Electronics Design for an Autonomous Helicopter

by

Christian A. Trott

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Bachelor of Science in Electrical Science and Engineering and Master of Engineering
in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 1997

Author .....
Department of Electrical Engineering and Computer Science
May 19, 1997

Certified by .....
Paul A. DeBitetto
Senior Member of Technical Staff, Charles Stark Draper Laboratory
Thesis Supervisor

Certified by .....
James K. Roberge
Professor of Electrical Engineeering, Massachusetts Institute of Technology
Thesis Supervisor

Accepted by .....
Arthur C. Smith
Chairman, Deparmental Committee on Graduate Theses

Electronics Design for an Autonomous Helicopter

by

Christian A. Trott

Submitted to the Department of Electrical Engineering and Computer Science
on May 19, 1997, in partial fulfillment of the
requirements for the degree of
Bachelor of Science in Electrical Science and Engineering and Master of Engineering in Electrical Engineering and Computer Science

Abstract

Over a two year period, two generations of electronics hardware for a fully functional autonomous aerial vehicle have been developed. Both custom-designed and off-the-shelf components have been integrated on a small radio controlled helicopter. This vehicle, the Draper Small Autonomous Aerial Vehicle (DSAAV), is now capable of fully autonomous take-off, way-point guidance, vision processing and landing. Its electrical systems include a 486 based computer, a differential GPS unit, a six degree of freedom inertial instrument, a sonar altimeter, a solid-state compass, a radio frequency modem, a power system, and a servo control system. The electronics necessary for the successful autonomous navigation of the vehicle have been designed, integrated, and tested under flight conditions involving intensive vibration and significant levels of electro-magnetic radiation. Specifically, a power system, inertial system, and sensor interface has been designed, built, tested, and integrated with commercial building blocks such as a computer and radio frequency modem. Building from a first year of experience, hardware has been modified and/or redesigned for a subsequent year of development. The most significant of these changes involved the inertial system, which has been completely redesigned for the purpose of improving the accuracy and bandwidth of the on-board navigation system. The revised inertial system has been built around a high speed impulse sampling chip and processors that take readings from the inertial instrument, process the data, and channel it to the main on-board computer.

Thesis Supervisor: Paul A. DeBitetto
Title: Senior Member of Technical Staff, Charles Stark Draper Laboratory

Thesis Supervisor: James K. Roberge
Title: Professor of Electrical Engineeering, Massachusetts Institute of Technology

Acknowledgements

The work that is described in this document was by no means done all by myself. There are many people who I would like to thank for direct or indirect contributions to this effort.

First, I would like to clarify who made direct contributions to the work described in this document. Unless stated otherwise, I was responsible for the work described. The receiver interface was designed by Bob Butler and layed out by Bob Menyhert. The altimeter hardware and test software was developed by Long Phan. The power board was layed out by Dmitry Brant. Some of the circuit construction and vehicle assembly work was done by Long Phan, Dmitry Brant, Chinsan Han, and Russel Sammon. Paul DeBitetto chose many of the electronic components. Eric Johnson helped with the development of the A/D board driver software.

I would like to thank those who were part of the DSAAV team. Paul DeBitetto was an excellent leader and contributed both technically and as a moral supporter. Eric Johnson made sure that we kept the "critical path" at the top of our priority list. He helped solve many of the problems that cropped up along the way. I admire Long Phan's willingness to march right into the middle of uncertainty and find ways to solve a problem. As an experienced electrical engineer, it was nice to have Bob Butler available as a critic. Mike Bosse, Dmitry Brant, Russel Sammon, Chinsan Han, and Valerie Lowe have all made contributions to this effort.

Other people at Draper Labs have also been available for help or moral support. In particular, Charles Tung was always one step ahead of me in finding the best hardware to use, or the best techniques for designing or building circuits. Bryan Koontz, Ryan Norris, Bill Kaliardos, and Mike Socha were all available for discussion of ideas or problems.

I would like to thank my MIT advisor, Professor James K. Roberge. As an academic advisor, professor, and thesis advisor, he has helped shape my academic pursuits. His excellence, enthusiasm, and style have all been inspirational.

Finally, I would like to thank my family for years of support and encouragement. It is to my mother, father, sister, and brothers that I dedicate this effort.

Christian Trott
May, 1997

Biographical Note

Christian Trott was born on October 7, 1973 in Saint Cloud, Minnesota. He spent the three years from 1978 to 1980 in Grand Haven, Michigan before Moving to Tupelo, Mississippi where his family still lives. After graduating, Valedictorian, from Tupelo High School, Christian studied Electrical Engineering for five years at the Massachusetts Institute of Technology. During the summer after his Sophomore year, he worked at the NSF Center for Computational Fluid Simulation in Starkville, MS, where he wrote software in C for Silicon Graphics workstations. The following summer, he worked for Maxim Integrated Products in Sunnyvale, CA, where he ran simulations of analog integrated circuit designs using PSPICE. At the beginning of his Senior year, he began work on the Draper Small Autonomous Aerial Vehicle at the Charles Stark Draper Laboratory in Cambridge, MA. This work was the focus of his research for the two years from the beginning of his senior year until the end of his graduate year. Contingent upon the acceptance of this thesis, Christian will be awarded a Bachelor's degree in electrical engineering and a Master's degree in engineering. Future plans include work at the Bose Corporation in Framingham, MA.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Introduction

The notion of a machine that can pro-actively interact with its environment has for many years been a source of fascination to scientists, engineers... and most everyone else. First popularized by science fiction novelists, the idea of a Robot[1], often referred to a machine that resembled human beings and was capable of interacting intelligently with people.

There are two main features that distinguish a robot from other machines. The first feature is the capacity for movement. This movement can either take the form of a vehicle which can relocate itself, or a stationary machine that is capable of physically manipulating objects. The second feature is some level of *autonomy.* In this context, a good definition for autonomy is given by the *American Heritage Dictionary.* An autonomous entity is defined as one that is "Independent in mind or judgment; self-directed." This means that the agent's behavior is a function only of its initial state and the environment with which it interacts—no human control is required.

Although the definition of a robot is fuzzy, there are certain machines that are clearly not robots. Hand tools, drill presses[2], automobiles, small and large rocks, elevators, and furniture do not qualify. Similarly, computer equipment does not qualify on the grounds that by itself it is not capable of movement. For the purposes of research, a more abstract concept of a robot may be useful. Software objects being used in computer simulations of robots may themselves be considered robots. However, the idea of a physical robot is always at the heart of the objectives of such research.

Of the two features that define a robot, the autonomy is the more difficult to achieve. People have been able to make machines that move for a long time. Being able to create a machine that can move intelligently is far more challenging. For this reason, most research into the subject of robots concentrates on the autonomy of the vehicles. Therefore, robots are often referred to as *autonomous agents* or *autonomous vehicles.*

For several years, this kind of research has been underway at the Charles Stark Draper Laboratory in Cambridge, MA. One product of this research is the Draper Small Autonomous Aerial Vehicle (DSAAV) which is an autonomous helicopter. This document concentrates on that vehicle.

[1] The term originates from the Czechoslovakian term, *Robota*, meaning drudgery [3]

[2] At least the ones available today...

1.2 Objective of This Thesis Document

In addition to being the deliverable for Masters level work at the Charles Stark Draper Laboratory, there are several objectives of this document.

1.2.1 Documentation of Existing System

Excepting this document, there is little documentation of the hardware on the DSAAV. There are two generations of the vehicle, the 1996 DSAAV and the 1997 DSAAV. A detailed description and explanation of both are provided. The level of detail provided may be far more than most readers will find useful or interesting. However, one objective of this document is to provide enough detail so that others will be able to re-create or service the existing systems.

1.2.2 Description of Design Process

In addition to knowledge about the hardware implementation on the DSAAV, it may be helpful for others to understand the design process that went into developing the hardware. In particular, the role of the major constraining factors are discussed. There will also be description of the mistakes that were made in the design process. The lessons learned in the development of this vehicle are extremely valuable, and need not be learned at the same expense by others. Otto von Bismarck has been quoted: "Fools you are... to say you learn by your experience... I prefer to profit by others' mistakes and avoid the price of my own." [4].

1.2.3 Analysis of Performance

In order to determine whether or not the DSAAV performed well, its performance is analyzed. In addition, methods for future performance analysis are suggested and discussed.

1.2.4 Listing of Potential Improvement Areas

Only a year and a half of development has been invested into the DSAAV. A great deal has been accomplished over that time period. However, there is great potential for future improvement. Many of the things that can be done to improve the vehicle are obvious. Others are more subtle. Some of these potential improvement areas are listed.

1.3 Autonomous Agents

As more work is being done in the field of autonomous systems, the term *autonomous agents* has often come to refer to a mobile platform– designed for land, sea, air, or a combination. In the United States, there are many groups that are working on developing autonomous agents. These groups include government agencies, government contractors, and private companies.

An early reason for pursuing research in autonomous vehicles was for space exploration. The high level of risk/cost associated with sending a person into space made the idea of a dispensable machine–explorer very attractive. The ultimate objective was to send a vehicle to another planet where it could collect data and samples to be returned to scientists on earth. The vehicle need not be retrieved, and destruction of the vehicle would not be a catastrophe.

Since then, other applications for autonomous agents have been targeted. These include search-and-rescue, hazardous area inspection, border patrol, delivery, and mapping. There are also many

military applications including bomblet detection, surveillance, and weapon deployment. Autonomous vehicles are well suited to applications that would place a human in high risk or to applications that would otherwise require more people than are readily available.

1.4 Autonomous Helicopters

Autonomous helicopters are an interesting sub-class of autonomous vehicles. They are often grouped with other aerial vehicles which are collectively referred to as *unmanned aerial vehicles* (UAV's). This group is still sometimes referred to as *autonomous aerial vehicles* (AAV's), but despite the desire for high levels of autonomy in these systems, the military does not foresee immediate applications in which there will be no human control over the vehicles. More realistic scenarios would involve a semi-autonomous vehicle that can direct itself from one way-point to another. A human "Pilot" would give the vehicle high level instructions from a remote control station. The fact that this type of vehicle would remove a human pilot from the danger of a mission is emphasized in the name, UAV.

1.4.1 Advantages of Helicopters Over Other Vehicles

Aerial vehicles have many advantages over ground vehicles for certain applications. The fact that they have three translational degrees of freedom gives them an advantage in avoiding obstacles. They are well suited for searching and surveillance applications because they can obtain a large view of an area by flying high. Finally, higher travel speed is usually achieved in the air than on the ground.

Helicopters in particular have characteristics that make them stand out from other UAV's. They have excellent dynamics that allow them to hover accurately, even in gusty conditions. This capability is useful in search, surveillance, and deployment applications. Blimps and tail-sitters are also capable of hover, but blimps have difficulty rejecting wind disturbances, and tail-sitters have less desirable dynamics than helicopters. Helicopters can be made small, and can achieve fairly high velocities.

1.4.2 Challenges Particular to Autonomous Helicopters

Even though helicopters are ideal for certain applications, there are many challenges that arise from this choice of vehicle platform. The most obvious challenge might be the failure mode of a helicopter. If something goes wrong with the vehicle, there is a good chance that it will completely destroy itself by crashing. This is usually not a problem with ground, sea, or even some air vehicles (blimps). It only takes a small mechanical or electrical problem to result in a crash.

Another challenge with helicopters is the inherent danger associated with them. Even small hobby helicopters can kill a person if the rotating blades come in contact with any vital organs. Therefore, great care has to be taken in testing this type of UAV. What can be tested on a ground vehicle in ten minutes in a lab may require several hours of preparation, travel, and set-up for a helicopter.

As with any air vehicle, helicopters have strict limits on how much baggage they can carry. For this reason, electronics must be designed to be light and energy-efficient (to minimize the number of batteries needed). For small vehicles, enclosures must be carefully designed to be light.

Finally, the vibration associated with a helicopter's engine is very stressful to the electronic components. Unless care is taken, the electronics will fail, likely leading to a devastating crash. Ways of preventing mechanical fatigue of electronic components include stress relieving, epoxying, solid construction techniques, and vibration isolation.

1.5 The Draper Small Autonomous Aerial Vehicle

The Draper Small Autonomous Aerial Vehicle (DSAAV) is an example of an autonomous helicopter. Its capabilities include take-off, way-point following, landing, and image processing. Figure 1-1 shows the vehicle in flight. Figure 1-2 shows an early version of the vehicle on display with the ground station computer. The vehicle is based on a TSK BlackStar model helicopter which sells for approximately $2500. It has a rotor disc diameter of approximately 10 feet. This vehicle was originally capable of remotely piloted flight. Electronics were added to make it capable of autonomous flight. In addition, several safety mechanisms were implemented.

There are actually two generations of the DSAAV. 1996 was the first year of development. In that year, a vehicle was designed, built, and demonstrated that could achieve all of the basic objectives of autonomy. In 1997 two major goals have emerged. The first is to increase the robustness and performance of the existing design. The second goal is to demonstrate two autonomous vehicles that can cooperatively search an area.

1.5.1 Objectives

The main objective for the DSAAV was to demonstrate a high level of autonomy with a small, dynamically complex vehicle. In order to motivate the effort, the first year's goal was participation in the 1996 International Aerial Robotics Competition. This annual competition brings together university students and faculty with corporate funding and expertise. The rules of the contest change yearly, but a typical mission involves flying over a field, locating specific objects, and retrieving one or more of the objects.

In 1996, the mission was to fly over a simulated toxic waste site in which drums were placed. The major axes of these drums could be oriented in any direction, and they could be partially buried. Each drum had a label on an upward facing surface. There were two types of labels: biohazardous labels and radioactive labels. The helicopter had to fly over the 120' by 60' field and locate the drums to within a meter of their true position. Further, it had to identify which type of drum (biohazardous or radioactive) was at each location. Finally, the vehicle was to retrieve a small disk from one of the drums and return it to the start position. All of the above had to be done with complete autonomy.

The first and most important objective, then, was to build a vehicle that could fly itself. Indeed, this task consumed an estimated 90% of the man-hours that went into the system's development. Much design and re-design had to go into developing a system that could perform its mission.

1.5.2 Components

A brief introduction to the electronics architecture of the vehicle will be given here. Other chapters will provide details on the design of the majority of the electronic sub-systems. At the center of the architecture is a 50MHz, 486 based computer. The computer used is an Ampro PC-104 module stack. The stack includes the processor, an ethernet card, and an analog-to-digital conversion board. Many peripherals are connected to the computer either directly or indirectly. These peripherals can be classified as sensors or interface units.

Four sensors are used to collect navigation data for the vehicle. One of these is a differential global positioning system (DGPS) receiver. The receiver has an on-board antenna which receives signals from orbiting satellites. In addition to the on-board receiver, there is a ground based receiver. By combining the data collected by both receivers, very accurate position and velocity estimates can be obtained from the system. The second sensor is a six degree of freedom inertial sensor (three rate gyros and three accelerometers). This instrument provides relatively high-bandwidth information to supplement the GPS estimate. Additionally, there is a solid-state compass and an

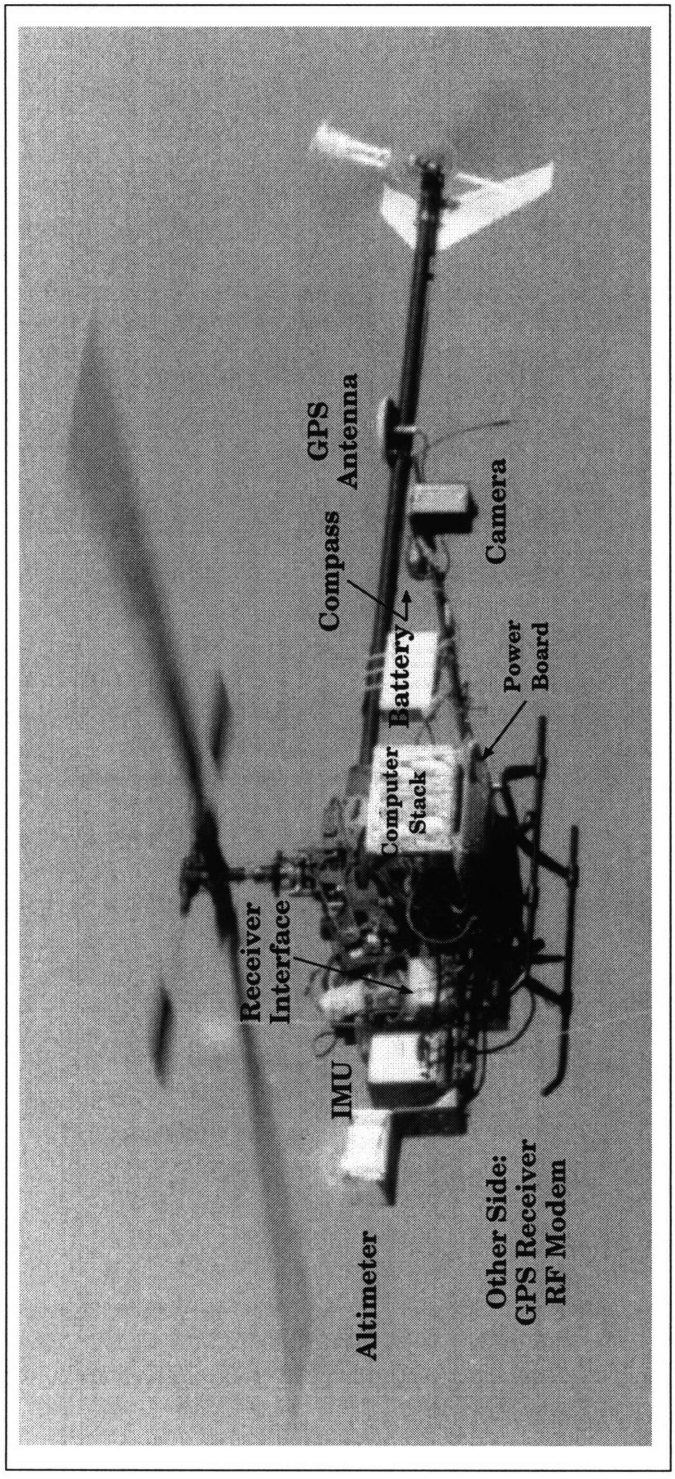

Figure 1-1: DSAAV In Action: 60″ From Nose to Tail

Figure 1-2: DSAAV on Display

acoustic altimeter. The compass corrects for long-term drift errors in the heading estimate. The altimeter provides high-accuracy altitude estimates for use when the helicopter is near the ground.

There are four main means of interfacing with the on-board electronics. The first method is via the receiver interface. The receiver interface provides a mechanism for switching between pilot-controlled flight and computer-controlled flight. Interface is also accomplished through a radio frequency modem. A ground based computer communicates with the on-board computer through this modem. State information and commands are transferred in this way. The third method of interface is through an ethernet cable that can be connected between the on-board computer and the ground computer when the helicopter is on the ground. Finally, the designated "Kill Operator" can interface with the helicopter by means of a separate radio that signals the engine choke to engage. Once the engine choke is engaged, the engine immediately stops and the helicopter is rendered ballistic.

Only two main pieces of hardware remain to be discussed. The first is a video camera which transmits raw video data to the ground in NTSC format. The second is the power system. The power system provides many different voltage levels to the electronic systems throughout the helicopter. There will be much more information given on the power system in Chapter 3.

1.6 Conclusion

The rest of this document will describe in detail the electronics design of the DSAAV. Chapters 2 through 4 will describe the design used in 1996. Chapters 5 and 6 describe important ideas that have affected the newer generation of the DSAAV. The final chapters describe the biggest changes that were made in 1997.

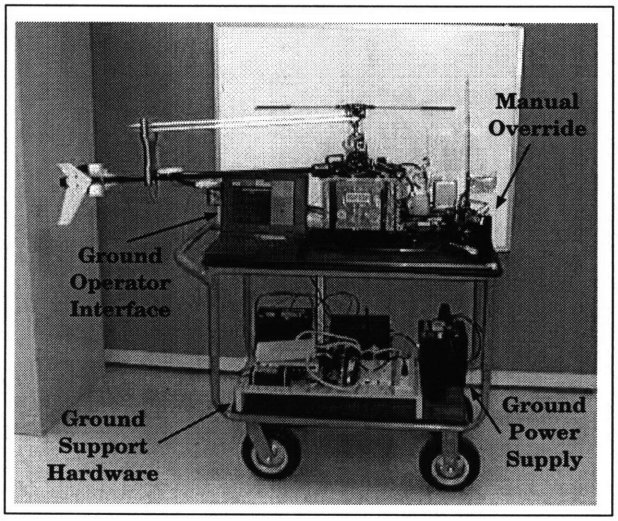

Figure 1-3: Helicopter with Support Hardware

Chapter 2

General Electronics Integration

The basic design philosophy behind the DSAAV was to use as many off-the-shelf components as possible in order to simplify the overall design effort and speed up the development time. Therefore, much of the work of getting the helicopter's electronic systems to function properly fell into the domain of electronics integration. Electronics integration is the process of taking high level building blocks and getting them to interact with each other as desired. This chapter describes the major electronic sub-systems and the way in which they are integrated. The highest level picture of the vehicle's integration is shown in Figure 2-1. This figure shows the major building blocks and the way in which they are inter-connected.

Most of the systems used on the vehicle are commercially available. Where possible, the name of the product is given. This name can be cross-referenced to Appendix B where the major components are listed as well as the manufacturers, distributors, prices, etc. Appendix C lists contact information for the manufacturers and distributors.

2.1 Computer

The on-board computer motherboard is based on a 50MHz Cyrix 486 processor. This board has all of the basic features needed for a PC based computer including serial ports, a parallel port, keyboard port, clock, etc. Additionally, a 386 floating point co-processor was added to the board. A picture of the computer stack that was used is shown in Figure 2-2. This picture shows a small protruding board on one side of the stack. This board holds the FPU.

Two other boards are included in the computer stack. One is an ethernet card. The other is an analog-to-digital conversion board. It has a conversion precision of 12 bits and can sample up to eight channels differentially (16 single-ended) at a rate of 100kHz. It has many nice features such as built in timers, DMA transfers, and 16 bits of Digital I/O.

These boards obey the PC-104 conventions, which is an interface standard for embedded processors. By claiming to be PC-104 compliant, these boards are guaranteed to fit within the 3.775" by 3.55" footprint specified for PC-104 boards. Additionally, the PC-104 standard provides a mechanism for interconnecting the boards. Each board has a connector along one edge that can plug into the board beneath it. The connector must also allow a board to fit on top. All data and power sent between the cards are sent through the edge connector. Finally, the PC-104 standard determines the location of holes that are used to mount stand-offs between the boards. Therefore, a PC-104 stack can be mechanically robust (as it must be).

The operating system used on the computer is the QNX operating system. It is referred to as a "real-time" operating system, meaning that the processes are guaranteed to be serviced regularly.

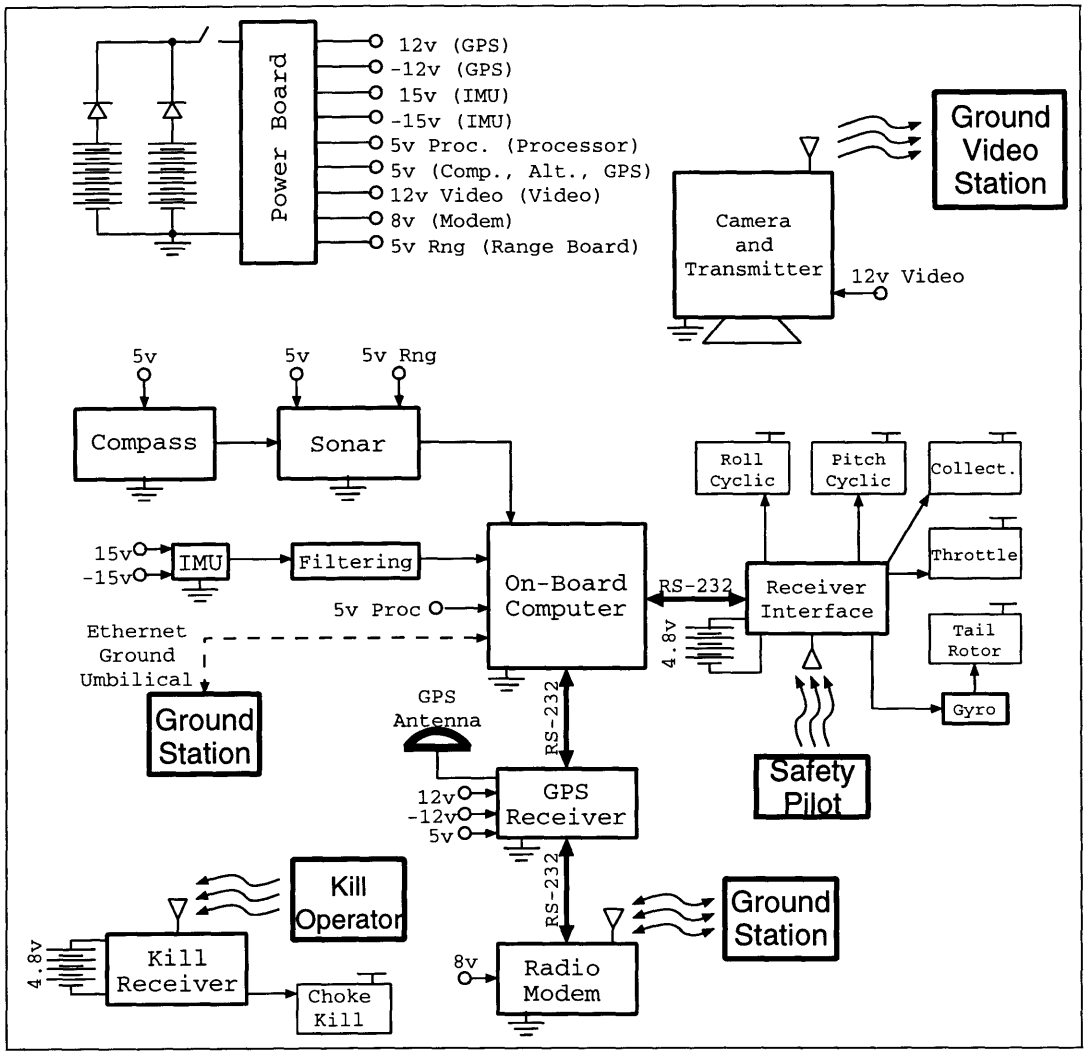

Figure 2-1: DSAAV 1996 Electronic Sub-Systems

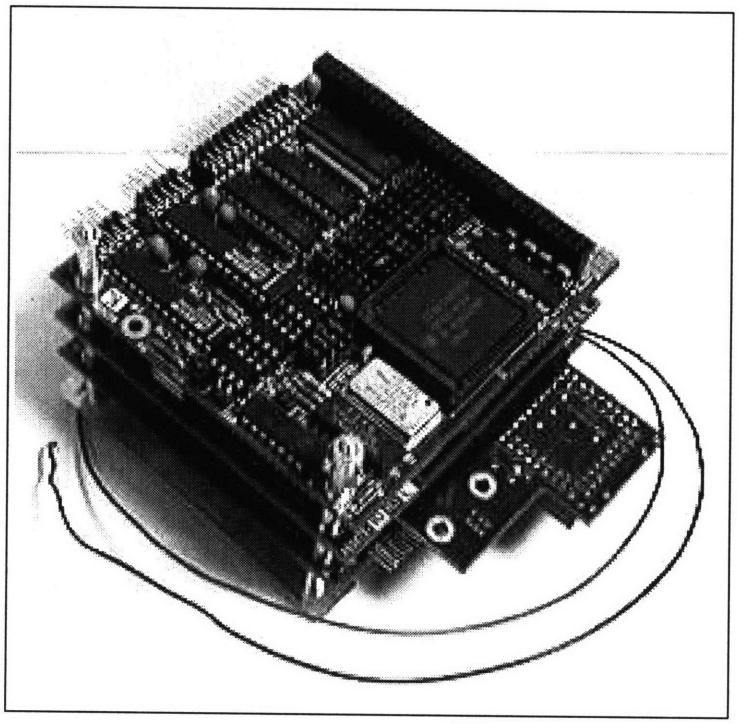

Figure 2-2: PC-104 Computer Stack

There is a bound on the amount of time that can pass before a process will be serviced by the operating system kernel. This feature can be important in real-time applications so that no critical process is ignored while others hog the CPU.

Because of the DSAAV software architecture, the real-time nature of QNX was not critical to the software's performance. However, it was still a good choice of operating system because it is very small and is Unix-like. Because it is unix-like, users need not learn a whole new operating system. It has man ("use") pages, user accounts, and most of the familiar commands found in Unix. A user who is familiar with Unix has little trouble adapting to the QNX environment. However, those who wish to use QNX will find it very useful to obtain the books available from QNX that describe some of its special commands.

Because there is no permanent storage associated with the on-board processor, it has to be booted via a network. In order to accomplish the booting process, a *boot ROM* is installed in the ethernet card. This ROM instructs the ethernet card to wait for a host to send the boot information to the card, and then oversees the process of loading the processor with the operating system. Once the processor has booted, it is communicated with just like a node on a network. In order to start the on-board software, the ground computer must be plugged into the on-board processor via ethernet, and the process must be run remotely.

2.2 Altimeter/Compass

The Altimeter and Compass are lumped together because they are controlled by the same set of micro-processors. However, the basic idea behind each of the units can be understood by itself. Then, the way in which they are integrated can be described.

2.2.1 Altimeter

The altimeter is based upon the sonar ranging module from the Polaroid Corporation. The module is actually composed of two parts. The first is a transducer which acts as a special speaker and microphone for sending and receiving a 50kHz acoustic pulse train. The second part of the system is the circuit board ("Ranging Board") which interfaces between a computer and the transducer. This equipment has been used for many years by hobbyists, and it is well documented. A picture of the ranging board and transducer is shown in Figure 2-3.

The ranging board can be powered with a 5 volt supply. In addition to power, three other signals are used to interface it to a processor. These are *Init*, *Binh*, and *Echo*. *Init* and *Binh* are inputs to the ranging board while *Echo* is an output from the ranging board. On a positive going edge of *Init*, the sonar system will transmit an acoustic pulse train.

Once a pulse train has been sent, the acoustic wave reflects from the ground and returns to the transducer. The transducer senses the return wave and pulls *Echo* low when the reflected wave is received. Problems with the system occur if transients on the transducer cause a false triggering of the *Echo* signal. In order to avoid these false signals, *Binh* (Blanking Inhibit) is used. For correct operation of the system, *Binh* is held at a low voltage level for a few milliseconds after the pulse train is sent. While *Binh* is low, no *Echo*s will be reported. Only after *Binh* is returned to a high state will return pulse will be considered echos.

It is up to the processor to time the interval from when a pulse is sent to when the echo is picked up. A picture of the relevant timing is shown in Figure 2-4. The times used on the helicopter are roughly as given in Table 2.1. τ_3 is the time of interest for purposes of distance measurement. Its value will be linear with distance. The relationship between τ_3 and distance can easily be found analytically. However, it is more reliable to determine the relationship experimentally by comparing the values of τ_3 with known distances and finding a linear expression to fit the data. The system

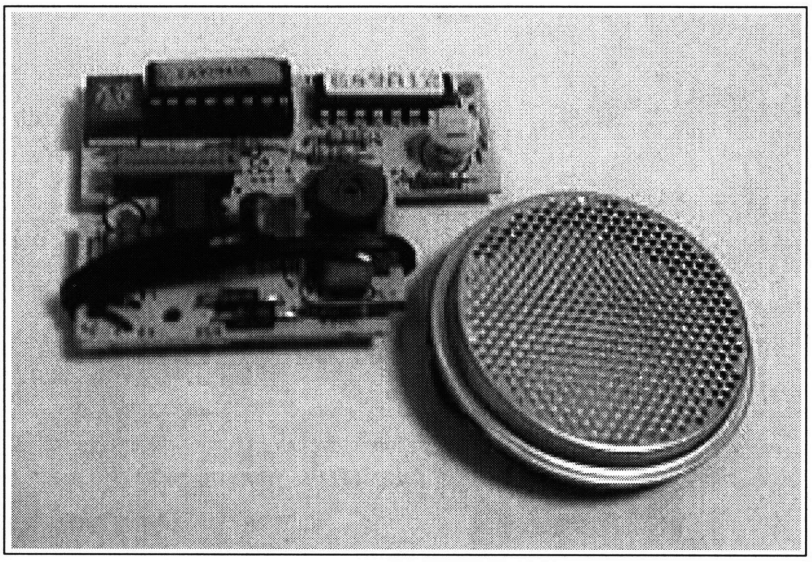

Figure 2-3: Sonar Ranging Board and Transducer

Parameter	Value
τ_1	30ms
τ_2	5ms

Table 2.1: Sonar Timing

on the DSAAV-1996 was found to have better than 99% linearity over the range from 6 inches to 6 feet.

There are some peculiarities to be careful of with the sonar system. The first is that the *Echo* signal is an open-collector pull-down. This means that it must be connected to the 5 volt supply through a moderate sized resistor ($\approx 10K\Omega$). The other pitfall to watch out for with the system is the fact that the ranging board pulls several hundred milliamps of current for a few micro-seconds every time a pulse is sent. This current impulse can easily cause problems with the 5 volt power supply. Unless care is taken, the added power supply noise can degrade the performance of equipment connected to the line, or worse. A good step toward fixing the problem is to bypass the power supply lines near the ranging board with a $.1\mu F$ ceramic capacitor and a $33\mu F$ electrolytic capacitor. In many cases, the bypassing may be enough to solve the problem. If very sensitive electronics are on the 5 volt line, it may be necessary to provide a separate power supply for the ranging board or the sensitive component. This was done on the DSAAV by adding a 5 volt linear regulator (LM340T5) which supplies the ranging board.

Another area of concern is a capacitor located in the upper left hand corner of the ranging board as seen in Figure 2-3. This capacitor has a relatively large mechanical moment arm. The vibration of a helicopter can easily cause the leads on this capacitor to fatigue. Therefore, this capacitor should be epoxied to the circuit board before it is used on a helicopter.

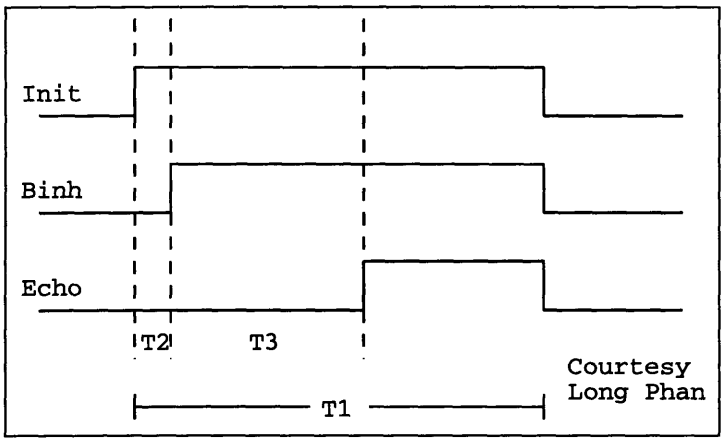

Figure 2-4: Sonar Timing Diagram

2.2.2 Compass

The compass used on the helicopter is a 2 degree of freedom flux-gate electronic compass. It has an accuracy of 2 degrees and an update rate of 5 Hz. The use of this instrument on the helicopter is complicated by the fact that there are two different versions of the instrument in use. In 1996, a Beta version of the instrument was used. In 1997, a commercial version of the instrument is used. The two versions have slightly different interfacing standards. The description in this chapter applies to the Beta version that was used on the DSAAV-1996. It is shown in Figure 2-5.

A brief discussion of interfacing with the unit will be given here. Precision Navigation provides a good set of application notes for the part. If others wish to use the part, they should obtain a copy of the application notes. The compass has timing requirements that must be met in order to function properly. These are as shown in Figure 2-6. *PC* is brought low in order to request a fresh compass reading. Soon afterwards, *EOC* (End Of Conversion) will be brought low by the compass and will remain low until a new reading has been taken. At that time, *EOC* will be asserted by the compass. After waiting for a few milliseconds, $\bar{SS}$ can be brought low, indicating that the compass' data is ready to be transmitted to the controller. 16 rising and falling clock edges are then transmitted by the controller on the *C-CLK* line. During each clock cycle, a bit of data is sent from the compass to the controller on the *C-SDO* line. The first 7 bits that are sent across are dummy values. Starting with the eighth bit, valid compass data is sent across, MSB first. After all 16 rising and falling edges have been sent and the data has been read, $\bar{SS}$ can be re-asserted, and the compass should once again be in its initial state, ready to start the whole process over when *PC* is brought low by the controller.

Application of the compass is fairly straightforward. It is simple to use and reliable. There is one main precaution associated with it, however. It is that the tilt of the compass has a large impact on the data returned by the compass. Software that uses the compass data should compensate for this effect.

2.2.3 Altimeter-Compass System

These two sub-systems, the altimeter and compass, are interfaced together using a couple of small, cheap microprocessors. The original design for this system was done by Long Phan, an undergraduate student who has worked at Draper Labs for several years. He did all of the early development work

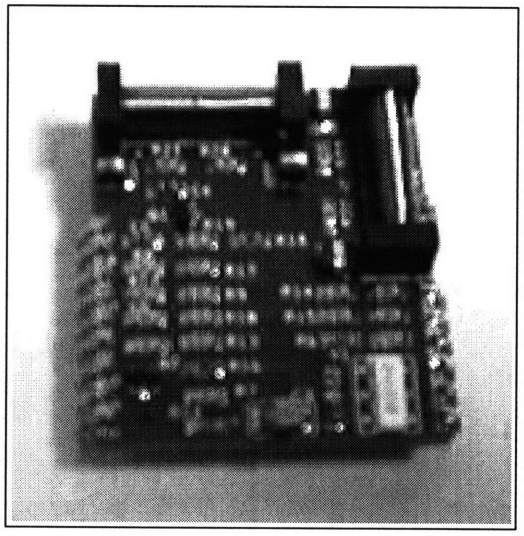

Figure 2-5: Electronic Compass

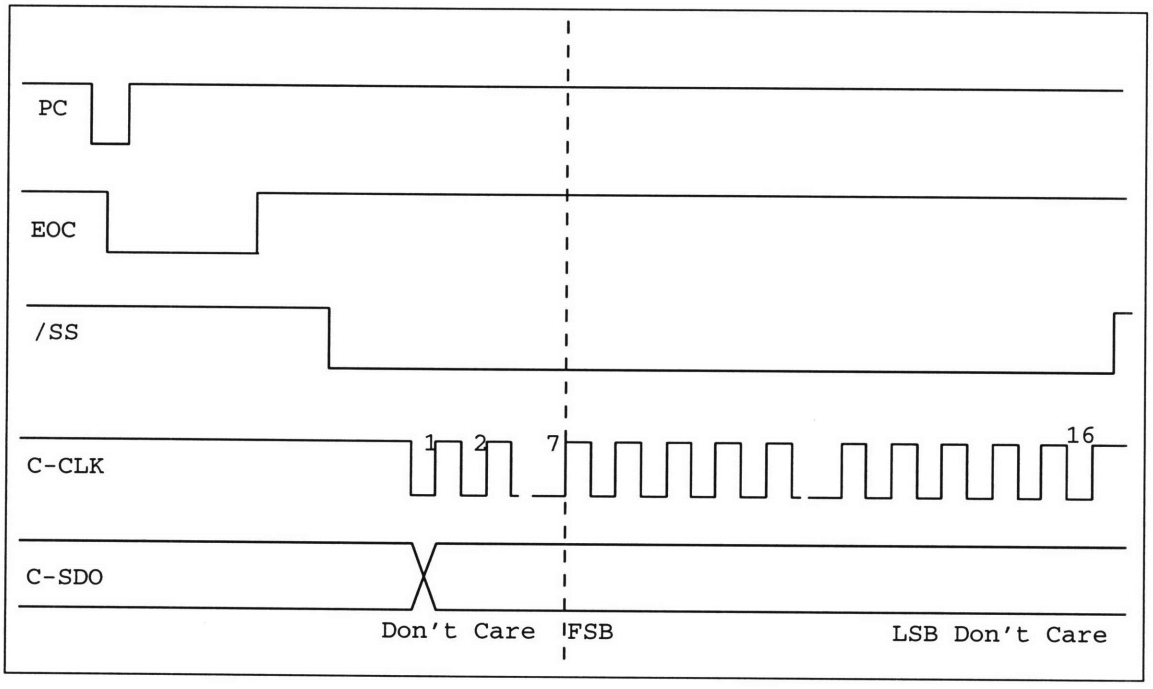

Figure 2-6: Compass Timing Diagram

μsync	*con1*	*con2*	Interpretation
0	X	X	Invalid Data
1	0	0	No Valid Data Available
1	0	1	Low Byte of Altitude Data Available
1	1	0	High Byte of Altitude Data Available
1	1	1	Compass Data Byte Available

Table 2.2: Altimeter-Compass Interface Signals

on getting the altimeter to work as a stand-alone system. His work showed that a pair of BASIC Stamp II processors were powerful enough to control the ranging board. These processors are digital controllers designed to be used for simple electronics projects. They have an instruction set with BASIC commands for serial communication, pulse measurements, digital I/O, etc. Later work showed that with a few simple additions to his design, the Basic Stamps could control both the altimeter and compass.

The basic idea behind the system is to have two microprocessors that control the instruments and send data back to the main CPU. Two microprocessors are necessary because of limitations of the BASIC Stamp instruction set. One processor is used to pulse the *Init* line at a regular rate. The other processor makes measurements of the *Echo* time, controls the compass, and sends data back to the main CPU.

The code for these processors is written in a form of BASIC known as Parallax BASIC. Appendix A has the listings of the programs that are used for the two processors. The processor that signals the altimeter to fire is referred to as the "Pinger." The processor that makes measurements and transmits data is referred to as the "Peripheral Stamp." Both code listings are very simple and can be easily understood by someone who studies the programming manual available from Parallax.

Figure 2-7 shows the schematic for the combined Altimeter/Compass system. In order for the "Peripheral Stamp" to interface with the main CPU, data is sent from the Stamp to an 8-bit shift register (DM74164). Three auxiliary signals, *μsync*, *con1*, and *con2* are used. These signals tell the main processor what kind of data is being sent and whether or not valid data is available. Table 2.2 summarizes the possible states of these signals. Although 9 bits of compass data are produced by the compass, only the high order 8 of these are sent to the CPU in order to reduce the number of bytes that must be transferred between the devices.

This scheme for transmitting data works fairly well. It makes it possible to minimize the number of wires between the BASIC Stamp and the main CPU (by placing the shift register near the CPU). The interface signal lines and the outputs of the shift register, labeled A-H in the schematic, are wired to the Digital I/O pins of the A/D converter. Table 2.3 shows the way in which these signals are wired.

2.3 GPS

The Global Positioning System (GPS) is composed of two main sub-systems. Each sub-system is made up of a GPS receiver and an antenna. With two receivers it is possible to achieve a much higher accuracy position estimate than with one receiver alone. Two receivers used together to improve accuracy in this way are referred to as a *Differential* GPS (DGPS). For the DSAAV, one sub-system is placed on the helicopter. The other sub-system (the ground reference station) is placed on the ground at a fixed position. The two receivers are designed to communicate with each other via a serial cable. Because tethers are not allowed between the ground and the helicopter, the DSAAV

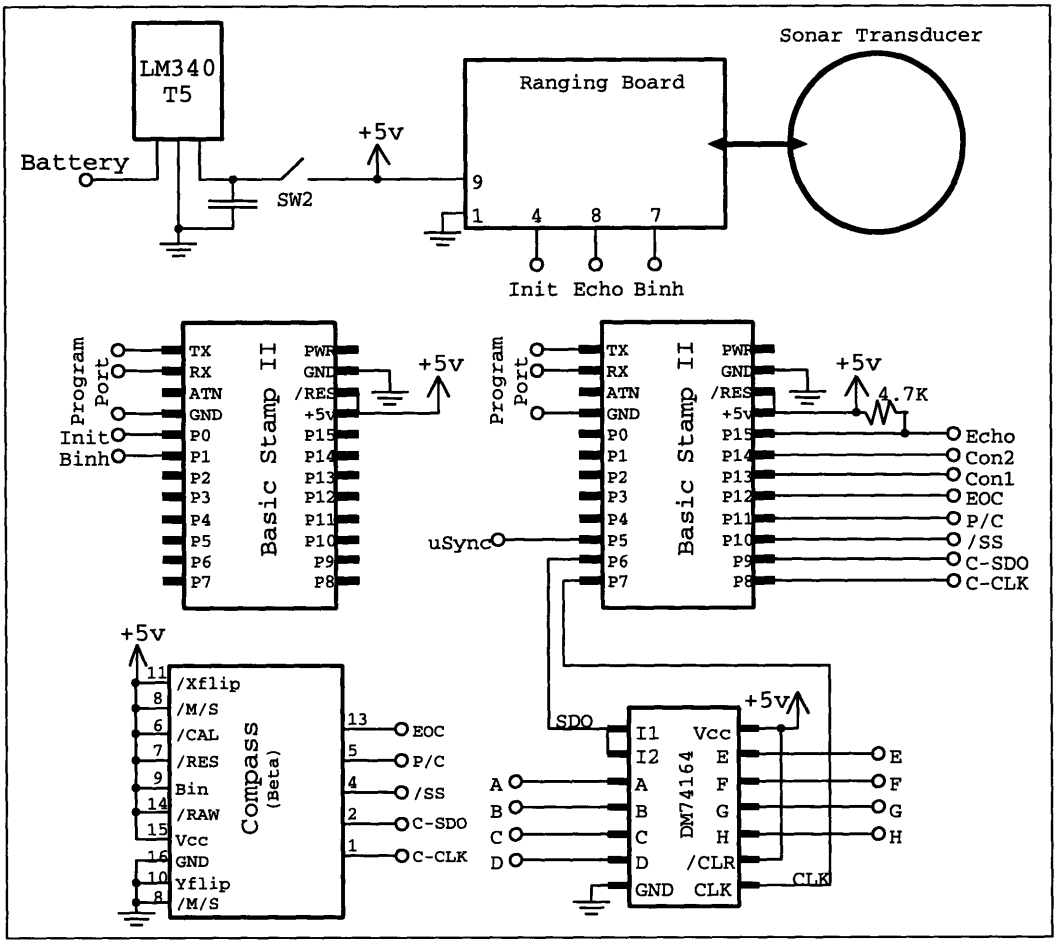

Figure 2-7: Sonar-Compass System Schematic

Signal	DM5406 Pin
H	23
G	25
F	27
E	29
D	31
C	33
B	35
A	37
µSync	24
Con1	26
Con2	28

Table 2.3: Pin Connection for Altimeter/Compass to DM5406 Interface (Connector P2)

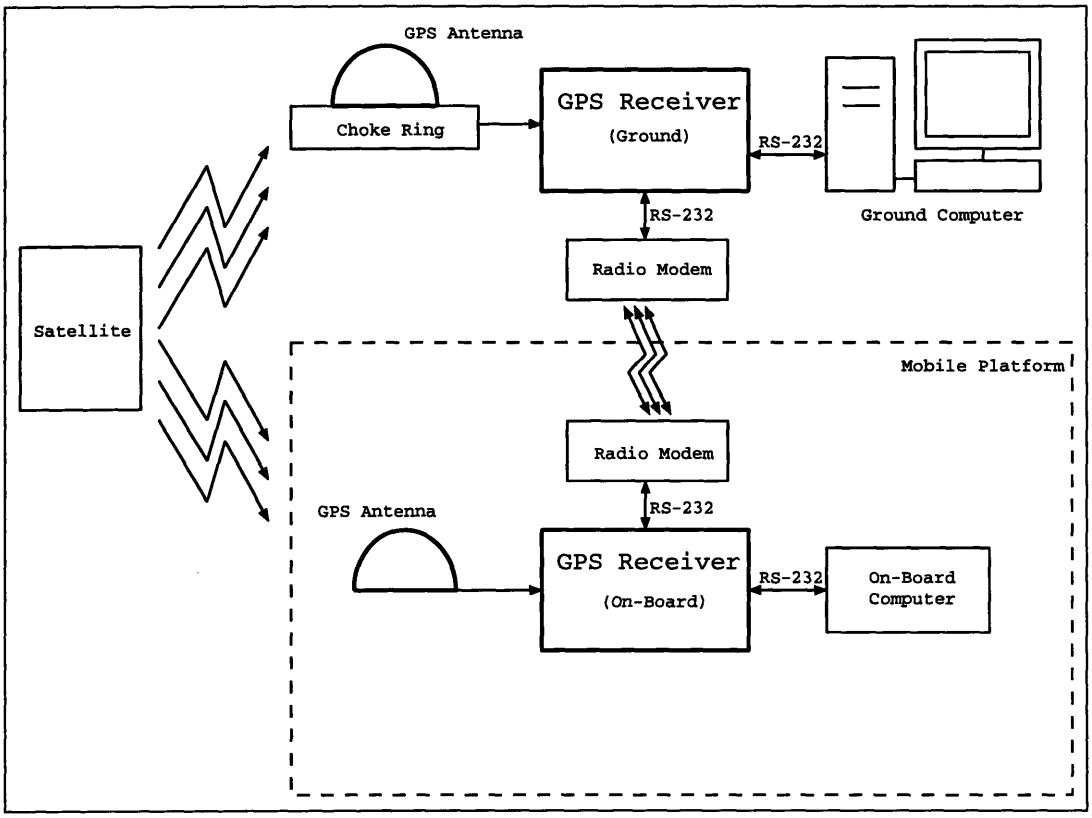

Figure 2-8: DGPS Topology

uses a radio frequency modem that acts like a serial cable but does not require a physical connection between the two receivers. Figure 2-8 shows the topology of the DGPS.

The GPS receiver used on board the helicopter is an OEM version which is small, lightweight, and easy to mount inside a custom enclosure. The ground station receiver is a larger unit that contains its own power converters. A *choke ring* is placed around the ground antenna. The choke ring reduces the effects of *multipath interference* which occur when satellite signals reflect from buildings or other objects and interfere with direct signals at the antenna.

The DGPS used with the DSAAV is capable of achieving 20 cm standard deviation of relative accuracy. The achieved performance of the DGPS depends on the number of satellites that are being tracked as well as the positions of those satellites. In practice, the achieved performance of the DGPS is something like $1\frac{1}{2}$ feet. This performance level is sufficient for all of the vehicle's maneuvers, including landing as long as the other sensors are performing well.

Each GPS receiver is connected to a computer via a serial cable. The on-board receiver is connected to the on-board computer and the ground receiver is connected to the ground station computer. These cables enable the computers to query the receivers for position and velocity estimates, as well as information about the performance of the DGPS.

2.4 Modem

The modems used for the helicopter system have two main purposes. First, they transfer DGPS correction data from the ground GPS receiver to the on-board GPS receiver. The second purpose is to transmit data between the on-board computer and the ground station computer. There are several types of data that are sent between the computers. First, navigation data is transmitted. This data is the on-board computer's estimate of its position, attitude, velocity, and angular rates. In addition, diagnostics are sent back to the ground computer including the main battery pack voltage, status of the altimeter (on/off), GPS system performance, etc. Finally, commands can be sent from the ground station computer to the on-board computer. In 1996, these commands were the way-points that the helicopter was to acquire.

Because there are two types of data being sent through the modems, it was at first thought that two independent modem links would be required. Significant weight saving was accomplished by only using one pair of modems. The trick to using one pair of modems is to use a feature offered by the GPS receivers called *pass through logs*. Pass through logs are messages formatted in such a way that the GPS receiver knows to send them on to the radio modem. On the other end, the radio modem will pass the message to the other GPS receiver which will send it to the computer to which it is serially connected.

The modem used on the DSAAV is a Proxlink II, made by Proxim, Inc. It operates in the 900MHz frequency range, which makes it legal to use without a license. It is a spread spectrum modem, meaning that it dynamically changes its frequency in order to improve the chances of getting its signal across on a clean channel. The output power of the modem is 500mW, and the rated range is 1000ft outdoors. In 1996, it was found that the achieved range of the modem on the DSAAV was something more like 100ft, and the range seemed to degrade over several months of time. This could be the fault of vibration damage to the modem.

Once the modems are configured, they behave just like a serial cable. Configuring the modems is straightforward. Simply connect them to a PC, run a terminal emulation program, and push the button on the front of the modem. A menu will appear that allows the user to choose from the many options offered by the modem. The only trick to the whole affair is that the user must be careful to use the right kind of serial cable between the computer and the modem. The correct cable has straight through connections on all of the pins except pins 9 and 1. Pins 9 and 1 must be left unconnected.

2.5 Receiver Interface

The DSAAV uses the servos and servo controller that come standard with the TSK model helicopter. For purposes of testing and for safety, it is necessary for the helicopter to be able to be controlled either by a human pilot or by the on-board computer. The interface between these two control sources is the Receiver Interface.

The receiver interface is responsible for sending the control signals to the servo controller, intercepting signals from the safety pilot radio, transmitting on-board computer commands to the servos, and establishing a fail-safe mode if necessary. The interface was designed by Bob Butler, a Draper Lab engineer.

Because the receiver interface controls the servos, its failure would guarantee a crash. In contrast, if any other system on the helicopter fails, there is a reasonable chance of the helicopter surviving. Because it is so critical, it was designed to be very robust, and it uses its own battery that is separate from the main vehicle power system. Therefore, if the main power system fails, the safety pilot will still be able to take control of the helicopter and pilot the vehicle down safely.

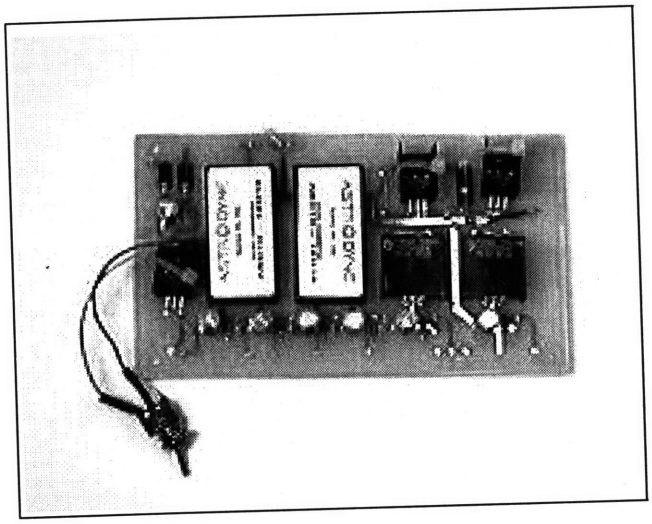

Figure 2-9: Power Board

2.6 Video

The video system is comprised of a stand-alone video camera with transmitter. It is powered by a 12 volt linear regulator. Other than the power supply, it has no interaction with the other on-board electronics. Video data is sent to the ground where it is processed by a Silicon Graphics workstation.

2.7 Power

Figure 2-9 shows a picture of the power board used in 1996. a combination of switching and linear regulators are used to regulate the 17volt battery to several voltages of interest on the vehicle. A detailed description of the power system is provided in Chapter 3.

2.8 Inertial System

Figure 2-10 shows a picture of the Inertial Measuring Unit (IMU) used on the Helicopter. The outputs of this unit are analog. They are filtered and sampled by the A/D converter on the computer stack. A detailed description of the inertial system is provided in Chapter 4.

2.9 Grounding

The grounding scheme is designed to rule out the possibility of a ground loop[1]. Because the main mounting hardware is made of aluminum, it was chosen to act as a ground for the vehicle. This choice is good because it creates a low resistance path for current to flow along the ground plane. It is also convenient to attach grounds from each component directly to the ground plane instead of running grounding wires around the vehicle. Ground wires are only allowed to run directly to

[1] A ground loop is any closed conductive path that serves as an inductive pick-up and hence introduces noise along the ground bus.

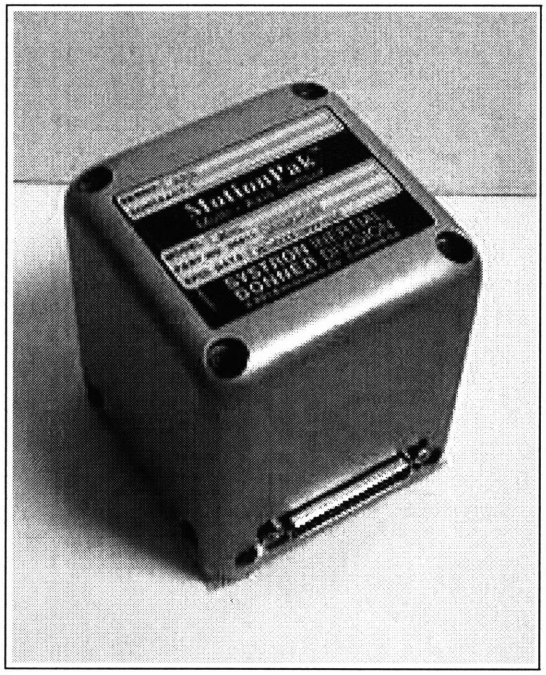

Figure 2-10: Systron Donner Inertial Sensor

the ground plane. This rule is applied to all except the compass ground, which is connected to the ranging board ground. Hence, no ground loops are possible.

2.10 A Word on Serial Cables

Serial cables are in general a messy business. It is surprising how much time can be spent getting something as silly as a serial cable to work. There are 25 pin (DB25) connectors, 9 pin (DB9) connectors, male connectors and female connectors. One must always double-check which pin is being soldered! Table 2.4 shows the pin assignment for a 9 pin serial connector.

The issue is further complicated by the fact that there are two types of *Devices* defined for use with a serial cable. These are DTE (Data Terminal Equipment) and DCE (Data Communications Equipment). Originally, the idea was that something like a computer would be a DTE, and a modem would be a DCE. A DTE would use the pin assignment shown in Table 2.4, while a DCE would have certain pins swapped so that a *straight through* cable could be used to connect the two. Therefore, one must always keep in mind if one is working with a DTE or a DCE. In order to connect two DTE's or two DCE's, a straight through cable will not work.

For many applications, a three wire serial cable is sufficient. This type of cable uses only pins 2,3 and 5 on the serial connector. The other pins on a serial connector are used for sophisticated transmission protocols that often are not necessary.

Pin	Assignment
1	DCD (Data Carrier Detect)
2	RX (Receive Data)
3	TX (Transmit Data)
4	DTR (Data Terminal Ready)
5	GND (Ground Signal)
6	DSR (Data Set Ready)
7	RTS (Request To Send)
8	CTS (Clear To Send)

Table 2.4: 9 Pin Serial Connector Pin Assignments

2.11 Ground Electronics

Ground support hardware is critical to a successful flight of the DSAAV. Fortunately, ground hardware is not constrained by the weight, power, or vibration constraints associated with hardware on the helicopter. A 486DX laptop serves as the main groundstation computer. The groundstation also consists of a radio modem, GPS receiver, and GPS antenna with choke ring. When testing is being done on the video system, A video receiver, television, VCR, and Silicon Graphics workstation are also used. In order to power all of the equipment, a 12 volt DC to 120 volt AC converter is used. A car battery is sufficient to power most of the equipment for two to four hours.

2.12 Conclusion

The integration of electronics requires detailed knowledge of all of the components of interest. The designer must know the ways in which the components may interact with one another. Communication protocols must be observed, power requirements must be met, and components must not interfere electro-magnetically with one another. The process of integration requires knowledge about the *whole* system, including its mechanical and electrical characteristics. Chapter 5 describes some of the problems that can develop in an integration effort.

Chapter 3

Power System Design

A critical part of system level design is providing the sub-systems with power at the proper voltage levels. Many components have fairly strict tolerances on their supplies. Restrictions include percentage error from nominal voltage level, peak-to-peak ripple voltage, and current level.

Power system design involves three main stages. First, the power requirements for the system must be evaluated. It is helpful to create a list of all of the components and their power requirements. Second, once the total amount of power needed is known, a power source can be selected. Finally, the electronics that convert the energy source into usable energy must be designed, built, and tested. Although it is useful to think of the design in these three distinct stages, they are in fact interrelated. Components may be rejected for the vehicle design based on unreasonable power demands. Similarly, The availability of voltage regulators may affect the design of the power source.

The power systems used in 1996 and 1997 are very similar. For simplicity, only the 1997 power system will be described in this chapter. The main difference between the two generations of power systems is brought about by the fact that a different modem is used in 1997 than was used in 1996. The new modem requires 15 volts as compared to the 8 volts needed in 1996. The other differences are due to the fact that in 1997, the main power board was PC boarded. In 1996, the main power board was only prototyped, and the prototype was used on the vehicle.

3.1 Power Requirements

The major components and their power requirements are shown in Table 3.1. Although the table is concise, it represents many weeks of work, as it takes a long time to determine what components should be used on the vehicle. Only after the components have been selected and data sheets are available can the real power system design begin. It is very helpful to have accurate numbers before the power system design is far underway.

3.2 Energy Source Selection

3.2.1 Power Source Type Selection

In general, there are many potential power sources for electronic components. A partial list includes wall outlets, solar cells, electro-mechanical generators, batteries, and nuclear generators. For small, mobile, power hungry vehicles, only a small number of power sources are available. For a vehicle like the DSAAV, only batteries or an electro-mechanical generator are feasible sources of power.

Sub-System	Voltage	Power (Watts)	Tolerances
Core Module (CPU)	+5	5.5	±5%
Ethernet	+5	1.2	±5%
A/D Board	+5	1.4	—
IMU	± 15	7.0	—
Modem	10.5-18	2.3	—
Compass	+5	.05	—
Video	+12	3.4	—
GPS	+12	1.0	±5%, 120mV
	-12	0.5	±5%, 120mV
	+5	4.5	+5%, -2.5%, 50mV
Ranging Board	+5	1.5	—
BASIC Stamps (2)	+5	.07	—
TOTAL	—	28.4	—

Table 3.1: Component Power Requirements

Although no careful analysis of generators has been done for the DSAAV, batteries were selected as the primary electrical power source. A generator could only be used while the engine is running. A battery, however, can be used before the engine is started, simplifying the testing of electronics. This feature is very useful for pre-flight testing and check-off.

3.2.2 Battery Chemistry Analysis

Many battery chemistries are commercially available. Battery chemistries are generally classified as either primary (nonrechargeable) or secondary (rechargeable). In order to make an intelligent selection of battery chemistry, criteria must be defined for the battery that will be used. The driving criterion for many components on the DSAAV was the weight of the component. This was true of the battery pack. Therefore, the *energy density* was the primary criterion of interest. In this context energy density is defined as the number of joules a battery can store divided by the weight of the battery[1].

Another criterion of interest is the internal resistance of the battery. The higher the internal resistance, the more power will be lost as heat within the battery itself. Because batteries are highly nonlinear chemical objects, the internal resistance is often a function of the current drawn from the battery. Because some of a batteries energy is always lost within the battery, it is useful to define the *usable energy* of a battery. This is the amount of energy that can be drawn from the battery's terminals. The usable energy is a function of the amount of current being drawn from the battery. A simple model for a battery is shown in Figure 3-1. It shows a voltage source in series with the internal resistance of the battery. Clearly, the internal resistor dissipates energy at a rate related to the current drawn from the battery. If the voltage source is thought of as having a finite number of milliamp-hours, then the resistor will reduce the total amount of energy that will be derived from the battery.

Another constraint on the cells is due to the fact that they will be used many times. Although it would be possible to buy many primary cells and throw them away after each use, it makes more sense to use secondary cells and recharge them after each use. For this reason, secondary

[1] This is in contrast to the usual definition of energy density as energy per unit volume

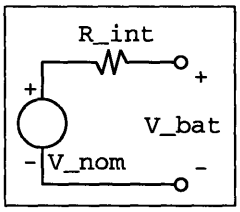

Figure 3-1: Battery Model

Chemistry	Voltage	R_{int}	C (mAh)	Weight (g)	Energy Density (J/g)	Usable J/g
Alkaline	1.5	.1Ω	8000	125	346	300
Li-Oxyhalide	3.5	NA	8500	120	893	≈ 800
Pb-acid	2.0	.01Ω	2000	180	80	79
NiMH	1.2	.02Ω	2000	35	247	239
NiCD	1.2	.01Ω	3500	130	116	114

Table 3.2: Battery Chemistry Analysis

cells are preferable to primary cells. Because the amount of usable energy is a function of the operating point of the battery, battery design is an iterative process. The chemistry affects the battery characteristics which affect the operating point which affects the battery performance which may make another chemistry more appropriate.

The battery industry uses a figure of merit, "C" (Capacity), to rate most battery cells. C is often expressed in units of mA×h (milliamp-hours). Hence, multiplying C by the rated cell voltage can give an estimate of the cell's milliwatt-hours (3.6 joules = 1 milliwatt-hour). However, this type of analysis only gives a rough approximation to the cell's energy content.

From the perspective of the power electronics, there are three main design parameters of interest. These parameters are operating current level, battery voltage, and usable energy. They are all coupled together by the chemistry of the cell and basic energy conservation. The availability of power regulators that can handle the operating point is also a consideration.

For a first pass at the design, the rated energy densities of different cell chemistries were compared. Table 3.2 shows the comparison data [5]. Comparison of cells in this way can be misleading, as it is often difficult to determine the conditions under which the cells were tested. However, getting rough engineering approximations for the figures of merit of the cells is worthwhile if detailed battery data is not readily available. To determine the numbers for usable energy density, an operating point of 2 amps was assumed, which is a realistic number for the DSAAV.

The two secondary cell types in the table are NiMH (Nickel Metal Hydride) and NiCd (Nickel Cadmium). Because the NiMH has a higher energy density than NiCd, it was chosen as a good first guess at the battery chemistry to use. In retrospect, the very high energy density of Li-Oxyhalide may make it a superior selection, even though it is not rechargeable. Future improvements to the DSAAV may make a change in the battery type.

NiMH cells were chosen as the battery chemistry for the DSAAV. This chemistry is fairly light, and can supply high current levels (several amps). They are not particularly expensive and readily available. The fact that they are rechargeable is a big advantage as well.

Cell Capacity	# Cells	Total Energy	Expected Life	Weight
1800 mA-h	14	91,000 joules	38 minutes	490g (1.1 lbs)
2500 mA-h	14	126,000 joules	53 minutes	640g (1.4 lbs)

Table 3.3: Battery Pack Design

3.2.3 Battery Pack Design

Once the cell chemistry has been selected, a battery must be designed with the cells. As was true for cell chemistry selection, criteria must be defined for the battery design. For the DSAAV, the criterion used was the expected life of a battery. Flight times of 30 minutes were anticipated. Therefore, a battery that could sustain flight for 30 minutes was the design goal. The data for the design came from GP Batteries[2].

The first step of the design is to approximate the total amount of power needed for the 30 minute flight. Table 3.1 shows that approximately 28 watts of power are needed by all of the components. This number was multiplied by 1.2 to account for the inevitable inefficiency of power regulators[3]. To be conservative, another factor of 1.2 was multiplied to account for losses in the batteries at the high operating point at which they are used. This brings the power level up to 40 watts.

30 minutes of flight at 40 watts requires 72,000 joules. Therefore, the battery used on the DSAAV should have 72,000 joules of energy. There are two main ways to increase the number of joules in a battery. First, larger cells may be used. Second, more cells may be used. For many cell chemistries, manufacturers discourage the use of cells in parallel with each other. Putting cells in parallel can cause the cells to charge/discharge each other. Therefore, if more cells are added to a battery, they should be added in series, thereby increasing the battery voltage and the battery resistance[4].

After a few design iterations, a battery of 14 cells was selected. This battery has a nominal voltage of 16.8 volts. This is based on the nominal cell voltage of 1.2 volts. However, the cell voltage may be anywhere in the range from .9 volts to 1.5 volts, depending on its health and level of charge. Therefore, the battery voltage may be anywhere from 12.6 volts to 21 volts. The switching regulators used on the DSAAV operate down to 9 volts, thereby utilizing almost all of the usable energy in the cells. The 15 and 12 volt linear regulators are allowed to go out of regulation as the battery voltage drops below their allowable operating voltage.

Two different battery packs were used for the DSAAV. One was based on an 2500 mA-h cell. The other was based on a 1800mA-H cell. Table 3.3 compares the two batteries. It was found that these cells performed roughly as predicted. The power draw of the on-board electronics was measured at 38 watts[5]. At this power level, 2.3 amps is drawn at the nominal battery voltage. Hence, the power lost due to the internal cell resistance is $P_{loss} = I^2 \times R$ ($R = 14 \times .02\Omega = .28\Omega$) = 1.5 watts. However, more power than this was lost in the batteries due to the higher internal resistance at the high operating current level.

Data was taken of the battery voltage while it powered the DSAAV. The battery that was tested was based on the 1800mA-h cells. The battery discharge curve for this pack is shown in Figure 3-2. The plot shows that the battery does not last as long as expected. This is attributed to there being more loss in the battery than predicted. In reality, the life of a battery depends on its general health, and one discharge does not show how long the battery may last if it is in poorer or better health.

[2] See Appendix C

[3] Switching regulators have efficiencies of about 80%. Linear regulators have efficiencies related to their operating voltage.

[4] Battery resistance can be approximated as the internal resistance of a cell times the number of cells.

[5] This is slightly higher than predicted due to the inefficiency of the linear regulators.

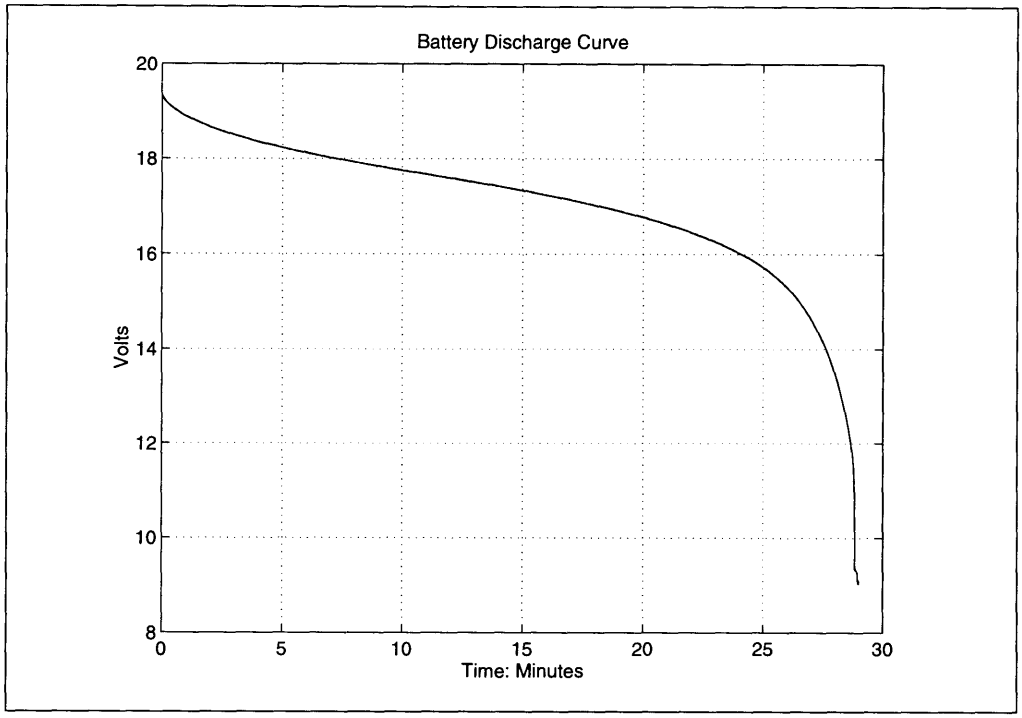

Figure 3-2: Battery Discharge Curve

Regulator	Type	Power Supplied	Maximum Suppliable Power	Total Power Used
5v Proc	Switching	8.1W	15.0W	9.3W
5v Main	Switching	4.6W	15.0W	5.3W
5v Range	Linear	1.5W	—	5.0W
+12v	Switching	1.0W	5.0W	1.2W
-12v	Switching	0.5W	5.0W	0.6W
± 15v	Switching	7.0W	10.0W	8.1W
12v Video	Linear	3.4W	—	4.8W
15v Modem	Linear	2.3W	—	3.1W
TOTAL	—	28.4W	—	37.4W

Table 3.4: Power Regulators and Associated Power

3.3 Power Electronics Design

Once a source of power has been selected, its power must be converted into a usable form for the electronic components. This is done with power converters or power regulators. Power regulators are classified as switching regulators or linear regulators.

Switching regulators convert one voltage to another by pumping current from the source to a load capacitor. This method for voltage conversion can be very efficient. Efficiencies of 85% are common. Depending on the design of the switching regulator, it may convert a lower voltage to a higher voltage, a higher voltage to a lower voltage, or have the capability of doing either. They can even convert a positive voltage into a negative voltage. On the down side, switching regulators have a ripple voltage associated with their output. This ripple voltage can affect the performance of the electronics that they power. The high speed switching also has the disadvantage of introducing electro-magnetic radiation into the environment that can interfere with other electronics.

Linear regulators on the other hand can only have an output voltage that is between their input voltage and ground. They are generally less efficient than switching regulators because they dissipate excess energy as heat. For example, consider a 5 volt linear regulator being powered from a 15 volt battery. If the regulator supplies 100mA of current, then it supplies 5 volts $\times$.1 amps = .5 watts. At the same time $(15v - 5v) \times .1\text{mA} = 1$ watt is being lost as heat. Designers must be careful that the linear regulator will be able to dissipate all of the heat that it generates. Otherwise, the regulator will overheat and malfunction. Linear regulators have good points, however. The have less ripple associated with their output voltage and have higher bandwidth than switching regulators. They are small and inexpensive.

For the DSAAV, switching regulators are used to generate all of the supplies that required large amounts of power and all negative supplies. Linear regulators are used for supplies that required little power. Ideally, no linear regulators would be needed. They were added as the design evolved to compensate for design changes. A total of seven regulators are used. Two of the regulators have dual outputs. This gives a total of 9 voltage supply lines. Table 3.4 lists the converters and the power associated with them. Table 3.5 gives information needed for locating the specific converters used on the DSAAV. Finally, Table 3.6 shows the specifications for the switching converters.

Designing a *power board* with these regulators is straightforward as long as a few simple rules and design goals are kept in mind. First, the ripple associated with switching regulators should be minimized. This is done by adding capacitors on the output of these regulators. Capacitors are also added to the input of the regulators in order to help the battery supply impulses of current when they are needed. Second, one must be careful about the startup condition of a regulator. Sometimes the regulators can develop a high voltage when they first turn on. This voltage could potentially

Regulator	Type	Manufacturer	Part No.
5v Proc	Switching	Power Trends	78HT205HC
5v Main	Switching	Power Trends	78HT205HC
5v Range	Linear	—	LM340AT-5.0
± 12v	Switching	Astrodyne	ASD10-12D12
± 15v	Switching	Astrodyne	ASD10-12D15
12v Video	Linear	—	LM340AT-12
15v Modem	Linear	—	LM340AT-15

Table 3.5: Power Regulator Reference Information

Converter	Maximum Power	Ripple Voltage	Input Voltage Range
78HT205HC	15 watts	50 mV (typ)	7 - 28 v
ASD10-12DXX	10 watts	100 mV (max)	9 - 36 v

Table 3.6: Switching Regulator Specifications

damage equipment. Therefore, zener diodes are added to the output of these regulators to clamp the output voltage.

The DSAAV power board design is shown in Figure 3-3. Power from the battery is routed through a power diode and a switch to the regulators. The purpose of the diode is to enable "Hot Swapping" of battery packs. This means that two battery packs can be connected to the power board at the same time. Therefore, when one battery pack is running low, another can be connected before the first is disconnected. This means that the electronics need not be powered down for battery swapping.

One may comment on the lack of fuses in the circuit. Fuses are not used because most of the regulators have current limiting circuitry. It was decided that the added complexity and weight associated with fuses was not worth the protection that they provide. Only a few times during work on the vehicle did a positive voltage get shorted to ground. These incidents usually resulted in a spark, but no real damage occurred.

The linear regulators in the circuit are thermally protected in two ways. First, the five volt regulator is protected with a resistor from the supply to the regulator input. This resistor dissipates energy that would otherwise be dissipated in the regulator. The five volt regulator wastes more energy than the other linear regulators, and hence needs more thermal protection than the others. The other protection is in the form of heat sinks that draw heat away from the regulator into the environment. Specification sheets for the regulators give plots for the amount of protection they need for different conditions.

In 1997, the power board was PC boarded. The layout for the board was done by an undergraduate student, Dmitry Brant, who helped out in many ways on the 1996 DSAAV. He used a program called EZ-Route that is available for PC's. The board itself was made by Alberta Printed Circuits, a Canadian based prototyping house. They offer double sided boards with plated through holes. The layout that was made for these boards is shown in Figures 3-4 through 3-6. In order, these figures show the first layer of metal, the second layer of metal, and the silk screen layer.

The system as a whole works fairly well. It reliably powers the sub-systems on the DSAAV. It is not perfect, however. Because the power requirements changed as time went on, some changes were made to the power board design that would not have been designed in originally. Ideally, linear

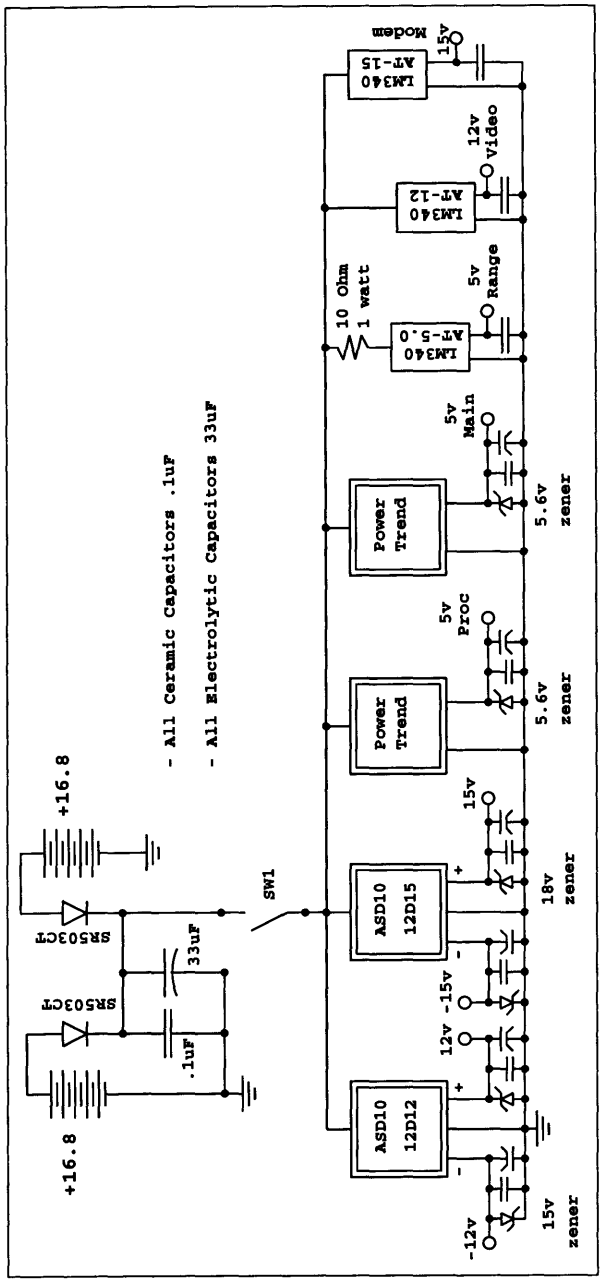

Figure 3-3: Power Board Schematic

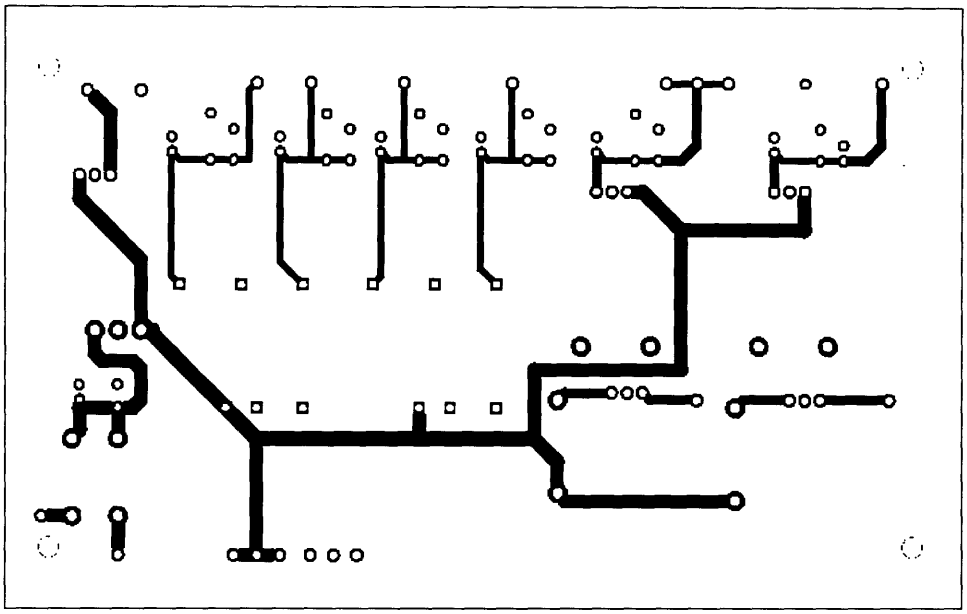

Figure 3-4: Power Board Layout — Metal Layer 1

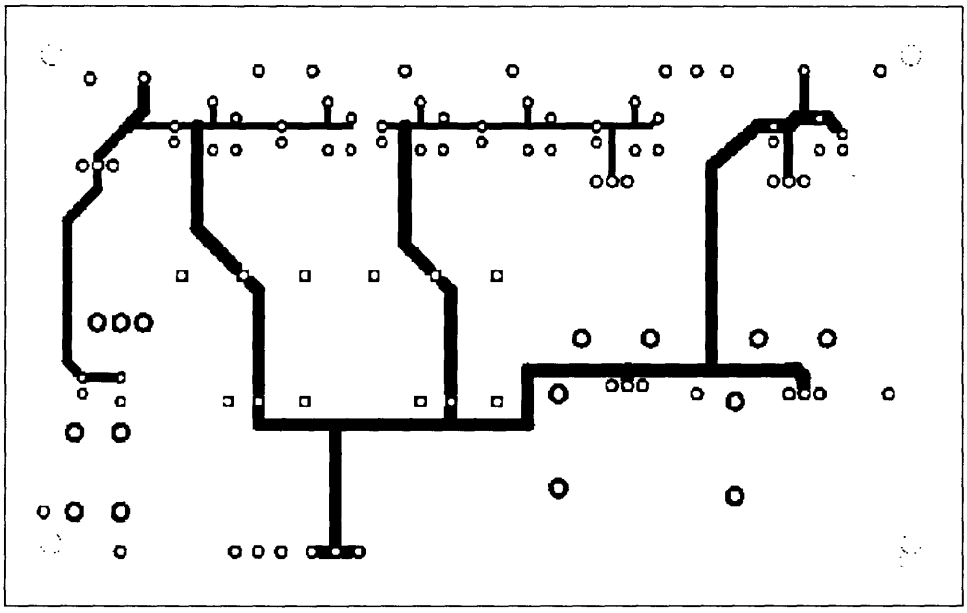

Figure 3-5: Power Board Layout — Metal Layer 2

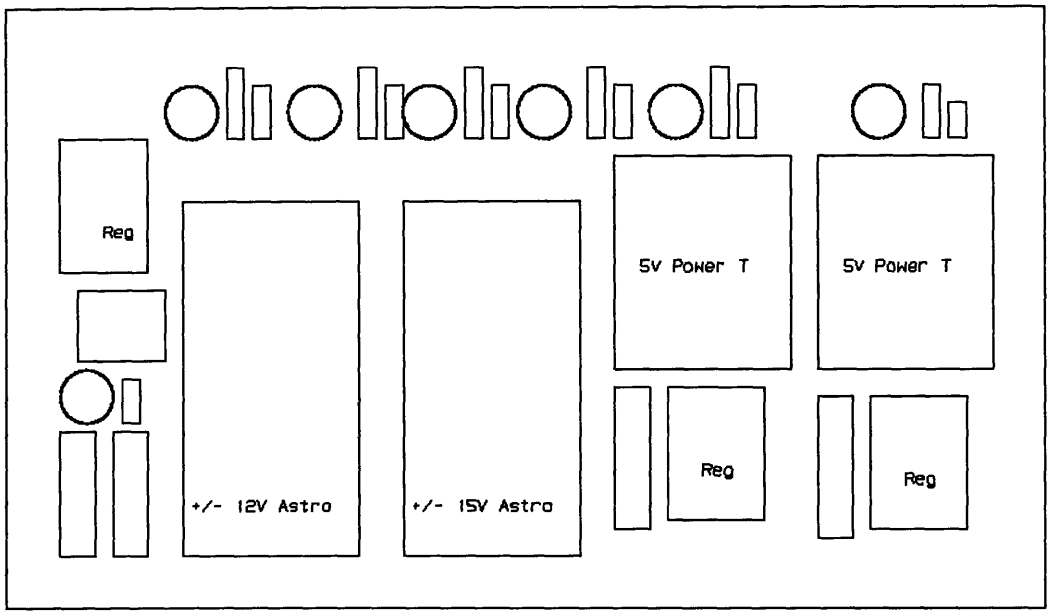

Figure 3-6: Power Board Layout — Silk Screen Layer

regulators would not be needed at all. They were added to supply odd voltages, or components that had special power needs. The ranging board, for instance, has its own 5 volt supply in order to reduce the affects of noise that it adds to its supply.

Another problem with the board is related to the Astrodyne converters. These converters like to have an equal amount of current drawn from the negative and positive supplies. If the currents are not equal, the voltages get skewed. This problem has never fully been addressed. Future versions of the power system should address this issue.

3.4 Thermal Considerations

All of the regulators used on the vehicle lose some energy in the form of heat. This heat must be drawn away from the regulators. For this reason, it is desirable to have a source of convection across the surface of the board. On the other hand, as section 5.4 on page 55 discusses, the power board must be electro-magnetically shielded from the other components. Therefore, bug screen is used for shielding. Bug screen has small enough holes and sufficient conductance to provide good shielding while still allowing air to flow to the regulators.

3.5 Interconnection

It is convenient to be able to connect and disconnect power to individual components without providing switches for each component or re-soldering wires every time. This is accomplished with connectors.

Although the issue of connectors is mundane, they are an important part of a robust design. They must be able to handle many insertion cycles without failure, must not become disconnected under vibration, etc. The connector issue is one that is worth researching. Many different types of

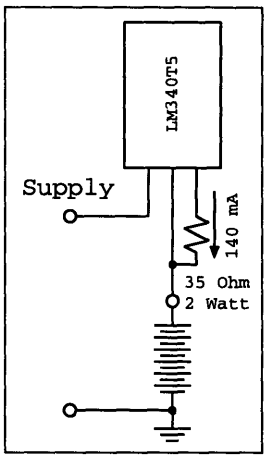

Figure 3-7: Battery Charger

connectors are available. The ones chosen for the 1996 DSAAV[6] worked well, but they may not be ideal. Further work should be done in this area to decide if another choice would be better.

3.6 Battery Charger

In addition to the power board, auxiliary electronics are needed to charge the batteries. NiMH batteries can safely be trickle charged at a C/10 rate[7]. Therefore, power current sources were used to charge the batteries. The design for the charger is shown in Figure 3-7. The 5 volt regulator places 5 volts across a 35 ohm power resistor resulting in a 140 milliamp current flow into the battery. Four of these circuits were used in one package to charge four batteries at a time.

3.7 Conclusion

Power system design for a vehicle like the DSAAV is not a trivial task. Much thought goes into the power requirements for the components, the vehicle energy source, and the power electronics. The power system on the DSAAV is adequate for the job, but there are improvements that can be made. Chapter 6 will describe potential improvements in more detail.

[6] See Appendix B

[7] That is, for a 1800mA-h cell, it can be charged at a rate of 180 mA indefinitely.

Chapter 4

Inertial System Design

The inertial measuring unit (IMU) is the most sensitive of the four sensors on the DSAAV. Its purpose to convert acceleration and angular velocity into signals that can be processed by the computer. It is an analog instrument, meaning that the voltage and current outputs of the unit are continuous variables. Its performance is affected by temperature, the amount of vibration, and electro-magnetic interference (EMI).

4.1 IMU Specifications and Performance

Weighing 890g (1.96lb), the IMU is housed in a thick metal casing. Inside are three rate gyros, three accelerometers, a temperature sensor, and power regulators. The inertial sensors are mounted orthogonally from each other. It is powered with ± 15 volts. Because there are internal regulators, the supplies to the IMU need not be highly regulated. Tolerances of 15% are associate with these supplies[1]. 250mA are drawn from the +15 volt supply. 200mA are drawn from the -15 volt supply. Hence, the total power used by the IMU is 6.8 watts.

The temperature sensor is an AD590, made by Analog Devices, Inc. It provides a current output that is proportional to absolute temperature. The scale factor is 1μA per degree Kelvin.

The output from the rate gyros is voltage proportional to angular rate. These outputs can range from -5 volts to +5 volts. The output from the accelerometers is current proportional to acceleration. These currents can be driven into fairly high impedances as long as the magnitude of the voltage does not exceed 5 volts. Many specifications are associated with these sensors, as shown in Table 4.1. The table is shows data taken with the IMU used in 1997. The values for the IMU used in 1996 are very similar.

When the IMU is powered, it begins to warm up. As it warms up, there are significant changes of the biases in the sensors. Therefore, if thermal compensation is not being used via the temperature sensor, the instrument should be allowed to fully warm up before being used. Normally, the warm-up time for the instrument is between thirty and forty minutes.

The bandwidth of the gyros is approximately 70 Hz. For the accelerometers, the bandwidth is significantly higher. Bode plots of the frequency responses of the sensors show that past their -3dB points, they have roughly single-pole roll-offs (-20dB/decade).

[1] This is good because of the asymmetry in the ± 15 volt regulator. Since unequal currents are drawn from the +15 volt supply and -15 volt supply, these voltages are not well regulated. See page 44 for more details.

	X Gyro	Y Gyro	Z Gyro	X Accel.	Y Accel.	Z Accel.
Range	$\pm 100°/s$	$\pm 100°/s$	$\pm 200°/s$	$\pm 15g$	$\pm 15g$	$\pm 15g$
Scale Factor	25.155 mV/°/s	25.136 mV/°/s	12.417 mV/°/s	2.887 mA/g	2.940 mA/g	2.947 mA/g
Temp. Performance	$< \pm 0.03\%/°C$	$< \pm 0.03\%/°C$	$< \pm 0.03\%/°C$	$-0.001\%/°C$	$-0.001\%/°C$	$-0.001\%/°C$
Bias ($22°C$)	0.09 °/s	-0.12 °/s	0.23 °/s	2.89 mg	0.75 mg	5.65 mg
Temp Performance	< 3 °/s	< 3 °/s	< 3 °/s	-46 $\mu g/°C$	-71 $\mu g/°C$	-10 $\mu g/°C$
Alignment	0.59 °	0.26 °	0.11 °	0.24 °	0.22 °	0.28 °
Bandwidth	68 Hz	72 Hz	71 Hz	1661 Hz	1605 Hz	1597 Hz
Damping	0.69	0.68	0.66	0.43	0.38	0.46
Noise (10-100Hz)	2.0 mVRMS	1.9 mVRMS	1.1 mVRMS	2.2 μARMS	1.8μARMS	2.5 μARMS

Table 4.1: Inertial Sensor Specifications

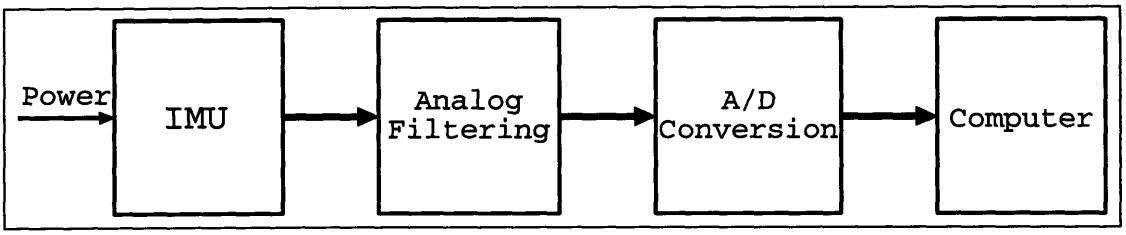

Figure 4-1: Inertial System Topology

4.2 Inertial System Design

The signals from the IMU must be conditioned in order to be useful to the computer. Figure 4-1 shows the topology that is used for the inertial system. The analog signals from the IMU are conditioned by analog filters before being sampled on the analog-to-digital (A/D) conversion board. The purpose of the system is to give the main computer information about the acceleration of the vehicle. This information is used directly, and is also integrated to obtain estimates for velocity, position, and angle.

Whenever an analog signal is being sampled, the issue of the sampling rate becomes important. Given adequate processing power, higher rate sampling is generally better than lower rate sampling. On the 1996 DSAAV, 50 Hz was the maximum sampling rate that could be achieved. A higher rate would have required more processing power. The Nyquist frequency for the system, then, is 25 Hz. The analog filters have their poles at 5 Hz, thereby providing an attenuation of approximately .04 at 25 Hz. It is important to prevent high frequency signals from being sampled, as they increase the standard deviation of the samples and provide no useful inertial information. Therefore, the cutoff frequency for the filters had to be chosen to be quite low. It would be nice to be able to obtain higher bandwidth data from the inertial sensors. The 50 Hz sampling rate, however, does not make this feasible. Chapter 7 will discuss this issue in much more detail.

The signals from the IMU have a natural bias associated with them. This bias must be subtracted out by software that uses the inertial information. To make the issue more complicated, the bias changes as a function of temperature. In 1996, no use was made of the temperature sensor inside the IMU. However, compensating for bias drift is one way in which the system performance can be improved.

Another point of concern with the IMU is its saturation levels. The instrument has maximum accelerations and angular velocities that it can measure. If the real accelerations and angular velocities exceed these levels, the sensor outputs will be pinned to plus or minus 5 volts. Therefore, it is

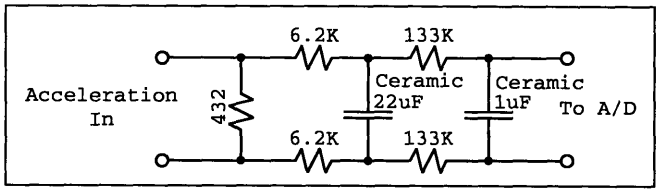

Figure 4-2: Acceleration Channel Filter

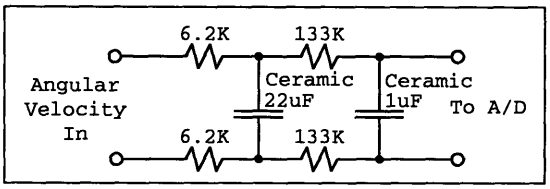

Figure 4-3: Angular Velocity Channel Filter

important to keep the accelerations down. The vibration of the helicopter can saturate the sensors if care is not taken to mechanically isolate the IMU.

4.3 Filtering

The analog filters have two main purposes. First, they convert the current outputs of the IMU into voltages which can be read by the A/D converter. Second, they reduce energy content of the signals in the higher frequency ranges. In this context, "higher frequency ranges" means anything above 5 Hz. The system is only capable of using information in the frequency band below 5 Hz. Signals outside this band only serve to increase the effective noise of the system. This leads to larger standard deviations and faster drift when the signals are integrated.

There are three main sources of signals outside the 5 Hz bandwidth of the system. The first is internal sensor noise, as shown in Table 4.1. The second is vibration concentrated at 20 Hz and harmonics of 20 Hz. The third is EMI. None of these three sources carries useful information, so they filters attempt to reject them.

The filters are simple by design. They are two-pole passive RC filters as shown in Figures 4-2 though 4-3. There are two main reasons for using passive filters. Originally, only a one pole passive filter was used. When it was determined that a second pole would be necessary, it was simplest and quickest simply to extend the all-passive design with another passive pole. Second, active filters are much more complex, requiring amplifiers and power supplies. Complexity translates into avoidable weight.

The filters have several important features. Because the signals from the IMU are differential[2], the filters were designed to be symmetric with respect to the inputs. Therefore, they filter the signal differentially. Ultimately, the A/D converter samples the signal differentially. Another important feature is the output impedance, which is two orders of magnitude less than the input impedance to

[2]Each sensor has its own signal line and its own return (ground) line. Therefore, the return lines carry little current and will not have large voltage drops

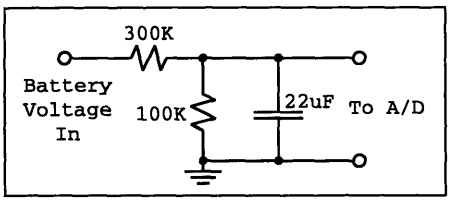

Figure 4-4: Battery Monitor Filter

the A/D converter. The relatively low impedance is necessary to reduce errors at the input to the A/D converter[3].

The damping coefficient for a passive filter can never be less than or equal to 1. This means that the roll-off will be slow, and the filter pass-band will not be particularly flat. In order to get as close as possible to the unity damping coefficient, the impedance of the first stage filter resistors are made small compared to the impedance of the second stage filter resistors. In this case, the ratio of the impedances is 20:1, so the damping coefficient will be close to 1.

For the acceleration channels, the 432Ω resistor converts the current to a voltage. This resistor value is small enough that it has only a small impact on the pole frequencies, which were designed to be near 5 Hz. The filter for the battery voltage is shown in Figure 4-4. This filter is located on the same board as the inertial system filters. In the case of this filter, the battery voltage is divided by four, and then low pass filtered before being sampled by the A/D converter.

4.4 Driver Software

Once signals have been conditioned for the A/D converter, they must be sampled and read in by the processor. Both of these processes requires software. In particular, low level driver software need to be available for the applications programmer to call. The driver used for the 1996 DSAAV is listed in section A.3. A detailed description of the code is not in order, as it is largely self documenting and will no longer be used in 1997. It may be useful to others, however, who have need of an example of functional driver software. As described in Chapter 2, the driver was written for the QNX operating system.

4.5 Conclusion

The inertial system provides a means for the main computer to get frequent updates of the vehicles acceleration and angular velocity. Analog signals from the IMU are conditioned by the filters before being sampled by the A/D converter. Once the signal is available to the A/D board, software takes control of the process, making it possible to perform useful computations with the resulting information.

[3] In retrospect, 2 orders of magnitude is not enough to do a really good job for a a 12 bit converter. However, the errors that arise from the output impedance of the filter should be linear with voltage, and therefore are compensated for in software.

Chapter 5

Major Challenges and Solutions

The development of an autonomous vehicle is a challenge. New hardware must be designed, built, and debugged. New software must be designed, written, and debugged. Then, the software must be made to work properly with the hardware. Each of these tasks is time consuming and challenging by itself. However, when everything is put together into a system, new challenges arise. These challenges have to do with unforeseen sub-system interactions and hardware limitations. On the DSAAV, there were three major challenges that presented themselves over the course of the first year of development. One after the other, they were encountered and had to be solved before progress could be made. These challenges were the vehicle weight limit, mechanical vibration, and electro-magnetic interference (EMI).

Before these three problem areas can be addressed, the methods used to test the vehicle should be discussed. Test was the main method of diagnosis and characterization of problems with the DSAAV. A careful test procedure can be very effective for solving problems. On a vehicle like the DSAAV, it is also important that the test procedures be in accordance with safety. Therefore, safety procedures were developed[1].

5.1 Test Methods and Procedures

There is a natural progression to testing as a project develops. Each system goes through several stages of development. To start with, the interface to each component must be understood. If custom electronics are necessary for use with the components, they must be designed. Each building block must be thoroughly checked as a stand-alone unit before any attempt is made to integrate it with other components. Therefore, the correct behavior of each block should be known and tested to whatever degree is reasonable. Then, one by one, the blocks can be hooked together. Again, whenever a new block is added, the whole system should be thoroughly checked out. Frequently, the testing of these partially integrated systems requires software. Whatever software is developed for hardware test will likely evolve into the hardware drivers. Therefore, thought should go into the design of this software from the outset.

Once the integrated systems work properly, the "bench test" stage is over. In the case of the DSAAV, the next stage of testing was the "cart test." Cart testing involves mounting the electronics onto the helicopter which is clamped to a mobile cart. The cart with helicopter was taken outside, where the DGPS must be used in order to track satellites. The ground system was also taken outside. Together the whole system was tested by pushing the cart around and seeing how well

[1] To date, no injuries have been associated with the DSAAV.

the navigation system could determine the cart position, how quickly it reacted, the ways in which errors occurred, etc. This stage of testing yielded many hardware and software revisions. It was at this stage of testing that problems with EMI were first encountered.

The third test stage involves turning the helicopter engine on while the helicopter is clamped to the cart. This is the "engine test" stage. At this stage, safety becomes a real issue. Several procedures emerged for managing the danger. People were posted around the vehicle who were responsible for observing the vehicle for different vantage points (from a considerable distance). The ground-station operator kept an eye on the diagnostic annunciators built into the operator interface. A "Kill Operator" held the radio that could kill the engine should anyone anticipate a problem and raise his arm. During the engine test stage of development, problems with mechanical vibration first emerged. These problems had to be solved before work could continue into the final stage of testing.

The final type of test is the "flight test." A flight test involves taking all of the necessary equipment[2] to a field and setting up the ground-station. The set up takes approximately 20 minutes for a group of four people. Once everything is set up, the on-board electronics are turned on and allowed to warm up. The range of the radio transmitters is tested by the kill operator and the safety pilot. The ground-station operator can determine if the electronic systems are functioning as they should be. If everything looks good, the engine is started and the safety pilot flies the helicopter for a brief test flight to make sure it is working well mechanically. Finally, the on-board computer is given control of the vehicle, and the ground-station operator can command control system or navigation system tests.

5.2 Vehicle Lifting Limitations (Weight Margin)

The TSK BlackStar model helicopter used in 1996 weighs about 15 pounds before it is loaded with electronics. There are two main environmental factors that affect how much weight a helicopter can lift. The first is the air density. As the temperature rises and the air becomes less dense, the rotor blades are able to displace less mass per unit time, resulting in less lift. The second factor is the air temperature which affects the efficiency of the engine. A higher temperature reduces the efficiency of the engine, decreasing the vehicles capability to lift.

It is not desirable to load down the vehicle with the maximum amount of load it can carry. Ideally, the vehicle will have a large *weight margin*. This is the difference between the maximum load it can lift and the load it is actually lifting. For a helicopter the size of the DSAAV, at least a couple pounds of weight margin is necessary in order for the vehicle to be able to maneuver well.

To find the original (unloaded) weight margin of the vehicle, it was loaded down with weights. Weights were added until the vehicle could not fly out of ground effect[3]. These tests were done in the middle of the winter, when the air was quite dense and the temperature was low. It was found that the vehicle's weight margin was approximately 10 pounds under these conditions. Allowing a 2 pound margin for hot conditions, and 2 pounds for maneuverability leaves 6 pounds available for electronics. This is very little weight considering the amount of electronics that are involved in the DSAAV!

The problem of the weight margin was known from the beginning of the project. Two approaches to dealing with the problem were pursued. The first was to adjust the clutch mechanism in the engine. By removing the springs in the centrifugal clutch, the frictional force in the clutch was increased, reducing the amount of clutch slippage. This change improved the vehicles weight margin by approximately 2 pounds.

[2] Surprisingly little equipment is required to perform a flight of the DSAAV. If the vision system is not being tested, the equipment can fit on an ordinary cart.

[3] Ground effect is the added lift and turbulence that a helicopter experiences when it flies near the ground.

Component	Weight (lbs.)
IMU	1.96
Battery	1.40
Power Board w/ Cabling	0.40
Computer Stack w/ case	1.32
GPS card	0.40
GPS antenna	0.31
Compass	0.02
Altimeter	0.03
Modem	0.74
Receiver Interface	0.42
Misc. Cabling/Mounting	1.00
TOTAL	8.00

Table 5.1: Major Components and Their Weights

The second approach was to minimize the weight added to the vehicle. There were many weights that could not be changed. Most of the electronic components had set weights. However, the packaging was one area in which there was great freedom. Instead of using aluminum enclosures for components, balsa wood frames were built and covered with mylar film. Thin sheets of laminated wood were used for surfaces to which connectors were mounted. The enclosures, then, were very light. Other methods of decreasing the weight added to the vehicle are minimizing the amount of custom electronics added and drilling lightening holes through surfaces where possible. Table 5.1 shows the weights of the major components on the vehicle. As can be seen from the total, the weight margin was a tough specification to meet.

To mount the components to the vehicle, a sturdy frame was necessary. This frame was made of perforated aluminum sheets and small aluminum L-brackets. This frame was used for both mechanical mounting of the electronics and as the ground plane for the power system and signal lines.

5.3 Mechanical Vibration

A small helicopter is a vibration intensive environment. A gasoline engine by itself introduces vibration into the system. More vibration is inevitable due to the fact that the main rotor and tail rotor blades cannot be perfectly balanced. An imbalance in these rotors leads to vibration. For the DSAAV, the fundamental vibrational frequency was 18 Hz. Significant vibration existed at harmonics of the fundamental as well. There are three main consequences of vibration on the DSAAV. These are fatigue, inertial noise, and GPS performance degradation.

5.3.1 Fatigue

Any time a piece of metal is bent even a small amount, the point at which the metal is bent becomes weaker. A paper clip bent through an angle of 90 degrees will break apart after about eight bends. When electronic components such as capacitors or linear regulators are subject to vibration, their leads can be bent a very small amount many thousands of times. Eventually, the leads break and the component no longer can perform its function. Most components play an essential role. Therefore, it only takes one failure to lose the functionality of part or all of the on-board electronics.

One contributing factor to the degree to which a component is susceptible to fatigue is the *moment arm* of the component. This is the distance from the point where the leads are soldered to the circuit board to the center of mass of the component. The longer the moment arm, the more quickly the leads will fail. Therefore, components should be kept as close as possible to the board. Similarly, components should be rigidly attached to the board at three non-collinear points. This can be accomplished by tie-wrapping the component down, or using epoxy to cement it to the board.

Another point where fatigue can crop up is at solder joints. A poorly soldered joint will not last long in a vibration intensive environment. All solder joints should be carefully inspected under a magnifying lens.

Part of the reason for performing an engine test is to check for components or joints that are likely to fail. Powering up the motor with the components on the helicopter is the most realistic way to check for potential problems. Another way to test for problems is with a *shaker table* in a lab. A shaker table can be made with a powerful speaker and an amplifier. It can be a useful piece of equipment, capable of vibration from about 20 Hz to a couple Kilohertz.

5.3.2 Inertial Noise

Vibration is the result of forces that accelerate the vehicle in a roughly periodic fashion. These accelerations are no different from accelerations due to translational movement of the vehicle. Hence, the IMU picks up vibrations just as it picks up other sources of acceleration. Because vibration represents real movement of the vehicle (movement back and forth), there is not an inherent problem between vibration and the inertial system. A truly linear inertial system would not struggle with vibration.

However, there are two sources of non-linearity that make vibration a problem. The first is the saturation of the IMU at high levels of acceleration. If the instrument is shaken too hard, its signals will become distorted and information about the lower frequency inertial trends of the vehicle will be lost. Second, since the signals are being sampled at a fairly low rate, high frequency information in the IMU signals only serve to alias the sampled signals.

When the effect of engine vibration was first observed on the DSAAV, three methods of solving the problem were pursued. This "three pronged attack" involved designing a mechanical isolation filter for the IMU, adding a second pole to the analog filters, and finding a way to double the sampling rate. All three of these solutions were implemented, and the resulting system worked well enough. However, there is potential for improvement in this area, as will be discussed more in chapter 7.

5.3.3 GPS Performance Degradation

Another impact of vibration on the vehicle's performance was observed in the GPS system. Whenever the engine was turned on, the performance of the GPS data became poor at best. It was at first thought that this was due to electrical interference from the spark plug, or the vibration of the antenna. However, after many tests, it was discovered that it was the vibration of the GPS card itself that caused the problem. Therefore, this problem was fixed by building a mechanical isolation filter for the GPS card.

Although the reason for this is not known for sure, it is suspected that the crystal oscillator used on the GPS card is affected by the vibration. The frequency of oscillation of crystal oscillators is known to be sensitive to acceleration[4].

[4] This principle is the basis for a class of accelerometers.

5.4 Electro-Magnetic Interference

There are many sources of strong electric and magnetic fields on the DSAAV. The most obvious of these is the modem which radiates 500 mW of power by design. Other sources include the computer and the switching power regulators. All sources of electro-magnetic interference (EMI) serve to raise the *noise floor* of the system. The noise floor is the average electro-magnetic signal strength that exists around the vehicle. It makes it more difficult for receivers to distinguish signals from noise.

The problem of EMI can be manageable if devices radiate over a narrow frequency band. On the DSAAV, however, it was found that the frequency range over which some devices radiate coincide with the frequency range over which some receivers receive signals. A symptom of this problem was observed when certain servos change their angle every time the radio modem transmitted data. It was found that the radio modem radiated significant energy outside its specified transmission frequency[5]. The type of radio modem used was changed to fix that problem. However, the range of the radios is still adversely affected by the high noise floor on the DSAAV.

The noise floor also affects the inertial system, as the analog signals from the IMU are carried to the A/D converter by wires that are susceptible to EMI. Hence, high frequency noise is added to these signals.

In order to reduce the noise floor on the vehicle, shielding was used. A shield is a grounded piece of metal that is impervious to electro-magnetic waves. There are two approaches to using shielding. Sensitive components can be protected with shielding, or highly radiative components can be muffled with a shield. On the DSAAV, the latter approach was taken in order to reduce the noise floor all around the vehicle. The computer and GPS card were shielded with copper tape that was placed around the mylar enclosures. The power board was shielded with bug screen[6]. Finally, wires that carried noisy signals were shielded by wrapping them in aluminum foil and insulating them with shrink wrap.

5.5 Conclusion

Debugging unexpected problems can be one of the most time-consuming parts of a project like the DSAAV. The solutions seem simple once they have been found and tested, but they seem less obvious when the problem is first encountered. By performing careful tests and consulting experienced engineers, the problems can be solved with a minimum of displeasure.

This chapter concludes the discussion of the 1996 DSAAV. Future chapters will describe on-going efforts to improve the system's performance.

[5] Testing was done by Valerie Lowe of Draper Labs. She used a spectrum analyzer with a "sniffer" to determine the signal strength of a transmitter in different frequency ranges.

[6] Bug screen is a metal mesh that is usually used on screen windows to allow air to flow while maintaining a barrier. It is light, it is an effective shield, and it allows for convection cooling of the power regulators.

Chapter 6

On Toward the 1997 DSAAV

The vehicle designed in 1996 was a significant engineering achievement. No other vehicle that is as capable as the DSAAV was developed with such a small budget or in as little time. The work in 1996 produced a functional autonomous aerial vehicle capable of take-off, way-point acquisition, vision processing, and landing.

However, the 1996 DSAAV was far from ideal. The performance and robustness of the vehicle left much room for improvement. Similarly, the architecture needed to be changed to allow for a more general mission. An analysis of the vehicle and ways in which it can be improved was in order. To evaluate which of the potential improvements were worthwhile, the revised objectives of the vehicle were considered.

6.1 New Objectives

In 1996, the International Aerial Robotics Competition was the motivating goal of the project. The competition required that the vehicle be able to take off, fly, and land autonomously. In addition, vision processing was an important capability for use in the competition. Therefore, the competition was useful for motivating the design of a vehicle with basic functionality. These goals were met in the summer of 1996. It was decided that for the following year of work, new, ambitious goals should be developed. Two such goals emerged. The first goal for 1997 was to demonstrate two AAV's that could interactively perform a mission. The second goal was to improve the performance and robustness of the vehicles. Performance in this context refers to the level of capability provided by the hardware. Clearly, much of the behavior of the system was determined by software, but the software is limited by the hardware's capability. Therefore, methods of improving the accuracy and bandwidth of sensor measurements are considered performance improvements.

With these two goals in mind, improvements were conceived and evaluated. Each of these potential improvements is worthwhile and should one day be implemented. Considering the limited amount of time available, some were not dealt with in 1997. The ones selected for implementation in 1997 made significant contributions toward the 1997 goals.

6.2 Potential Improvements

6.2.1 Physical Layout

In 1996, the electronic components were distributed all around the helicopter. Most components were attached to brackets that were in turn connected to the frame of the vehicle. This layout made

it easy to gain access to most of the components, should work need to be done on them, but it had its shortcoming. One shortcoming was that the components had to be tested on the vehicle. It was difficult to remove all of them and test them on a bench. Instead, work always had to be done around the vehicle itself. Another shortcoming was the need to run power and signal cabling all around the vehicle. Such cabling looked sloppy, was a reliability hazard, and added weight.

In 1997, the majority of the components are being mounted together on one platform beneath the vehicle. This platform will make it possible to put the components close together, while still providing shielding where necessary. The platform will be removable to allow work on the electronic systems or to transfer them to another platform. Short cables will provide the necessary interconnection.

6.2.2 Weight Margin

The issue of weight was a major design constraint in 1996. It was the driving factor in many of the decisions that were made for the 1996 DSAAV. A larger weight margin would have many positive results. For one, the vehicle would be able to command larger control forces, thereby improving its flight performance. Weight margin can also be used to add superior shielding and more robust mounting hardware. Since the DSAAV may be used as a test platform for new hardware such as vision processing equipment and inertial sensors, it may also be important for the vehicle to be able to carry a payload.

The issue of weight margin is being dealt with by using a helicopter that is capable of lifting more payload. The helicopter that will be used in 1997 is a twin industrial helicopter made by Bergen Machine and Tool Company. It uses a two-cylinder, two-cycle, 32 cubic centimeter engine. It is rated for a payload of twenty pounds.

6.2.3 Processing Power

For a real-time application like the DSAAV, more processing power is always desirable. A faster processor would make it possible to sample the sensors at a higher rate or to run a more sophisticated control system, or both. As pentium processors become available on boards that follow the PC-104 conventions, it may be worthwhile to switch to a faster processor. The only disadvantages to switching are the time and money it takes to integrate any new piece of hardware, and the additional power consumed by a faster processor.

6.2.4 Communications Range

The 1996 DSAAV barely had the remote control and communications link range it needed to be able to safely fly its mission in the 60′ by 120′ contest field. For more ambitious missions, much more range is desirable. Eventually, the hope is the develop a vehicle that can pilot itself many miles. One step toward accomplishing the long term goals for the vehicle is to increase the communications range between the ground-station and the vehicle. Approaches to this improvement include providing better shielding on the vehicle, and using equipment that is capable of sending and receiving signals over a greater distance.

Communications range is being improved in two ways. First, a new modem is being used in 1997. This modem uses a different kind of data encoding that is more immune to noise than the Proxlink modems. They are rated for transmission over 10 miles at 1 watt of output power. In order to decrease the amount of EMI on the vehicle, the modems will be operated at 500mW.

The second way in which the range is being improved is by having better shielding on the vehicle. Metal boxes will separate sensitive components from noisy components. These boxes will be well grounded and should provide ample shielding.

6.2.5 Robustness

There is the constant fear of a devastating crash associated with the DSAAV, although no such crash has yet been encountered. The excellent survival record for the vehicle is due to careful treatment, conservative procedures, redundant safety mechanisms, and solid construction techniques. It is sobering to realize that any small problem with an electronic component could destroy weeks worth of effort. For this reason, it is highly desirable to build very robust hardware.

Robustness is being achieved by PC-boarding the custom electronics, and simplifying the layout. A simpler layout means that less interconnection will be required. Hopefully, less interconnection will imply a more robust system.

6.2.6 Inertial System

The heart of the avionics is the inertial system. The other sensors mainly serve to correct errors that accumulate in the inertial system's navigation estimates. Therefore, improvements in the inertial system would change the fundamental performance of the system. Potential improvements were increasing accuracy of the system, increasing the system bandwidth, or both.

A complete re-design of the inertial system has been undertaken. The analysis and design of this system is the topic of the remaining chapters of this document.

6.3 Conclusion

The topology that has evolved with these improvements in mind is shown in Figure 6-1. It is very similar to the topology used in 1996, but it has some subtle differences; some of the power requirements are different, and the interfacing of the IMU, compass, and sonar have changed.

Even though the first year of the DSAAV was a success, a great deal of work remains to be done. In addition to the hardware improvements discussed here, a significant re-design of the vehicle's software is required. It is not enough to enjoy the success of the first year of the project's development. Significant improvements must be implemented in the second year in order to maintain the lead that Draper Lab achieved after its first year of work.

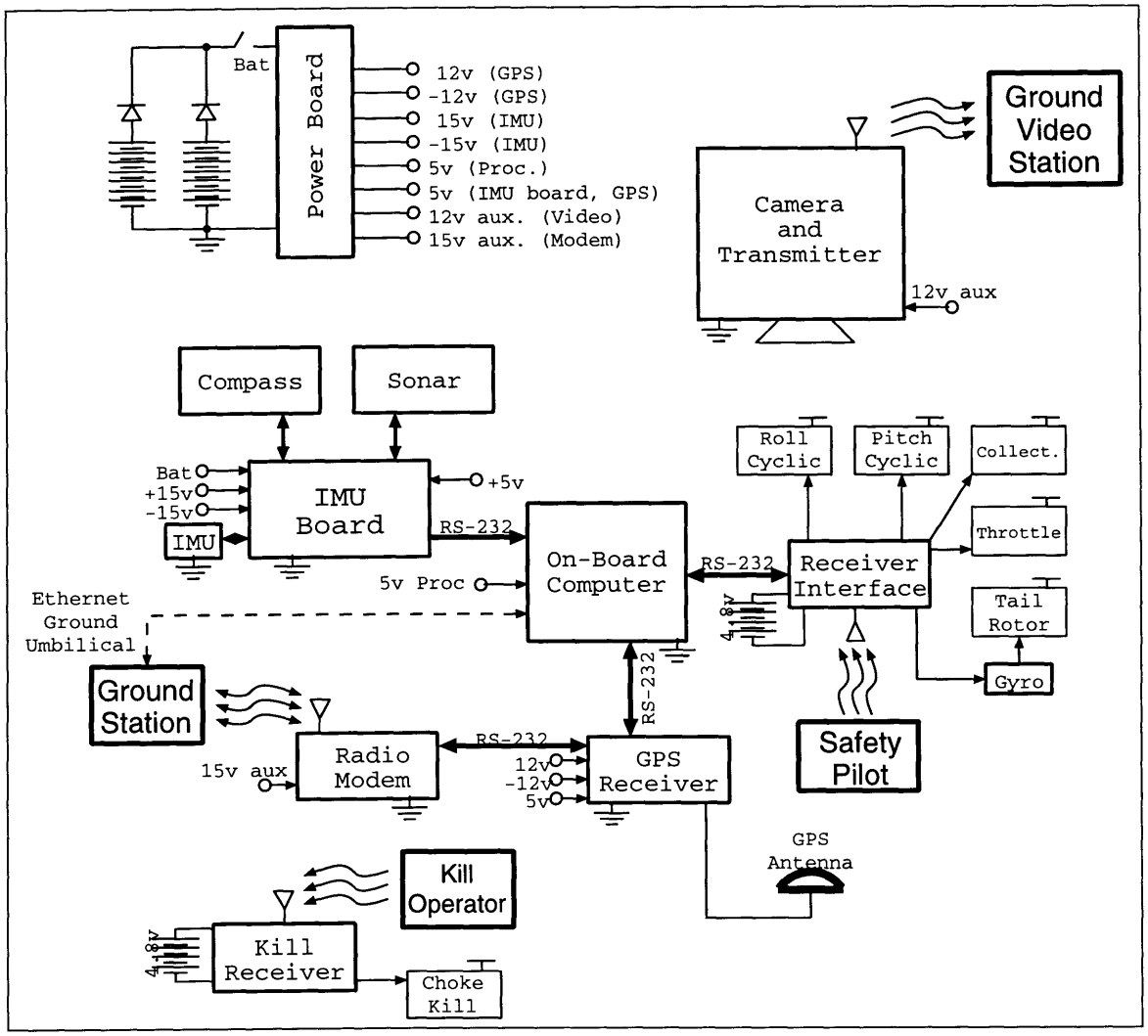

Figure 6-1: 1997 Block Diagram

Chapter 7

Inertial System Analysis

7.1 Introduction

After the first year's experience with the system, the inertial system was targeted as having potential for significant improvement. There were two main ways in which its performance could be improved. First, the bandwidth of the system could be increased. Second, the accuracy of the system could be increased. In order to determine what type of approach should be used toward accomplishing these goals, an analysis of the system was undertaken.

7.2 Motivation for the Project

7.2.1 Drift and Bandwidth

The inertial system used in the fiscal year 1996 DSAAV was adequate for the performance level required. It provided the information needed to fly the helicopter in spite of the low update rate from the GPS receiver. Because the helicopter relied heavily on GPS to correct position estimate errors, the drift performance of the inertial system did not need to be particularly good. Similarly, since the helicopter was flown at low speeds and under fairly benevolent environmental conditions, the bandwidth of the system did not need to be very high.

Although the first generation inertial system was adequate, there were advantages to making improvements. If a lower drift system could be developed, the helicopter's susceptibility to bad or intermittent GPS data could be reduced. Similarly, a higher bandwidth inertial system would make it more feasible to increase the bandwidth of the control system and therefore the maneuverability of the vehicle.

7.2.2 Alternatives to the Present System

A preliminary investigation into other options for the inertial system exposed two approaches with potential to improve the system's performance. These approaches are high speed sampling and voltage to frequency (V-F) conversion.

High Speed Sampling

The first possible approach involves increasing the rate at which the inertial instrument is sampled and improving the analog noise reduction filters used. This approach might involve the design of an independent sampling board that would talk to the main processor through a serial port. Depending

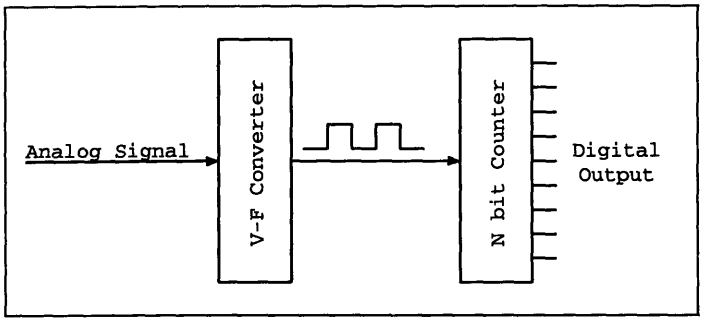

Figure 7-1: V-F Converter Topology

on the cost analysis, it might also merely involve small changes to the present system that would make it possible to sample at a higher rate[1].

Voltage to Frequency Converters

Another possible approach involves the use of V-F converters. These devices take an analog voltage as input and output a pulse waveform with a frequency proportional to the input. If this output is counted, it can be used to determine the average value of the analog input signal. Figure 7-1 shows how the V-F converter works in block diagram form.

7.3 Inertial System Simulation

In order to understand the behavior of the inertial system. A simple simulation of the system was developed using Matlab. The simulation models the mechanical and electrical properties of the system, including noise, finite sampling rate, and the non-linearity of the sensor. This simulation was useful in understanding the sources of error in the system.

7.3.1 Problem Definition

The mechanical portion of the inertial system is shown in Figure 7-2. Acceleration from the helicopter is coupled to the 2 pound sensor through a spring and damper. The spring has a spring constant, k, and the damper has a damping factor of b. Neither k nor b is known in advance, but it is known that the damped natural frequency of the system is 5 Hz and the oscillations die away in approximately 1 second.

The accelerometer is modeled as having a constant ratio of acceleration in to voltage out. This ratio is maintained until the instrument saturates at ± 5 volts. Saturation is modeled as a region of zero slope. This behavior is shown in Figure 7-3. The signal coming from the sensor is filtered by a first order RC filter before it is sampled.

[1] Such changes might involve switching to a faster main processor, or using a different operating system for which DMA transfers can be implemented between the A/D board and the main processor memory.

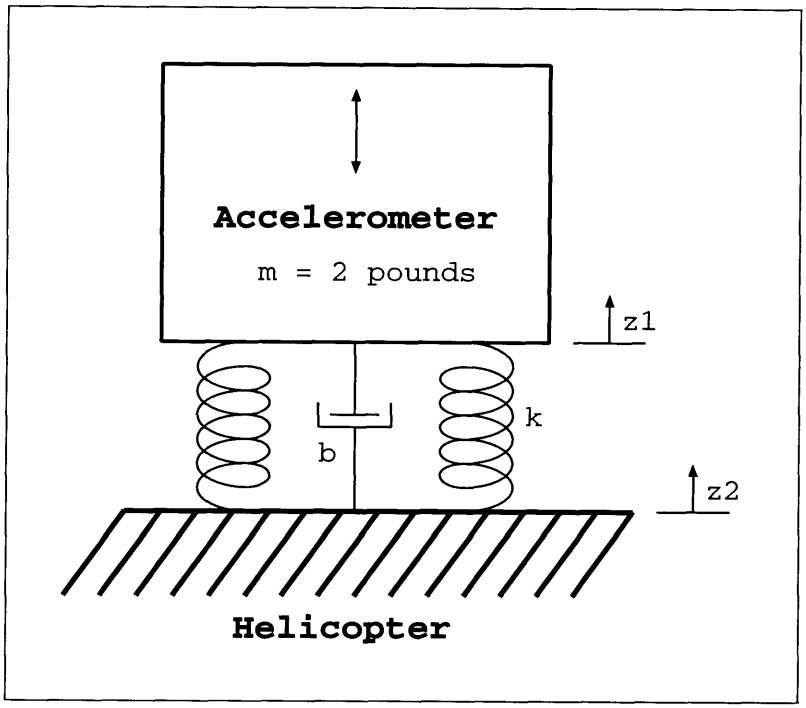

Figure 7-2: Mechanical System

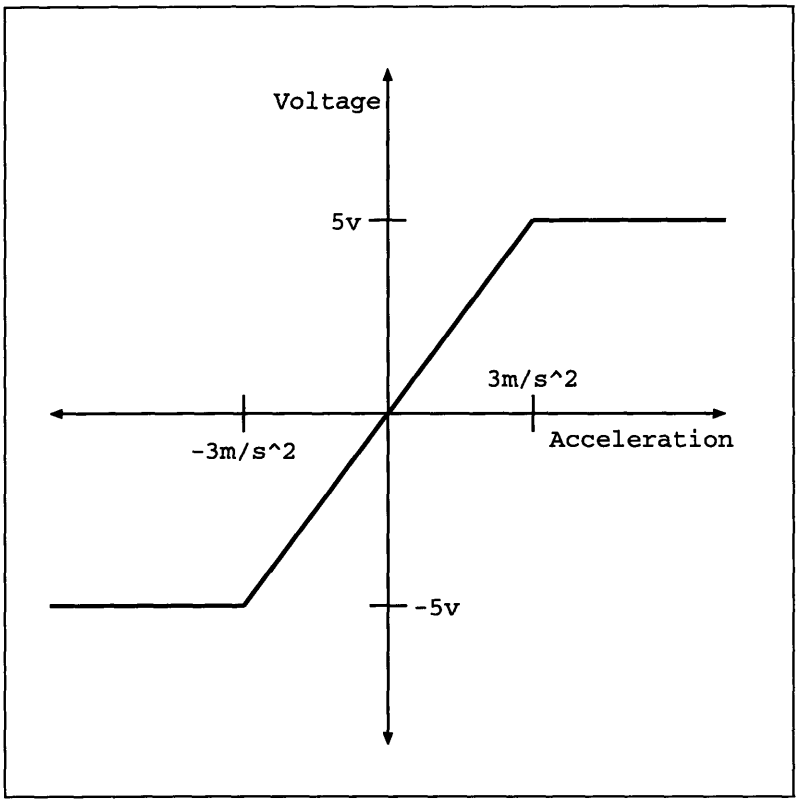

Figure 7-3: Large Signal Accelerometer Transfer Function

7.3.2 Model Development

The overall block diagram for the system is shown in Figure 7-4. Each part of the diagram is described individually below. See Appendix B for code listings.

Mechanical System Modeling

The first part of the model to develop was the mechanical system. This is a simple Mass-Spring-Damper system. Its equation of motion was easily found to be

$$m\ddot{z}_1 + b\dot{z}_1 + kz_1 = b\dot{z}_2 + kz_2. \tag{7.1}$$

Transforming to the frequency domain, the transfer function is given by

$$\frac{z_1(s)}{z_2(s)} = \frac{bs + k}{ms^2 + bs + k}. \tag{7.2}$$

In order to simplify the mechanical system analysis, it is modeled as an electrical system. The analogy can easily be found by inspection. First, notice that at DC, the transfer function has a gain of 1. Next, note that it is a second order system. Finally, observe that the spring and damper are connected in parallel, and so the resistor and inductor in the electrical circuit will also be connected in parallel. Figure 7-5 shows the electrical circuit that models the mechanical system. A further look at the circuit shows that in this analogy, current is analogous to force and voltage is analogous to velocity.

If this circuit is analyzed from scratch, the transfer function for it is found:

$$\frac{V_o(s)}{V_i(s)} = \frac{\frac{1}{R}s + \frac{1}{L}}{Cs^2 + \frac{1}{R}s + \frac{1}{L}} \tag{7.3}$$

If this expression is compared to the transfer function for the mechanical system, it is observed that the two are identical as long as $R = 1/b$ and $L = 1/k$. These relationships agree with intuition as well. Therefore, in order to simulate the behavior of the mechanical system, the circuit in Figure 7-5 was simulated using a simple forward Euler algorithm as is shown in the Appendix[2]. In addition to modeling the purely mechanical aspects of the system, the simulation also included the effects of the 75 Hz pole associated with the sensor. It also models the saturation voltages by constraining the output to be within the range from +5 volts to -5 volts.

The value of m was known from the outset, but values for k and b had to be determined. These values were determined by trying different values for k and b and changing them until the step response of the system had the desired properties (5 Hz oscillating frequency and 1 second of ringing to within 5% final value). The values were found to be k=1000, b=10.

Filtering Model

Two general purpose filters were used with the simulation for many different purposes. These were a first order low pass filter and a first order high pass filter. The schematics for these two circuits are shown in Figures 7-6 and 7-7.

These filters were also simulated with Forward Euler algorithms and are shown in the Appendix as lowpass.m and highpass.m. They were very handy for frequency shaping of many of the signals in the simulation. Also, when a high order filter was needed, these filters were run multiple times.

[2]File mech.m

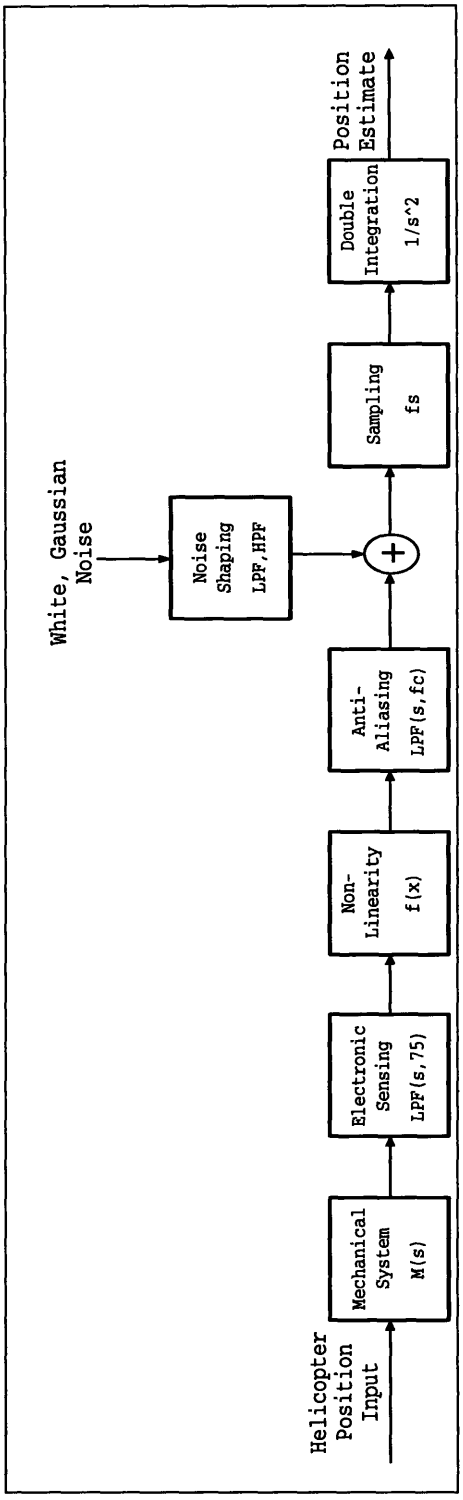

Figure 7-4: Accelerometer System Block Diagram

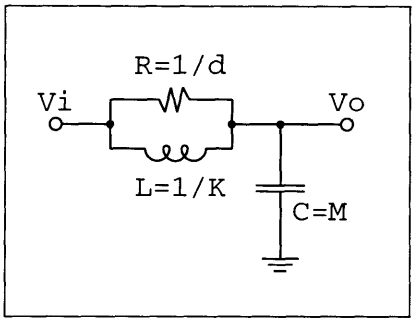

Figure 7-5: Electrical Model of Mechanical System

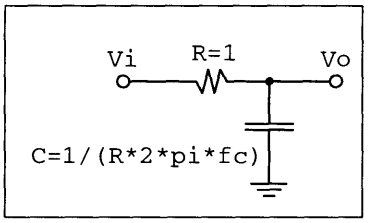

Figure 7-6: Low Pass Filter Schematic

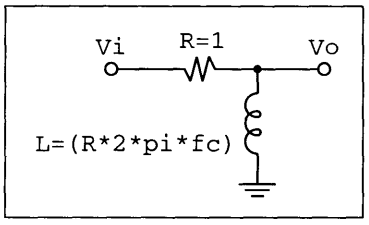

Figure 7-7: High Pass Filter Schematic

Noise

Noise was added to the signal by creating a vector of gaussian, zero mean, white noise. This vector could be added to the signal directly, or it could be filtered first in order to study the effects of the frequency content of the noise on the system. The noise being modeled by this vector included both the intrinsic noise of the sensor, and noise that was added to the signal by vibration or EMI.

Sampling

The process of sampling was simulated with the function, resample.m, which simply picks out elements of a vector at a constant rate. The other elements of the vector are simply discarded.

Integration

Finally, the integration was performed with the Matlab command, cumsum. This function takes the cumulative sum of the elements in a vector. Hence, if the cumsum is multiplied by the sampling period, a discrete-time integration is performed.

7.3.3 Results

Several different system properties were studied. First, the frequency response for the system was found. Then, the dependence of the position estimate on noise, non-linearities, and sampling rate was studied. The following sections describe the results of the analysis.

Many of the analyses performed required simulating an input to the system. Finding a good input was important, since it was not desirable to supply an input that was non-realistic. In particular, it was desirable to choose a continuously differentiable input. Since sine waves are always good inputs, a sine wave was chosen as the starting point input. At time=0, $\sin(2\pi t)u(t)$ has zero position and zero acceleration, but it has a step in velocity. In order to cancel out the step in velocity, another term was added to the input: $I(t) = (\sin(2\pi t) - 2\pi t)u(t)$. This input almost solves the problem. The only remaining problem is that the input tends toward $-\infty$ as t goes to ∞. Therefore, a factor was added to make the second term decay as time went on. The final input used was

$$I(t) = (\sin(2\pi t) - 2\pi t 10^{-t})u(t) \tag{7.4}$$

System Frequency Response

The frequency response for the system was found by assuming that the system was being operated in its linear mode. Then, the transfer functions for all of the sub-systems were multiplied to get an aggregate system transfer function. The result of this is shown in the bode plot of Figure 7-8. As the plot shows, The system's response is dominated by the mechanical modes around 5 Hz = 32 rps.

Impact of Noise

The first non-ideality that was studied was noise. Noise of different amplitudes and frequency contents was added to the signal, and the effect of the noise on the integration was observed. For the purpose of noise study, an infinite sampling rate was assumed, and non-linearities were turned off.

The first type of noise analyzed was white noise. It was expected that white noise would cause the output to undergo a random walk as time progressed. Figure 7-9 shows that this is in fact what resulted from the white noise. The bold line in the top plot shows the input to the inertial system. The other line in the top plot shows the position estimate of the system. The bottom plot shows the acceleration estimate that was integrated.

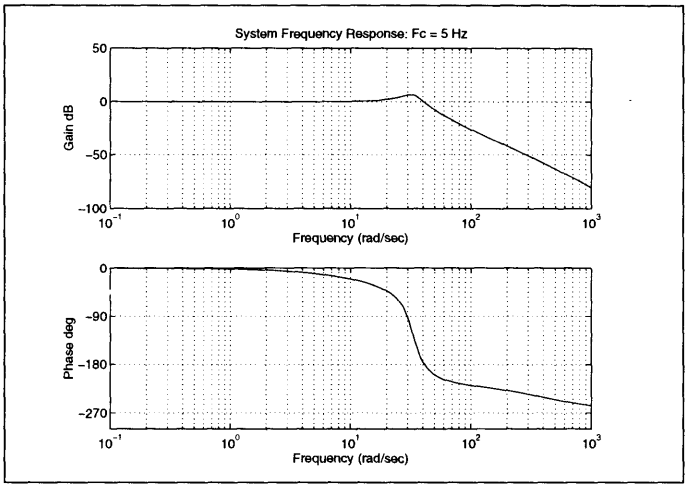

Figure 7-8: Overall System Frequency Response

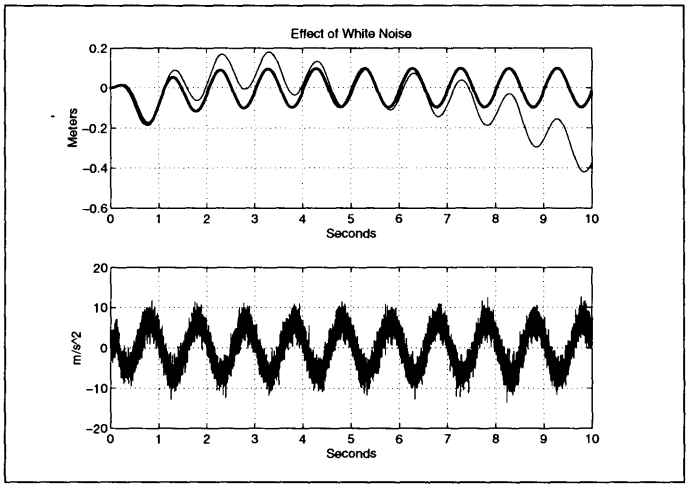

Figure 7-9: Effect of White Noise on the System

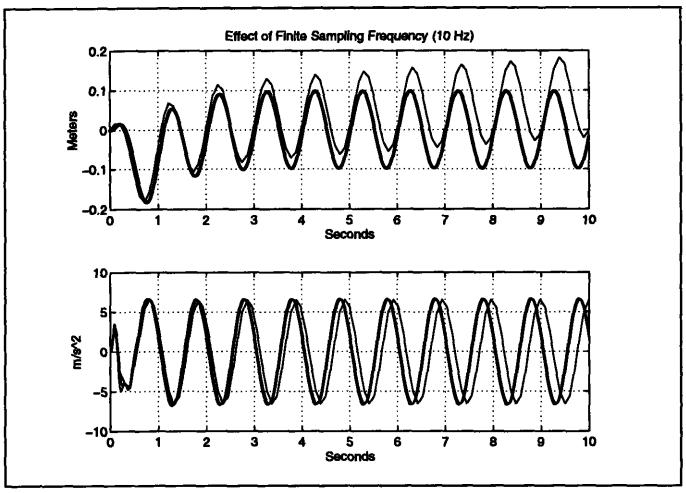

Figure 7-10: Impact of Finite Sampling Rate

Noise was also studied in the case of low frequency noise and high frequency noise. In this case of infinite sampling frequency, low frequency noise tended to be more harmful to the integration, since the integrators have more time to accumulate faulty readings from low frequency errors.

Impact of Sampling Rate

System theory tells us that the higher the sampling rate, the more capable the system will be of re-constructing the input signal. This effect was studied in the case of the inertial system. Several different sampling frequencies were used. Once the sampling rate got much below 25 Hz, the system was noticeably affected. As one case of this, Figure 7-10 shows what happens with a sampling rate of 10 Hertz. It was observed that the position estimate was slowly drifting away from the true position.

Impact of Non-linearities

The third non-ideality that was studied was the non-linearity of the inertial sensor. The system was driven hard enough to drive the sensor into saturation, and the impact on the position estimate was observed. Figure 7-11 shows the result of this simulation. The position estimate has a much lower amplitude than the true position, and the mean of the estimate drifted away from the true mean.

Impact of Noise and Finite Sampling Frequency

One of the most interesting cases to study was the case where there was both noise and a finite sampling frequency. In real applications, both noise and sampling frequency are major factors in the design of the system. Therefore, understanding the interaction of the two is important. Intuitively, it seems that high frequency noise ought to be worse than low frequency noise. Low frequency noise (if it is zero mean) can be averaged out by the system. High frequency noise, on the other hand, will result in an erratic "random walk" of the estimate. Two plots were generated to confirm these suspicions. First, Figure 7-12 shows the effect of low frequency noise on the system. Figure 7-13 shows the effect of high frequency noise on the system.

For both of these cases, the sampling frequency was set at 50Hz. To obtain the low frequency noise source, white noise was filtered with a 10Hz low pass filter. For the high frequency case, white

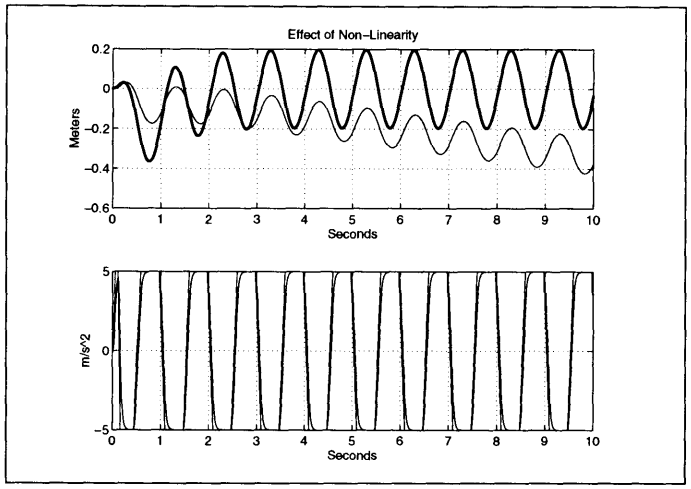

Figure 7-11: Impact of Non-Linearities

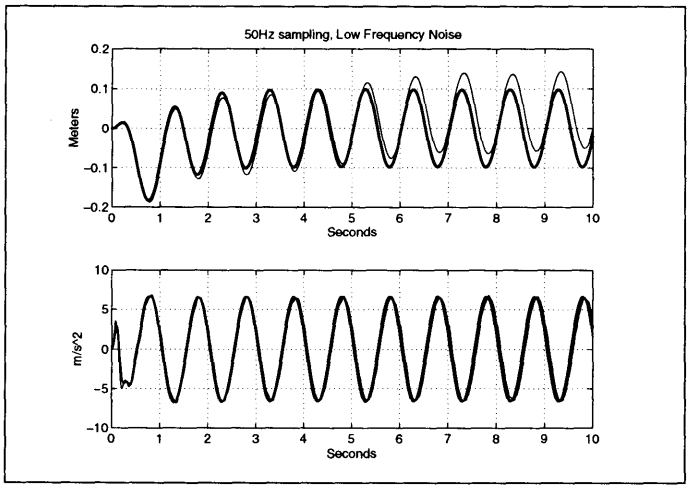

Figure 7-12: Impact of Low Frequency Noise with Finite Sampling Frequency

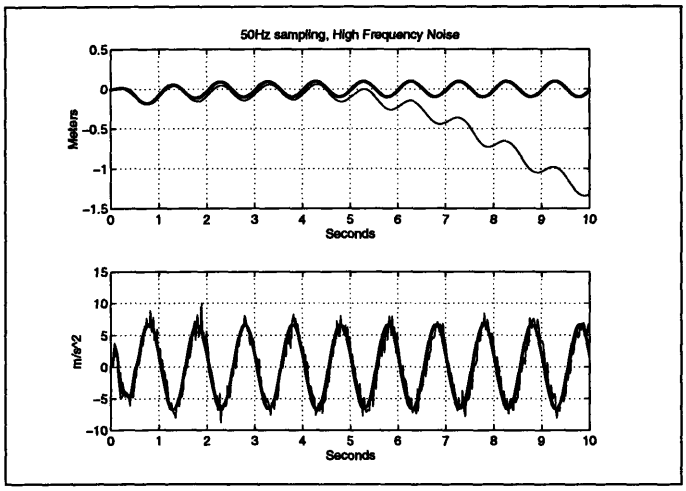

Figure 7-13: Impact of High Frequency Noise with Finite Sampling Frequency

noise was filtered with a 990 Hz high pass filter.

7.3.4 Conclusion of Simulation

The results obtained here were useful for clarifying some of the issues related to the DSAAV inertial system. The effects of noise, non-linearity, and sampling rate were studied, and the ways in which these factors affect the system performance were observed clearly in graphical form. In particular, it was observed that it is important to maintain an adequate sampling rate and to keep the level of noise down as far as possible. The new inertial system is designed in order to minimize the impact of these effects.

7.4 Analysis of Drift Performance

In order to decide whether the V-F converters or Impulse sampling system should be used, an expectation of the level of performance that can be achieved was developed. The first parameter of interest was the drift of the inertial system.

7.4.1 Measure of Performance

Some quantitative method for evaluating the performance of the inertial system is needed. Intuitively, this measure ought to be related to the expected magnitude of drift over some interval of time.

To make this idea quantitative, consider the continuous variable $\tilde{P}(t)$, which represents a measurement of the inertial system's output. This quantity is decomposed as

$$\tilde{P}(t) = P(t) + P_{err}(t) \tag{7.5}$$

where P(t) represents that part of the signal that carries useful inertial information. $P_{err}(t)$ rep-

resents noise that tends to corrupt the angle[3] estimate. The true angle of the vehicle is then given by

$$I(t) = \int_0^t P(\tau)d\tau \tag{7.6}$$

Since I(t) must be compared to the discrete-time estimate of the angle, it is useful to define

$$I[k] = I(kT) \tag{7.7}$$

where T is the sampling interval in seconds. Hence, I[k] gives the true angle for each k.

If $\tilde{P}(t)$ is sampled and integrated with a first order forward-rectangular integration algorithm, the value of the angle estimate will be given by

$$\hat{I}[k] = T\sum_{n=0}^{k} \tilde{P}[n], \tag{7.8}$$

$$\tilde{P}[n] = \tilde{P}(nT). \tag{7.9}$$

The difference between the two quantities,

$$I_\Delta[k] = I[k] - \hat{I}[k], \tag{7.10}$$

is the drift of the system at any moment, k.

Because the drift cannot be found deterministically, it must be analyzed probabilistically. An application of basic probability theory is in order [1]. In particular, a good quantity to use as the measure of performance is the standard deviation of the drift:

$$\textbf{Drift Performance Measure} = \sigma_{\mathbf{I_\Delta[k]}}. \tag{7.11}$$

In order to find a value for $\sigma_{I_\Delta[k]}$, the sources of error in the system must be considered.

7.4.2 Sources of Error

There are many possible sources of error in the system. Some of the more obvious ones are

- Built in sensor noise (P_S)
- Mechanical vibrational noise (P_V)
- EMI induced noise (P_{EMI})
- Sampling quantization (P_Q)
- Integration errors (I_E)
- Instrument non-linearity[4]
- Instrument bias drift
- Mis-calibration (Scale Factor Error, Bias Estimate Error)

[3] or velocity

[4] Including saturation under mechanical vibration

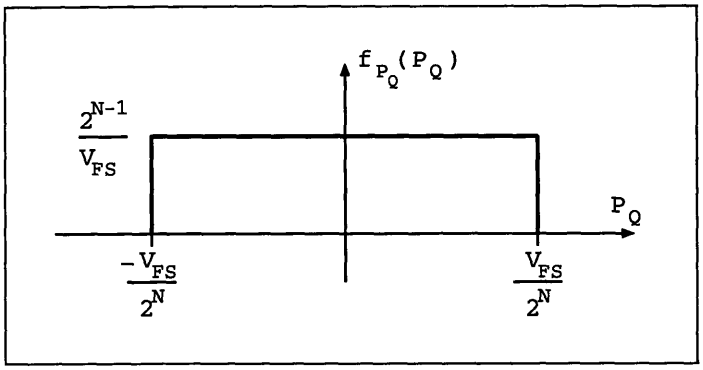

Figure 7-14: Quantization Noise PDF

The first five (P_S, P_V, P_{EMI}, P_Q, and I_E) are errors that can be reduced by improving the sampling method. Instrument non-linearity, if significant, might be corrected for at the software level. The same is true of bias drift. For the purposes of this discussion, it will be assumed that the system has been properly calibrated.

7.4.3 Noise Models

Each of the relevant sources of error needs to be analyzed probabilistically. The contribution of each source to the overall measure of performance, $\sigma_{I_\Delta[k]}$, must then be determined.

P_S, P_V, P_{EMI}

The first three sources of noise are assumed to have normal distributions and zero mean. Each of these three sources is continuous in nature and one parameter, σ_C, is sufficient to represent the error introduced into the system by these sources at each time step. The overall drift error incurred is given by

$$T\sum_{n=0}^{k}(P_S[n] + P_V[n] + P_{EMI}[n]) \tag{7.12}$$

Hence, the error is characterized by $\sqrt{KT}\ \sigma_C$. Note that the product, KT is a period of time. The implication of this is that amount of error introduced by these sources is independent of the sampling period and is proportional to the square root of the amount of time passed.

P_Q

Quantization noise is a due to the finite precision of the analog to digital conversion process. The analog signal is rounded to the nearest discrete value that can be represented. Because there are so many possible values, the probability density function for quantization error is uniformly distributed over a quanta. Figure 7-14 shows the probability density function for the quantization noise. V_{FS} is the full scale voltage output (i.e. $P_{max}(t)$). N is the number of bits reported by the analog to digital converter.

This function has an expectation of zero and a standard deviation given by

$$\sigma_{P_Q} = \frac{L \times V_{FS}}{\sqrt{3}\ 2^N} \tag{7.13}$$

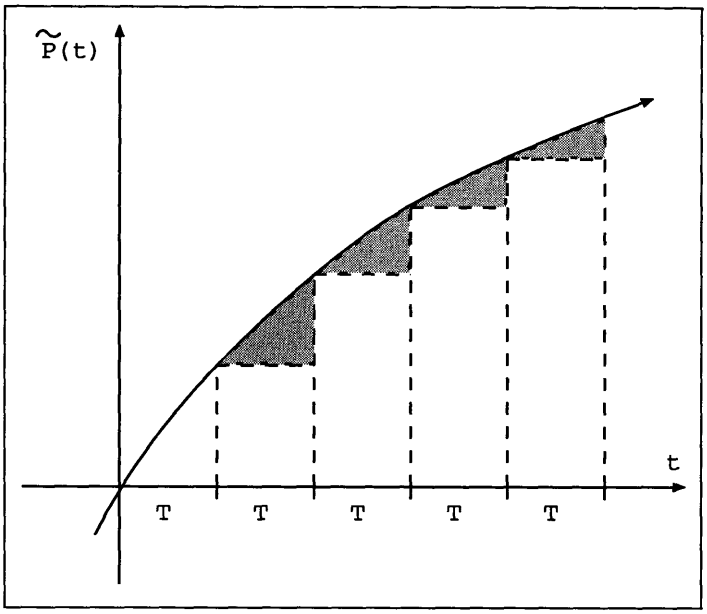

Figure 7-15: Integration Errors

Here, L, is a constant that relates the voltage of the inertial sensor to the angular velocity or acceleration of the unit. Since the quantization noise will add some error to the integration at each time step, the contribution to the overall drift will be characterized by $\sqrt{KT}\,\sigma_{P_Q}$. Again, the error is seen to be independent of the sampling frequency.

Integration Errors

Figure 7-15 shows the main idea behind integration errors. They are due to the fact that a continuous function ($\tilde{P}(t)$) is being represented in discrete time. The shaded regions represent area under the curve of $\tilde{P}(t)$ which is not included in the discrete time estimate of $\int \tilde{P}(t)$, assuming a forward Euler integration algorithm. If it can be assumed that the slope of $\tilde{P}(t)$ is a random variable with a normal distribution, then the standard deviation of the integration error can be computed. Let the standard deviation of the slope of $\tilde{P}(t)$ be σ_δ. Then the contribution to the overall error from this source will be characterized by

$$\frac{1}{2}T^2\sqrt{K}\,\sigma_\delta \tag{7.14}$$

This analysis is not completely accurate since consecutive measurements of the slope of $P(t)$ cannot be considered statistically independent. What this means is that positive integration errors will tend to be canceled by negative integration errors as the function evolves over a period. However, there will be some value of σ_δ such that the above expression accurately characterizes the amount of error contributed from this source.

If the expression is re-written as

$$\frac{1}{2}T^{3/2}\sqrt{KT}\,\sigma_\delta, \tag{7.15}$$

the amount of error is seen to be dependent on the sampling frequency (or the sampling period, T).

Full Model

When all of the error sources are added, the drift performance measure can be computed. It is given by

$$\sigma^2_{\tilde{I}[k]} = KT(\sigma^2_C + \sigma^2_{P_Q} + \frac{1}{4}T^3\sigma^2_\delta) \tag{7.16}$$

Since σ_{P_Q} can be readily computed from V_{FS} and N, only two parameters, σ_C and σ_δ, need be found experimentally.

7.4.4 Analysis of Current Inertial System in Terms of the Models

At this point, it becomes possible to analyze the existing inertial system in light of the model. The behavior that has been observed on the DSAAV-2 should be consistent with the model.

Effect of EMI

On the existing system, it was observed that an increase in the level of EMI decreased the performance of the inertial system. In the model, this would correspond to an increase in σ_C and hence a decrease in performance.

Effect of Mechanical Vibration

It was found that the mechanical vibration of the helicopter had a big impact on the drift performance of the inertial system. In the model, this would correspond to an increase in both σ_C and σ_δ.

Effect of Sampling Rate

The model clearly shows that an increase in the sampling rate will tend to decrease the drift of the system by reducing the impact of integration errors. Sampling at a higher rate also makes it possible to have more effective anti-aliasing filters and to reduce the standard deviation of the other sources of error. This, also, was observed experimentally on the helicopter.

7.4.5 Effect of Analog Filtering

The existing inertial system uses anti-aliasing filters between the inertial sensor and the sampling board. If the model developed is accurate, the usefulness of these filters lies in their noise reduction function more than in any anti-aliasing function they may perform.

Until now, this effect has been ignored in the model. In order to account for their impact, a simple model for the filters is used. This is because a more accurate model involving power spectral densities of the noise would be difficult to use, and it is unlikely that a more complete model would add a great deal to this analysis. Therefore, the filters are modeled as having two bands—the pass band and the rejection band. As Figure 7-16 shows, the pass band has a magnitude of 1 and extends from DC to f_c, the cutoff frequency. The rejection band includes the remainder of the spectrum and has a magnitude[5] of $M_r \ll 1$.

If the value of M_r is chosen well, this model can roughly account for the fact that some EMI is introduced to the system *after* the filter. In fact, M_r may not be significantly less than one if there is much noise added after the filtering stage. When applied to the model, it is seen that the effect of this filter is to reduce the value of σ_C by a factor of $\sqrt{M_r}$ as long as all of the noise is above the

[5]The phase of the filter is will not play a significant role in the drift of the system, but will become an issue when the bandwidth of the system is discussed.

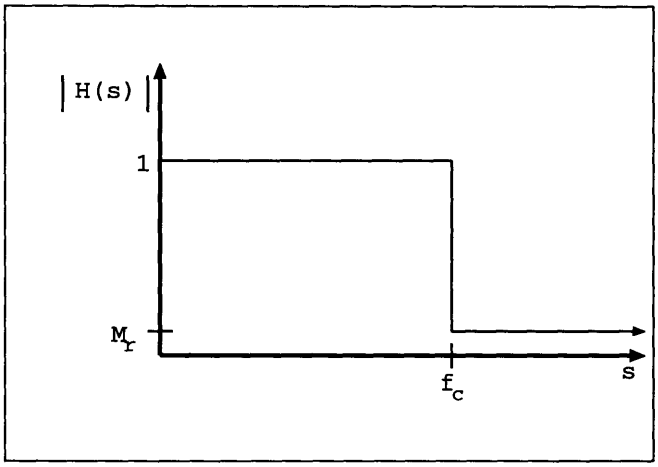

Figure 7-16: Filter Model

cutoff frequency. It is important to note that this filter model is *not* a good model for the existing analog filters. The transition band of the current filters includes a part of the spectrum where there is significant mechanical vibrational noise.

Although this discussion has been mostly qualitative, and does not directly apply to the existing system, it is useful for the purpose of improving the system's performance. In particular, it highlights the importance of minimizing the amount of EMI introduced into the signal path after the filters. It also gives a rough level of expectation for the degree to which improved filtering can improve the overall system.

7.4.6 Application of the Models to Sampler and V-F Converter

Impulse Sampler

The model developed was based on the assumption of an impulse sampler. Therefore, the model applies directly to this scenario. All that really needs to be investigated for this case is the effect of an increased sampling rate on the system's performance.

An increased sampling rate will affect the system performance in two different ways, as shown in the model. The model explicitly shows that the term associated with integration error will be decreased if the sampling rate is increased (T is decreased). Implicitly, the increased sampling rate will affect the drift error introduced in the $KT\sigma_c^2$ term. This is because the portion of the continuous time noise associated with mechanical vibration will be reduced as the sampling rate is increased. That is, the system will be able to meaningfully integrate the mechanical vibration, and it will not appear as random additive noise. The noise associated with the sensor itself and with EMI is assumed to be high frequency[6], so these additive noise sources cannot be reduced with a reasonable sampling rate. Therefore, for a very high sampling rate (roughly an order of magnitude above the mechanical vibrational frequency range), an upper bound on the performance that can be expected is given by

$$\sigma^2_{\tilde{I}[K]} = KT(\sigma_c^2 + \sigma^2_{P_Q}) \tag{7.17}$$

[6] Temperature changes, however, introduce low frequency noise into the inertial sensor which should be compensated for in software

Where σ_C is associated only with IMU internal noise and EMI. If the system is to be improved, the standard deviations of the noise sources must be reduced.

V-F Converter

In order to apply the model to a V-F converter system, a bit more modeling must be done since the V-F converter differs from the impulse sampler for which the model was developed. The V-F converter can be modeled as an integrator cascaded with an impulse sampler. This model is at first confusing, since the impulse sampling system might be implemented as a single pole filter followed by an impulse sampler. Why might the V-F converter system be better if it uses a lower bandwidth filter (bandwidth of 0!) where the impulse sampling system uses a pole that is not at the origin? The answer is that the V-F conversion system is measuring a different quantity (dimensionally) than the impulse sampling system. The impulse sampling system must read a (time-averaged) signal and integrate it, thereby introducing phase in addition to the phase of its analog filter. The V-F converter system, however, does the integration and filtering in one step, reading in the integrated signal. A total of 90 degrees of phase are introduced by the V-F converter system.

As far as performance goes, the V-F converter system looks comparable to a high speed sampling system. Its performance will be at least as good and may be better for two main reasons. The first reason is that the V-F converter system will avoid almost all integration errors, regardless of the sampling system, whereas the impulse sampling system must have a very high sampling rate to avoid these. The second reason is that in the V-F converter system, all of the analog processing (including filtering) is done on a single chip. Therefore, very little continuous time noise will be added to the signal between the filtering and sampling stages[7]. This level of noise rejection may be difficult to achieve in a practical impulse sampling system.

7.4.7 Model Verification and Parameter Evaluation

Now that a model has been developed and analyzed, values for the parameters need to be found, and the model should be verified experimentally. These two goals can be achieved at the same time by finding parameter values under various environmental conditions and comparing the results to the behavior predicted by the model.

In order to determine the values for the two parameters, σ_c and σ_δ, the recursive least squares method will be used [2]. The performance measurement equation is written as

$$\sigma^2_{I_d[K]} = E(I_d^2[k]) = KT(\sigma_c^2 + \sigma^2_{P_Q}) + \frac{1}{4}KT^4\sigma_\delta^2. \tag{7.18}$$

or

$$\sigma^2_{I_d[K]} = y_K^2 \approx A_K \cdot \hat{x}_K \tag{7.19}$$

for

$$\mathbf{A}_K = [\ KT \quad \tfrac{1}{4}KT^4\ ], \qquad \hat{\mathbf{x}}_K = \begin{bmatrix} \hat{\sigma}_c^2 + \hat{\sigma}^2_{P_Q} \\ \hat{\sigma}_\delta^2 \end{bmatrix} \tag{7.20}$$

($\mathbf{A}$ is a 1x2 row vector and $\hat{\mathbf{x}}$ is a 2x1 column vector which is the estimate of the parameters at time KT.) Here, y_K is a measurement of the instrument drift at time KT. If the system is initialized at zero, and it is not moved during the experiment, the value of the drift (y) is simply the system's estimation of the current state.

[7] As was later discovered, the amount of noise added after the analog filtering is very significant for the impulse sampling system. This noise-rejection characteristic of a V-F converter design makes it a very good choice for the design. Unfortunately, this impact was discovered too late.

Now, applying the results for recursive least squares estimation,

$$Q_{K+1} = Q_K + A'_{K+1}A_{K+1}, \quad Q_{-1} = 0, \tag{7.21}$$

$$\hat{x}_{K+1} = \hat{x}_K + Q^{-1}_{K+1}A'_{K+1}(y^2_{K+1} - A_{K+1}\hat{x}_K). \tag{7.22}$$

Therefore, by taking many data points, an estimate for the performance parameters can be obtained. If this is done multiple times to verify consistency, the results will be believable.

7.5 Analysis of Bandwidth

In order to discuss the bandwidth of the two types of systems, it must be clear what is meant by bandwidth in this context. Since the concern here is to accurately estimate angle (or velocity), the desired definition will be related to the rate at which changes in angle are detected by the system. In particular, the definition that will be used is the traditional 3dB bandwidth of the transfer function from angular inputs to angle estimate outputs:

$$H(s) = \frac{\Delta\hat{\theta}(s)}{\Delta\theta(s)}. \tag{7.23}$$

7.5.1 Bandwidth of the Impulse Sampler

Since the IMU is approximately first order with a pole at 450 rps, the instrument will contribute a pole to the system at $s = -450$. The next source of dynamics will be the analog filter. If a second order filter is assumed with two poles at $s = -p_0$, then another term will multiply the transfer function: $1/(s + p_0)^2$. The final contribution to the system dynamics is the rate at which the main processor reads the value of the signal. This effect can be approximated by a pure delay of the form $e^{s\tau}$ for a worst case of $\tau = T$. This term models the fact that there is a delay from the time when the converter reads a value until the time the main processor is made aware of the result. Hence, the overall transfer function for this system is

$$\frac{\Delta\hat{\theta}(s)}{\Delta\theta(s)} = \frac{e^{s\tau}}{(s/450 + 1)(s/p_0 + 1)^2}. \tag{7.24}$$

If the value of p_0 is chosen large enough, the approximate bandwidth of the system will be 450 rps. Otherwise is will be slightly less than p_0. The value that can be chosen for p_0 depends on the sampling rate. p_0 must be significantly smaller than the sampling rate. The effect of the delay should not be ignored. It will add significant phase at higher frequencies resulting in a slower response.

7.5.2 Bandwidth of the V-F Converter System

For this system, the bandwidth model is simpler. The integration is being done by the analog circuitry instead of by the processor. When the response of this system is characterized, its frequency response turns out to be

$$\frac{\Delta\hat{\theta}(s)}{\Delta\theta(s)} = \frac{e^{s\tau}}{(s/450 + 1)}. \tag{7.25}$$

Here there are no filter poles to worry with. The delay and the sensor bandwidth limitations are the only sources of dynamics in the frequency range of interest.

Component	Area (in^2)	Power (W)	Expense ($)
PIC	.56	.05	15.00
Crystal Oscillator	.40	small	2.00
6 Filters	3.0	small	10.00
A/D converter	.81	.085	28.00
Power Supply	.45	.255	2.25

Table 7.1: Cost Matrix for Impulse Sampling System

Component	Area (in^2)	Power (W)	Expense ($)
PIC	.56	.05	15.00
Crystal Oscillator	.40	small	2.00
6 V-F Converters	4.0	2	30.00
FPGA	1.5	.20	30.00
Power Supply	.45	small	2.25

Table 7.2: Cost Matrix for V-F Converter System

7.5.3 Comparison of Bandwidth

Both systems are capable of achieving good bandwidth. For the impulse sampler, getting good bandwidth requires a high sampling rate, high p_0, and a high data throughput rate to the main processor. For the V-F converter system, getting good bandwidth does not require any special effort–it comes automatically as long as the throughput rate to the processor is sufficiently high.

7.6 Cost Estimates

In order to make a good decision about what type of system should be developed, the cost associated with each system should be predicted. Four main categories of expense have been determined.

7.6.1 Weight

Weight is added by the PC board, chips, and interconnection to and from the board. Assuming the interconnection cost will be the same for both types of systems, PC board weight and chip weight should be determined.

PC board weight is a function of the area of the board required. This will be approximated as the area of the chips needed times 5. That ought to give a rough idea of the area required for routing traces, mounting the board, etc. Table 7.1 shows the relevant data for all of the cost categories for the impulse sampling system. Likewise, Table 7.2 shows the data for the V-F converter system.

Although the data is rough, an estimate can be obtained for the weight of each system by considering the total area that each system will use. In the lab it was found that a PC board weighs about $2.2g/in^2$. Here, the factor $5g/in^2$ will be used to help compensate for the fact that the weight of the chips is not known. The result is that the estimate for the impulse sampling system is 29 grams. For the V-F converter system, the result is 37 grams. Although these numbers are very rough, they show that comparable weights would be expected for the two systems.

7.6.2 Expense

Expense includes cost of the components; time to design, build, test, and debug; and manufacturing. Numbers are difficult to determine in advance, but some qualitative insight will be helpful. The fact that the V-F converters might require someone to design an FPGA makes it very expensive. Alternatively, if counters are used instead of the FPGA, the board will become a little more heavy and significantly more complex (leading to more development time). The cost of the components and manufacturing should come to less than $200 for both systems.

7.6.3 Level of Effort

This was already mostly covered under expense. The V-F converter design, although more interesting, would require more effort. This effort would either take the form of learning new design tools to design the FPGA, or dealing with the complexity of the counters.

7.6.4 Power

The power estimate for the two systems is, again, rough, but still useful. For the impulse sampling system, the estimate is .6 watts. On the other hand, the V-F converter system has an estimate of 2.1 watts.

7.7 Conclusions

The performance of the existing system is not as good as it can be. The discussion has shown how both its drift performance and bandwidth can be improved. In the area of bandwidth in particular, significant improvement is expected. Before a good estimate of the expected drift improvement can be determined, values for the parameter values must be determined. However, the model shows that there is potential for a good bit of improvement. The effects of almost all of the noise sources can be mitigated to some degree with a better system.

The cost of developing an improved system is not prohibitive, even if only for test purposes. The development of either of the proposed systems would be possible. However, the higher cost of the V-F converter system weighted with only slightly better expected performance makes the impulse sampling system seem more attractive. Therefore, the impulse sampling system will be developed.

Chapter 8

Sensor Board Design

8.1 Introduction

A custom impulse sampling inertial system does away with the need for the PC-104 A/D converter. However, in 1996, the sonar/compass system relied on the A/D converter's Digital I/O pins for communication with the main computer. Therefore, the revised inertial system required a revised sonar/compass system. As the design for these two systems was carried out, it became clear that the easiest solution was to integrate the two systems onto a single PC board. This board was designed as part of the 1997 effort and is referred to as the Sensor Processing Unit.

8.2 Sensor Board Functions

The Sensor Processing Unit has four functions. First, it samples the Inertial sensor at a high rate. It also takes data readings from the sonar and the compass. Last, it sends data back to the main computer. In order to simplify debugging, a pair of LED's on the board indicate if the board has detected an error condition with either the IMU or the compass[1].

If an error condition is detected by the board, that information is passed along to the main computer which has the option of resetting the board in-flight. Because the board is capable of rebooting very quickly (on the order of .1 seconds), in-flight resets are a reasonable way to correct errors.

8.3 Interface

The sensor board communicates with the main processor via an RS-232 cable. Data is sent only *from* the board *to* the main processor, so a 2 conductor cable is sufficient. Additionally, one wire from the main processor's parallel port can be used to send a reset signal to the sensor board.

8.3.1 RS-232 Protocol

The RS-232 cable is a uni-directional data pipeline; data is only sent from the sensor board to the main computer and cannot be sent in the reverse direction. The protocol used is:

- 1 Start Bit

[1] Because of the way the sonar is controlled, there is no easy way to detect an error with the sonar.

- 8 Data Bits
- 1 Stop Bit

The Data transfer rate is set at 19.2 Kilobaud.

8.3.2 Packet Format

The sensor board sends data in the form of packets. One packet is sent every 40 milliseconds yielding a frequency of 25Hz[2]. Each packet has the following form:

- 3 *Start* bytes (0xAB 0xCD 0xEF)
- 1 *Size* byte (Number of bytes in packet excluding *Start* bytes)
- 1 *Status* byte (Gives status of sensors)
- 2 *Sonar* bytes (High byte then low byte – Straight binary)
- 2 *Compass* bytes (High byte then low byte – Straight binary)
- 12 *IMU* bytes (6 channels [Rx Ry Rz Ax Ay Az]; 2 bytes per channel; high byte then low byte; two's complement form)
- 1 *Temperature* byte (Temperature in straight binary)
- 1 *Battery* byte (Battery voltage in straight binary)
- *N Other* bytes (Readings from other sources not yet dreamed up...)
- 1 *Checksum* byte (Sum of all binary numbers in packet excluding *Start* bytes and *Checksum*)

Currently, $N = 0$, so the *Size* byte is equal to 21.

8.3.3 *Status* Byte

The *status* byte contains information about the status of the sensors:

- **Bit 0**: IMU Status. 1 $\Longrightarrow$ IMU alive; 0 $\Longrightarrow$ IMU dead.
- **Bit 1**: Compass Status. 1 $\Longrightarrow$ Compass alive; 0 $\Longrightarrow$ Compass dead.
- **Bit 2**: Sonar Data. (1 means fresh sonar data in packet)
- **Bit 3**: IMU Data. (1 means fresh IMU data in packet)
- **Bit 4**: Compass Data. (1 means fresh Compass data in packet)

[2] Originally, data was to be sent back at 50Hz. It was later found that the main processor was not powerful enough to read data in at such a high rate.

8.4 Hardware Design

8.4.1 Component Selection

The first stage of designing the sensor board was selecting components to use on the board. To decide what components were necessary, the board's functions had to be determined. The board must accept six inertial, analog signals from the IMU, filter them, and sample them with an accuracy of 12 bits. It must collect data from the compass and from the sonar. It also must be able to flash LEDs, sample the battery voltage, sample the IMU temperature, and send data to the main processor. It is also desirable for the board to have the capability of future expansion[3]. As always with the DSAAV, the weight of the board must be kept to a minimum.

A/D Converter Selection

Because the purpose of the board is centered around sampling analog signals, the first component to be selected was the A/D converter. It was assumed that a 12 bit converter would be used. This is because 8 bits is not enough to give reasonable accuracy, whereas it would be difficult to make the board noise-free enough to make any sense of 16 bit conversions.

There are many A/D converters on the market today. These cover a broad range of functionality, size, speed, accuracy, and cost. Several key functions were important to the selection. First, a 12 bit A/D converter was necessary. Second, at least a six channel analog multiplexer was required to make it possible to sample all six inertial sensors. Third, a precision reference was needed as the metric for A/D conversions. Some of these functions could be performed externally to the A/D chip, but it is more efficient in terms of weight, design time, and cost to buy a chip that includes as much functionality as is needed.

The chip that was selected for this purpose was the AD7891 made by Analog Devices. Although it is a fairly large chip (44 pin PLCC), it has all of the features necessary: 12 bit A/D converter with a conversion time of 1.6μs, eight channel multiplexer, $\pm$5 volt input range, reference, single 5 volt power supply, and a track and hold amplifier. This chip supports both a serial and a parallel communications interface.

One of the issues that was faced with the A/D converter selection was that of single-ended versus differential sampling. The A/D board used in 1996 could sample eight differential signals, so all of the inertial signals were sampled differentially. This helped reduce the effects of voltage drops along the lines that carried signals from the IMU to the sampling board. However, if the signals were to be sampled differentially on the new board, a great deal of complexity would be added to its design. While investigating this issue, it was discovered that all of the return signals on the inertial instrument were shorted together anyway! Therefore, as long as the ground wire between the sensor and the sensor board was kept short, it would be acceptable to use a common ground for all of the signals and sample them in a single-ended sense.

Microprocessor Selection

The requirements of the board implied that a significant amount of computation had to be carried out. Many sophisticated processor features were required such as timing, interrupts, digital I/O, data processing, etc. Other work at Draper Labs had involved the use of a microprocessor called the PIC, made by Microchip. Because there was already experience with this processor, it was investigated as a possible choice. After considering the computational needs of the board, the PIC16C73A was chosen as the processor to be used. It has many features that are critical to the success of the

[3] This expansion may take the form of controlling new sensors, or sampling more signals

board including a 5MHz instruction rate, several interrupts, three timers, a synchronous serial port, a UART, 192 bytes of RAM, 22 I/O pins, a five channel A/D converter (8 bit), and UV erase-ability.

The use of this microprocessor was simplified because the laboratory already had the necessary hardware and software necessary to program it. Also, there was already a knowledge base of how to use the processor.

Operational Amplifier Selection

Because the A/D converter is capable of sampling at a very high rate, simple anti-aliasing filters can be used. The roll-off of these filters need not be spectacular as long as their 3dB points are well below the sampling frequency. Hence, single-pole passive filters are sufficient for this purpose. However, the specification sheet for the A/D converter indicates that the inputs to the multiplexer are not high impedance. This means that these inputs must be driven with a low impedance. It is not possible to use an all-passive filter and still drive the A/D converter with a low impedance. Therefore, amplifiers were necessary. Because the signals going into these amplifiers are relatively tame, they need not have especially good specs.

At least one operational amplifier must be used for each channel. Hence, eight amplifiers should be included if all eight A/D channels are ever to be usable. To minimize size, surface mount quad op-amps were selected. In particular, the LM614 was selected as the op-amp to use. These are supplied by ± 15 volts.

Connector Selection

Although the issue of connectors is not always a very interesting one, it plays a very important role in a system such as the DSAAV. Reliability of the connectors is critical. Reliability is maximized when the number of wires being connected is minimized, and when the connectors have a positive locking mechanism to keep them from becoming disconnected.

For the sensor board, a family of connectors made by Amp was selected. These "Mate-and-Lock" connectors have a male housing that is soldered onto the PC board, and a female mate with a positive locking mechanism. These connectors are large and are capable of carrying far more current than is needed by the sensor board, but their robustness is worth the penalty in size and weight.

Other Components

An assortment of other components are necessary to complete the parts list for the sensor board. In order to drive the serial line, a MAX232 serial line transceiver is used. Surface mount resistor packs were selected in hopes of minimizing the board area used up by resistors. A surface mount LED package was found that contains a green LED and a red LED. By getting two LEDs in one package, a little board area was saved. A 20MHz crystal oscillator is used as the clock for the processors. Finally, an LM340T5 is used to provide a clean 5 volt supply for the A/D converter.

8.4.2 System Level Design

Before the circuits were designed, an expectation of performance was developed. Due to limitations in the vehicle hardware itself, it was determined that the highest useful rate at which data could be used was 50 Hz. It was decided that the highest possible data rate would be used as long as the rate did not exceed 50 Hz. Later experiments showed that 25 Hz was the highest rate that the main processor could support.

However, the hardware selected for the board is capable of taking data at a much higher rate than 25 Hz. In order to take advantage of the hardware's power and improve the system's performance, data was taken at a much higher rate (1.6 kHz per channel). With such a high sampling rate, the

analog filters need not have particularly steep roll-off. Therefore, first order RC filters were sufficient for the job. The pole of these filters was placed at 50Hz in order to make sure that sufficiently high bandwidth information would ultimately make it to the main computer.

To make the system work, the disparity between the sampling rate and the packet transmission rate had to be resolved. With a 1.6 kHz sampling rate and a 25 Hz data transmission rate, 64 samples are taken for each packet. If these 64 samples of 12 bit data are added together, the result is an 18 bit value which can be divided by 64 by the main processor to produce a very accurate reading. However, 18 bits is 2 bytes plus 2 bits. It is more convenient to throw away the least significant two bits of each reading, and add up the 64 readings to result in a 16 bit result which can be transmitted in 2 bytes[4].

8.4.3 Circuit Design

The hardware design for the sensor processing unit is divided into two main sections. They are the analog section and the digital section and are discussed independently below.

Digital Section Design

Figure 8-1 shows the schematic diagram for the digital circuitry. The circuit is built around two PIC16C73A micro-controllers. One of them, the *IMU-PIC*, is responsible for controlling the A/D converter and sending sampled data back to the *Periph-PIC*. The Periph-PIC is responsible for all of the other board functions including sampling the battery voltage, sampling the IMU temperature sensor, controlling the sonar, controlling the compass, controlling the LEDs, receiving data from the IMU-PIC and sending data to the main processor. The idea behind using two PICs instead of one was that it would be easier to sample the inertial sensor at a very steady frequency if a controller was dedicated to that purpose. However, if enough effort were put into the software, one PIC would probably be powerful enough to perform all of the necessary board functions[5].

A 20MHz crystal oscillator provides the clock for the microprocessors[6]. Pin 1 on both processors is connected to the *Reset* line which can be pulled low by the main processor to reset the board. This line is pulled up to 5v by a 14KΩ resistor to make sure the board is not accidentally reset.

Most of the pins on the IMU-PIC are used to communicate with the A/D converter. In all cases but one, this is accomplished by simply connecting the pin on the IMU-PIC to the appropriate pin on the A/D converter. In the case of pin 6 (RA4), the pin does not have a full output line driver. Instead, it only has an open-collector output[7], so this pin must be tied to +5V through a resistor. A 5K resistor was chosen for this purpose. Pin 14 is used as the clock for the data pipeline between the two PICs. Similarly, pin 16 is the data line for the communications channel. Pins 25 through 28 on the IMU-PIC were left available for whatever needs may be encountered in the future.

On the Periph-PIC, pin 2 is used to sample the battery voltage after it is been divided down by a factor of 4. Pin 3 is used to sample the IMU temperature sensor after it has been low-pass filtered by a single pole RC filter. Each of the signals on pins 2 and 3 are sampled to 8 bits of precision, which is more than sufficient for the nature of the data being sampled on these pins.

Pins 4 and 5 control the LEDs. The LEDs have a forward bias voltage of approximately 2 volts. 15 milliamps is enough current for the LEDs to radiate almost all of their rated output power.

[4] Because of the amount of noise in the system, there is very little useful information stored in the least significant two bits, anyway.

[5] In some ways it would be easier to use one PIC, since code would not have to be written to allow communication between the two PICs

[6] The processors have four different allowed oscillator configuration options. For the oscillator used, the correct option to use is **XT**.

[7] See the Microchip 16C7X manual for more details.

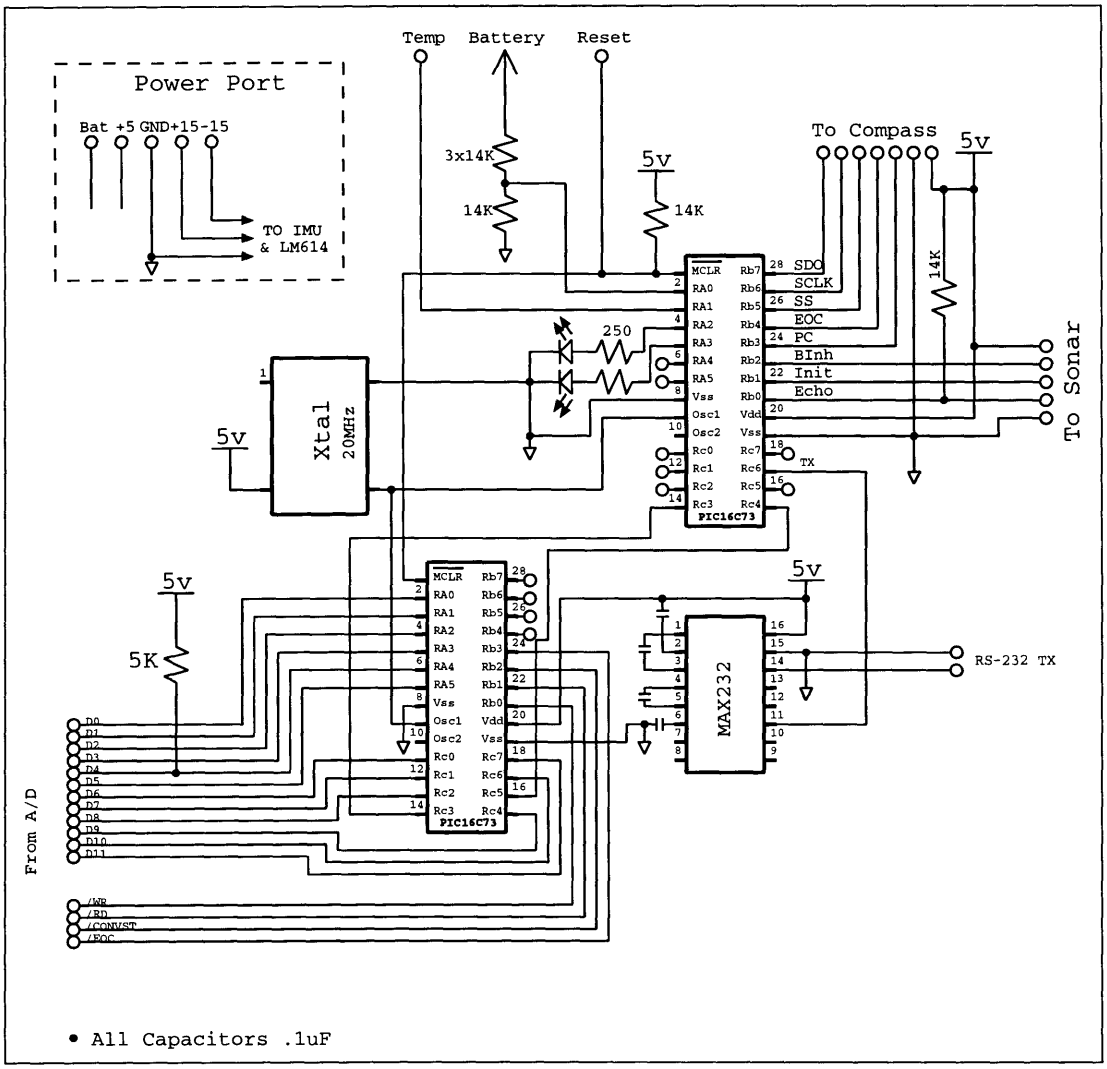

Figure 8-1: Sensor Board Circuitry

Therefore, 250Ω resistors are used, giving LED currents of 12 milliamps. At this current level, sufficient light is radiated.

As for the IMU-PIC, pin 14 on the Periph-PIC is used as the clock line for the PIC-to-PIC communication, while pin 15 is the data line. Pin 17 is the RS-232 data out line which is sent to the MAX232. The MAX232 is wired up according to the application notes for the part to function as an RS-232 data transceiver. The remaining pins on the Periph-PIC are used as either control lines to the compass and sonar, as power, or as expansion pins. Pin 21 is tied to +5V through a 14KΩ because the *Echo* line on the sonar is an open-collector output and must be tied to +5V through a resistor.

Analog Section Design

Figure 8-2 shows the schematic diagram for the analog circuitry. The bulk of the circuit is the A/D converter (AD7891), two packages of quad op-amps, and a linear regulator (LM340T5).

As discussed in Chapter 4, the outputs of the rate gyros are voltages in the range from -5 volts to +5 volts. These signals are low-passed filtered by single-pole RC filters with a cutoff frequency of 50Hz. The output of the filter is buffered with an op-amp. Each buffer has a resistor between its output and the - terminal of the amplifier in order to compensate for bias current in the amplifier. This precaution may not be necessary, since biases in the signals can (and will) be canceled in software. The outputs of the buffers are fed to the input channels on the A/D converter which are configured to be bipolar in the range from -5v to +5v.

The acceleration channels are identical to the gyro channels except that they include an additional resistor to convert the IMU output current into a voltage in the same way that the filters used in 1996 converted current to voltage. The addition of the current-to-voltage resistor changes the cutoff frequency of the filter slightly, but not enough to make a significant difference.

Two *Expansion* channels are available for the future when other instruments may need to be sampled. The circuitry for these channels is designed in such a way that a current-to-voltage conversion resistor can be left out (in the case of a voltage input), included (in the case of a current input), or shorted to ground (in the case of no input)[8].

The A/D converter is wired up according the the guidelines given in the application notes for the AD7891. Capacitors are used to supply power supply by-passing. Both digital and analog grounds are supplied to the part at the appropriate pins. The 5 volt supply used by the converter is generated from the battery voltage by a linear regulator in order to minimize the amount of ripple that will be coupled in to the analog signals. Data and control lines are wired to the IMU-PIC which is responsible for controlling the A/D converter.

8.4.4 Layout

The layout of a sensitive circuit like the sensor processing unit is an important part of its design. Care must be taken to keep down the amount of noise that is coupled from the digital electronics to the analog electronics. Ground loops should be avoided, and the overall size of the board should be kept to a minimum in order to minimize the weight of the board.

Six connectors are used to connect the board to the outside world. The first of these is the IMU connector. This connector brings in the three accelerometer signals, the three gyro signals, the temperature sensor signal, and the analog ground. The power connector is connected to the battery voltage, the system ground, a 5 volt switching regulator, and the ±15 volt supply. The compass connector supplies power to the compass and transfers data between the compass and the

[8] It is important that the inputs are not allowed to float or the input to the A/D converter may be saturated, thereby interfering with the achieved accuracy on other channels.

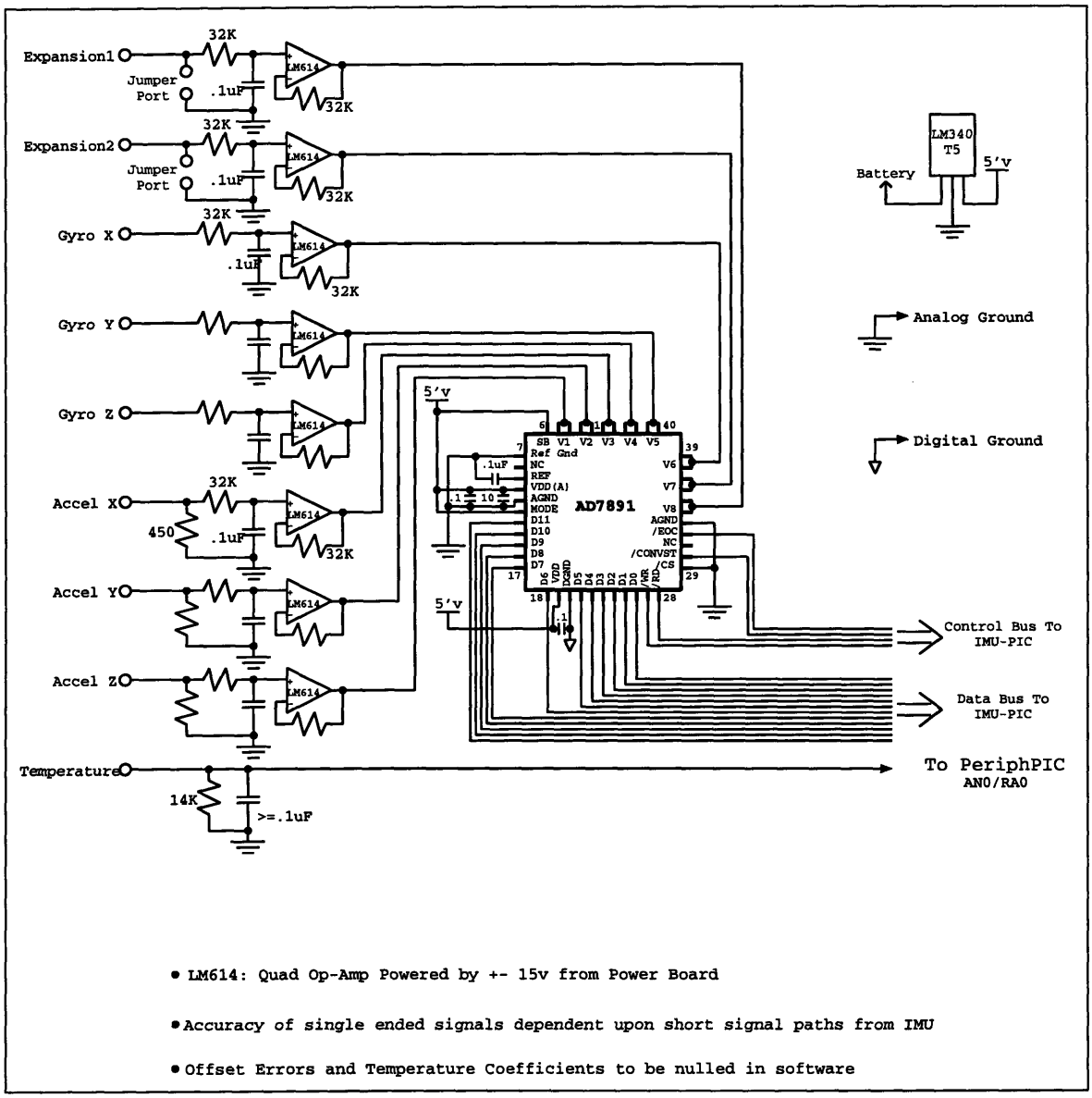

Figure 8-2: A/D Conversion Circuitry

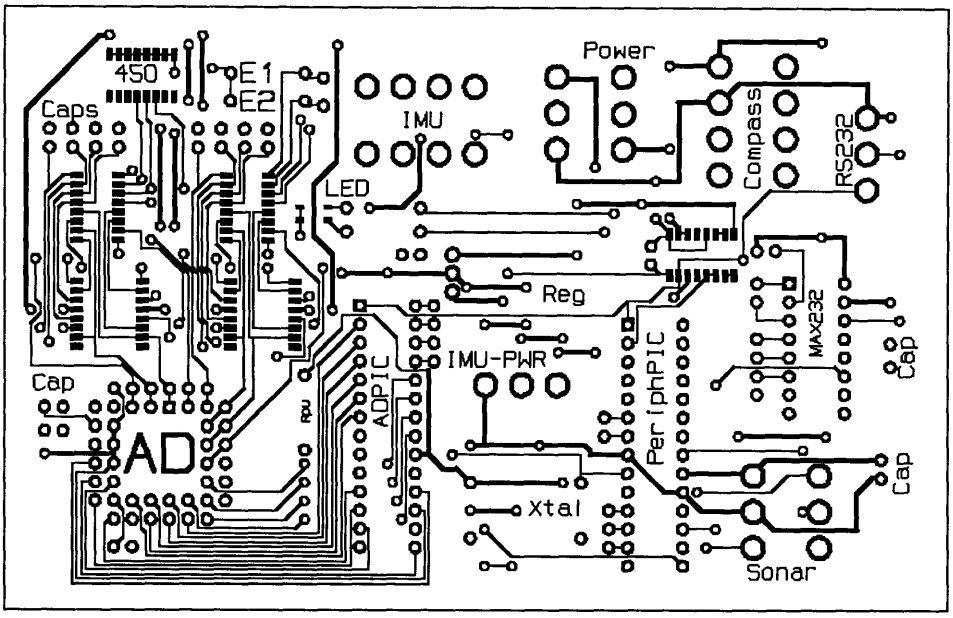

Figure 8-3: First Layer Metal — Sensor Board

Periph-PIC. Similarly, the sonar connector supplies power to the sonar and has data lines between the sonar ranging board and the Periph-PIC. The IMU-PWR connector supplies power to the IMU in the form of ground, +15 volts, and -15 volts. Finally, the RS232 connector makes connection to the RS-232 ground, RS-232 signal, and the reset lines.

The process used to manufacture the board allows for two layers of conductor. While the top layer is used only for interconnection, the bottom layer is used for interconnection when necessary, and also for ground planes. Two ground planes are used on the board in order to maintain very low impedance ground paths around the board. The analog ground plane is drawn below the analog section, while the digital ground plane is drawn below the digital section. The two ground planes are only connected at one point, inside the IMU. In order to understand how the ground path is layed out, consider the ground starting at the negative terminal of the battery. This ground makes contact to the board at the GND pin of the Power connector. The GND pin makes direct contact to the digital ground plane. The IMU-PWR connector also has a GND pin which supplies the IMU. This GND pin is connected to the digital ground plane. Inside the IMU, the ground is used as a reference point for all of the sensors. It is brought out of the IMU as a return signal and makes connection to the analog ground plane at the GND pin of the IMU connector. Because all of the chips make contact to the ground at the appropriate ground plane, and the ground planes are only connected at one point, there should be no ground loops in the system.

In order to keep the board dimensions small (They are 5" by 3.25"), surface mount components are used whenever possible. Many of the resistors used on the board are packaged as surface mount resistor arrays. The op-amps are also surface mount components.

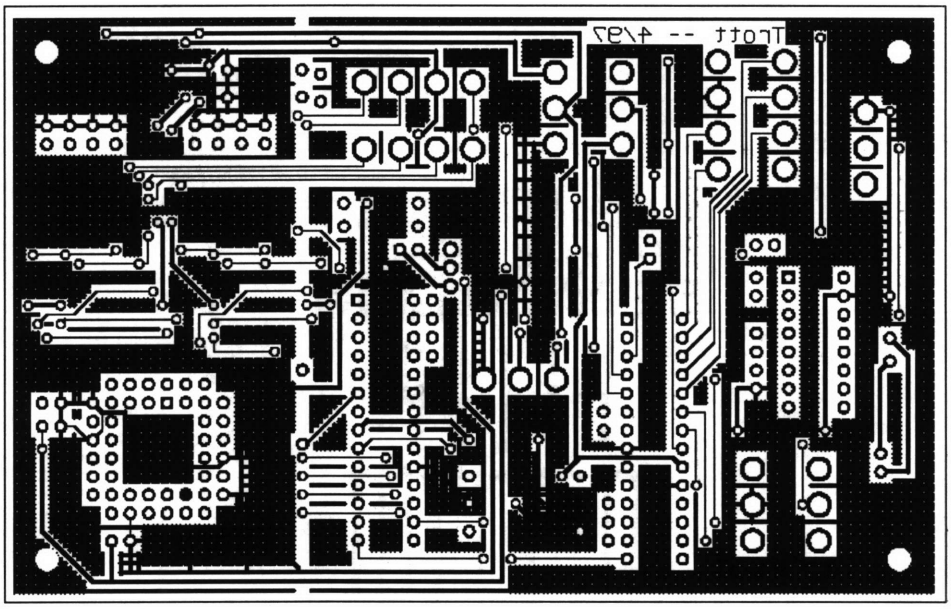

Figure 8-4: Second Layer Metal — Sensor Board

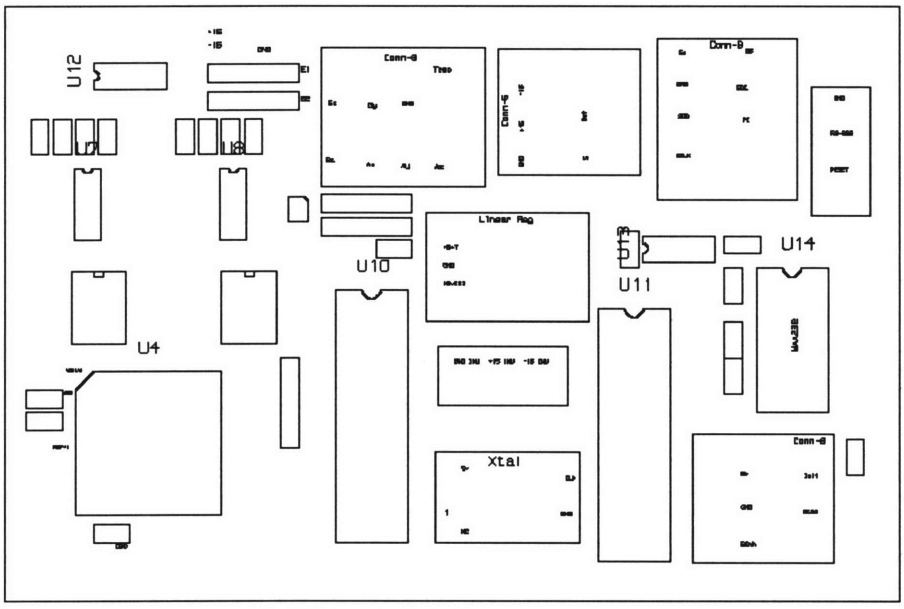

Figure 8-5: Silk-screen Layer — Sensor Board

8.5 Software Design

The software used for the PICs is somewhat complicated, mostly due to the fact that it is written in assembly language, and also by the fact that it takes advantage of most of the features available in the PICs. It has been included in Appendices A.6, A.5, and A.7. In order to understand this code, the manual on the PIC16C7X must be thoroughly understood. This manual is readily available from Microchip. A brief discussion of the software will be given here, but the software is well documented with comments, so it will be left to the reader to completely understand the code.

8.5.1 Periph-PIC Code

This code is based upon a 1ms time scale. Timer1 is used to generate an interrupt every millisecond. This interrupt routine then increments various timers, and resets some variables to get ready for a fresh millisecond of processing. Each of the main loops is run once during each millisecond period.

Conceptually, the code is broken into two main sections. Those being the standard, sequential running code and the interrupt code. The code is highly dependent upon the interrupts for just about all of its features. It has been written carefully to make sure that an interrupt can occur at any instant (during the standard sequential code) and everything should be fine. Interrupts are automatically disabled during interrupt routines (avoiding recursive interrupts!).

The first task of the sequential running code is to initialize all of the micro-processor states. Once the processor is initialized, the main return is begun which calls each of the major sub-routines once every millisecond. The major subroutines include the A/D converter routine, the compass routine, the transmission routine, and the sonar routine. The A/D converter routine takes readings of the battery voltage and IMU temperature 4 times each second. The compass routine checks for special timing conditions and controls the compass. The sonar routine drives the sonar control lines with the appropriate timing. Finally, the transmission routine builds packets and sends them at the right time.

In addition to many state variables used to store flags and counters, there are three main blocks of memory used to store data that needs to be transmitted. The first of these three blocks is used when the Periph-PIC is first reading data from the IMU PIC. Once an entire packet has been read from the IMU PIC, the data is transferred to the main data holding block of memory. The routines for Sonar, compass, battery, etc, update the main data holding memory block directly. When the time comes for a new packet to be sent to the ampro (every 20ms), the data from the main data holding block is transferred to the transmission data block for transmission. This is done so that once a packet is created, the data in the packet won't be updated by interrupt routines until the time comes to form a new packet.

The code checks for two error conditions. First, it checks and makes sure that new data is being returned by the compass. If too much time has elapsed between compass readings, the red LED will be turned on. If no new data has been sent from the IMU-PIC to the Periph-PIC, the green LED will be turned on. Similarly, the appropriate bits in the *Status* byte of the data packet will be cleared to indicate that a problem exists. The main processor can use that information to reset the sensor processing unit if it deems it necessary.

8.5.2 IMU-PIC Code

This code is structurally similar to the Periph-PIC code. interrupts are generated by Timer1 every 30.2μs. These interrupts are used as the timing source for the code. The code is broken into the normal running section and the interrupt section, as for the Periph-PIC code.

8.5.3 Programming Procedure

The process of programming a PIC is fairly straightforward, but it is worthwhile to briefly describe it. First, code is written or edited within an text editor. Once the code is ready to be tested, it is assembled with an assembler such as mpasmwin. The assembler creates a HEX file which is loaded into a programmer such as warp-3. In the programmer input window, a few options are set such as the type of processor being programmed, the type of oscillator that is being used, and whether or not the watch-dog timer should be enabled. Then, an erased PIC is placed in the programming socket and the software is told to program the chip. Chips that have previously been programmed can be cleaned by being placed in a UV eraser for 10 to 20 minutes (or longer).

8.6 Performance

There are several importance performance criteria that must be verified before that board can be ready for use on the DSAAV. Two of the easiest specifications to verify are power consumption and weight. The power consumption was found by adding up the specified power consumption for each of the components on the board. Additionally, power was budgeted for waste in the linear converter. From these calculations, the total power usage by the board was found to be approximately 440mW. This is a reasonable level of power consumption, particularly when it is noted that the A/D converter from 1996 used 1.6 watts. The weight of the board was measured to be approximately 80 grams which is equal to .18 pounds[9].

In addition to being light enough and power efficient enough, the board must be able to perform its functions. This means that it must be able to control the sensors, obtain accurate readings, and send data to the main computer. All of these functions must be performed at the proper rate.

The board was found to be able to control the sonar and the compass and return readings from both of them. Similarly, The timing constraints for the board have been met. This means that it is able to return packets at a rate of 50 Hz while still regularly taking readings from all of the information sources.

The only remaining issue, then, is how well does that analog portion of the board perform? To start with, the bandwidth of the RC filters was measured, and it was very close to the desired 50 Hz. In order to determine how accurate the A/D readings were, five minutes of data was taken from the gyro X channel. Similarly, five minutes of data was taken from the gyro X channel with the 1996 system. These two data sets were compared.

At first, the result of this comparison was disappointing. the standard deviation of the new system was significantly higher than that of the old system. In order to check out what was going on, the inputs to the A/D converter was probed with an oscilloscope. Even though these inputs enter the A/D converter after being filtered by the RC filters, they were *very* noisy. These signals had approximately 60mV of RMS noise on them. To put this number into perspective, the precision of the system should be determined. With 12 bits of precision, a 10 volt signal ($\pm$ 5 volts) can be resolved down to 2.44 mV. Hence, 60mV of noise is the same as 25 units out of 4096.

This noise problem is due to the fact that noise enters the signal path *after* the filters. Recalling from Chapter 7, one of the nice things about the V-F converter design was that it was difficult for noise to enter the signal path after the filtering was performed. However, the impulse sampling design is far more susceptible to noise. The problem was combatted in two ways. First, where there had been only a ground plane beneath the analog circuitry, a second ground plane was added below the digital circuitry. The hope was that this new ground plane would contain the fields generated by the digital electronics and keep fields from affecting the sensitive analog lines. Secondly, the sample

[9] Under the usual $9.8 m/s^2$ acceleration

averaging was implemented. The averaging serves to attenuate the high frequency noise that caused most of the problem.

Once these measures were taken, the performance of the system looked much better. Data was taken with both systems and compared. The new system showed a standard deviation of 4mV, whereas the older system showed a standard deviation of 12mV. For this reason, it is believed that the new system will outperform the old system in terms of drift performance, even though there has not yet been an opportunity to test this theory directly.

There are two ways in which the new system's performance may break down. The first is that noise coupled into it from other electronic components on the helicopter may adversely affect its performance. To date, it has only been tested as a stand-alone unit. Secondly, its performance may be degraded when it is powered by switching regulators. However, it is hoped that each of these effects will be minor in comparison to the noise that is coupled to the signal from the sensor processing unit itself.

8.7 Conclusion

The sensor processing unit is fairly straightforward to understand. The hardware is not very complicated, but it requires attention to detail to make it work properly. In order to produce a functioning piece of hardware like the sensor processing unit, the chip specifications must be well understood, the hardware must be carefully designed, the board must be painstakingly layed out, and the code must be written with care. The success of this hardware should be an important part of the DSAAV-97. It will simplify the overall system topology, reduce the system weight, increase robustness, and improve the navigation system's performance.

Chapter 9

Conclusion

The Draper Small Autonomous Aerial Vehicle is currently a year and a half into development. Many challenges and successes have been experienced over that time period. This chapter describes one of the vehicle's major successes and describes some of the improvements which ought to be pursued in the future.

9.1 The 1996 International Aerial Robotics Competition

The goal of the first year of development of the DSAAV was to win the 1996 International Aerial Robotics Competition. This competition was held at Disney World in Orlando, Florida in July of 1996. Originally, over 20 teams from around the world entered the contest. However, by the day of the competition, only seven teams were able to show up with a vehicle.

The objective of the competition was to fly over a 60′ by 120′ field and locate the positions of five drums that were placed on the field. Labels on the drums identified them as radioactive or biohazardous drums. The vehicle was to take off, fly over the field, locate the drums and determine their types, and land back in the starting area. The vehicle could also recover a disk from one of the drums.

The DSAAV was the first of the vehicles to fly on the day of the competition. Seven autonomous flights were made during the alloted hour. On the best of the seven attempts, all five drums were correctly located, and two of them were correctly identified. Although valiant attempts were made by some of the other vehicles, none of them were able to even fly autonomously. The DSAAV was far and away the winner of the competition. It was able to perform all of the desired tasks except for the disk retrieval.

9.2 Future Improvements

9.2.1 Sensor Processing Unit Improvements

There are many ways in which the sensor processing unit can be improved. The work that has so far been done on it is really just the leg work of a very interesting project. The intial stage of concept verification has been completed. Now, effort can be put into optimizing its performance.

The first and most important way in which it can be improved is in the area of noise. There are two approaches to this. First, it could be re-designed with the V-F conversion topology. This change would almost certainly reduce the noise of the system. Second, the present system could be

studied in order to determine exactly where the noise is coming from and how it can be eliminated. It is likely that with some simple, yet clever design changes, much improvement could be achieved.

Another potential problem with the current design lies in the communication from the IMU-PIC to the Periph-PIC. Currently, a communication known as Serial Peripheral Interface (SPI) is used. After this protocol was implemented, it was discovered that it could not recover from certain error conditions. In particular, if one of the PICs loses sync with the other PIC (i.e. misses a clock cycle), there will be no way for the two to re-synchronize unless the main processor sends a reset signal to the board. Another communications protocol might not suffer from this problem. The Inter-Integrated Circuit protocol, for instance, can easily be implemented with the PICs and does not have this problem.

Another improvement that could be made to the system would be to implement two-way communication between the board and the main processor. Presently, data is only sent from the board to the main processor, but it would be useful to be able to send commands from the main processor to the sensor processing unit. Commands might tell the board to change the rate at which the sonar is being pinged, or to send a signal to some component that is not yet being used with the board.

There is also potential for improvements with the type of processor being used. As processors become smaller and more powerful, it may become easy to do the entire task of the board with one processor instead of two. Also, it would be nice to use a processor with built-in EEPROM, so that it would not be necessary to use a UV eraser.

Great improvements could be made in the way in which the IMU data is processed. The inertial sensors could be modeled more completely by adding error terms and non-linearities. If these behaviors were known, better use of the instrument's information could be made. This is especially true of the temperature information that comes from the IMU's temperature sensor. So far, no analysis has been done of the effect of the temperature on bias. This analysis would not be very hard, and ought to be done in the near future.

9.2.2 Power System Improvements

The power system also has potential for improvements. As mentioned in the chapter on the power system, it may be possible to find a better energy source for the vehicle. Primary batteries are one option.

The power board exhibits some quirky behavior that should one day be resolved. In particular, the Astrodyne switching regulators fail to regulate well when differing current levels are drawn from the dual-ended supplies. Either the problem should be fixed, or a new power regulator should be selected that does not exhibit this problem.

Finally, Interesting work could be done in the area of EMI on the power board. EMI filters could be designed that would eliminate the high frequency components on the power lines. This would reduce the amount of interference coupled in to other on-board systems.

9.2.3 Other Improvements

There are many other ways in which the vehicle could be improved. Better methods of mounting components, lighter weight materials, simpler connectors, and more robust construction techniques can all be explored. The field of autonomous aerial vehicles is still very young, and there will be many challenging areas in which to develop the DSAAV for years to come.

9.3 Conclusion

The Draper Small Autonomous Aerial Vehicle has been a very challenging, exciting, and rewarding project. It requires a multi-disciplinary approach to design. This approach is useful for gaining an

appreciation for all of the factors that affect a real design. The project has been very successful, in that it has completed its first year's goals, and is well on its way to completing its second-year goals. A lot of hard work has gone into it, and must continue to go into it, but the hard work is rewarded with the sounds and sights of a small aerial vehicle that flies — all on its own.

Appendix A

Code Listings

A.1 Pinger BASIC Stamp Code

```
x var word
output 0
output 1
output 2
x=0
low 0
low 1
low 2

ping: high 0
      pulsout 2,320
      high 1
      for x=1 to 22
      next
      low 0
      low 1
      for x=1 to 22
      next
      goto ping
```

A.2 Peripheral BASIC Stamp Code

```
heading var word
time    var word
output 5   'Sync
output 6   'SDO
output 7   'Clock
output 13  'con1
output 14  'con2
input  12  'eoc
input  9   'output
```

```
output 8   'clock
output 10  'ss
output 11  'p/c
input  15  'echo
BigLoop:
        gosub DoHeading
        gosub SendHeading
        pause 40
        gosub SendAltLow
        gosub DoAltitude
        gosub SendAltHigh
        goto BigLoop

DoAltitude:
        pulsin 15,1,time
        return
SendAltHigh:
        low 5
        shiftout 6,7,LSBFIRST,[time.highbyte]
        high 13
        low 14
        high 5
        return

SendAltLow:
        low 5
        shiftout 6,7,LSBFIRST,[time.lowbyte]
        low 13
        high 14
        high 5
        return

SendHeading:
        low 5
        shiftout 6,7,LSBFIRST,[heading.lowbyte]
        shiftout 6,7,LSBFIRST,[heading.highbyte 2]
        high 13
        high 14
        high 5
        return

DoHeading:
        low 11
        pause 10
        high 11
        high 8

HLoop:
        if in12 = 1 then ready
        goto HLoop
```

```
ready:
        low 10
        pause 10
        shiftin 9,8,2,[heading 16]
        high 10
        return
```

A.3 DM5406 Driver Code

```
#include "dm406.h"

#define SAMPLES 2
#define NCHANNELS 7

volatile unsigned int   ChVal[SAMPLES][NCHANNELS]; // Data read from converter
volatile float          imuData[NCHANNELS];        // output voltages
volatile int            dio_value = 0;             // historical
volatile char           current_channel = 0;       // Channel next in read queue
int  ADClockRate = 300;
char ADSampleSubCount = 0;
int  ADSampleCount = 0;
int  ADSamples = SAMPLES;
int  MyCount = -1;

int  InitStat = 0;

int sonarData, compassData;
int newSonar = 0, newCompass = 0;

#define ABS( var ) ( var < 0.0 ? -(var) : (var) )

/*****************************************************************/
/******** External Functions *************************************/
/*****************************************************************/


/*********************************************/
/**** ADSetClock(int rate) *******************/
/**** Must be used in conjunction with   *******/
/**** ADInit().  rate is sampling        *******/
/**** Frequency of one of the  channels  *******/
/*********************************************/
void ADSetClock(int rate)
{
        ADClockRate = NCHANNELS*rate;
}
```

```
/*********************************************/
/**** ADInit(int mode) ************************/
/**** mode can be 0: No Samples        **********/
/****             \>1: Scan 6 Channels **********/
/****             \>2: Channel 7       **********/
/****             \>3: Channel 8       **********/
/*********************************************/
void ADInit(int mode)
{
        printf( "Initializing A/D...\n" );
        if((ADClockRate < 3) || (ADClockRate > 6000))
        {
                printf("ADClockRate out of valid range\n");
                exit(0);
        }

        current_channel = 0;
        dm406_init(mode);
        printf( "done\n" );
}



/*****************************************************************/
/******** Internal Functions *************************************/
/*****************************************************************/


void SetExternalTrigger(unsigned char TriggerStatus)
{
        unsigned char B;

        B = inp(DM406_PPIB);                            // Read current byte
        B = B & 191;                                    // Clear B6
        B = B | TriggerStatus * 64;                     // set trigger bit
        outp(DM406_PPIB, B);                            // write new byte
}

void SetIRQStatus(char IRQStatus)
{
        unsigned char B;

        B = inp(DM406_PPIB);                            // Read Current Byte
        B = B & 127;                                    // Clear B5
        B = B | IRQStatus * 128;                        // Set IRQ select bit
        outp(DM406_PPIB, B);                            // Write new byte
}

void SetInterruptSource(unsigned char Source)
```

```
{
        unsigned char B;

        B = inp(DM406_SCN_BRST);                        // Read Current Byte
        B = B & 207;                                    // Clear bits 4 and 5
        B = B | Source * 16;                            // set new bits
        outp(DM406_SCN_BRST, B);                        // Writ new byte
}


void SetUserClock(float Rate)
{
        ClockMode(0, 2);
        ClockDivisor(0, 16);
        ClockMode(1, 2);
        ClockDivisor(1, (500000.0 / Rate));

        ClockMode(2, 2);
        ClockDivisor(2, 2);
}

void ClockMode(unsigned char Clock, unsigned char Mode)
{
        unsigned char StatusByte;

        StatusByte = (Clock * 64) + (Mode * 2) + 48;
        outp(DM406_TIMER_CTRL, StatusByte);
}


void ClockDivisor(unsigned char Clock, unsigned int Divisor)
{
        unsigned char MSB, LSB;
        unsigned int PortID;

        PortID = DM406_TIMER_A + Clock;
        LSB = Divisor % 256;
        MSB = Divisor / 256;
        outp(PortID, LSB);
        outp(PortID, MSB);
}


float int_to_volt(int data)
{
        float fdata;

        if (data > 2047)
           data -= 4096;                          // Convert From Two's Complement
        fdata = 0.0025*data;
```

```
        return fdata;
}


int read_data(int *rdata)
{
        int hdata, ldata, data;

        if(!((inp(DM406_CONVST) & 0x01)))              // If conversion is not
            return 0;                                  // complete, return 0

        hdata  = inp(DM406_ADC_VAL) & 0x0F;            // Read High Byte
        ldata  = inp(DM406_ADC_VAL);                   // Read Low Byte
        *rdata = (hdata*256) + ldata;                  // Get 12 bit value

        return 1;
}

void start_conversion(void)
{
        outp(DM406_CONVST, 0x00);
}


void dm406_init(int mode)
{
        int data;

        if (qnx_name_attach(0, "DM406") == -1)
                InitStat = 1;

        outp(DM406_RESET, 0x00);
        read_data(&data);
        outp(DM406_PPICTL, 0x99);

        switch (mode)
        {
                case 0:
                        outp(DM406_SCN_BRST, 0x00);
                        outp(DM406_PPIB, 0x00);
                        break;
                case 1:
                        outp(DM406_SCN_BRST, 0x86);            // 7 channels
                        outp(DM406_PPIB, 0x00);
                        SetExternalTrigger(ENABLED);
                        SetIRQStatus(1);
                        SetInterruptSource(SOURCE_AD_START);
                        break;
                case 2:
```

```
                        outp(DM406_SCN_BRST, 0x00);
                        outp(DM406_PPIB, 0x06);
                        break;
                case 3:
                        outp(DM406_SCN_BRST, 0x00);
                        outp(DM406_PPIB, 0x07);
                        break;
        }

        inp(DM406_EOI);
        outp(DM406_PPIC, 0xff);
        if( InitStat == 0 )
                qnx_hint_attach(DIO_IRQ, &interrupt_handler, FP_SEG(&dio_value));

        SetUserClock(ADClockRate*SAMPLES);              // Set Sampling Rate

}
/*************************************************************************/
#pragma off(check_stack);
pid_t far interrupt_handler(void)
{
        int hdata, ldata;

        hdata = inp(DM406_ADC_VAL)&0x0F;
        ldata = inp(DM406_ADC_VAL);
        ChVal[ADSampleSubCount][current_channel] = hdata*256 + ldata;
        current_channel++;
        if(current_channel >= NCHANNELS)
        {
          current_channel = 0;
          ADSampleSubCount++;
        }
        if( ADSampleSubCount >= SAMPLES ) {
          ADSampleCount++;
          ADSampleSubCount = 0;
        }
        inp(DM406_EOI);
    return 0;
}
#pragma on(check_stack);
/*************************************************************************/

int ADUpdate( void ) {

        int i, j, ticks;

    while( ADSampleCount == MyCount )
                printf( "" );

        ticks = ADSampleCount - MyCount;
```

```
    MyCount = ADSampleCount;

        for( j=0; j<NCHANNELS; j++ ) {
                imuData[j] = 0.0;
                for( i=0; i<SAMPLES; i++ ) {
                        imuData[j] += int_to_volt( ChVal[i][j] )/SAMPLES;
                }
        }

        return ticks;

}

#define VALID_DATA      0x02
#define LOW_SONAR_DATA  0x40
#define HIGH_SONAR_DATA 0x80
#define COMPASS_DATA    0xc0
#define TYPE_INFO       0xc0

int ADUpdateSonarCompass( void ) {

        static char highsonar;
        char adata, cdata, answer;
        int newSonarData;
        static char mustKeepCompass = 0;
        static char mustKeepSonar   = 0;

        /* attempt to read data */
        cdata = inp(DM406_DIOC);
        if( cdata&VALID_DATA ) {
            adata = inp(DM406_DIOA);
            if( inp(DM406_DIOC) == cdata ) {

                /* have a valid data byte */

                switch( cdata&TYPE_INFO ) {
                case COMPASS_DATA:
                    if( ( ABS( compassData - adata ) < 5 ) ||
                                !mustKeepCompass ) {
                            compassData = adata;
                            newCompass = 1;
                            mustKeepCompass = 2;
                    } else
                    mustKeepCompass--;
                    break;

                case HIGH_SONAR_DATA:
                    highsonar = adata;
                    break;
```

```
                case LOW_SONAR_DATA:
                    newSonarData = highsonar*256 + adata;
                    if( ( ABS( newSonarData - sonarData ) < 800 ) ||
                                        !mustKeepSonar ) {
                            sonarData = newSonarData;
                            newSonar = 1;
                            mustKeepSonar = 4;
                    } else
                    mustKeepSonar--;
                    break;

                }
            }
        }

        answer = newSonar+2*newCompass;
        newSonar = newCompass = 0;

        return answer;
}


#ifndef ON_BOARD
void main( int narg, char *arg[] ) {

#define RATE 25
        int count = 0, i;
        float w[3], a[3], wrms[3], arms[3];
        float voltage;
        float wm[3], am[3];

    ADSetClock( RATE );
    ADInit( 1 );


        while( 1 ) {
                ADUpdate();

                switch( ADUpdateSonarCompass() ) {
                case 1:   /* sonar data */
                        printf( "Sonar = %d\n", sonarData );
                        break;
                case 2:   /* compass data */
                        printf( "Compass = %d\n", compassData );
                        break;
                }

                w[0] = imuData[0]*1000.0/24.8;
                w[1] = imuData[1]*1000.0/25.1;
                w[2] = imuData[2]*1000.0/12.6;
```

```
                a[0] = imuData[3]*32.2/1.4;
                a[1] = imuData[4]*32.2/1.3;
                a[2] = imuData[5]*32.2/1.4;
                voltage = imuData[6]*4.15;

                for( i=0; i<3; i++ ) {
                        wm[i] = ( w[i] + wm[i]*count )/( count+1 );
                        am[i] = ( a[i] + am[i]*count )/( count+1 );
                    wrms[i] = ( pow( w[i]-wm[i], 2 ) +
                                wrms[i]*count )/( count+1 );
                    arms[i] = ( pow( a[i]-am[i], 2 ) +
                                arms[i]*count )/( count+1 );
                }

                count++;
                if( count == 10*RATE ) {
                        count = 0;
                }

        }

}
#endif
```

A.4 DM5406 Driver Code Header File

```
/* dm406.h */

#ifndef DM406_H
#include "library.h"
#include <stdio.h>

/* description of the DM406 board: */
#define DM406_BASE        0x300
#define DM406_CONVST      DM406_BASE+0
#define DM406_DAC_UPDATE  DM406_BASE+1
#define DM406_ADC_VAL     DM406_BASE+1
#define DM406_RESET       DM406_BASE+2
#define DM406_SCN_BRST    DM406_BASE+3
#define DM406_PPIA        DM406_BASE+4
#define DM406_PPIB        DM406_BASE+5
#define DM406_PPIC        DM406_BASE+6
#define DM406_PPICTL      DM406_BASE+7
#define DM406_TIMER_A     DM406_BASE+8
#define DM406_TIMER_B     DM406_BASE+9
#define DM406_TIMER_C     DM406_BASE+10
#define DM406_TIMER_CTRL  DM406_BASE+11
#define DM406_DAC1        DM406_BASE+12
```

```
#define DM406_DAC2        DM406_BASE+14
#define DM406_EOI         DM406_BASE+14
#define DM406_DIO         DM406_BASE+4
#define DM406_DIOA        DM406_BASE+4
#define DM406_DIOC        DM406_BASE+6

#define DIO_IRQ           (0x06)
#define DM406_AIO_MAX     (4096)

void  dm406_init(int);
pid_t far interrupt_handler(void);
void start_conversion(void);              // Starts single conversion
int read_data(int *);                     // Get value from AD board
float int_to_volt(int);                   // Convert 2's complement to voltage
void CalibrateIMU(float *);               // Find Drigt to Cancel
void SetUserClock(float Rate);            // Set Clock Rate
void ClockMode(unsigned char, unsigned char);
void ClockDivisor(unsigned char, unsigned int);

#define ENABLED 1
#define DISABLED 0

#define SOURCE_AD_START 0
#define SOURCE_DMA_DONE 1
#define SOURCE_TRIGGER 2
#define SOURCE_PACER_CLOCK 3

#define DM406_H
#endif
```

A.5 IMU-PIC Code

```
        processor 16c73

#include "16c73.h"

;;;;;;;;;;;;;;;;;;;;;;;;
; Set up Pin Definitions
;;;;;;;;;;;;;;;;;;;;;;;;

#define DB0     _porta,0              ; Pin 2
#define DB1     _porta,1              ; Pin 3
#define DB2     _porta,2              ; Pin 4
#define DB3     _porta,3              ; Pin 5
#define DB4     _porta,4              ; Pin 6
#define DB5     _porta,5              ; Pin 7
#define DB6     _portc,0              ; Pin 11
#define DB7     _portc,1              ; Pin 12
```

```
#define DB8     _portc,2                ; Pin 13
#define DB9     _portc,4                ; Pin 15
#define DB10    _portc,6                ; Pin 17
#define DB11    _portc,7                ; Pin 18

#define Format  _porta,0                ; Pin 2
#define SWSTBY  _porta,1                ; Pin 3
#define SWCONV  _porta,2                ; Pin 4
#define A0      _porta,3                ; Pin 5
#define A1      _porta,4                ; Pin 6
#define A2      _porta,5                ; Pin 7

#define WR      _portb,0                ; Pin 21
#define RD      _portb,1                ; Pin 22
#define CONVST  _portb,2                ; Pin 23
#define EOC     _portb,3                ; Pin 24

#define TEST    _portb,7                ; Pin 28
#define TEST2   _portb,6                ; Pin 27
#define TEST3   _portb,5                ; Pin 26
#define TEST4   _portb,4                ; Pin 25


;;;;;;;;;;;;;;;;;;
; define variables
;;;;;;;;;;;;;;;;;;

#define TIME            0x20
#define Service_Stat    0x21
#define DataInHigh      0x22
#define DataInLow       0x23
#define W_Save          0x24
#define Status_Save     0x25
#define Fsr_Save        0x26
#define FH_TIME         0x27
#define Fsr_Pointer     0x28
#define Temp            0x29

; Data Storage Variables
; Each Variable Followed by It's "Partner" variable ('S' variable)
; Which is used to store the variables contents during transfers...

        CBLOCK 0x45
                Start1, S1, Start2, S2, Start3, S3, RxH, S10
                RxL, S11, RyH, S12, RyL, S13, RzH, S14, RzL, S15
                AxH, S16, AxL, S17, AyH, S18, AyL, S19, AzH, S20
                AzL, S21, Checksum, S24
        ENDC
        ; Checksum @ 64
        ; S24 @ 65
```

```
;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;
;;; Start Code;;;
;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;


        ORG 0x0000      ; Reset Start Location
        GOTO Startup

        ORG 0x0004      ; Interrupt Vector Location
        GOTO Interrupt_Handler


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Special Register Initializations
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

Startup:
; Initialize Option Register
        BSF _rp0
        MOVLW 0xc8
        MOVWF _option
        ; Weak pullups disabled
        ; Rising edge interrupt on INT
        ; WDT has 1:1 prescale

; Initialize Port A
; 1 --> input
; 0 --> output
        BCF     _rp0
        CLRF    _porta
        BSF     _rp0
        MOVLW   0xFF
        MOVWF   _trisa
        ; Make all of them inputs

; Initialize Port C
        BCF     _rp0
        CLRF    _portc
        BSF     _rp0
        MOVLW 0xD7
        MOVWF _trisb
        ; Make all of them inputs
        ; Except RC3 & RC5 (Serial Out Data Port)

; Initialize A/D Status (RA0, RA1 and RA3)
        BCF     _rp0
        MOVLW   0x00
```

```
        MOVWF   _adcon0
        ; A/D converter off

        BSF     _rp0
        MOVLW   0x07
        MOVWF   _adcon1
        ; All A/D port Pins set to Digital I/O mode

; Initialize Interrupts
        BSF _rp0
        MOVLW 0x0C
        MOVWF _pie1
        ; A/D interrupt disabled
        ; SSP interrupt enabled
        ; CCP1 interrupt enabled
        ; USART transmit interrupt NOT enabled

        BCF _rp0
        MOVLW 0x7F              ; subtract 24 clock cycles for delay...
        MOVWF _ccpr1l
        MOVLW 0x00
        MOVWF _ccpr1h
        ; Get value into compare module for
        ; interrupts at 30.2us intervals

        BCF _rp0
        MOVLW 0x00
        MOVWF _pir1
        ; Reset all interrupt flags

        BSF _rp0
        MOVLW 0x00
        MOVWF _pie2
        ; Disable CCP2 interrupt

        BCF _rp0
        MOVLW 0x00
        MOVWF _pir2

        MOVLW 0x40
        MOVWF _intcon
        ; INT disabled
        ; Peripheral Interrupts disabled
        BSF   _intcon,7
        ; Global interrupts enabled



; Initialize Port B
        BCF     _rp0
```

```
        CLRF    _portb
        BSF     _rp0
        MOVLW   0x08
        MOVWF   _trisb
        ; Make all of them outputs
        ; Except RB3 ==> EOC


; Initialize Asynchronous Transmitter
        BSF     _rp0

        ; Transmitter Control Word Stuff
        MOVLW 0x00
        MOVWF _txsta
        ; 8 bit mode
        ; Transmitter disabled
        ; Asynchronous mode
        ; Low speed transmission

        ; Baud Rate Generator Stuff
        MOVLW 0x07
        MOVWF _spbrg
        ; 19.2Kbaud generator (for 10MHz Clock)

        BCF     _rp0            ; Switch to Bank 0
        BCF     _spen           ; disable Port

; Initialize Timer 1
        BCF _rp0                ; Use Bank 0
        MOVLW 0x01
        MOVWF _t1con
        ; Turn Timer 1 on
        ; Prescale 1:1
        ; Use internal clock

; Initialize Timer 2
        BCF _rp0
        MOVLW 0x00
        MOVWF _t2con
        ; Timer 2 off
        ; Postscale 1:1
        ; Prescale 1:1

; Initialize CCP1
        BCF _rp0
        MOVLW 0x0a
        MOVWF _ccp1con
        ; Compare mode --> generate interrupt on compare
```

```
; Initialize CCP2
        BCF _rp0
        MOVLW 0x00
        MOVWF _ccp2con
        ; CCP2 off

; Initialize SSP
        BCF _rp0
        MOVLW 0x32
        MOVWF _sspcon
        ; SSP enabled for SPI mode master.  Fosc / 64
        ; Send on falling edge
        BSF _rp0
        BCF _trisc,3    ; SCK Output
        BCF _trisc,5    ; SDO Output
        BCF _rp0


;;;;;;;;;;;;;;;;;;;;;;;;;;
; Variable initializations
;;;;;;;;;;;;;;;;;;;;;;;;;;

        CLRF Service_Stat
        MOVLW 0xAB
        MOVWF Start1
        MOVLW 0xCD
        MOVWF Start2
        MOVLW 0xEF
        MOVWF Start3

Main_Loop:
        BCF _rp0

        BTFSS Service_Stat,1
        CALL Send_Data
        BTFSS Service_Stat,0
        CALL Read_Data
        GOTO Main_Loop

Send_Data:
        BSF Service_Stat,1
        MOVLW 0x01
        SUBWF TIME,0
        BTFSS _status,2
        RETURN                  ; Wait for TIME == 0
        MOVLW 0x01
        SUBWF FH_TIME,0         ; Wait for 50 Hz timer
        BTFSS _status,2
        GOTO Send2
        ; Time to send new packet
```

```
        MOVLW 0x45              ; Get Set up for indirect addressing
        MOVWF _fsr
        ; Buffer data variables
        BCF _intcon,7           ; Disable ALL interrupts
DoItOver:
        MOVFW _indf
        INCF _fsr,1
        MOVWF _indf
        INCF _fsr,1
        MOVLW 0x63
        SUBWF _fsr,0
        BTFSS _status,2
        GOTO DoItOver
        BCF TEST
        BSF _intcon,7           ; Re-enable interrupts
        Call Compute_Checksum
        CLRF RxH
        CLRF RxL
        CLRF RyH
        CLRF RyL
        CLRF RzH
        CLRF RzL
        CLRF AxH
        CLRF AxL
        CLRF AyH
        CLRF AyL
        CLRF AzH
        CLRF AzL
        MOVLW 0x46
        MOVWF _fsr
        MOVFW _indf             ; Get ready to send 1st byte
        MOVWF _sspbuf
        INCF _fsr,1
        INCF _fsr,1
        MOVFW _fsr              ; Reset Pointer for Interrupt routine
        MOVWF Fsr_Pointer
        RETURN

Send2:
        MOVLW 0x20
        SUBWF FH_TIME,0
        BTFSS _status,2
        RETURN
        ; FH_TIME = .02 seconds (50 Hz)... reset it
        CLRF FH_TIME
        RETURN


Compute_Checksum:
        MOVLW 0x4C
```

```
        MOVWF _fsr
        MOVFW _indf                     ; Start with RxH byte
        MOVWF 0x64                      ; Put it into Checksum Variable...
        INCF _fsr,1
        INCF _fsr,1
Loop:   MOVFW _indf                     ; Add in other Bytes
        ADDWF 0x64,1
        INCF _fsr,1
        INCF _fsr,1
        MOVLW 0x64
        SUBWF _fsr,0
        BTFSS _status,2
        GOTO Loop
        RETURN                          ; All Done... go back


Read_Data:
        MOVLW 0x02
        SUBWF TIME,0
        BTFSS _status,2
        GOTO Read_Data3
        ; TIME == 2 units ... read in Accel Z
        CALL Init_Output
        BCF A0          ; Select Channel 0
        BCF A1
        BCF A2
        BSF WR          ; Write Address to A/D
        NOP
        NOP
        NOP
        BCF CONVST
        BSF CONVST
        CALL Init_Input
        CLRF DataInHigh
        CLRF DataInLow
        ;BTFSC DB0
        ;BSF DataInLow,0
        BTFSC DB1
        BSF DataInLow,0
        BTFSC DB2
        BSF DataInLow,1
        BTFSC DB3
        BSF DataInLow,2
        BTFSC DB4
        BSF DataInLow,3
        BTFSC DB5
        BSF DataInLow,4
        BTFSC DB6
        BSF DataInLow,5
        BTFSC DB7
```

```
        BSF DataInLow,6
        BTFSC DB8
        BSF DataInLow,1
        BTFSC DB9
        BSF DataInHigh,0
        BTFSC DB10
        BSF DataInHigh,1
        BTFSC DB11
        BSF DataInHigh,2
        MOVFW DataInLow
        ADDWF AzL,1
        BTFSC _status,0         ; Carry 1 if necessary
        INCF AzH,1
        MOVFW DataInHigh
        CALL Sign_Extend_W
        ADDWF AzH,1
        RETURN

Read_Data3:
        MOVLW 0x03
        SUBWF TIME,0
        BTFSS _status,2
        GOTO Read_Data4
        ; TIME == 3 units ... Accel Y
        CALL Init_Output
        BSF A0          ; Select Channel 1
        BCF A1
        BCF A2
        BSF WR          ; Write Address to A/D
        NOP
        NOP
        NOP
        BCF CONVST
        BSF CONVST
        CALL Init_Input
        CLRF DataInHigh
        CLRF DataInLow
        ;BTFSC DB0
        ;BSF DataInLow,0
        BTFSC DB1
        BSF DataInLow,0
        BTFSC DB2
        BSF DataInLow,1
        BTFSC DB3
        BSF DataInLow,2
        BTFSC DB4
        BSF DataInLow,3
        BTFSC DB5
        BSF DataInLow,4
        BTFSC DB6
```

```
        BSF DataInLow,5
        BTFSC DB7
        BSF DataInLow,6
        BTFSC DB8
        BSF DataInLow,7
        BTFSC DB9
        BSF DataInHigh,0
        BTFSC DB10
        BSF DataInHigh,1
        BTFSC DB11
        BSF DataInHigh,2
        MOVFW DataInLow
        ADDWF AyL,1
        BTFSC _status,0         ; Carry 1 if necessary
        INCF AyH,1
        MOVFW DataInHigh
        CALL Sign_Extend_W
        ADDWF AyH,1
        RETURN

Read_Data4:
        MOVLW 0x04
        SUBWF TIME,0
        BTFSS _status,2
        GOTO Read_Data5
        ; TIME == 4 units ... read in Accel x
        CALL Init_Output
        BCF A0          ; Select Channel 2
        BSF A1
        BCF A2
        BSF WR          ; Write Address to A/D
        NOP
        NOP
        NOP
        BCF CONVST
        BSF CONVST
        CALL Init_Input
        CLRF DataInHigh
        CLRF DataInLow
        BTFSC DB0
        ;BSF DataInLow,0
        ;BTFSC DB1
        BSF DataInLow,0
        BTFSC DB2
        BSF DataInLow,1
        BTFSC DB3
        BSF DataInLow,2
        BTFSC DB4
        BSF DataInLow,3
        BTFSC DB5
```

```
        BSF DataInLow,4
        BTFSC DB6
        BSF DataInLow,5
        BTFSC DB7
        BSF DataInLow,6
        BTFSC DB8
        BSF DataInLow,7
        BTFSC DB9
        BSF DataInHigh,0
        BTFSC DB10
        BSF DataInHigh,1
        BTFSC DB11
        BSF DataInHigh,2
        MOVFW DataInLow
        ADDWF AxL,1
        BTFSC _status,0         ; Carry 1 if necessary
        INCF AxH,1
        MOVFW DataInHigh
        CALL Sign_Extend_W
        ADDWF AxH,1
        RETURN

Read_Data5:
        MOVLW 0x05
        SUBWF TIME,0
        BTFSS _status,2
        GOTO Read_Data6
        ; TIME == 5 units ... read in Gyro Z
        CALL Init_Output
        BSF A0          ; Select Channel 3
        BSF A1
        BCF A2
        BSF WR          ; Write Address to A/D
        NOP
        NOP
        NOP
        BCF CONVST
        BSF CONVST
        CALL Init_Input
        CLRF DataInHigh
        CLRF DataInLow
        BTFSC DB0
        ;BSF DataInLow,0
        ;BTFSC DB1
        BSF DataInLow,0
        BTFSC DB2
        BSF DataInLow,1
        BTFSC DB3
        BSF DataInLow,2
        BTFSC DB4
```

```
        BSF DataInLow,3
        BTFSC DB5
        BSF DataInLow,4
        BTFSC DB6
        BSF DataInLow,5
        BTFSC DB7
        BSF DataInLow,6
        BTFSC DB8
        BSF DataInLow,7
        BTFSC DB9
        BSF DataInHigh,0
        BTFSC DB10
        BSF DataInHigh,1
        BTFSC DB11
        BSF DataInHigh,2
        MOVFW DataInLow
        ADDWF RzL,1
        BTFSC _status,0         ; Carry 1 if necessary
        INCF RzH,1
        MOVFW DataInHigh
        CALL Sign_Extend_W
        ADDWF RzH,1
        RETURN

Read_Data6:
        MOVLW 0x06
        SUBWF TIME,0
        BTFSS _status,2
        GOTO Read_Data7
        ; TIME == 6 units ... read in Gyro y
        CALL Init_Output
        BCF A0          ; Select Channel 4
        BCF A1
        BSF A2
        BSF WR          ; Write Address to A/D
        NOP
        NOP
        NOP
        BCF CONVST
        BSF CONVST
        CALL Init_Input
        CLRF DataInHigh
        CLRF DataInLow
        BTFSC DB0
        ;BSF DataInLow,0
        ;BTFSC DB1
        BSF DataInLow,0
        BTFSC DB2
        BSF DataInLow,1
        BTFSC DB3
```

```
        BSF DataInLow,2
        BTFSC DB4
        BSF DataInLow,3
        BTFSC DB5
        BSF DataInLow,4
        BTFSC DB6
        BSF DataInLow,5
        BTFSC DB7
        BSF DataInLow,6
        BTFSC DB8
        BSF DataInLow,7
        BTFSC DB9
        BSF DataInHigh,0
        BTFSC DB10
        BSF DataInHigh,1
        BTFSC DB11
        BSF DataInHigh,2
        MOVFW DataInLow
        ADDWF RyL,1
        BTFSC _status,0         ; Carry 1 if necessary
        INCF RyH,1
        MOVFW DataInHigh
        CALL Sign_Extend_W
        ADDWF RyH,1
        RETURN

Read_Data7:
        MOVLW 0x07
        SUBWF TIME,0
        BTFSS _status,2
        GOTO Read_Data8
        ; TIME == 7 units ... read in Gyro x
        CALL Init_Output
        BSF A0          ; Select Channel 5
        BCF A1
        BSF A2
        BSF WR          ; Write Address to A/D
        NOP
        NOP
        NOP
        BCF CONVST
        BSF CONVST
        CALL Init_Input
        CLRF DataInHigh
        CLRF DataInLow
        BTFSC DB0
        ;BSF DataInLow,0
        ;BTFSC DB1
        BSF DataInLow,0
        BTFSC DB2
```

```
        BSF DataInLow,1
        BTFSC DB3
        BSF DataInLow,2
        BTFSC DB4
        BSF DataInLow,3
        BTFSC DB5
        BSF DataInLow,4
        BTFSC DB6
        BSF DataInLow,5
        BTFSC DB7
        BSF DataInLow,6
        BTFSC DB8
        BSF DataInLow,7
        BTFSC DB9
        BSF DataInHigh,0
        BTFSC DB10
        BSF DataInHigh,1
        BTFSC DB11
        BSF DataInHigh,2
        MOVFW DataInLow
        ADDWF RxL,1
        BTFSC _status,0         ; Carry 1 if necessary
        INCF RxH,1
        MOVFW DataInHigh
        CALL Sign_Extend_W
        ADDWF RxH,1
        RETURN

Read_Data8:
        MOVLW 0x14              ; 1.6KHz sampling => 30.2us * 20
        SUBWF TIME,0
        BTFSS _status,2
        RETURN
        INCF FH_TIME,1          ; Increment Fifty Hertz Timer
        CLRF TIME
        RETURN

Sign_Extend_W:
        MOVWF Temp
        BTFSS Temp,2            ; If bit 3 is set, set higher order bits
        RETURN
        BSF Temp,7
        BSF Temp,6
        BSF Temp,5
        BSF Temp,4
        BSF Temp,3
        MOVFW Temp
        RETURN

Init_Output:
```

```
        BSF WR
        BSF RD
        BSF _rp0
        MOVLW 0x00
        MOVWF _trisa
        BCF _rp0
        BCF SWCONV
        BCF SWSTBY
        BSF Format
        BCF WR
        RETURN

Init_Input:
        NOP
        NOP
        NOP
        NOP
        NOP
        NOP
        BSF WR
        BSF RD
        BSF _rp0
        MOVLW 0xFF
        MOVWF _trisa
        BCF _rp0
        BCF RD
        RETURN

Interrupt_Handler:
        MOVWF W_Save            ; Save Key Registers
        MOVFW _status
        MOVWF Status_Save
        MOVFW _fsr
        MOVWF Fsr_Save
        BCF _rp0
        BTFSC _pir1,2           ; Time interrupt?
        CALL Time_int
        BTFSC _pir1,3           ; ssp interrupt?
        CALL Transmit_int
        MOVFW Fsr_Save
        MOVWF _fsr
        MOVFW Status_Save       ; Restore Key Registers
        MOVWF _status
        MOVFW W_Save
        RETFIE

Transmit_int:
        BCF _pir1,3             ; Clear interrupt flag
        MOVFW Fsr_Pointer       ; Make sure _fsr is right!
        MOVWF _fsr
```

```
        MOVLW 0x66
        SUBWF _fsr,0
        BTFSC _status,2         ; done transmitting this packet
        RETURN
        MOVFW _indf             ; Send out next byte
        MOVWF _sspbuf
        INCF _fsr,1
        INCF _fsr,1
        MOVFW _fsr              ; Save _fsr for next time around...
        MOVWF Fsr_Pointer
        RETURN

Time_int:
        CLRF Service_Stat
        BSF TEST2
        BCF _pir1,2
        CLRF _tmr1l
        CLRF _tmr1h
        INCF TIME,1
        CLRWDT
        BCF TEST2
        RETURN
        END
```

A.6 Periph-PIC Code

```
        processor 16c73

#include "16c73.h"

;;;;;;;;;;;;;;;;;;;;;;;;
; Set up Pin Definitions
;;;;;;;;;;;;;;;;;;;;;;;;

; Sonar Definitions
#define Init _portb,1           ; Pin 22
#define Binh _portb,2           ; Pin 23
#define Echo _portb,0           ; Pin 21

; LED
#define LED  _porta,2           ; Pin 4
#define LED2 _porta,3           ; Pin 5
#define TEST _porta,4           ; Pin 6
#define TEST2 _porta,5          ; Pin 7

; Compass Definitions
#define SCLK _portb,6           ; Pin 27
```

```
#define SDO  _portb,7          ; Pin 28
#define SS   _portb,5          ; Pin 26
#define PC   _portb,3          ; Pin 24
#define EOC  _portb,4          ; Pin 25

; Serial
#define SERIAL _portc,6        ; pin 17

; A/D
#define Temperature _porta,0   ; Pin 2
#define Battery     _porta,1   ; Pin 3


;;;;;;;;;;;;;;;;;;
; define variables
;;;;;;;;;;;;;;;;;;


; Global variables
#define TIME                   0x20    ; System Time in ms
#define Temporary              0x21    ; General Purpose Temp. Var.
#define Service_Stat           0x22    ; Which Periph. have been serviced
#define Status_Save            0x23    ; Save Key Registers during int.
#define W_Save                 0x24
#define Fsr_Save               0x25
#define Fsr_Pointer            0x26

; Sonar variables
#define Sonar_TIME             0x30
#define Sonar_Status           0x31
#define Sonar_Counter_Low      0x32
#define Sonar_Counter_High     0x33

#define Battery_TIME           0x35
#define Transmit_TIME          0x36


; Data Storage Variables
; Each Variable Followed by It's "Partner" variable ('S' variable)
; Which is used to store the variables contents during transfers...

        CBLOCK 0x45
                Start_1, S1, Start_2, S2, Start_3, S3, Size, S4
                Status, S5, Sonar_Data_High, S6, Sonar_Data_Low, S7
                Compass_Data_High, S8, Compass_Data_Low, S9, RxH, S10
                RxL, S11, RyH, S12, RyL, S13, RzH, S14, RzL, S15
                AxH, S16, AxL, S17, AyH, S18, AyL, S19, AzH, S20
                AzL, S21, Temperature_Data, S22, Battery_Data, S23
                Checksum, S24
        ENDC
```

```
        ; Checksum @ 0x73
        ; S24 @ 0x74


; IMU variables (All in Bank 1!!)
#define Incoming 0x20
#define Receive_Status 0x21


;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;
;;; Start Code;;;
;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;


        ORG 0x0000      ; Reset Start Location
        GOTO Startup

        ORG 0x0004      ; Interrupt Vector Location
        GOTO Interrupt_Handler


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Special Register Initializations
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

Startup:

; Initialize Port A
; 1 --> input
; 0 --> output
        BCF     _rp0
        CLRF    _porta
        BSF     _rp0
        MOVLW   0xC3
        MOVWF   _trisa
        ; Make all of them inputs
        ; Except RA2 ==> LED
        ; And RA3 ==> LED2
        ; And RA4 ==> TEST
        ; And RA5 ==> TEST2

; Initialize Option Register
        BSF _rp0
        MOVLW 0xc8
        MOVWF _option
        ; Weak pullups disabled
        ; Rising edge interrupt on INT
        ; WDT has 1:1 prescale
```

```
; Initialize Port C
        BCF     _rp0
        CLRF    _portc
        BSF     _rp0
        MOVLW 0xbF
        MOVWF _trisb
        ; Make all of them inputs
        ; Except RC6 ==> SERIAL

; Initialize A/D Status (RA0, RA1 and RA3)
        BCF     _rp0
        MOVLW   0x81
        MOVWF   _adcon0
        ; A/D converter on and channel 0 selected
        ; No conversion in process

        BSF     _rp0
        MOVLW   0x04
        MOVWF   _adcon1
        ; RA0, RA1, RA3 set as analog inputs

; Initialize Interrupts
        BSF _rp0
        MOVLW 0x4C
        MOVWF _pie1
        ; A/D interrupt enabled
        ; SSP interrupt enabled
        ; CCP1 interrupt enabled
        ; USART transmit interrupt NOT enabled

        BCF _rp0
        MOVLW 0x88
        MOVWF _ccpr1l
        MOVLW 0x13
        MOVWF _ccpr1h
        ; Get value into compare module for
        ; interrupts at 1ms intervals

        BCF _rp0
        MOVLW 0x00
        MOVWF _pir1
        ; Reset all interrupt flags

        BSF _rp0
        MOVLW 0x00
        MOVWF _pie2
        ; Disable CCP2 interrupt
```

```
        BCF _rp0
        MOVLW 0x00
        MOVWF _pir2

        MOVLW 0x50
        MOVWF _intcon
        ; INT enabled
        ; Peripheral Interrupts Enabled
        BSF   _intcon,7
        ; Global interrupts enabled



; Initialize Port B
        BCF     _rp0
        CLRF    _portb
        BSF     _rp0
        MOVLW   0x91
        MOVWF   _trisb
        ; Make all of them inputs
        ; Except RB1 ==> INIT
        ;        RB2 ==> BINH
        ;        RB3 ==> PC
        ;        RB5 ==> SS
        ;        RB6 ==> SCLK

; Initialize Asynchronous Transmitter
        BSF     _rp0

        ; Transmitter Control Word Stuff
        MOVLW 0x20
        MOVWF _txsta
        ; 8 bit mode
        ; Transmitter enabled (Notice subtlety with TXIF, pg. 99)
        ; Asynchronous mode
        ; Low speed transmission

        ; Baud Rate Generator Stuff
        MOVLW 0x0F
        MOVWF _spbrg
        ; 19.2Kbaud generator (for 10MHz Clock)

        BCF     _rp0            ; Switch to Bank 0
        BSF     _spen           ; Enable Port

; Initialize Timer 1
        BCF _rp0                ; Use Bank 0
        MOVLW 0x01
        MOVWF _t1con
        ; Turn Timer 1 on
```

```
        ; Prescale 1:1
        ; Use internal clock

; Initialize Timer 2
        BCF _rp0
        MOVLW 0x00
        MOVWF _t2con
        ; Timer 2 off
        ; Postscale 1:1
        ; Prescale 1:1

; Initialize CCP1
        BCF _rp0
        MOVLW 0x0a
        MOVWF _ccp1con
        ; Compare mode --> generate interrupt on compare


; Initialize CCP2
        BCF _rp0
        MOVLW 0x00
        MOVWF _ccp2con
        ; CCP2 off

; Initialize SSP
        BCF _rp0
        MOVLW 0x35
        MOVWF _sspcon
        ; SSP enabled for SPI mode slave.  SS disabled
        ; Receive on Rising edge


;;;;;;;;;;;;;;;;;;;;;;;;;
; Variable initializations
;;;;;;;;;;;;;;;;;;;;;;;;;

        BCF _rp0
        CLRF TIME
        CLRF Battery_TIME
        CLRF Status
        CLRF Sonar_TIME
        BSF _rp0
        CLRF Receive_Status
        BCF _rp0
        MOVLW 0xAB
        MOVWF Start_1
        MOVLW 0xCD
        MOVWF Start_2
        MOVLW 0xEF
        MOVWF Start_3
```

```
        MOVLW 0x15
        MOVWF Size
        MOVLW 0xFF
        MOVWF Service_Stat
        MOVLW 0x76
        MOVWF _fsr
        BSF _rp0
        MOVLW 0x00
        MOVWF RxH
        MOVWF RxL
        MOVWF RyH
        MOVWF RyL
        MOVWF RzH
        MOVWF RzL
        MOVWF AxH
        MOVWF AxL
        MOVWF AyH
        MOVWF AyL
        MOVWF AzH
        MOVWF AzL
        BCF _rp0



;;;;;;;;;;;;;;;;;;;;;;;;
; Output Initializations
;;;;;;;;;;;;;;;;;;;;;;;;

        BCF _rp0

; Initialize Compass Outputs
        BSF SCLK        ; Start Compass Clock High
        BSF SS          ; Start Compass Select High
        BSF PC          ; Start PC High


; Initialize LED
        BSF LED         ; Assume Compass fails until we get data
        BSF LED2        ; Assume IMU fails until we get data

;;;;;;;;;;;;
; Start Loop
;;;;;;;;;;;;

; Main Routine
Main:
        BCF _rp0
        BTFSS Service_Stat,0    ; Skip if compass already serviced this ms
        Call Service_Compass
        BTFSS Service_Stat,1
```

```
        Call Send_Data
        BTFSS Service_Stat,2
        Call Service_Sonar
        BTFSS Service_Stat,3
        Call Service_AD
        GOTO Main

Service_AD:
        BSF Service_Stat,3
        MOVLW 0xFA
        SUBWF Battery_TIME,0
        BTFSS _status,2
        RETURN                  ; Go Back
        ; Battery_TIME == 250 ==> Start New Conversion
        BSF _adcon0,2           ; Start Conversion
        RETURN                  ; Go Back


Service_Sonar:
        BSF Service_Stat,2      ; Sonar has been serviced
        MOVLW 0x01
        SUBWF Sonar_TIME,0
        BTFSS _status,2
        GOTO Sonar2
        ; Sonar_TIME == 1ms  ==> Ping Sonar
        BSF Init
        CLRF Sonar_Status       ; Reset Sonar Status

        MOVWF Sonar_Counter_Low ; Reset Sonar Counter
        MOVWF Sonar_Counter_High
        RETURN

Sonar2:
        MOVLW 0x02
        SUBWF Sonar_TIME,0
        BTFSS _status,2
        GOTO Sonar3
        ; Sonar_TIME == 2ms  ==> Set Binh
        BSF Binh

        RETURN

Sonar3:
        MOVLW 0x1E
        SUBWF Sonar_TIME,0
        BTFSS _status,2
        GOTO Sonar4
        ; Sonar_TIME == 30ms ==> Reset Sonar
        BCF Init
        BCF Binh
```

```
        BTFSC Sonar_Status,0
        RETURN
        ; Save data (As Max Range) if No new data
        BCF _intcon,4           ; Disable Echo interrupt for now
        MOVFW Sonar_Counter_Low
        MOVWF Sonar_Data_Low
        MOVFW Sonar_Counter_High
        MOVWF Sonar_Data_High
        BSF Status,2            ; "fresh" data available
        BSF Sonar_Status,1      ; After 30 ms, an echo is invalid
        BSF _intcon,4           ; Re-enable Echo interrupt
        RETURN

Sonar4:
        MOVLW 0x3C
        SUBWF Sonar_TIME,0
        BTFSS _status,2
        RETURN                  ; Non-Significant time... go back
        ; Sonar_TIME == 60ms  ==> Start Cycle over
        CLRF Sonar_TIME
        RETURN

Send_Data:
        BSF Service_Stat,1
        MOVLW 0x14
        SUBWF Transmit_TIME,0
        BTFSS _status,2
        RETURN                  ; Nothing to do now...
        ; Transmit_TIME == 20ms ==> Start another transmission
        MOVLW 0x45              ; Get Set up for indirect addressing
        MOVWF _fsr
        CLRF Transmit_TIME      ; Reset Transmit Timer
        ; Buffer all data variables:
        BCF _intcon,7           ; Disable ALL interrupts
DoItOver:
        MOVFW _indf
        INCF _fsr,1
        MOVWF _indf
        INCF _fsr,1
        MOVLW 0x73
        SUBWF _fsr,0
        BTFSS _status,2
        GOTO DoItOver
        BSF _intcon,7           ; Re-enable interrupts
        Call Compute_Checksum
        MOVLW 0x46
        MOVWF _fsr
        MOVFW _indf             ; Get ready to send 1st byte
        MOVWF _txreg            ; (Thereby clearing _txif)
        INCF _fsr,1
```

```
        INCF _fsr,1
        MOVFW _fsr              ; Reset Pointer for Interrupt routine
        MOVWF Fsr_Pointer
        BSF _rp0
        BSF _pie1,4             ; Enable TXIE
        BCF _rp0
        BCF Status,0            ; Reset Status bits: IMU "Dead"
        BCF Status,2            ; Sonar "Stale"
        BCF Status,3            ; IMU "Stale"
        BCF Status,4            ; Compass "Stale"
        RETURN

Compute_Checksum:
        MOVLW 0x4C
        MOVWF _fsr
        MOVFW _indf                     ; Start with Size byte
        MOVWF 0x74                      ; Put it into Checksum Variable...
        INCF _fsr,1
        INCF _fsr,1
Loop:   MOVFW _indf                     ; Add in Sonar-Battery Bytes
        ADDWF 0x74,1
        INCF _fsr,1
        INCF _fsr,1
        MOVLW 0x74
        SUBWF _fsr,0
        BTFSS _status,2
        GOTO Loop
        RETURN                          ; All Done... go back

Service_Compass:
        BSF Service_Stat,0
        MOVLW 0x05
        SUBWF TIME,0
        BTFSS _status,2
        GOTO Compass2
        ; TIME == 5  ===>  Start Compass Cycle
        BCF PC
        RETURN

Compass2:
        MOVLW 0x0F
        SUBWF TIME,0
        BTFSS _status,2
        GOTO Compass3
        ; TIME == 15 ==> Reset PC, check EOC
        BSF PC
        BTFSC EOC               ; Error if EOC not clear
        CALL Compass_Err
        RETURN
```

```
Compass3:
        MOVLW 0x64
        SUBWF TIME,0
        BTFSS _status,2
        GOTO Compass4
        ; TIME == 100 ==> EOC high?
        BTFSS EOC                       ; Error
        CALL Compass_Err
        RETURN

Compass4:
        MOVLW 0x6E
        SUBWF TIME,0
        BTFSS _status,2
        GOTO Compass5
        ; TIME == 110 ==> Set SS low
        BTFSC LED
        RETURN                          ; Return if in error state
        BCF SS
        RETURN


Compass5:
        MOVLW 0x79
        SUBWF TIME,0
        BTFSS _status,2
        GOTO Compass6
        ; TIME == 120ms  ==>  Read in Compass Data
        CLRF Compass_Data_Low
        CLRF Compass_Data_High
        ;BTFSC LED
        ;RETURN                 ; Return if in error state
        Call Clock1in           ; 1st 7 Bits Meaningless
        Call Clock1in
        Call Clock1in
        Call Clock1in
        Call Clock1in
        Call Clock1in
        Call Clock1in
        Call Clock1in           ; MSB
        BTFSC SDO
        BSF Compass_Data_High,0
        Call Clock1in
        BTFSC SDO
        BSF Compass_Data_Low,7
        Call Clock1in
        BTFSC SDO
        BSF Compass_Data_Low,6
        Call Clock1in
        BTFSC SDO
```

```
        BSF Compass_Data_Low,5
        Call Clock1in
        BTFSC SDO
        BSF Compass_Data_Low,4
        Call Clock1in
        BTFSC SDO
        BSF Compass_Data_Low,3
        Call Clock1in
        BTFSC SDO
        BSF Compass_Data_Low,2
        Call Clock1in
        BTFSC SDO
        BSF Compass_Data_Low,1
        Call Clock1in
        BTFSC SDO
        BSF Compass_Data_Low,0
        Call Clock1in           ; For good measure ;)
        Call Clock1in
        Call Clock1in
        Call Clock1in


        BSF Status,4            ; "Fresh" Compass data available
        BSF Status,1            ; Compass "alive"
        BCF LED

        BSF SS

        RETURN

Compass6:
        MOVLW 0xFA
        SUBWF TIME,0
        BTFSS _status,2
        RETURN                  ; Non-Significant TIME.... go back
        ; TIME == 250 ==> Restart Cycle
        BCF LED                 ; Reset Compass error indicator
        BSF LED2                ; Set IMU failure indicator
        BSF SS
        CLRF TIME
        RETURN

Clock1in:
        BCF SCLK
        NOP
        NOP
        NOP
        NOP
        NOP
        NOP
```

```
        BSF SCLK
        Return

Interrupt_Handler:
        MOVWF W_Save            ; Save Key Registers
        MOVFW _status
        MOVWF Status_Save
        MOVFW _fsr
        MOVWF Fsr_Save
        BCF _rp0
        BTFSC _pir1,2           ; 1 ms interrupt?
        CALL Milli_int
        BTFSC _intcon,1         ; INT (Echo) interrupt?
        CALL Echo_int
        BTFSC _pir1,6           ; AD interrupt?
        CALL AD_int
        BTFSC _pir1,4           ; USART interrupt?
        CALL Transmit_int
        BTFSC _pir1,3           ; SSP (IMU) Interrupt?
        Call IMU_int
        BCF _rp0
        MOVFW Fsr_Save
        MOVWF _fsr
        MOVFW Status_Save       ; Restore Key Registers
        MOVWF _status
        MOVFW W_Save
        RETFIE

IMU_int:
        BCF _pir1,3
        MOVFW _sspbuf
        BSF Status,0                    ; IMU is ALIVE
        BSF _rp0
        MOVWF Incoming
        MOVLW 0x00
        SUBWF Receive_Status,0          ; Skip ahead if waiting for 0xAB
        BTFSC _status,2
        GOTO IMU2
        MOVLW 0x01
        SUBWF Receive_Status,0          ; Skip ahead if waiting for 0xCD
        BTFSC _status,2
        GOTO IMU3
        MOVLW 0x02
        SUBWF Receive_Status,0          ; Skip ahead if waiting for 0xEF
        BTFSC _status,2
        GOTO IMU4
        ; Waiting for IMU data... get it
        MOVLW 0x03
        SUBWF Receive_Status,0          ; Get RxH
        BTFSC _status,2
```

```
GOTO IMU5
MOVLW 0x04
SUBWF Receive_Status,0          ; Get RxL
BTFSC _status,2
GOTO IMU6
MOVLW 0x05
SUBWF Receive_Status,0          ; Get RyH
BTFSC _status,2
GOTO IMU7
MOVLW 0x06
SUBWF Receive_Status,0          ; Get RyL
BTFSC _status,2
GOTO IMU8
MOVLW 0x07
SUBWF Receive_Status,0          ; Get RzH
BTFSC _status,2
GOTO IMU9
MOVLW 0x08
SUBWF Receive_Status,0          ; Get RzL
BTFSC _status,2
GOTO IMU10
MOVLW 0x09
SUBWF Receive_Status,0          ; Get AxH
BTFSC _status,2
GOTO IMU11
MOVLW 0x0A
SUBWF Receive_Status,0          ; Get AxL
BTFSC _status,2
GOTO IMU12
MOVLW 0x0B
SUBWF Receive_Status,0          ; Get AyH
BTFSC _status,2
GOTO IMU13
MOVLW 0x0C
SUBWF Receive_Status,0          ; Get AyL
BTFSC _status,2
GOTO IMU14
MOVLW 0x0D
SUBWF Receive_Status,0          ; Get AzH
BTFSC _status,2
GOTO IMU15
MOVLW 0x0E
SUBWF Receive_Status,0          ; Get AzL
BTFSC _status,2
GOTO IMU16
CLRF Receive_Status             ; Get ready for fresh packet....
; Copy all variables over to bank 0 locations (Now that
; they're read in)

MOVLW 0x37
```

```
SUBWF RxH,0
BCF _rp0
BTFSS _status,2
BSF _rp0

MOVFW RxH
BCF _rp0
MOVWF RxH

BSF _rp0
MOVFW RxL
BCF _rp0
MOVWF RxL

BSF _rp0
MOVFW RyH
BCF _rp0
MOVWF RyH

BSF _rp0
MOVFW RyL
BCF _rp0
MOVWF RyL

BSF _rp0
MOVFW RzH
BCF _rp0
MOVWF RzH

BSF _rp0
MOVFW RzL
BCF _rp0
MOVWF RzL

BSF _rp0
MOVFW AxH
BCF _rp0
MOVWF AxH

BSF _rp0
MOVFW AxL
BCF _rp0
MOVWF AxL

BSF _rp0
MOVFW AyH
BCF _rp0
MOVWF AyH

BSF _rp0
```

```
        MOVFW AyL
        BCF _rp0
        MOVWF AyL

        BSF _rp0
        MOVFW AzH
        BCF _rp0
        MOVWF AzH

        BSF _rp0
        MOVFW AzL
        BCF _rp0
        MOVWF AzL

        BSF Status,3            ; Set IMU "fresh" bit
        BCF LED2                ; Reset IMU error indicator

        RETURN

IMU2:
        MOVLW 0xAB
        SUBWF Incoming,0
        BTFSS _status,2
        RETURN          ; Not start byte 0xAB... go back and wait for it
        INCF Receive_Status,1
        BCF _rp0
        BSF TEST2
        BSF _rp0
        RETURN

IMU3:
        MOVLW 0xCD
        SUBWF Incoming,0
        BTFSS _status,2
        GOTO Clear_it           ; Bad start sequence... re-frame data
        INCF Receive_Status,1
        RETURN

IMU4:
        MOVLW 0xEF
        SUBWF Incoming,0
        BTFSS _status,2
        GOTO Clear_it           ; Bad start sequence... re-frame data
        INCF Receive_Status,1
        RETURN

Clear_it:
        CLRF Receive_Status
        RETURN
```

```
IMU5:
        MOVFW Incoming
        MOVWF RxH               ; Put it in Temporary RxH (Bank 1)
        INCF Receive_Status,1
        RETURN

IMU6:
        MOVFW Incoming
        MOVWF RxL               ; Put it in Temporary RxL (Bank 1)
        INCF Receive_Status,1
        RETURN

IMU7:
        MOVFW Incoming
        MOVWF RyH               ; Put it in Temporary RyH (Bank 1)
        INCF Receive_Status,1
        RETURN

IMU8:
        MOVFW Incoming
        MOVWF RyL               ; Put it in Temporary RyL (Bank 1)
        INCF Receive_Status,1
        RETURN

IMU9:
        MOVFW Incoming
        MOVWF RzH               ; Put it in Temporary RzH (Bank 1)
        INCF Receive_Status,1
        RETURN

IMU10:
        MOVFW Incoming
        MOVWF RzL               ; Put it in Temporary RzL (Bank 1)
        INCF Receive_Status,1
        RETURN

IMU11:
        MOVFW Incoming
        MOVWF AxH               ; Put it in Temporary AxH (Bank 1)
        INCF Receive_Status,1
        RETURN

IMU12:
        MOVFW Incoming
        MOVWF AxL               ; Put it in Temporary AxL (Bank 1)
        INCF Receive_Status,1
        RETURN

IMU13:
        MOVFW Incoming
```

```
        MOVWF AyH               ; Put it in Temporary AyH (Bank 1)
        INCF Receive_Status,1
        RETURN

IMU14:
        MOVFW Incoming
        MOVWF AyL               ; Put it in Temporary AyL (Bank 1)
        INCF Receive_Status,1
        RETURN

IMU15:
        MOVFW Incoming
        MOVWF AzH               ; Put it in Temporary AzH (Bank 1)
        INCF Receive_Status,1
        RETURN

IMU16:
        MOVFW Incoming
        MOVWF AzL               ; Put it in Temporary AzL (Bank 1)
        INCF Receive_Status,1
        BCF _rp0
        BCF TEST2
        BSF _rp0
        RETURN


Transmit_int:
        MOVFW Fsr_Pointer       ; Make sure _fsr is right!
        MOVWF _fsr
        MOVLW 0x76
        SUBWF _fsr,0
        BTFSC _status,2         ; done transmitting this packet
        GOTO Transmit2
        MOVFW _indf             ; Send out next byte
        MOVWF _txreg
        INCF _fsr,1
        INCF _fsr,1
        MOVFW _fsr              ; Save _fsr for next time around...
        MOVWF Fsr_Pointer
        RETURN

Transmit2:
        BSF _rp0
        BCF _pie1,4
        BCF _rp0
        MOVWF _txreg            ; Put dummy value in txreg to reset _txif
        RETURN

AD_int:
        BCF _pir1,6             ; Clear AD interrupt flag
```

```
        MOVFW _adres            ; Move result to W
        MOVWF Battery_Data      ; Move result to Battery_Data
        CLRF Battery_TIME       ; Clear time base
        RETURN                  ; Done!


Echo_int:
        BCF _intcon,1           ; Clear Interrupt Flag
        ; Add additional time to Sonar Counters
        ; Additional Time is calculated as Tmr1 time / 128
        ; (Right Shifted by 7 bits)
        BTFSC Sonar_Status,1
        RETURN                  ; Forget it if after 30 ms... go back
        CLRF Temporary
        BTFSC _tmr1l,7
        BSF Temporary,0
        BTFSC _tmr1h,0
        BSF Temporary,1
        BTFSC _tmr1h,1
        BSF Temporary,2
        BTFSC _tmr1h,2
        BSF Temporary,3
        BTFSC _tmr1h,3
        BSF Temporary,4
        BTFSC _tmr1h,4
        BSF Temporary,5
        BTFSC _tmr1h,5
        BSF Temporary,6
        BTFSC _tmr1h,6
        BSF Temporary,7
        ; Additional Time is in Temporary... Add it to counter
        MOVFW Temporary
        ADDWF Sonar_Counter_Low,1
        BTFSC _status,0
        INCF Sonar_Counter_High,1
        ; Time has been added... store time
        MOVFW Sonar_Counter_Low
        MOVWF Sonar_Data_Low
        MOVFW Sonar_Counter_High
        MOVWF Sonar_Data_High
        BSF Sonar_Status,0      ; New Sonar Data Available
        BSF Status,2            ; Set Sonar "Fresh" Bit
        RETURN                  ; Jobs Done... go back

Compass_Err:
        BSF LED                 ; Turn LED on
        BCF Status,1            ; Compass "Dead"
        RETURN

Milli_int:
```

```
        BCF _pir1,2             ; Clear Interrupt Flag
        CLRF _tmr1l             ; Clear Timer
        CLRF _tmr1h
        ; Add 1 ms to Sonar Time
        ; Sonar Time is based on 25.6 us Time base
        ; Clock period divided by 64 for 10 MHz external clock
        ; 1 ms == 39 units (25.6us units)
        MOVLW 0x27                ; Put 39 in W register
        ADDWF Sonar_Counter_Low,1 ; Add it to counter
        BTFSC _status,0           ; Check for carry
        INCF Sonar_Counter_High,1 ; Increment high byte if carry occured
        INCF TIME,1             ; Increment Time 1 ms
        INCF Sonar_TIME,1       ; Increment Sonar Time 1 ms
        INCF Battery_TIME,1     ; Increment Battery Time 1 ms
        INCF Transmit_TIME,1    ; Increment Transmit Time 1 ms
        CLRF Service_Stat       ; Reset Service Stat
        CLRWDT                  ; Clear Watch-Dog Timer
        RETURN

        END
```

A.7 PIC Code Header File

```
                NOLIST

;*************************************************************************
;                               PIC16Cxx Header File
;
;       Special Function Register Definitions
;
;
;        By     :       Amar Palacherla
;        Date   :       01 Mar 1993
;
;        Edited By:     Christian Trott
;        Date:          22 Jan 1997
;
;*************************************************************************/


        CBLOCK 0x00
               _indf,_tmr0,_pcl,_status,_fsr,_porta,_portb,_portc,_portd
               _porte,_pclath,_intcon,_pir1,_pir2,_tmr1l,_tmr1h,_t1con
               _tmr2,_t2con,_sspbuf,_sspcon,_ccpr1l,_ccpr1h,_ccp1con,_rcsta
               _txreg,_rcreg,_ccpr2l,_ccpr2h,_ccp2con,_adres,_adcon0
        ENDC
```

```
        CBLOCK 0x00
                _indf,_option,_pcl,_status,_fsr,_trisa,_trisb,_trisc,_trisd
                _trise,_pclath,_intcon,_pie1,_pie2,_pcon,_uu1,_uu2,_uu3
                _pr2,_sspadd,_sspstat,_uu4,_uu5,_uu6,_txsta,_spbrg,_uu7,_uu8
                _uu9,_uu10,_uu11,_adcon1
        ENDC


; Porta Bits
#define         _ra0            _porta,0
#define         _ra1            _porta,1
#define         _ra2            _porta,2
#define         _ra3            _porta,3
#define         _ra4            _porta,4
#define         _ra5            _porta,5

#define         _t0cki  _porta,4

#define         _an0            _porta,0
#define         _an1            _porta,1
#define         _an2            _porta,2
#define         _an3            _porta,3
#define         _an4            _porta,5

#define         _vref           _porta,3
#define         _ss             _porta,5

; Portb bits
#define         _rb0            _portb,0
#define         _rb1            _portb,1
#define         _rb2            _portb,2
#define         _rb3            _portb,3
#define         _rb4            _portb,4
#define         _rb5            _portb,5
#define         _rb6            _portb,6
#define         _rb7            _portb,7

#define         _int            _portb,0

; PortC bits
#define         _rc0            _portc,0
#define         _rc1            _portc,1
#define         _rc2            _portc,2
#define         _rc3            _portc,3
#define         _rc4            _portc,4
#define         _rc5            _portc,5
#define         _rc6            _portc,6
#define         _rc7            _portc,7
```

```
; STATUS Reg Bits
#define         _carry  _status,0
#define         _c              _status,0
#define         _dc             _status,1
#define         _z              _status,2
#define         _pd             _status,3
#define         _to             _status,4
#define         _rp0            _status,5
#define         _rp1            _status,6
#define         _irp            _status,7

; INTCON Reg Bits
#define         _rbif   _intcon,0
#define         _intf   _intcon,1
#define         _rtif   _intcon,2
#define         _rbie   _intcon,3
#define         _inte   _intcon,4
#define         _rtie   _intcon,5

#define         _adie   _intcon,6               ; 16C71
#define         _eeie   _intcon,6               ; 16C84

#define         _gie    _intcon,7


; OPTION Reg

#define         _ps0    _option,0
#define         _ps1    _option,1
#define         _ps2    _option,2
#define         _psa    _option,3
#define         _rte    _option,4
#define         _rts    _option,5
#define         _intedg _option,6
#define         _rbpu   _option,7


; T1CON Reg
#define         _t1ckps1        _t1con,5
#define         _t1ckps0        _t1con,4
#define         _t1oscen        _t1con,3
#define         _t1sync         _t1con,2
#define         _tmr1cs         _t1con,1
#define         _tmr1on         _t1con,0

; TXSTA Reg
#define         _csrc           _txsta,7
#define         _tx9            _txsta,6
#define         _txen           _txsta,5
#define         _sync           _txsta,4
```

```
#define         _brgh           _txsta,2
#define         _trmt           _txsta,1
#define         _tx9d           _txsta,0

; RCSTA Reg
#define         _spen           _rcsta,7
#define         _rx9            _rcsta,6
#define         _sren           _rcsta,5
#define         _cren           _rcsta,4
#define         _ferr           _rcsta,2
#define         _oerr           _rcsta,1
#define         _rx9d           _rcsta,0



_ResetVector    set     0x00
_IntVector      set     0x04
;

W       equ     0
w       equ     0

TRUE    equ     1
FALSE   equ     0

LSB     equ     0
MSB     equ     7

;

        LIST
```

Appendix B

Matlab Simulation Files

B.1 sim.m

: This script file is the main script which was used to run simulations of the inertial system.

```
echo on

Vi = zeros(1,1000);
Vt = 0;
h = .001;

for i = 2:1000,
Vt = Vt + h;
Vi(i) = Vi(i-1) + Vt*h;
end

TIME = 10;
h = .001;

T = linspace(0,10,10/h);
Vi = .08*(sin(2*pi*T) - 2*pi*T.*10.^(-T));
Vi = lowpass(Vi,h,5);

[Vo Clip] = Mech(Vi,.001,0);
V1 = lowpass(Vo,h,5);
randn('seed',sum(100*clock));
Noise = 2*randn(1,length(V1));
Noise = lowpass(Noise,h,10);
Noise = lowpass(Noise,h,10);
%Noise = highpass(Noise,h,990);
%Noise = highpass(Noise,h,990);
%Noise = zeros(1,length(V1));
V2 = V1 + Noise;
sample = .02;
V2 = resample(V2,sample/.001);
temp = 5*ones(1,length(V2));
```

```
V2 = min(V2,temp);
temp = -1*temp;
V2 = max(V2,temp);
V3 = sample*sample*cumsum(cumsum(V2));

Ti = linspace(0,TIME,length(Vi));
T0 = linspace(0,TIME,length(Vo));
T1 = linspace(0,TIME,length(V1));
T2 = linspace(0,TIME,length(V2));
T3 = linspace(0,TIME,length(V3));

figure(2)
subplot(2,1,1)
hold off
plot(Ti,Vi,'b.')
hold on
plot(T3,V3*3/5,'c');
grid on
title('50Hz sampling, Low Frequency Noise')
xlabel('Seconds')
ylabel('Meters')


subplot(2,1,2)
hold off
plot(T0,Vo,'y')
hold on
plot(T1,V1,'g')
plot(T2,V2,'r')
grid on
zoom on
xlabel('Seconds')
ylabel('m/s^2')
```

B.2 lowpass.m

: This file simulates a single pole RC filter

```
function Y = lowpass(Signal, h, Freq)
% Y = lowpass(Signal, h, Freq)
% lowpass filters the signal with timesteps, h,
% by a first order low pass filter with cutoff frequency
% Freq.

N = length(Signal);
TIME = N*h;

R = 1;
C = 1/R/2/pi/Freq;
```

```
V = [];
Vo = 0;
M = h/C;
V = zeros(1,N);


for i=1:N,
I = (Signal(i) - Vo);
Vo = Vo + I*M;
V(i) = Vo;
end


Y = V;
```

B.3 highpass.m

: This file simulates a single pole LR filter.

```
function Y = highpass(Signal, h, Freq)
% Y = highpass(Signal, h, Freq)
% high pass filters the signal with timesteps, h,
% by a first order high pass filter with cutoff frequency
% Freq.


N = length(Signal);
TIME = N*h;

R = 1;
L = R*2*pi*Freq;

V = [];
Vo = 0;
M = h/L;
V = zeros(1,N);
IL = 0;

for i=1:N,
IL = IL + Vo*M;
Vo = Signal(i) - IL;
V(i) = Vo;
end
Y = V;
```

B.4 mech.m

: This file was used to simulate the all aspects of the inertial sensor including its mechanical and electrical properties.

```
function [Vo, Clip] = Mech(Vi,h,NONLINEAR);
% Vo = Mech(Vi, h, NONLINEAR)
% Returns the response to the position input, Vi,
% of the inertial sensor.  The output is the voltage
% of the sensor which is roughly proportional to
% acceleration.  h is the timestep size.
% NONLINEAR == 1 if non-linearities are to be modeled...
% Clip is a vector that is 1 or -1 for each element where
% a non-linearity was encountered.

M = 2 / 2.2;
k = 1000;
d = 10;

R = 1/d;
L = 1/k;
C = M;

TIME = length(Vi)*h;

V = 0;
I = 0;
Vo = zeros(1,TIME/h);
IL = 0;
M1 = h/C;
M2 = h/L;


for i=2:TIME/h,
Vo(i) = Vo(i-1) + (IL + (Vi(i-1)-Vo(i-1))/R) * M1;
IL = IL + (Vi(i-1)-Vo(i-1))*M2;
end

Vo = Diff(Vo,2)/h/h;

R = 1;
C = 1/R/150/pi;
N = length(Vo);
Vo2 = zeros(1,N);

for i=2:N,
Vo2(i) = Vo2(i-1) + (Vo(i-1)-Vo2(i-1))/R*h/C;
end

Vo2 = Vo2*5/3;
```

```
Clip = zeros(1,length(Vo2));
if NONLINEAR == 1
for i=1:length(Vo2),
if Vo2(i) > 5.0
Vo2(i) = 5.0;
Clip(i) = 1;
elseif Vo2(i) < -5.0
Vo2(i) = -5.0;
Clip(i) = -1;
end
end
end

if sum(abs(Clip)) > 0
echo on
Warning_Clipping_Occurred = 1
echo off
end

Vo = Vo2;
```

B.5 resample.m

: This file resamples a vector at a lower rate.

```
function Y = resample(X,r)
% Y = resample(X,r)
% resamples the data in X with 1/r the original rate
% length(Y) = length(X)/r if r evenly divides length(X)


Y = zeros(1,floor(length(X)/r));

for i=1:floor(length(X)/r),
Y(i) = X((i-1)*r+1);
end
```

Appendix C

List of Electronic Components

Component	Manufacturer	Distributor	Part Number	Price
Main Computer	Ampro Computers, Inc.	Trilogic, Inc.	CoreModule/486-II	$1,073.00
Ethernet Card	Ampro Computers, Inc.	Trilogic, Inc.	MiniModule/Ethernet-TP	$275.00
A/D board	Real Time Devices, Inc.	Trilogic, Inc.	DM5406-1	$489.00
I/O Board	Ampro Computers, Inc.	Trilogic, Inc.	MiniModule/SSP	$235.00
IMU (MotionPak)	Systron Donner Inertial Division	—	MP-GDDEQVVV-100	$13,194.00
GPS Antenna	NovAtel Communications, Ltd.	—	511	$365.00
GPS Receiver	NovAtel Communications, Ltd.	—	OEM RT-20	$9,995.00
GPS Antenna (GND)	NovAtel Communications, Ltd.	—	501	$595.00
GPS Receiver (GND)	NovAtel Communications, Ltd.	—	PowerPak 3151-R	$6,660.00
Radio Modem	Proxim, Inc.	New England Digital Distribution, Inc.	Proxlink II	$795.00
Radio Modem	FreeWave Technologies, Inc.	—	DGR-115	$800.00
Sonar Transducer	Polaroid Corporation	—	607281	$18.00
Ranging Board	Polaroid Corporation	—	615077	$23.50
Compass	Precision Navigation, Inc.	—	Vector2X	$49.00
Battery Pack	Periphex, Inc.	—	1800mAh, 14 cells	$107.49
Regulator	Astrodyne Corporation	—	ASD10-XXDXX	$54.00
Custom PC Board	Alberta Printed Circuits, Inc.	—	Custom Fabricated Board	$122.49
Boot ROM	—	T&T Computer Products, Inc.	PN3405	$95.00
Regulator	Power Trends, Inc.	Digi-Key Corporation	78HT205HC	$22.05
BASIC Stamp II	Parallax, Inc.	Digi-Key Corporation	BS2	$49.00
Regulators	—	Digi-Key Corporation	LM340AT-XX-ND	$2.28
Connectors	Amp, Inc.	Digi-Key Corporation	WM12XX-ND	—
PLCC Socket	—	Digi-Key Corporation	A2122-ND	$1.98
LED	—	Digi-Key Corporation	LU604215CT-ND	$1.08
SMT Resistors	—	Digi-Key Corporation	4816P-1-XXX-ND	$1.12
Quad Op-Amp	—	Digi-Key Corporation	LM614IWM-ND	$1.84
Microcontroller	Microchip Corporation	FAI Electronics Corporation	PIC16C73/JW	$15.07

Appendix D

List of Manufacturers and Distributors

1. **Alberta Printed Circuits, LTD.**
   #3, 1112 - 40th Ave. N.E.
   Calgary, AB Canada T2E-ST8
   TEL: (403) 250-3406

---

2. **Ampro Computers, Incorporated**
   990 Almanor Avenue
   Sunnyvale, CA 94086
   TEL: (408) 522-2100
   FAX: (408) 720-1305

---

3. **Astrodyne Corporation**
   300 Myles Standish Blvd.
   Taunton, MA 02780
   TEL: (800) 823-8082

---

4. **Digi-Key Corporation**
   701 Brooks Ave. South
   Thief River Falls, MN 56701-0677
   TEL: (800) 344-4539

---

5. **FAI Electronics Corporation**
   41 East Main Street
   Bolton, MA 01704
   TEL: (508) 779-3111

6. **Freewave Technologies, Inc.**
   1880 South Flatiron Court
   Boulder, CO 80301
   TEL: (303) 444-3862
   FAX: (303) 786-9948

---

7. **GP Batteries, Inc.**
   11235 West Bernardo Court
   San Diego, CA 92127-1638
   TEL: (619) 674-5620
   FAX: (619) 674-7237

---

8. **New England Digital Distribution**
   185C New Boston St.
   Woburn, MA 01801-6203
   TEL: (617) 935-3424

---

9. **NovAtel Communications, Ltd**
   6732 - 8th Street N.E.
   Calgary, Alberta, Canada
   T2E 8M4
   TEL: (800) 280-2242
   FAX: (403) 295-4900

---

10. **Parallax, Inc.**
    3805 Atherton Road, #102
    Rocklin, CA 95765
    Tel: (916) 624-8333
    Fax: (916) 624-8003

---

11. **Periphex, Inc.**
    300 Centre St.
    Holbrook, MA 02343
    TEL: (617) 767-5516

---

12. **Polaroid Corporation**
    5601 Fulton Industrial Blvd, SW
    Atlanta, GA 30378
    TEL: (800) 225-1000

---

13. **Power Trends, Inc.**
    1101 North Raddant Road
    Batavia, IL 60510
    TEL: (708) 406-0900
    FAX: (708) 406-0901

14. **Precision Navigation, inc.**
    1235 Pear Avenue, Suite 111
    Mountain View, CA 94043
    Tel: (415) 962-8777
    Fax: (415) 962-8776

---

15. **Proxim, Inc.**
    295 North Bernardo Ave.
    Mountain View, CA 94043
    TEL: (415) 960-1630
    FAX: (415) 964-5181

---

16. **QNX Software Systems, Ltd.**
    175 Terenece Matthews Crescent
    Kanata, Ontario K2M 1W8
    Canada
    TEL: (613) 591-0931

---

17. **Real Time Devices, Inc.**
    Post Office Box 906
    State College, Pennsylvania 16804
    TEL: (814) 234-8087
    FAX: (814) 234-5218

---

18. **Systron Donner Inertial Division**
    2700 Systron Drive
    Concord, CA 94518-1399
    TEL: (510) 671-6464

---

19. **T&T Computer Products, Inc.**
    P.O. Box 14010
    Tulsa, OK 74159-1010
    TEL: (800) 921-5563

---

20. **Trilogic, Incorporated**
    187 Ballardvale Pk. Suite, A220
    Wilmington, MA 01887
    TEL: (508) 658-3800

Bibliography

[1] Alvin W. Drake: *Fundamentals of Applied Probability Theory* McGraw-Hill Book Company, New York, 1988.

[2] Munther Dahleh, George Verghese, and Mohammed Dahleh: *6.241–Dynamic Systems* Course Notes. MIT, 1996.

[3] *The American Heritage College Dictionary* Third edition. Houghton Mifflin Company, Boston, 1993.

[4] Robert A. Pease: *Troubleshooting Analog Circuits* Butterworth-Heinemann, Boston, 1991.

[5] Paul Horowitz and Winfield Hill: *The Art of Electronics* Cambridge University Press, New York, 1993.

[6] Johnson, DeBitetto, Trott and Bosse: *The 1996 MIT/Boston University/Draper Laboratory Autonomous Helicopter System* The Charles Stark Draper Laboratory, Cambridge, MA, 1996.