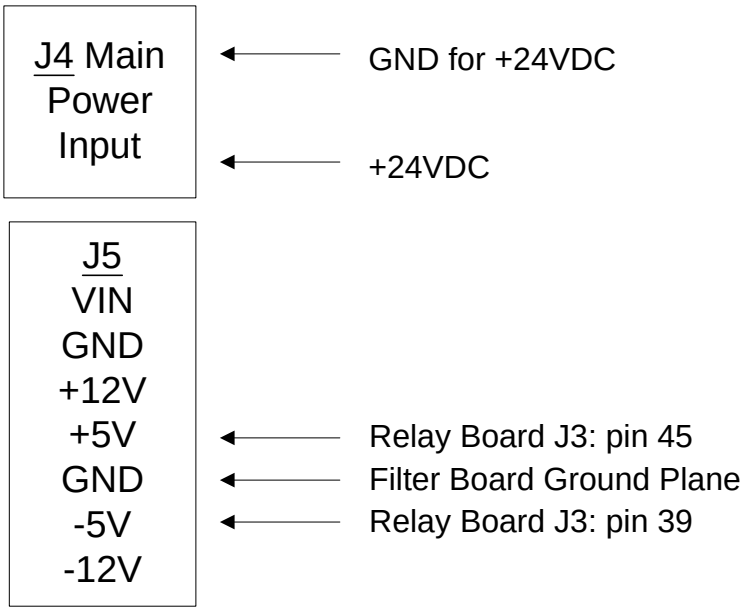

# Excitation Circuit

Circuit shown for just one tube

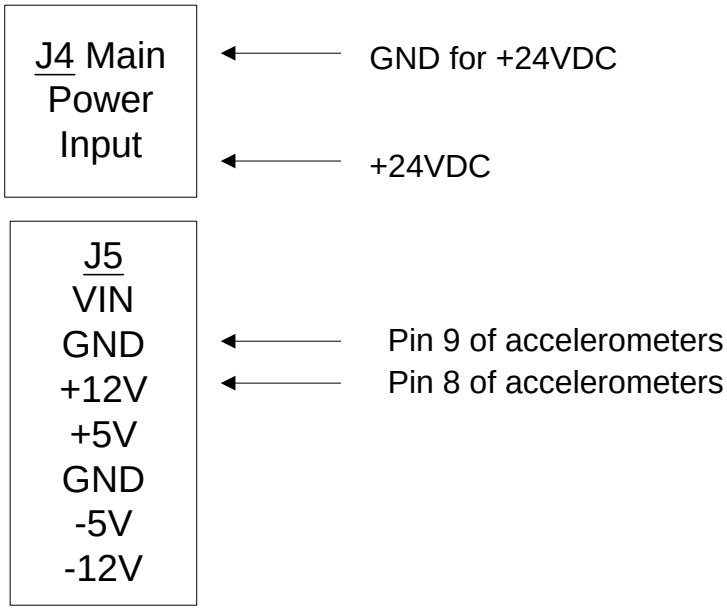


# Power Supply Connections

## Data Acquisition Computer Supply Board




## Imaging Computer Supply Board



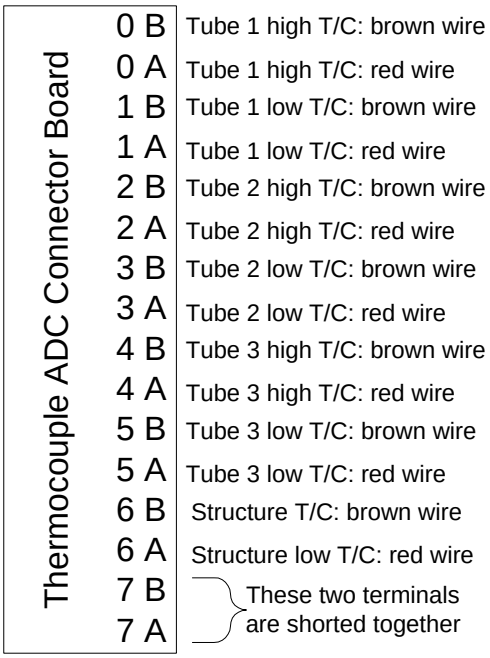
# Relay Board J3

| | | |
|---|---|---|
| + control input for oven #1 power relay | 1 | 2 |
| Imaging Timer board Pin 49: +5V | 3 | 4 |
| | 5 | 6 |
| Tube 1 Pin-puller white lead | 7 | 8 |
| Imaging Timer board Pin 49: +5V | 9 | 10 |
| | 11 | 12 |
| Tube 1 solenoid valve yellow lead | 13 | 14 |
| Positive voltage terminal of Tube 1 oven battery cell | 15 | 16 |
| | 17 | 18 |
| + control input for oven #2 power relay | 19 | 20 |
| Imaging Timer board Pin 49: +5V | 21 | 22 |
| | 23 | 24 |
| Tube 2 Pin-puller white lead | 25 | 26 |
| Imaging Timer board Pin 49: +5V | 27 | 28 |
| | 29 | 30 |
| Tube 2 solenoid valve yellow lead | 31 | 32 |
| Positive voltage terminal of Tube 2 oven battery cell | 33 | 34 |
| | 35 | 36 |
| -5V input of filter boad | 37 | 38 |
| -5V output power of data acquisition computer power supply | 39 | 40 |
| | 41 | 42 |
| +5V input of filter boad | 43 | 44 |
| +5V output power of data acquisition computer power supply | 45 | 46 |
| | 47 | 48 |
| | 49 | 50 |

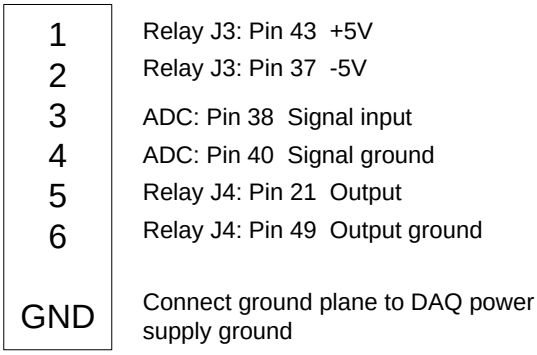
# Relay Board J4

| | | |
|---|---|---|
| + control input for oven #3 power relay | 1 | 2 |
| Imaging Timer board Pin 49: +5V | 3 | 4 |
| | 5 | 6 |
| Tube 3 Pin-puller white lead | 7 | 8 |
| Imaging Timer board Pin 49: +5V | 9 | 10 |
| | 11 | 12 |
| Tube 3 solenoid valve yellow lead | 13 | 14 |
| Positive voltage terminal of Tube 3 oven battery cell | 15 | 16 |
| | 17 | 18 |
| Solid color wire for tube 1 transformer | 19 | 20 |
| Filter board positive output lead and relay J4: Pins 27 and 33 | 21 | 22 |
| | 23 | 24 |
| Solid color wire for tube 2 transformer | 25 | 26 |
| Relay J4: Pins 21 and 33 | 27 | 28 |
| | 29 | 30 |
| Solid color wire for tube 3 transformer | 31 | 32 |
| Relay J4: Pins 27 | 33 | 34 |
| | 35 | 36 |
| Positive Input for LED circuit | 37 | 38 |
| Imaging Timer board Pin 49: +5V | 39 | 40 |
| | 41 | 42 |
| | 43 | 44 |
| | 45 | 46 |
| | 47 | 48 |
| Filter board negative output lead and stiped colored wire for each transformer | 49 | 50 |

# Data Acquisition Computer T/C ADC Board




# Filter Board



# Appendix H: Single Tube Experiment Results

Upon completion of 1/3 of the experiment structure and associated equipment (oven, latch, pressure system, and pin-puller), a single tube experiment was conducted. The experiment routine was adjusted to choose the lowest temperature from the two thermocouples for temperature thresholding. The test was conducted using 15 sec inflation timer and a 3 min cool down.

The tube was successful in its inflation but the structural analysis results were not so good. The data collected during the excitation of the tube could not resolve a transfer function. The modal frequencies can be seen as little spikes in the PSD plots of the vibration signals. The cause of this low vibration could be the result of the actuator patch performance degraded by heating or the tube being attached to the structure.

The resulting plots from the test can be seen below. The inflation modeling has some error due to the rotation of the accelerometer.

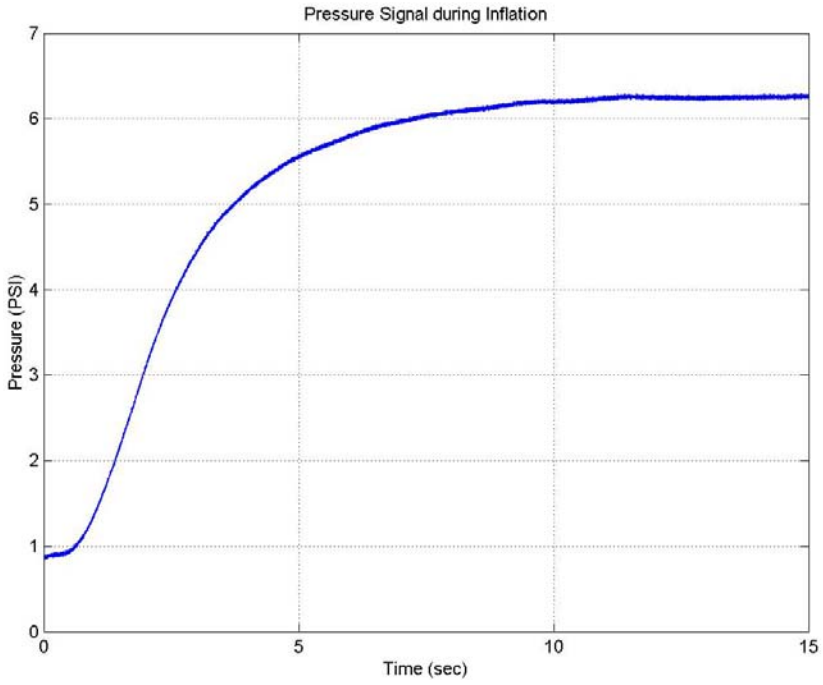



Figure H.1: Inflation pressure

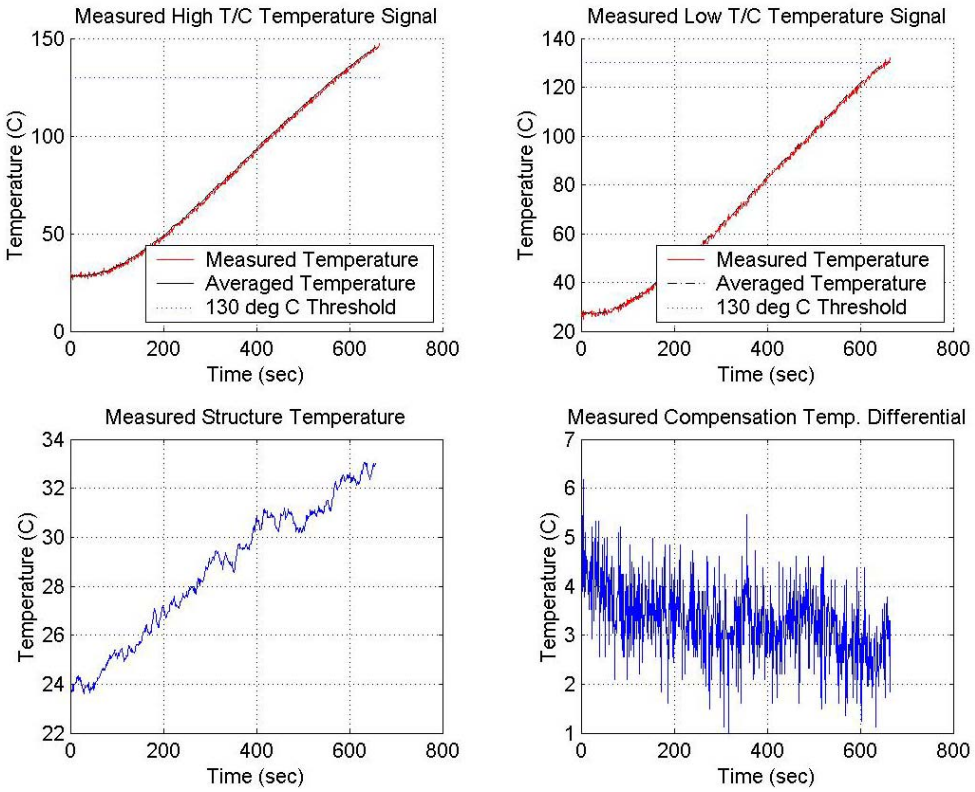



Figure H.2: Tube heating

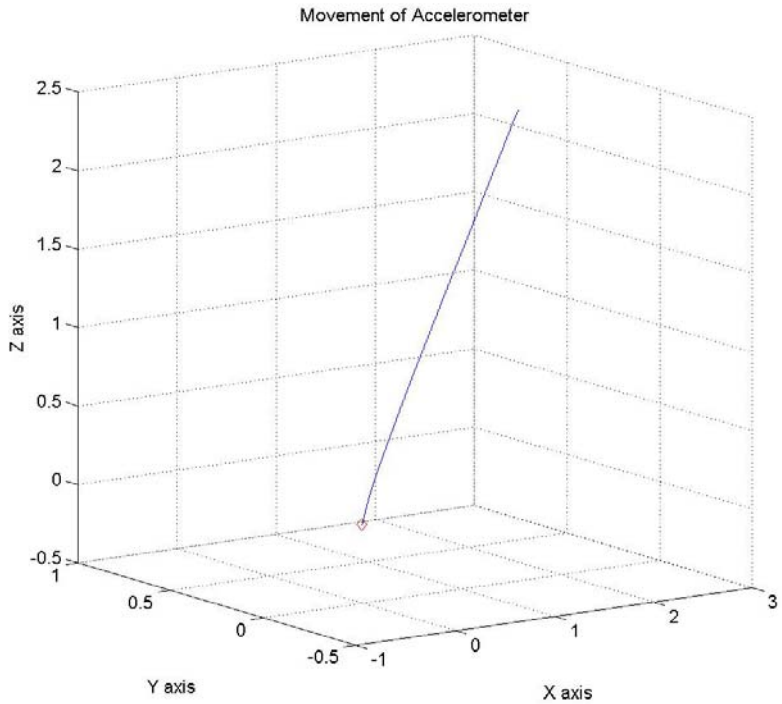



Figure H.3: Inflation model

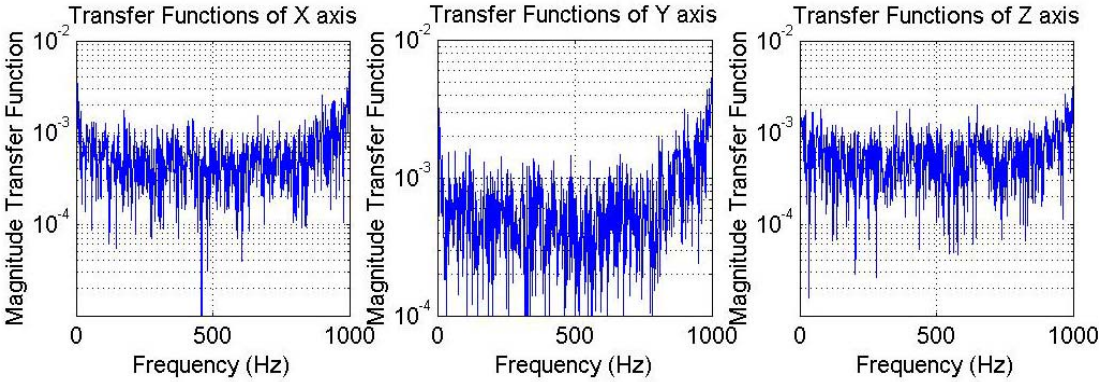



Figure H.4: Transfer function estimation

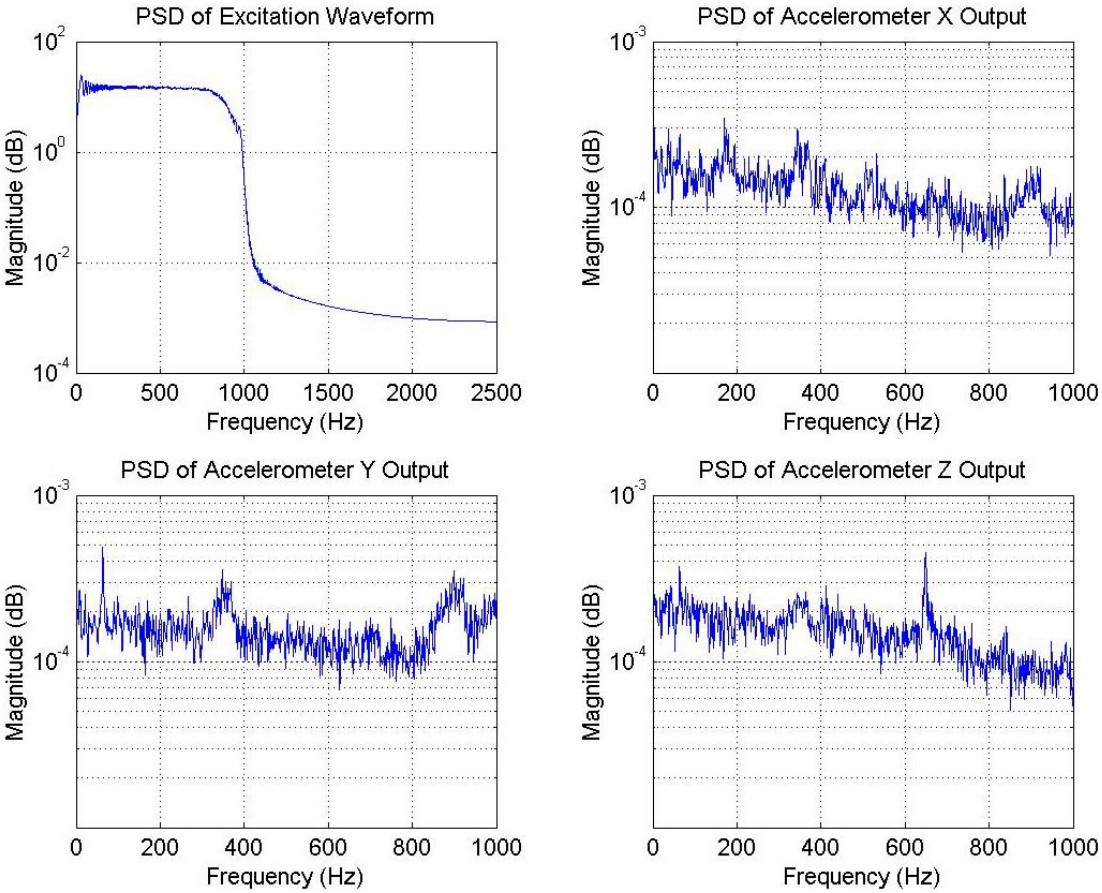



Figure H.5: PSD of excitation signals

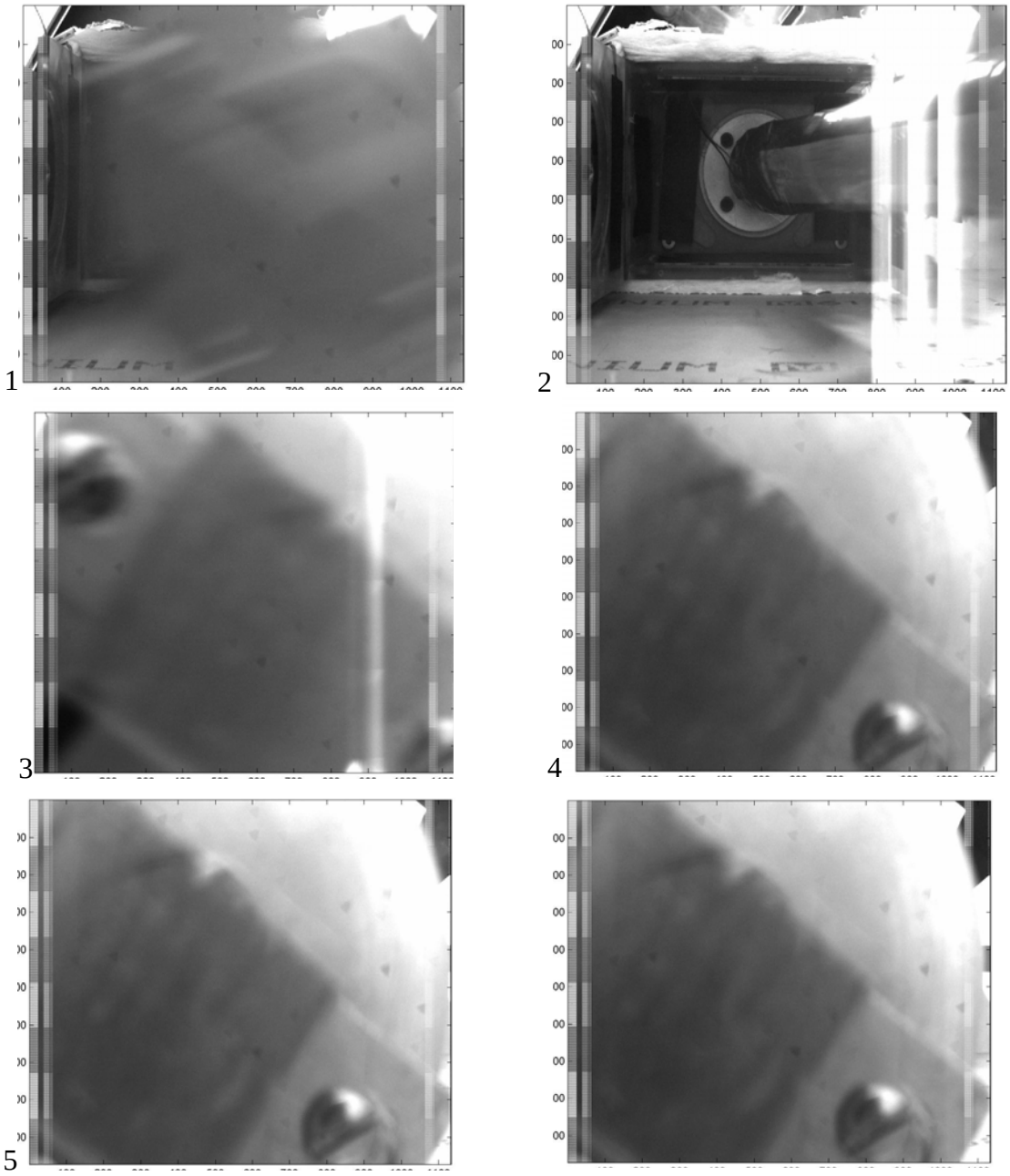



Figure H.6: Inflation images 1 – 5 and final rest state image

## Bibliography


[AD98] Analog Devices, Inc. *Thermocouple Conditioner and Setpoint Controller, AD596/597 Data Sheet.* 1998.

[Bo98] Boeing North America Inc. *Shuttle Orbiter/Small Payload Accomodation Interfaces NSTS-21000-IDD-SML.* Feburary 1998.

[Dar76] Daryanani, Gobind. *Principles of Active Network Synthesis and Design*. 1976.

[DMM00] Diamond Systems Corp. *Diamond-MM-32-AT User Manual V2.3*. 2000.

[DiS01] DiSebastian III, John Daniel. *RIGEX: Preliminary Design of Rigidized Inflatable Get-Away-Special Experiment*. Master's Thesis, Air Force Institute of Technology, Dayton, OH, March 2001.

[Dur00] Duracell. *Alkaline-Manganese Dioxide Battery, Size: D*, May 2000.

[Elec97] ELECTRIM Corp. *EDC-1000U Computer Camera Technical Manual*. 1997.

[End03] ENDEVCO Corp. *Piezoresistive Pressure Transducer Model 8510C*. No date listed.

[Fr98] Freeland, R. E. "Inflatable Deployable Space Structures Technology," *49th International Astronautical Congress.* Number IAF 98-1.5.01.1998.

[Fr97] Freeland, R.E. "Large Inflatable Deployable Antenna Flight Experiment Results," *48th International Astronautical Federation Congress*. 1997.

[FrBi92] Freeland, R. E. and G. Bilyeu. "In-Step Inflatable Antenna Experiment," *International Astronautical Federation IAF 92-0301*. 1992.

[GuH95] Guru, Bhag S and Hüseyin R. Hiziroglu *Electric Machinery and Transformers 2nd Edition.* 1995.

[Hu01] Huang, John. "The Development of Inflatable Array Antennas," *IEEE Antennas and Propagation Magazine, Vol. 43, No. 4.* August 2001.

[Ir96] Irwin, J. David. *Basic Engineering Circuit Analysis 5th Ed.* 1996.

[JMM01] Diamond Systems Corp. *Jupiter-MM 50W Power Supply User Manual V1.3*. 2001.

[Lim90] Lim, Jae S. *Two-Dimensional Signal and Image Processing*. 1990.

[Ma03] Matlab®. r*egionprops Help File.* No date listed

[Max96] Maxim Integrated Products. *4th- and 8th-Order Continuous Time Active Filters, MAX274/275 Data Sheet.* 1996.

[MiS03] Microcomputer Systems, Inc. *MSI-P440 User Manual*. Apr. 2003.

[MI99] MINCO Products Inc. *250ºC All-Polymide Thermofoil® Heaters*. Jan. 1999.

[MI01] MINCO Products Inc. *CT198 Heaterstat™ Instructions.* 2001.

[MiS01] Mitra, Sanjit K. *Digital Signal Processing, A Computer-Based Approach 2nd Ed.* 2001.

[NAG99] NASA, Goddard Space Flight Center. *Hitchhiker, Customer Accommodations & Requirements Specifications 740-SPEC-008.* 1999.

[NAG93] NASA, Goddard Space Flight Center. *Gas Experimenter's Guide to the STS Safety Review Process and Data Package Preparation*, September 1993.

[NAL89] NASA, Lyndon B. Johnson Space Center. *Safety Policy and Requirements for Payloads using the STS*, January 1989.

[NAL85] NASA, Lyndon B. Johnson Space Center. *Manned Space Vehicle Battery Safety Handbook*, September 1985.

[NaI03] National Instruments. *LabVIEW 7 Express User Manual.* Apr. 2003.

[NPH00] National Plastic Heater Sensor & Control Co. *Technical Letters: How Thermocouples Work.* 2000.

[PaH03] Parker Hannifin Corp. *Series 9, 2 and 3 way High Performance Valves.* No date listed.

[PCCon96] PC/104 Consortium. *PC/104 Specifications, Version 2.3.* June 1996.

[PMM01] Diamond Systems Corp. *Pearl-MM 16 Relay Digital Output PC/104 Module User Manual V1.1*. 2001.

[Phi03] Philley Jr, Thomas Lee. *Development, Fabrication, and Ground Test of an Inflatable Structure Space-Flight Experiment*. Master's Thesis, Air Force Institute of Technology, Dayton, OH, March 2003.

[Pi96] Pierret, Robert F. *Semiconductor Device Fundamentals*. Mar 1996.

[Pon03] Ponziani, Kevin. *Image Processing for the Rigidized Inflatable Get-Away-Special Experiment.* Intern Report, Air Force Institute of Technology, Dayton, OH, Unpublished.

[PrBo03] Pratt, Timothy and Charles W Bostian. *Satellite Communications 2nd Ed.* 2003.

[PrM96] Proakis, John G and Dimitris G Manolakis. *Digital Signal Processing, Principles, Algorithms, and Applications 3rd Edition.* 1996.

[QMM01] Diamond Systems Corp. *Quartz-MM PC/104 Format Counter/Timer & Digital I/O Module Users Manual V1.5*. 2001.

[ShB88] Shanmugan, K. Sam and A. M. Breipohl *Random Signals, Detection, Estimation and Data Analysis*. 1988.

[SiT02] Single, Thomas G. *Experimental Vibration Analysis of Inflatable Beams for AFIT Space Shuttle Experiment.* Master's Thesis, Air Force Institute of Technology, Dayton, OH, March 2002.

[Skl00] Sklar, Bernard. *Digital Communications, Fundamentals and Applications 2nd Edition*. 2000.

[Str88] Strang, Gilbert. *Linear Algebra and its Applications 3rd Ed.* 1988.

[SuI02] Summit Instruments. *34200A Triaxial Accelerometer Data Sheet*. Dec. 2002.

[Th92] Thomas, Mitchell. "Inflatable Space Structures: Redefining aerospace design concepts keeps costs from ballooning," *IEEE Potentials December*, Dec 1992.

[TiA03] TiNi Aerospace, Inc. *P5 Series Pinpullers.* No date listed.

[TriM01] Tri-M Engineering Inc. *MZ104+ Manual*. 2001.

[ZF01] ZF Micro Devices. *ZFx86 Data Book 1.0 Rev. B*, 2001

[Email03] Email Conversation with Electrim Corp. Representative, Walker, Jo N., about camera subroutine object files (*USUBCAM.obj*).

## Vita

First Lieutenant David Charles Moody graduated from Gulf Breeze High School in Gulf Breeze, Florida in 1995. He entered undergraduate studies at the University of South Alabama in Mobile Alabama where he graduated with a Bachelor of Science Degree in Electrical Engineering in May of 2000.

His first assignment was as a squadron communications engineer for the 54th Combat Communications Squadron, 5th Combat Communications Group, Robins AFB, GA. His responsibilities as squadron communications engineer was to design and implement deployable communications base infrastructures to support Air Force and Department of Defense tactical communications requirements.

Lt. Moody began the Graduate Electrical Engineering program at the Air Force Institute of Technology, Wright Patterson AFB, OH in August of 2002. Upon graduatation, he will be assigned to the 333rd Training Squadron, Keesler AFB, MS.

# REPORT DOCUMENTATION PAGE

*Form Approved*
*OMB No. 074-0188*



| 1. REPORT DATE *(DD-MM-YYYY)* | 2. REPORT TYPE | 3. DATES COVERED *(From – To)* |
|---|---|---|
| 23-03-2004 | Master's Thesis | March 2003 – March 2004 |

**4. TITLE AND SUBTITLE**

**MICROPROCESSOR-BASED SYSTEMS CONTROL FOR THE RIGIDIZED INFLATABLE GET-AWAY-SPECIAL EXPERIMENT**

**5a. CONTRACT NUMBER**

**5b. GRANT NUMBER**

**5c. PROGRAM ELEMENT NUMBER**

**6. AUTHOR(S)**

Moody, David C., 1st Lieutenant, USAF

**5d. PROJECT NUMBER**
04-299 04-948

**5e. TASK NUMBER**

**5f. WORK UNIT NUMBER**

**7. PERFORMING ORGANIZATION NAMES(S) AND ADDRESS(S)**
Air Force Institute of Technology
Graduate School of Engineering and Management (AFIT/EN)
2950 Hobson Way, Building 640
WPAFB OH 45433-8865

**8. PERFORMING ORGANIZATION REPORT NUMBER**

AFIT/GE/ENG/04-17

**9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)**
IMINT/RNTS
Attn: Maj Dave Lee
14675 Lee Road DSN: 898-3084
Chantily VA 20151 email: david.lee@nro.mil

**10. SPONSOR/MONITOR'S ACRONYM(S)**

**11. SPONSOR/MONITOR'S REPORT NUMBER(S)**

**12. DISTRIBUTION/AVAILABILITY STATEMENT**

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

**13. SUPPLEMENTARY NOTES**

**14. ABSTRACT** As the demand for space based communications increases and the need for faster data throughput, satellites are becoming larger. Larger satellite antennas help to provide the needed gain to increase communications in space. Compounding the performance and size trade-offs are the payload weight and size limit imposed by the launch vehicles. Inflatable structures offer a cost saving opportunity since the structure is significantly lighter and has a reduced storage volume. This allows for smaller launch vehicles and/or increased performance capabilities. Inflatable structures offer possibilities for increased satellite lifetimes, increased communications capacity, and reduce launch costs.

This thesis develops and implements the computer control system and power system to support the Rigidized Inflatable Get-Away-Experiment. The autonomous computer system controls the flow of the experiment while at the same time collecting and recording temperature, pressure, vibration, and image data.

The computer system consists of two processors, one for experiment control and sensor data collection and the second for image data collection. These two systems can work simultaneously to control the flow of the experiment and meet the experiment's requirements. Examples of the data collection include heating curves, pressure, tube transfer function plots and images. This thesis also develops the Matlab® tools required to analyze the data collected by the computers.

This thesis lays the groundwork for future inflatable structures research. This pioneering effort has been selected for flight testing on-board the U.S. Space Shuttle.

**15. SUBJECT TERMS**

Microcomputers, Space Sciences, Space Technology, Composite Structures, Inflatable Structures, Control, Design of Experiments, Experimental Design

| 16. SECURITY CLASSIFICATION OF: | | | 17. LIMITATION OF ABSTRACT | 18. NUMBER OF PAGES | 19a. NAME OF RESPONSIBLE PERSON |
|---|---|---|---|---|---|
| a. REPORT | b. ABSTRACT | c. THIS PAGE | | | Richard A. Raines, Ph.D. |
| U | U | U | UU | 253 | 19b. TELEPHONE NUMBER *(Include area code)* (937) 255-6565, ext 4278 (Richard.raines@afit.edu) |

**Standard Form 298 (Rev. 8-98)**
Prescribed by ANSI Std. Z39-18