Design and Development of a GPS Intermediate Frequency and IMU Data Acquisition System for Advanced Integrated Architectures

Except where reference is made to the work of others, the work described in this thesis is my own or was done in collaboration with my advisory committee. This thesis does not include proprietary or classified information.

Michael Linton Newlin

Certificate of Approval:

John Y. Hung, Co-chair
Professor
Electrical and Computer Engineering

David M. Bevly, Co-chair
Assistant Professor
Mechanical Engineering

Stanley J. Reeves
Professor
Electrical and Computer Engineering

Joe F. Pittman
Interim Dean
Graduate School

Design and Development of a GPS Intermediate Frequency and IMU Data Acquisition System for Advanced Integrated Architectures

Michael Linton Newlin

A Thesis

Submitted to

the Graduate Faculty of

Auburn University

in Partial Fulfillment of the

Requirements for the

Degree of

Master of Science

Auburn, Alabama
December 15, 2006

Design and Development of a GPS Intermediate Frequency and IMU Data Acquisition System for Advanced Integrated Architectures

Michael Linton Newlin

Signature of Author

Date of Graduation

Vita

Michael Linton Newlin was born in Birmingham, AL and raised in Macon, GA. In Macon, Michael graduated from First Presbyterian Day School in 2000. In the fall of that year, he began studying Electrical Engineering at Auburn University, graduating with a Bachelors of Science degree in that field of study in December 2004. Shortly thereafter, he married Lindsay Kay Talbot from Greenville, MS whom he met while in school at Auburn. He began his graduate studies in Electrical Engineering in January of 2005. While pursuing his Masters of Science degree, he worked as a Graduate Research Assistant in the GPS and Vehicle Dynamics Lab at Auburn University where he researched GPS/INS integration algorithms. In September of 2006, he accepted a position with Taranis in Birmingham, Alabama where he and his wife now live.

Thesis Abstract

Design and Development of a GPS Intermediate Frequency and IMU Data Acquisition System for Advanced Integrated Architectures

Michael Linton Newlin

Master of Science, December 15, 2006
(B.E.E./B.W.E., Auburn University, 2004)

170 Typed Pages

Directed by David M. Bevly & John Y. Hung

Advanced levels of GPS and INS integration, including Deeply Integrated and Ultra-Tightly Coupled, have been reported to provide significant gains in anti-jamming capability and reduced susceptibility to loss of GPS signal lock. This thesis provides the design and development of a GPS intermediate frequency and IMU data acquisition system that can be used for the implementation of these advanced GPS and INS integrated algorithms. Details of the design of the system are covered, including GPS chip set selection, signal conditioning, and data collection. The system is capable of synchronizing IMU measurement collection to the collection of digitized GPS intermediate frequency in real-time using a derivative of the GPS IF sampling clock to sample the IMU measurements. This synchronization is necessary for proper integrated algorithm implementation. The system was installed on a moving platform and the results of the experiment are shown using a GPS software receiver for the validation of the digitized GPS IF and the plotting of the IMU measurements.

Acknowledgments

First and foremost I thank Jesus Christ for His grace and mercy. Second, I thank my wife, Lindsay, for her patience and support. I also thank my parents, Linton and Janice, and my brother, Chris, for their encouragement.

I wish to express my gratitude to my advisers, Dr. David M. Bevly and Dr. John Y. Hung, for their technical and financial support, and to the members of the GPS and Vehicle Dynamics Lab for their eagerness to openly share their knowledge and experience. Also, I would like to thank Dr. Demoz Gebre-Egziabher and Dr. Dennis Akos for providing much needed technical support.

Finally, I thank the U.S. Army Aviation and Missile Research, Development, and Engineering Center (AMRDEC) for their financial and technical support of the research presented here.

Style manual or journal used IEEE Transactions on Aerospace and Electronic Systems (together with the style known as "aums"). Bibliography follows van Leunen's *A Handbook for Scholars.*

Computer software used The document preparation package TEX (specifically LATEX) together with Bibtex and the departmental style-file `aums.sty`.

Table of Contents

List of Figures

Chapter 1

Introduction, Motivation, and Contributions

1.1 Introduction

Beginning in the early 1970's, the United States Department of Defense (DOD) in conjunction with the United States Air Force (USAF), under the United States Joint Programs Office (JPO), began the development of a system referred to at the time as NAVSTAR, but today is more commonly known as the Global Positioning System (GPS). GPS is a satellite based time-of-arrival (TOA) radio frequency navigation system based on a trilateration of the distances estimated from a user to satellites based on the time it takes a satellite's signal to reach a user. Inherent in the satellite signal design is low received signal power, rendering the GPS signal buried in noise upon reception. Furthermore, in high-jamming environments, whether intentional or unintentional, and during periods of high platform dynamics, loss of lock in signal tracking is commonly encountered.

Efforts have been made in improving the level of precision and robustness in GPS based navigation, particularly in the area of integration with Inertial Measurement Unit (IMU) based Inertial Navigation Systems (INS). Currently low levels of GPS/INS integration are being implemented with higher levels still in stages of development and testing. The development of Deeply Integrated GPS/INS is at the leading edge of this technology. In order to test and validate Deeply Integrated (DI) GPS/INS algorithms, a system capable of producing the necessary level of GPS data is needed. This thesis

covers the development and testing of such a system, capable of providing raw GPS IF data synchronized with IMU data for DI research and design.

1.1.1 Global Positioning System Overview

The Global Positioning System (GPS) can be divided into three major segments: the control segment (CS), the space segment (SS), and the user segment (US). Each segment has its own distinct purpose, with all three segments working together to provide a space-based all-weather radio-frequency time-of-arrival global positioning system capable of providing a user's position at any point on Earth. Following is a brief discussion of each of the three segments, including the tasks required of each.

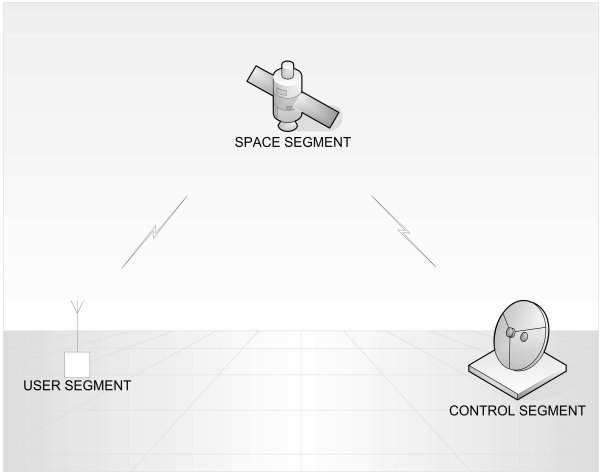

Figure 1.1: Three Segments

The control segment consists of five fixed location earth-based monitor stations. They are located at Colorado Springs, Ascension Island, Diego Garcia, Kwajalein, and Hawaii. The control segment has four major objectives [7]: (1) Maintain each of the satellites in its proper orbit through infrequent small commanded maneuvers, (2) make

corrections and adjustments to the satellite clocks and payload as needed, (3) track the GPS satellites and generate and upload the navigation data to each of the GPS satellites, and (4) command major relocations in the event of satellite failure to minimize impact. In short, the control segment's job is to maintain the satellites in the space segment and provide them with the necessary data and corrections to provide accurate location and timing information to the user segment.

In order to best provide a minimum of four operational satellites at any point on earth at any given time, a 24 satellite constellation was selected (although there are 29 Block II/IIA/IIR/IIR-M satellites today) [8]. These 24 satellites make up the space segment and rest on six orbital planes with four satellites on each plane, as seen in Figure 1.2. The purpose of each of the 24 satellites is to transmit the information essential to the operation of GPS.

Figure 1.2: Figure of Satellite Orbital Planes (Courtesy of [1])

Each satellite transmits at two frequencies, the L1 band and the L2 band, located at 1.57542 GHz and 1.2276 GHz on the radio frequency spectrum, respectively. The satellites' signals include the satellites' time and well as the satellites' position, both provided as accurately as possible. The signal is modulated by a code, specific to each satellite, and a carrier, in order to propagate the signal the necessary distance. Error corrections are also included and will be discussed in detail later in the chapter.

The user segment is the final segment of GPS, consisting of all users, military and civilian, commercial and individual, who receive, process, and utilize the GPS signal. Applications range from shipping, to recreation, to marine navigation, to travel, etc. However, despite the application, the same general process of demodulating the code and carrier from the navigation data must be employed, be it in a hardware or software GPS receiver.

1.1.2 Inertial Navigation System Overview

Inertial measurement units (IMU's) are often used in conjunction with GPS navigation data. Historically, IMU's were used long before GPS was available and were therefore the only available device used for determining user location beyond use of a compass and dead reckoning. When GPS became available, it was suggested early on that the two could be combined to provide a more accurate and capable navigation solution than ever before [9].

The most common IMU consists of six accelerometers and six gyrometers. This gives the IMU six degrees of freedom (6-DOF). That is, the acceleration in the x, y, and z positions can be measured, as well as the roll, pitch, and yaw rates. These measurements are illustrated in Figure 1.3.

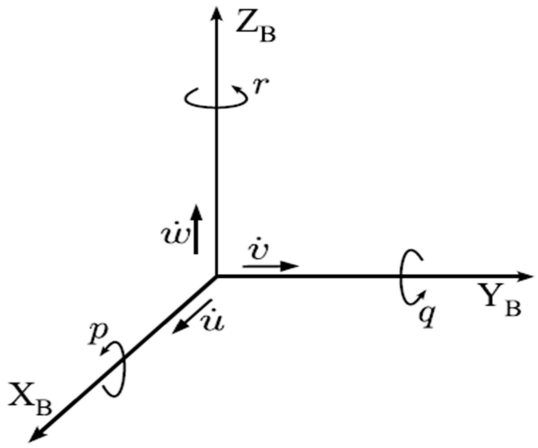

Figure 1.3: 6-DOF IMU Model (from [2])

The advantages of a GPS navigation solution combined with an inertial navigation system can be seen by the defining characteristics of the two systems. GPS is capable of giving a user position within meters of accuracy anywhere on the globe. Although the user's path may be a best fit estimate due to satellite line-of-sight outages and loss of signal lock, a rough path trajectory is available that is corrected upon reacquisition of satellite data signal.

An IMU is capable of providing precise movement of the user well below the meter level of accuracy provided by GPS [10]. Furthermore, there are no IMU outages. An IMU will continue to provide position data regardless of the environment and its conditions. However, integration of IMU rates leads to inherent drifts which must be corrected periodically. Combing the capabilities of the two systems, the drawbacks of each system are counteracted by the complementary system, providing very precise and robust user position.

1.1.3 Integration Architectures Overview

As previously mentioned, different levels of integration algorithms have been developed. The three most common of these are Loosely Coupled (LC), Tightly Coupled (TC), and Deeply Integrated (DI) or Ultra-Tightly Coupled (UTC). The method used for classifying an algorithm's level of integration is based on the point at which the GPS signal data is combined with IMU measurements. A brief discussion of these three levels and their defining characteristics follows in order of depth of integration, the lowest being first.

Loosely Coupled

The first level of integration developed and widely implemented by the general public is termed Loosely Coupled GPS/INS Integration. Loosely Coupled, sometimes further defined as either Uncoupled or Loosely Coupled, combines the navigation solution provided by GPS (position, velocity, and time) and the navigation solution provided by INS (position, velocity, attitude) [11]. This approach implements a Kalman filter (or extended Kalman filter, EKF) in order to update the navigation solution provided by the IMU with the navigation solution provided by GPS. The Kalman Filter estimation procedure can found in [12] and throughout navigation estimation literature and will be referenced throughout this writing. While this is an improvement over previous forms of navigation, there is no aiding in the tracking of the GPS signal as the two systems operate independently, leaving the GPS susceptible to RF jamming and signal loss and requiring a minimum of four satellites for full operability [13, 14].

Tightly Coupled

The first major enhancement in terms of integration is Tightly Coupled GPS/INS Integration, and, as of this writing, the most prevalent of the GPS/Inertial (GPSI) systems [15]. A Tightly Coupled system combines the pseudoranges and pseudorange rates (also known as delta pseudoranges or delta-ranges) from a GPS navigation processor with an INS navigation solution, using an extended Kalman Filter. This approach also provides aiding to the GPS tracking loops, thus increasing the performance.

Deeply Integrated/Ultra-Tightly Coupled

The most advanced level of integration proposed as of the time of this writing is Deeply Integrated GPS/INS Integration (DI), with similar architectures referred to as Ultra-Tightly Coupled (UTC). The exact methods for DI are not as widely agreed upon as are the methods for Loosely Coupled and Tightly Coupled. Currently there are three patents recognized as implementing what is agreed upon as the basis of DI/UTC algorithms. They are held by The Charles Stark Draper Laboratory, Inc. [3], Raytheon Company [4], and The Aerospace Corporation [5], which refers to its particular method as Ultra-Tight. Each of these methods will be discussed later in the chapter.

1.1.4 Implementing Deeply Integrated GPS/INS

As mentioned before, the level of the integration is defined by the construct of the GPS signal used, i.e. the GPS inputs to the integration algorithm. According to the Draper patent, "The measurements to be used are referred to herein as 'raw' measurements, consisting of a[t] least in-phase (I) and quadrature (Q) data obtained from one or more correlators operating on received GPS signals" [3]. The Aerospace

Corporation patent reads, "In the ultra-tight method, the quadrature I and Q samples from a correlator are sent to a Kalman prefilter that processes the samples..." [5]. The Raytheon patent reads, "In accordance with the present invention, a procedure is provided which uses integrate and dump techniques operating on I and Q data to directly produce residuals which are input to a Kalman filter used to correct navigation errors without the use of intermediate tracking loops," [4].

As can be seen from the literature, the inputs to a Deeply Integrated GPS/INS algorithm are the I and Q samples taken from a GPS correlator and the raw measurements from an IMU. These I and Q samples can be acquired using a software correlator which will be discussed in detail in Chapter 4 of this thesis. The GPS samples necessary for producing the I and Q samples in a software based correlator are the sampled GPS intermediate frequency (IF) and can be represented by a digitized 2-bit binary stream corresponding to sign and magnitude of the IF.

The process of attaining these samples is discussed in Chapter 3. Once these samples are gathered and synchronized with raw IMU measurements, they can be sent to a DI algorithm for testing and validation. The acquisition of this combined data set is discussed in Chapter 6.

1.2 Research Motivation

As previously mentioned, it has been claimed that Deeply Integrated GPS/INS provides a number of advantages beyond those of Loosely and Tightly Coupled [16, 17, 18]. The claims include "...reliable and accurate GPS-base navigation solutions in high interference and dynamic environments, at a performance level which has heretofore be unobtainable," [3]. More specifically, Gustafson claims that "...improvements of up to 15

dB in broadband anti-jam capability can be expected relative to current gain scheduled tightly-coupled tracking loops for some scenarios," [18]. Furthermore, "...the navigation system architecture and processes employed yield significant improvements in navigation system performance, both in code tracking and reacquisition, and in carrier tracking and reacquisition. The improvements are particularly significant at low signal/noise ratios, where conventional approaches are especially susceptible to loss of code lock or carrier lock," [3].

With such purported advantages as improved anti-jamming capabilities, improved signal lock and reacquisition, and a minimal number of satellites necessary for operation, clearly such a system would prove to be far more robust than any other systems currently used. However, while these claims are somewhat generally accepted, they have not yet been independently verified as to whether or not they do possess such advantages in performance [14]. In addition, there are few, if any, provided publications providing a relevant amount of mathematical detail pertaining to the algorithms necessary for DI [19, 20]. Therefore, it is desirable to pursue the study and development of such a system and its algorithms. If such a system can be realized and implemented, the advantages are clear. Currently there are no such systems available for the public market. Entities mentioned previously are currently developing and testing their own systems, but these will not be available to the general public for some time [21].

1.3 Thesis Contributions

This thesis presents the design of a low-cost test bed capable of providing the data measurements necessary for advanced integration architectures. This design uses a GPS front-end chip set embedded in a commercially available GPS system by tapping into

the board. Although there are preassembled commercial off the shelf (COTS) systems that are capable of producing the digitized raw GPS IF, these systems are limited by the amount of detail given in the internal operation of one of these systems. Furthermore it is desirable to have the ability to control certain system parameters that are not available in a COTS unit. Therefore a system with these capabilities was designed and built for this thesis. These abilities are discussed in Chapter 6.

The system designed for this thesis is capable of providing digitized GPS IF samples, both directly in a two bit-stream manner, or spread across 32 channels for future real-time algorithm capabilities. Furthermore the system produces a GPS synchronized 1 pulse-per-second (PPS) clock as well as a GPS ADC sampling-clock-synchronized 100 Hz clock. The system is capable of receiving analog IMU measurements, sampling them, and synchronously logging them with GPS data in real-time, i.e. without the need for post-process correlation and synchronization, as is a common method. Furthermore, the 100 Hz clock may be used by a digital INS whose data may be logged using a separate data acquisition system.

The advantages of this system includes the ability to run advanced integration algorithms both in post-process and in real time. Additionally, the system developed in this thesis costs a great deal less than a preassembled COTS unit which is furthermore incapable of direct IMU synchronization.

1.4 Thesis Outline

Chapter 2 will provide a more detailed discussion of GPS, its segments and its signals, followed by a discussion of inertial navigation systems (INS) and their significance to GPS/IMU synchronization necessary for advanced algorithms. The descriptions and

modeling of GPS errors will be included, along with the effects of these errors on DI and how the advancing levels of integration seek to minimize them. The chapter will conclude with the different levels at which these two systems can be integrated. Chapter 3 discusses in detail the design of a system that is capable of producing the digitized raw GPS IF samples necessary for Deeply Integrated GPS/INS Integration (DI) and how it can be synchronized with INS data. In Chapter 4, the process of taking a sampled GPS signal and acquiring the navigation solution through tracking and navigation algorithms will be covered. The calculations for producing the in-phase and quadrature (I and Q) signals will be presented here in order to provide insight for the validation of the GPS IF Acquisition process. The integration of data from the GPS IF Acquisition System and IMU data presented in Chapter 3 will be covered in Chapter 5 along with the data collection, analysis, and validation of the system. Finally, in Chapter 6, future work and proposed future research will be discussed with the intention of continuing efforts in the study of Deeply Integrated GPS/INS using the system developed in this thesis.

Chapter 2

GPS and INS Integration Background

2.1 Global Positioning System Background

Before further discussion concerning the specifics of the GPS Intermediate Frequency and IMU Data Acquisition System, a more detailed review of GPS and the levels of integration is necessary. First is a review of GPS: its satellites, signal structure, navigation data, satellite position information, pseudoranges, and user position calculations. The chapter concludes with detailed descriptions of the three levels of integration employing GPS and INS briefly mentioned before.

2.1.1 Satellites

As previously stated there are a total of 29 Block II/IIA/IIR/IIR-M satellites making up today's GPS satellite constellation. Historically, each satellite broadcasts a signal at two frequencies, L1 and L2, however, additional broadcasting frequencies are being developed. Each signal must contain both the precise clock time as well as the satellite's position in the form of navigation data so that the satellite's time and position may be determined by the user. The signals broadcasted by each satellite will be discussed in the following section.

Each satellite is periodically updated by the Control Segment with satellite clock and position corrections. The most important aspect of a satellite's performance is the stability of the satellite's clocks. Each satellite carries up to four atomic clocks, which are necessary for the required level of accuracy due to error build-up and update periodicity.

The 29 satellites rest on six orbital planes, designated A through F, with up to six slots per plane. This gives a total of 36 possible satellites on the current constellation pattern. Each plane is inclined with respect to the equator by 55 degrees.

Block II consists of nine satellites, space vehicle numbers (SVN) 13 through 21, one of which (II 9) is still in service. These were the first full scale operational satellites designed to provide 14 days of operation without contact from the Control Segment. The Block II satellites were developed by Rockwell International and launched from February 1989 through October 1990.

Block IIA consists of 19 satellites, SVNs 22 through 40, 14 of which are still in service. These were designed to provide 180 days of operation without contact from the Control Segment. The Block IIA satellites were also developed by Rockwell International and launched from November 1990 through November 1997.

The Block II and IIA satellites were designed to be in service for 7.3 years. Each satellite in these blocks contains four atomic clocks: two Cesium (Cs) and two Rubidium (Rb), and has the capabilities of Selective Availability (SA) and Anti-Spoofing (A-S).

Block IIR consists of 13 satellites, SVNs 41 through 62, only 12 of which are in service due to the unsuccessful launching of SVN 42. These were designed to be the operational replenishment satellites, and currently make up nearly half of the GPS satellite constellation, replacing most of Block II and a portion of Block IIA. Block IIR satellites were designed to provide 14 days of operation without contact from the Control Segment and up to 180 days of operation when operating in the autonomous navigation (AUTONAV) mode.

The Block IIR satellites have been designed to be in service for 7.8 years. Each satellite in this block contains three Rubidium (Rb) clocks and has the SA and A-S

capabilities. The Block IIR satellites were developed by Lockheed Martin and launched from January 1997 through November 2004.

The newest block of satellites in the process of being designed and launched at the time of writing is the IIR-M. The only satellite in this block in current orbit is the IIR-M1, SVN 53. It was launched in September of 2005. This satellite is unique in that it is the first to broadcast the two new military signals (M-code) and a second civil signal (L2C).

All GPS satellite vehicle information is available at the United States Naval Observatory website [8].

2.1.2 Signal Structure

There are three main components to the GPS signal structure: the carrier signal, the code signal, and the navigation data. In order for the carrier signal to traverse the minimum distance of 20,162.61 km between satellite and user, a high frequency must used [7]. The carrier signal is then modulated with the navigation data and the code signal, particular to each satellite, using Binary Phase-Shift Keying.

Binary Phase-Shift Keying (BPSK) is considered one of the simplest methods for transmitting digital information [22]. For example, x(t) is a digital sequence defined as:

$$x(t) = \begin{cases} 1, \text{if b[n]} = 0 \\ -1, \text{if b[n]} = 1 \end{cases} \tag{2.1}$$

For nT $\leq t < (n+1)T$.

When a digital sequence similar to x(t) is used to modulate a sinusoidal carrier wave, the resulting high-frequency wave has the form

$$m(t) = x(t)\cos(\omega t) \tag{2.2}$$

In the case of GPS, there are two carrier signals which all satellites broadcast: L1 at 1575.42 MHz (154 x 10.23 MHz) and L2 at 1227.6 MHz (120 x 10.23 MHz). L1 is modulated with the navigation data and two code signals: the C/A code and the P code. The C/A code, or coarse/acquisition code, is a civilian access code. The P code, or precision code, is more accurate but is not available to civilian users. The two signals, C/A and P, are transmitted orthogonally on L1, that is, they are 90 degrees out of phase to minimize interference. L2 is modulated with the navigation data and only the P code. Therefore L1 is the carrier frequency of interest to this thesis as the C/A code demodulation scheme is publicly available.

In the near future, two new civilian access signals will be broadcast and available for use. The first of the two is the L2C code, broadcast at the L2 frequency. As of this writing, one satellite with L2C capability is already in orbit. The second new civilian access signal will be known as L5 and will be broadcast at 1176.45 MHz. Both new signals are scheduled to be available for initial operability by 2012 and at full operational capability around 2015 [23]. This will provide the general public with increased position accuracy due to techniques that employ more than one carrier frequency. As is discussed in Chapter 5, ionospheric propagation delay errors can be nearly eliminated using dual-frequency techniques.

In order for GPS to function properly, all satellites must be continuously broadcasting their own unique signal at the same frequency band of the radio frequency spectrum and the user must be able to receive all viewable signals without interference. This

requires a multiple access (MA) scheme. The scheme chosen for GPS is code division multiple access (CDMA). Time division multiple access (TDMA), frequency division multiple access (FDMA), are space division multiple access (SDMA) are other available methods, however, they do not meet the requirements of GPS signal broadcasting techniques due to their inherent characteristics. A signal is periodically interrupted in time using TDMA, multiple frequency bands are required for FDMA, and a smart antenna for each user would be required on every satellite for SDMA - a physical impossibility.

CDMA allows for multiple signals to be broadcast at the same frequency while creating little interference from signal to signal. In order for CDMA to operate correctly, each satellite must modulate its signal with a coding scheme that is unique to the satellite and has very low cross-correlation to the other codes. Once the signal is received by the user, the code can then be demodulated and the satellite can be determined.

As previously stated, there are currently two coding schemes for GPS signal modulation, the C/A-code and the P-code. Both C/A and P codes are created using the same technique: two pseudorandom noise sequences generators' outputs are multiplied together. The delay between the two generators corresponds to the particular satellite for which the code is being created. Thus these types of codes are referred to as 'product codes' [24].

In the case of P-code, the product of the two random sequences is so long that given its clock rate of 10.23 MHz, the P-code's period could last for a little over 38 weeks. However, each week the code generator is reset so that the P-code has a period of exactly one week. Furthermore, the P-code is published in the GPS-ICD [25] and is available to the general public. Therefore the P-code can be replaced by the Y-code which is only available to those authorized by the United States government. This is

referred to as the AS, or antispoof, mode of operation, and makes the Y-encrypted-P code, or the P(Y) code, unusable to the public.

The C/A-code is similarly created using two pseudorandom noise sequence generators. However, the codes used for C/A are taken from a family of sequences known as the Gold codes, and are created by taking the product of two 1023 bit pseudorandom noise codes. The Gold codes are created at 1.023 MHz giving the C/A-code a period of 1 ms. The Gold codes were chosen for their low cross-correlation with other codes and a low auto-correlation with themselves.

In current development is another coding scheme intended for use by the military, referred to as the M-code. It will be broadcast on the existing L1 and L2 carrier frequencies and will be designed so that it has no interference effects on the current coding schemes. The M-code will possess several advantages over the current P-code, such as lower susceptibility to jamming through increased signal power and a Binary Offset Carrier (BOC) modulation scheme in which the signal is multiplied by the rectangular subcarrier frequency, dividing the signal into two parts located on either side of the carrier frequency [26]. Like the P-code, the M-code will be available only to United States military authorized users.

The modulation of a carrier wave with a square-pulse signal wave, such as the code signal or the navigation data, results in what is called a spread-spectrum transmission. The initial signal power of the carrier wave is spread out over a frequency band proportional to the frequency of the code signal and navigation data. In Figure 2.1, an illustration of a 1.575 GHz carrier wave modulated by a pseudorandom 1.023 MHz square-pulse wave binary code sequence and further modulated by 50 Hz square-pulse wave binary navigation data is shown.

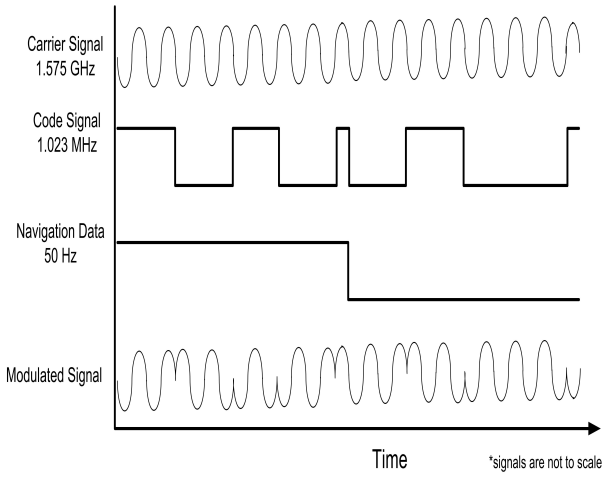

Figure 2.1: Signal Modulation

The resulting signal is shown to be spread across a frequency band proportional to the frequency of the binary data. This method of using a known sequence at a high data rate with a data message of a relatively low data rate is known as direct-sequence spread-spectrum transmission [27, 28].

Direct-sequence spread-spectrum transmission is used in GPS because it makes it possible to recover the carrier signal along with the precise timing necessary to GPS. The carrier signal recovery is necessary because it allows for precision differential delay and Doppler measurements that are capable of accuracies within 1 percent of a carrier wavelength, or roughly 2 mm ($3x10^8$ / 1575 MHz x 0.01) [24].

2.1.3 Navigation Data

The navigation data broadcast from each satellite contains the information necessary for determining user position, user velocity, and UTC (Universal Coordinated Time). The information consists of data on satellite health status, ephemeris (satellite

position and velocity), clock bias parameters, and an almanac giving ephemeris data on all satellites at a lower level of precision. The following is a discussion of the navigation data message and how this information is extracted.

The navigation data is transmitted at 50 bits per second, or 50 bps, which is modulated with the C/A and P codes and carrier as discussed earlier. Once the C/A code, or P code, are demodulated from the signal, along with the carrier wave, the data stream is formatted into 30-bit words. The words are grouped into 10-word subframes and the subframes are grouped into 5-subframe pages, or frames. A superframe, or Master Frame, consists of 25 frames, or pages, and has a message duration of 12 minutes, 30 seconds. That means it takes 12 and a half minutes for one full navigation message to be sent. This structuring is illustrated in Figure 2.2.

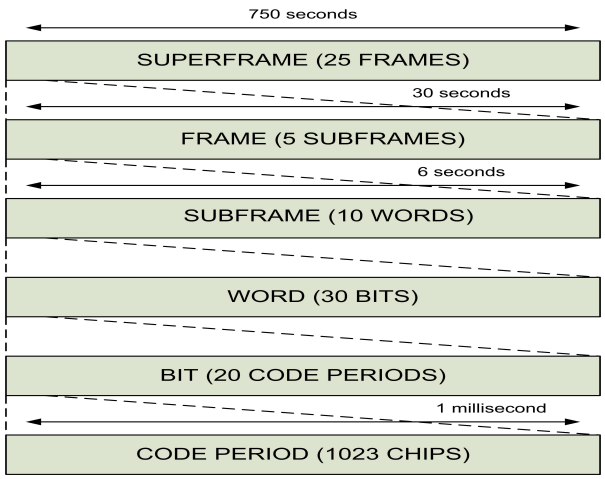

Figure 2.2: Navigation Data Structure

The first two words of each subframe are used for synchronization, hand-over-word (HOW), and C/A code time ambiguity removal. The rest of subframe 1 provides clock corrections, subframes two and three contain ephemeris data used to determine satellite

position, and subframes four and five contains almanac data, that is, information pertaining to all satellites, including ephemeris data and clock corrections [29]. An illustration of subframe data structure is shown in Figure 2.3.

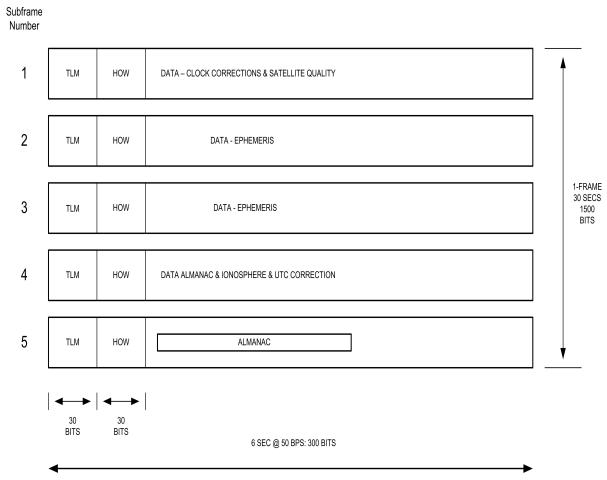

Figure 2.3: Subframe Structure

2.1.4 Satellite Position, Pseudorange, and User Position

The orbit of a GPS satellite can roughly be defined using Kepler's three laws of planetary motion. However, there are forces that inhibit the satellite from strictly following these laws. The three contributing forces are (1) a non-central gravitational force field, (2) gravitational fields of the Sun and Moon, and (3) solar radiation pressure [30]. In order to determine a satellite's position at any given point in time, an accurate model of the satellite's orbit is necessary.

The corrections made to this perturbed Keplerian orbit are provided in the satellite ephemeris of the navigation data. There are thirteen ephemeris parameters included in each navigation message [30]. These parameters are used to compute an accurate

estimation of the satellite orbit. Once the position vector of the satellite is calculated a subsequent transformation to the Earth-Centered Earth Fixed (ECEF) coordinate system is necessary for further calculations. This discussion is continued in Chapter 4.

In addition to satellite position, a measurement called the pseudorange must be known. The pseudorange is defined as the transit time multiplied by the speed of light. Transit time of a signal is defined as the difference between signal reception time, as measured by the receiver clock, and the transmission time of the satellite, as marked on the signal [30]. The transit time is physically measured as the amount of time shift required to align the locally generated C/A-code replica created by the receiver with the C/A-code embedded in the received signal from the satellite.

A model for the measured pseudorange is given in Equation (2.3),

$$\rho^{(k)}(t) = r^{(k)}(t, t-\tau) + c[\delta t_u(t) - \delta t^{(k)}(t-\tau)] + I^{(k)}(t) + T^{(k)}(t) + \epsilon_\rho^{(k)}(t) \tag{2.3}$$

In the equation, $r(t, t-\tau)$ is the actual distance between the receiver antenna at signal reception time, t, and the satellite antenna at signal transmission time, $(t-\tau)$, and is modeled in Equation (2.4),

$$r^{(k)} = \sqrt{(x^{(k)} - x)^2 + (y^{(k)} - y)^2 + (z^{(k)} - z)^2} \tag{2.4}$$

The variables $\delta t_{u(t)}$ and $\delta t_{u(t-\tau)}$ are the receiver and satellite clock offsets, relative to GPS time; I and T are the ionospheric and tropospheric propagation delays; and $\epsilon_\rho(t)$ accounts for modeling errors (e.g., satellite clock modeling error and orbit prediction error) and unmodeled effects (e.g., receiver noise/interference and multipath) [30].

Using Equation (2.4), Equation (2.3) can now be rewritten as:

$$\rho_c^{(k)} = \sqrt{(x^{(k)} - x)^2 + (y^{(k)} - y)^2 + (z^{(k)} - z)^2} + b + \epsilon_\rho^{(k)} \tag{2.5}$$

It can be seen that there are four unknowns: x-, y-, and z- positions, and clock bias, b. Therefore four equations are needed, requiring four satellites. At this point, root mean squares (RMS) methods can be used to determine user position. A more detailed discussion of calculating user position and velocity measurements will be covered in Chapter 4. However, the previous discussion should give a good understanding of how GPS navigation data is transmitted to the user and how it may be used to determine position.

2.1.5 GPS Error Modeling

This section discusses common errors found in GPS and the modeling techniques used to minimize the errors' effects. Previous chapters have covered GPS signal properties and their reception, acquisition, and tracking. There are error sources corresponding to each step of the process and at each level of integration. Defining these errors, studying them, and determining methods in minimizing their effects is essential to furthering the abilities of GPS.

First, error sources common to all GPS equipment will be defined as well as the methods for modeling them. In following sections, the advantages and disadvantages of Deeply Integrated (DI) and Ultra-Tightly Coupled (UTC) GPS/INS concerning error sources will be discussed. It has been discussed in previous chapters that DI and UTC architectures are capable of minimizing or canceling out completely several error sources.

However, no system is completely immune to error and so remaining DI and UTC error sources must be investigated.

GPS Errors and Modeling Techniques

The first error source comes from the satellite clock. Although the satellites' atomic clocks are highly accurate, there is still a possible offset as large as 1ms, which translates to 300-km in pseudorange error [31]. The Control Segment determines the satellite clock error and transmits parameters in the navigation data of the satellite signal that are used by the receiver to adjust the satellite clock parameter. The residual satellite clock error can be modeled by averaging residuals from several satellites with different ages of data (AODs), that is, time since last Control Segment update. Based on data presented in [32], the nominal 1-sigma clock error in 2004 was 1.1 meter.

The second error source comes from errors in the ephemeris data. As covered in Chapter 4, the ephemeris data is uploaded by the Control Segment and used by the receiver to determine a satellite's position in reference to time. Because these ephemeris parameters are determined by the control segment using a curve fit analysis, the path predicted may be different from the true path. Also based on data presented in [32], the 1-sigma error due to ephemeris prediction errors is on the order of 0.8 meters.

Because the GPS satellite vehicle and the receiver lie at different gravitational potentials and because they are both moving with respect to the Earth-centered inertial (ECI) frame, both Einstein's general and special theories of relativity are employed to model the effects of this third error source [33]. Both of these effects are reduced by adjusting the satellite clock's frequency to 10.22999999543 MHz. Once the satellite is in orbit, the clock will appear to the user to be at 10.23 MHz, due to relativistic effects.

An error that must be compensated for by the user is the result of the satellite's relative velocity and gravitational potential at perigee and apogee, making the clock to appear to run slower or faster, respectively [33]. In order to minimize this effect, Equation (2.6) is used [31].

$$\Delta t_r = F e \sqrt{a} \sin E_k \tag{2.6}$$

where:

F = -4.442807633 x 10^{-10} s/$m^{1/2}$

e = satellite orbital eccentricity

a = semimajor axis of the satellite orbit

E_k = eccentric anomaly of the satellite orbit

The values for the above parameters can be found in Chapter 4, in Table 4.1 and Equations (4.34) - (4.37). According to [34], this relativistic delay effect can have a deviation as much as 70 nanoseconds, or 21 meters.

Another relativistic error source is known as the Sagnac effect and is covered in detail in [35]. In summary, the Sagnac effect is a result of the finite rotation a receiver clock experiences in reference to the ECI frame. If left alone, this effect can cause up to 30 meters of error. A common method of correcting this error is iteratively calculating the time of reception such that the ECEF and ECI coordinate frames seem instantaneously fixed for each iteration.

The fourth error source is atmospheric delay and can be defined by two separate delays: ionospheric delay and tropospheric delay. Ionospheric delay is a result of the

ionosphere located between roughly 70 km and 700 km from the Earth's surface [36]. The ionosphere has interesting effects on radio wave propagation, and in the case of GPS, causes the signal information to be delayed and the carrier phase to advance. Consequently, the errors' magnitudes are the same for the carrier-phase measurement and the pseudorange, but the sign of the errors are opposite.

Because ionospheric delay is frequency dependent, dual-frequency receivers that utilize L1 and L2 signals can estimate the ionospheric delay on L1, as seen in Equation (2.7).

$$\Delta S_{iono,corr_{L1}} = \left(\frac{f_{L2}^2}{f_{L2}^2 - f_{L1}^2} \right) (\rho_{L1} - \rho_{L2}) \tag{2.7}$$

Unfortunately, L2 is not available to the average user and so the Klobichar model can be used, which removes roughly 50% of the delay [31]. In the near future, the L5 and L2C civilian frequency bands will be broadcasted, negating the need for such estimation models.

Tropospheric delay is caused by the lower part of the atmosphere known as the troposphere. The troposphere is susceptible to local temperature, pressure, and humidity, which affects the tropospheric delay. Various methods have been developed in order to estimate the delay. However, an optimal method for estimating tropospheric delay is still debated because of the dependency on local conditions. One method uses values from predefined tables to interpolate local values for parameters such as pressure, temperature, and humidity, based on receiver latitude [37]. These parameters are substituted into models that are used to predict the local tropospheric delay. Another method uses meteorological sensors to determine these parameters.

The next error source, known as receiver noise and resolution errors, is comprised of several different error sources but can be categorized into two types: tracking loop errors and RF interference errors. Tracking loop errors are a result of phase-lock loop (PLL), frequency-lock loop (FLL), and delay-lock loop (DLL)/coarse-acquisition (C/A) loop errors. PLL errors are a result of phase jitter and dynamic stress error. Phase jitter induced error can be offset by increasing the order of the PLL and narrowing the bandwidth. Dynamic stress error cannot be counteracted unless the PLL is externally aided using knowledge of the LOS dynamic stress. These are computationally intensive methods and are discussed in more detail in [38]. FLL errors are similarly a result of frequency jitter and dynamic stress error, although the FLL is less susceptible to dynamic stress, making it a desirable back-up to a PLL, as mentioned before.

Code, or C/A, tracking loops typically use a delay-lock loop (DLL), which are affected by thermal noise, range error jitter, and dynamic stress error. The performance of a DLL can be increased by widening the tracking bandwidth or reducing the correlator spacing. However, if the correlator spacing is reduced, the front-end bandwidth must be increased, which may lead to increased RF interference. DLL jitter can be reduced by lowering the filter noise bandwidth and carrier-aided code techniques are commonly used to significantly reduce the dynamic stress from the code tracking loop. Reducing tracking loop errors is the subject of the next section.

RF interference errors consist of antenna multipath and jamming. Antenna multipath is a result of the presence of reflective sources propagating delayed signals in addition to a direct line-of-sight (LOS) broadcast from satellite to user. Whenever a reflected signal is received by an antenna, it is processed along with the direct LOS signal

and all other reflected signals received by the antenna. There are several techniques used to deal with multipathing [39, 40, 41, 42, 43], but will not be discussed in this thesis.

RF interference known as jamming comes in two forms: intentional and unintentional. Intentional interference (commonly referred to as just "jamming", when making the distinction from unintentional interference) is the result of a replica GPS signal being broadcast with the intent of disrupting the reception of a true GPS signal. Two common forms of intentional jamming are smart spoofers and repeat-back spoofers. There are also intentional narrowband and wideband jammers that do not spoof. Smart spoofers attempt to have the target receiver lock-on to the jamming signal, giving the target erroneous data. The repeat-back spoofers literally repeat back, or rebroadcast, the signals its antennas receive in order to mislead the target receiver. Directional null-steering antenna techniques are used to counter these jamming signals by determining the source's location.

Unintentional jamming is simply the presence of RF signals in the same bands as GPS signals, i.e. L1, L2, and, in the near future, L5. While the L1 band is relatively clear, the L2 and L5 bands have a large number of possible unintentional jamming signals [44].

Advantages of Advanced Integration Algorithms

According to sources referenced in Chapter 2, advanced integration algorithms such as Deeply Integrated and Ultra-Tightly Coupled GPS/INS provide the highest levels of performance in terms of accuracy and robustness when compared to simpler architectures, i.e. Loosely Coupled and Tightly Coupled. This is due primarily to the Doppler

aided GPS receiver tracking loops, and more specifically, carrier tracking loops. Typically, carrier tracking loops (Costas PLL's) require a wider bandwidth to avoid loss of lock in situations of high dynamics and a narrower bandwidth in order to minimize phase jitter. Chapter 4 discusses that the Doppler aiding is derived from inertial navigation system (INS) measurements and is used to remove the Doppler caused by receiver platform dynamics. This aiding allows a reduction in the carrier tracking loop bandwidth, reducing phase jitter, and, as discussed in the previous section, results in improved tracking performance, dynamic range, and accuracy of measurements.

In code tracking loops, where DLL's are typically used, more accurate Doppler and pseudorange measurements are available due to improvements in dealing with dynamic stress and noise rejection [45]. These errors were mentioned in the previous section. In the method of Deeply Integrated (DI) GPS/INS proposed by Gustafson and Dowdle [18], the DI filter sets the code tracking loop bandwidth instantaneously at each measurement as a function of expected rms delay error, the jammer/signal power ratio (J/S), and true delay error. The value for J/S is measured at the antenna and the true delay error measurement is determined by the correlator outputs. According to Gustafson and Dowdle, code tracking loops implemented in this DI architecture are capable of anti-jam (AJ) improvements of up to 15 db compared to fixed-gain tightly-coupled tracking loops, a significant improvement.

Error Modeling Conclusion

Despite using advanced architectures and tracking methods to minimize the errors present in Loosely Coupled and Tightly Coupled systems' tracking loops, several of the error sources presented in the chapter are still present, such as satellite clock error,

ephemeris, relativistic delay, and atmospheric errors. Most of these error sources can be offset or minimized using the discussed methods. However, there remain sources of error that are currently considered acceptable due to limitations in hardware, processing, or understanding.

Even with the significant advancements that have been made in GPS tracking error management, the methods are not yet optimal and future improvements will continue to be made. In [45], limitations in processing power are bypassed using a federated Kalman filter architecture to improve Ultra-Tightly Coupled GPS/INS Integration. The limitation concerning the use of individual scalar DLL's for each visible satellite for code tracking is improved upon by using what are known as vector delay-lock loops (VDLL) [45, 2]. VDLL improve performance by reducing noise in the tracking channels making them less likely to fall below threshold and thus lose lock. This expands upon the concept discussed in Ultra-Tightly Coupled Integration [5].

The future of GPS-based navigation research will inevitably focus on tracking loop architectures improving accuracy by minimizing resolution error. The methods of Deeply Integrated and Ultra-Tightly Coupled Integration are currently at the forefront of this navigation generation.

2.1.6 Satellite Based Augmentation Systems

In addition to the standard GPS service, there exists the Satellite Based Augmentation System (SBAS) which is capable of providing increased precision to civilian users by transmitting a Differential GPS (DGPS) signal. There are several SBAS's around the world: the Wide Area Augmentation System (WAAS) in the United States, the Canada Wide Area Augmentation System (CWAAS) in Canada, the European Geostationary

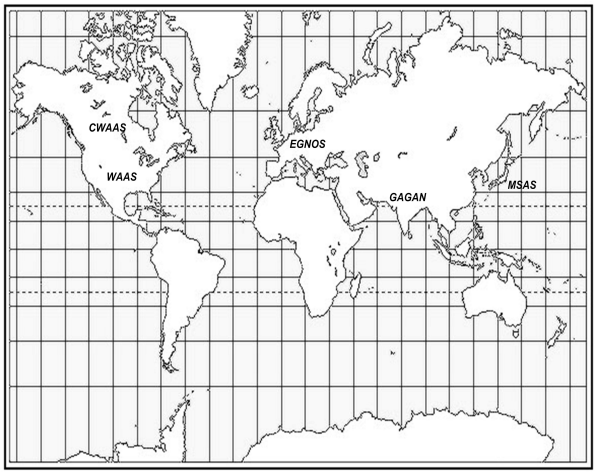

Figure 2.4: Satellite Based Augmentation Systems

Navigation Overlay System (EGNOS) in Europe, the Multifunctional Transport Satellite (MTSAT)-based Augmentation System (MSAS) in Japan and Southeast Asia, and the GPS and (GEO) Augmented Navigation (GAGAN) system in India [46].

Each of these systems uses its own terminology when referring to the different segments of its SBAS. However, each system generally uses the same four major components: monitoring receivers, central processing facilities, satellite uplink facilities, and one or more geostationary satellites and/or ground-based broadcast antennas. In the US-based WAAS, these components are referred to as wide-area reference stations (WRS's), wide area master stations (WMS's), ground Earth stations (GES's), and Inmarsat-3 satellites. At the time of this writing there are 25 WRS's, two WMS's, three GES's, and two geostationary satellites.

In the WAAS system, the 25 WRS's determine the corrections based on the GPS signal they receive and their known position. These corrections are sent to the WMS's which feed the corrected signal to the geostationary satellites. The DGPS corrections are

broadcasted and are available to all WAAS-enabled receivers, providing an accuracy level much greater than previously available. Currently, WAAS is capable of precision of less than three meters in the vertical and less than two meters in the horizontal according to a study done by the WAAS Group at the William J. Hughes Technical Center in October of 2005 [47].

2.1.7 International Positioning Systems

Although not of particular interest to this work, it is important to mention the existence of systems similar to the Global Position System (GPS). In Europe, the European Union began development of the GALILEO system. It is designed to act independently of the GPS system while having full interoperability capabilities. GALILEO is scheduled for full operational capability in 2008.

Similar to the United States, in the mid-1970's, Russia began the developed of the Global Navigation Satellite System (GLONASS). Like the United States' system, GLONASS is designed to offer both military and civilian support. At the time of this writing, the Russian government is working with the United States and the European Union to ensure interoperability among the three systems.

China is developing the BeiDou system. Unlike GPS, GALILEO, and GLONASS, the BeiDou system uses a radio determination satellite service (RDSS), that is, the satellites and users use two-way communication to determine user position. Furthermore, this system uses geostationary satellites placed over China, limiting the system to users in and around China.

2.2 Inertial Measurement Unit Integration

This section briefly covers the relevance of inertial measurement units (IMU's) in GPS/INS integration along with the classification and performance of various grades of IMU's. GPS/INS integration requires proper acquisition of inertial measurement unit (IMU) data which will be integrated into the system, using methods discussed in detail in the following section.

Inertial measurement units are classified based on precision, or amount of error that accumulates over time. There are four major classifications of inertial measurement units: navigational, tactical, automotive, and consumer [48]. Navigational units, being the most precise, are also the most expensive, and usually range around $100,000US a unit. These are followed, in order of descending precision and cost, with tactical at around $20,000US, automotive at around $1,000US, and consumer costing less than $1,000US a unit.

The error previously mentioned results from several error sources such as constant bias, moving bias, random error, gravitational error, cross-coupling error, and sensor scale factor error. The modeling of these errors is not discussed in this work, however, they can be referred to in [10]. A table listing the typical characteristics used to determine classification of IMU's is seen in Tables 2.4 and 2.5.

Accelerometer	Attribute	Units	Specification
Consumer	Random Walk	$g/\sqrt{Hz}$	0.003
	Bias Time Constant	sec	100
	Bias Variation	g	2.4 x 10^{-3}
Automotive	Random Walk	$g/\sqrt{Hz}$	0.001
	Bias Time Constant	sec	100
	Bias Variation	g	1.2 x 10^{-3}
Tactical	Random Walk	$g/\sqrt{Hz}$	0.0005
	Bias Time Constant	sec	60
	Bias Variation	g	50 x 10^{-5}

Table 2.1: Error Quantities for Various Grade Accelerometers [48]

Rate Gyro	Attribute	Units	Specification
Consumer	Random Walk	$^\circ/sec/\sqrt{Hz}$	0.05
	Bias Time Constant	sec	300
	Bias Variation	$^\circ/hr$	360
Automotive	Random Walk	$^\circ/sec/\sqrt{Hz}$	0.05
	Bias Time Constant	sec	300
	Bias Variation	$^\circ/hr$	180
Tactical	Random Walk	$^\circ/sec/\sqrt{Hz}$	0.0017
	Bias Time Constant	sec	100
	Bias Variation	$^\circ/hr$	0.35

Table 2.2: Error Quantities for Various Grade Gyroscopes [48]

The two IMU's used for this thesis are the Bosch and the HG-1700, which fall into the automotive and tactical categories, respectively.

2.3 Integration Architectures Background

As was briefly discussed in a previous section, there are three levels of GPS/INS integration that are generally accepted with regards to level of integration and implementation. The level of integration can be seen as guided by what GPS navigation solution elements are required by the integration algorithm, while implementation is guided by the algorithm's method. In following is a more in depth discussion of the levels of integration and implementation of Loosely Coupled GPS/INS Integration, Tightly Coupled GPS/INS Integration, and Deeply Integrated GPS/INS.

2.3.1 Loosely Coupled

Introduced in the early 1980s, Loosely Coupled GPS/INS Integration was the first level of integration developed and used by the vast majority of the navigation community [49]. In Loosely Coupled, the GPS and INS systems are completely independent of each other, that is, their navigation solutions are independently calculated within each system and then combined, typically, with a Kalman Filter. This type of Loosely Coupled Integration is sometimes referred to as Uncoupled because of the nature of its integration and to distinguish it from a Loosely Coupled architecture in which feedback loops are used [15].

The Kalman Filter uses a state space model to estimate the state of a system using knowledge of the system's dynamics along with the statistical behavior of the system's errors. In the general case of Loosely Coupled Integration, a GPS unit with

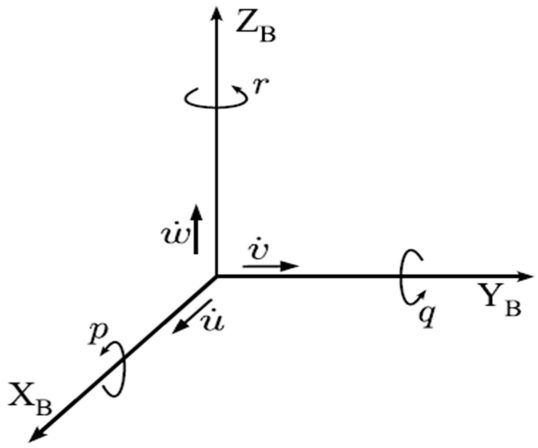

Figure 2.5: IMU Measurements (from [2])

an 8-state Kalman Filter-based Navigation Processor producing Position, Velocity, and Time measurements is combined with an IMU producing translational acceleration and angular velocity measurements. A diagram depicting these IMU measurements can be seen in Figure 2.5. These measurements are combined in a navigation processor with a 15- to 18-state Kalman Filter. An illustration of Loosely Coupled Integration is seen in Figure 2.6.

The eight states associated with the GPS unit are position in x, y, and z (x, y, z), velocity in x, y, and z (δx, δy, δz), GPS receiver clock bias (t_u), and clock drift ($\dot{t}_u$). The states are updated using pseudorange and pseudorange rate measurements from within the GPS receiver. Furthermore the GPS navigation processor is aided by the velocity vector of the integrated solution.

The IMU measurements of translational acceleration and angular velocity are fed to the integrated navigation processor where an IMU navigation solution of Position,

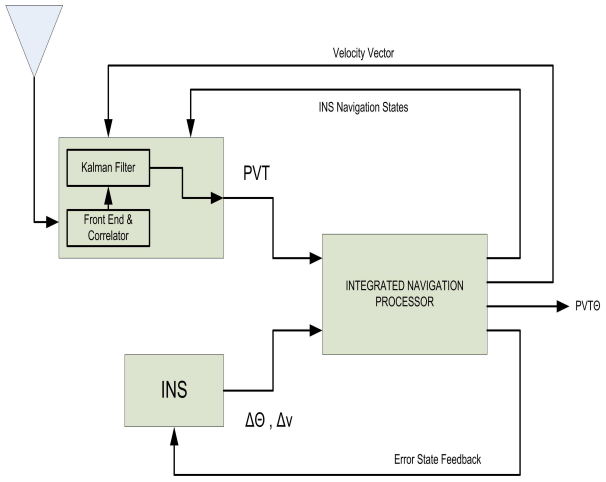

Figure 2.6: Loosely Coupled Integration

Velocity, and Attitude is calculated by integrating the rate measurements. It is here in the integrated navigation processor where the two systems' measurements are combined in a Kalman Filter to produce an integrated solution. The states of this Kalman Filter include, but are not limited to, platform position (x, y, z), platform velocity ($\dot{x}$, $\dot{y}$, $\dot{z}$), roll (θ), pitch (ϕ), yaw (ψ), roll rate (p), pitch rate (q), yaw rate (r), and generally some IMU error terms. In applications involving sensors beyond IMU and GPS, the navigation processor's Kalman Filter may require a much larger number of states [50].

As can be seen in Figure 2.6, there can be up to three types of feedback aiding associated with Loosely Coupled architectures, hence the distinction between Uncoupled and Loosely Coupled architectures. The most common is the aiding of the GPS navigation processor with the velocity vector produced by the integrated navigation processor. The velocity measurements are used to propagate the navigation solution state forward by providing a much more accurate estimate of the platform's true acceleration. The second form of aiding uses inertial measurement-based states of position, velocity, and

attitude to aid the GPS navigation processor. The third feedback loop is the aiding of the INS with an error-state such that the position and velocity solutions as well as the alignment may be reset [11].

Although it is relatively simple to implement, Loosely Coupled integration has severe drawbacks. Because of the independence of GPS and INS in the Loosely Coupled Integration, errors from both systems can more easily propagate through the system due to the lack of any sort of aiding (that is, in the Uncoupled architecture). This causes instability in the overall system and limits its capabilities and overall robustness. Furthermore full operation of GPS is required at all times, that is, if GPS drops below four satellites, or has an outage for any reason, the systems reverts back to a stand alone INS until GPS is restored. However, according to [11], in the case of one or more satellites outages, a Loosely Coupled system is not required to drop GPS aiding entirely, even if the GPS measurements are heavily degraded. This is left to the discretion of the system architect.

2.3.2 Tightly Coupled

The level of integration most widely used today, according to [15], is known as Tightly Coupled GPS/INS Integration as illustrated in Figure 2.7. In contrast to Loosely Coupled, Tightly Coupled negates the need for a Kalman Filter in the GPS navigation processor by sending the pseudorange and pseudorange rate data for the GPS receiver directly to the integrated navigation processor. As discussed earlier in the section, pseudorange and pseudorange rates are a direct result of the GPS receiver's correlators, and the number of measurements is dependent on the number of satellites in view.

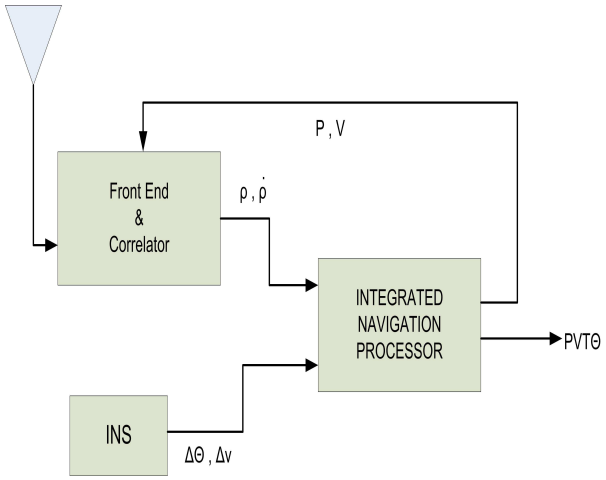

Figure 2.7: Tightly Coupled Integration

The Tightly Coupled system's integrated navigation processor uses navigation equations to first translate the IMU translational accelerations and angular velocities into ECEF coordinates and then combines the Position, Velocity, and Attitude with the GPS pseudorange and pseudorange rate measurements in a Kalman Filter. The Kalman Filter produces an estimated error state vector to correct the output of the IMU navigation equations, as seen in Figure 2.8.

The calculation of the pseudorange and pseudorange rate measurements in a Tightly Coupled system is aided by the Position and Velocity measurements of the integrated navigation processor. In the simplest of terms, the Position and Velocity measurements are first translated into LOS measurement corrections ($pseudorange_{INS}$, $pseudorangerate_{INS}$). These pseudorange and pseudorange rate corrections are then used to aid the code and carrier tracking loops, respectively.

Aiding of the code loop is much easier than carrier loop aiding due to the high accuracy requirements of using a phase lock loop (PLL) on the small GPS carrier signal

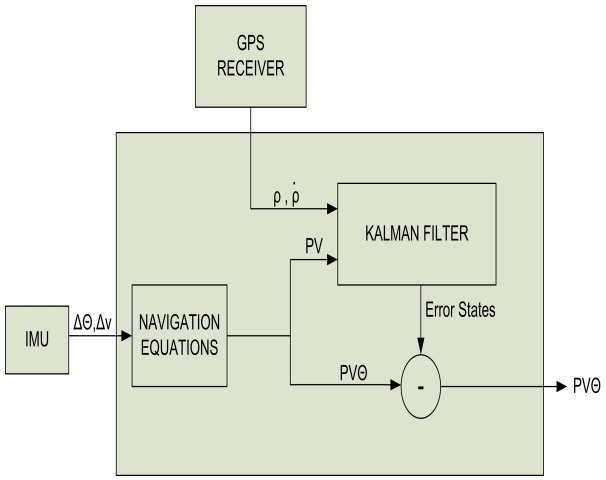

Figure 2.8: Tightly Coupled Integration Processor

wavelength [15]. Typically, the code loop is continuously aided while the carrier loop aiding is much more susceptible to error. In application, a PLL is commonly used to track the carrier and a frequency lock loop (FLL) is used as a back-up.

As a result of this aiding process and the removal of the Kalman Filter in the GPS navigation processor, Tightly Coupled GPS/INS offers an increased level of anti-jamming capability by allowing the tracking loops to use a tighter bandwidth which helps mitigate jamming. Furthermore, as the levels of integration recede further back into the hardware, processing errors are reduced, e.g., errors that would otherwise result from the GPS Kalman Filter's estimation are eliminated.

2.3.3 Deeply Integrated/Ultra-Tightly Coupled

The most advanced level of GPS/INS integration proposed as of the time of this writing is Deeply Integrated GPS/INS, or alternatively referred to as Ultra-Tightly Coupled. A diagram of a Deeply Integrated system is shown in Figure 2.9. This level of

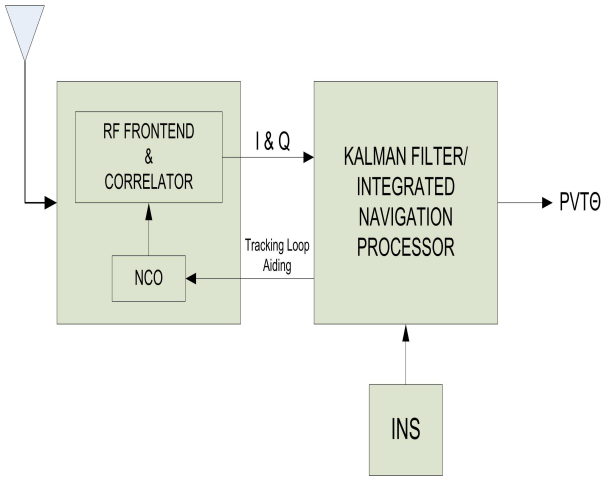

Figure 2.9: Deeply Integrated

integration was first proposed as advantageous to former levels by [51]. Copps suggested, "...that a best estimator for position, velocity, and clock phase and frequency error should be equivalent to a best estimator for the set of satellite code phases and Dopplers," [15]. Copps' architecture was not necessarily dependent on INS measurements, however it did introduce vector tracking to GPS. Vector tracking removes the traditional independent tracking loops from a GPS receiver and replaces them with tracking loops that are coupled through their response to the platform's position, velocity, and clock errors. A continued discussion on vector tracking can be found in [2] and [52].

Although Copps' work was a breakthrough in GPS tracking, there remained a need for common methods to integrate GPS and IMU measurements at this level. As mentioned earlier in the chapter, at the time of writing, there are three patents recognized as commercial and governmental pursuits in implementing what is agreed upon as the basis of Deeply Integrated algorithms. They are held by The Charles Stark Draper Laboratory, Inc. [3], Raytheon Company [4], and The Aerospace Corporation [5]. The

recognized inventor of the DI algorithm, Donald Gustafson, is responsible for publishing the algorithm's only recognized performance documents [17, 16, 18]. It is important to note that at the time of this writing, sources containing mathematical explanation of these algorithms are limited [19, 20].

The method authored by Gustafson is presented only in its patent [3]. The patent covers a method employing a carrier numerically controlled oscillator and a code numerically controlled oscillator (NCO) used to aid the Doppler removal process and the bank of correlators, respectively, as seen in Figure 2.10. The Doppler removal process provides the discrete-time sampled in-phase and quadrature signals, I_s and Q_s, which are then processed by the bank of correlators, providing the discrete-time 50 Hz in-phase and quadrature signals, I_{50} and Q_{50}. These in-phase and quadrature signals along with a signal-to-noise ratio (SNR) estimate and a propagation update vector, whose construction is aided by the IMU, are sent to the measurement updating filter. This measurement update filter generates an updated state vector containing a navigation solution, such as platform position and velocity, clock errors, atmospheric propagation delays, and satellite errors. Furthermore, the measurement update filter generates a covariance matrix of estimation errors, required to estimate the quality of the navigation solution.

The inertial measurements, such as the vector change in velocity of the platform and the vector change in attitude angles, are applied to the delayed update state vector to provide a propagated state vector. The propagated error covariance matrix is generated in a manner similar to the state vector. According to [18], this method has been implemented and, when compared to current tightly-coupled algorithms, improvements of up to 15 dB in broadband anti-jam capability can be expected.

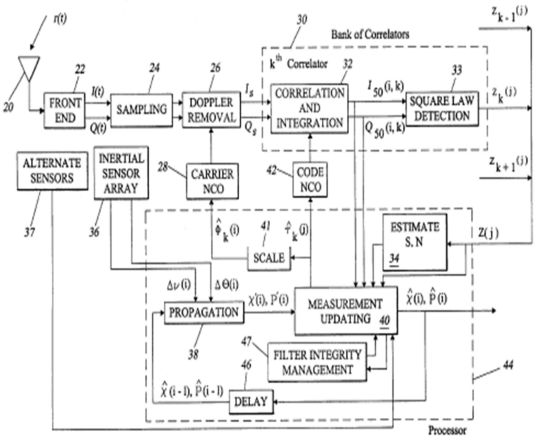

Figure 2.10: Draper Labs Patent (from [3])

In a similar manner, the Raytheon Patent [4] discusses aiding of the carrier and code tracking loops using a line of sight geometry function that maps position and velocity vector information from a navigation function, as seen in Figure 2.11. The navigation function takes measurement updates from the inertial sensors and propagates them with the range error and velocity error signals generated from the Kalman Filter. The aided carrier and code tracking provides in-phase and quadrature samples and GPS (pseudo)range and (pseudo)range rate residuals, respectively, which are used by the Kalman Filter to estimate navigation state vector corrections.

The third and final patent, [5], uses an integration Kalman Filter to update inertial measurement data to provide the navigation solution and can be seen in Figure 2.12. The integration Kalman Filter is fed error residuals generated by federated Kalman prefilters. Pseudorange and pseudorange rate data based on the navigation solution and ephemeris

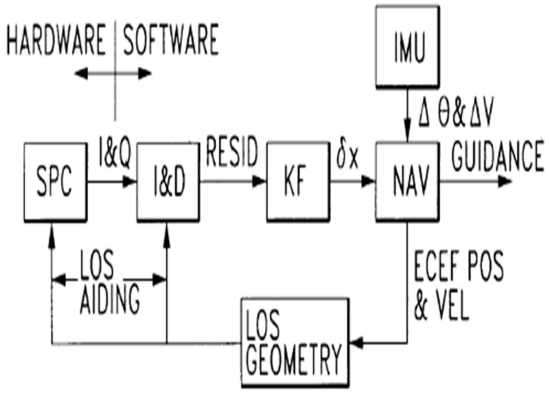

Figure 2.11: Raytheon Patent (from [4])

data is computed to generate early, prompt, and late in-phase and quadrature signals used in code and carrier tracking.

The pseudorange and pseudorange rate residual errors are used to update the navigation state vector. The navigation state vector consists of best estimates of user position, velocity, attitude, and clock errors, and IMU calibration coefficients. The federation of the Kalman Filters in the design improves estimation of the error state vector based on the dual frequency code and carrier in-phase and quadrature sampling.

These methods require proficient knowledge of Kalman Filtering techniques, signal processing, and receiver design. Two significant technical difficulties in implementing a UTC/DI design are presented in [15]. They are (1) the modeling of the code loop nonlinearity by the Kalman Filter and (2) loss of lock detection. These issues, among many others, must be thoroughly covered before initiating any serious design efforts.

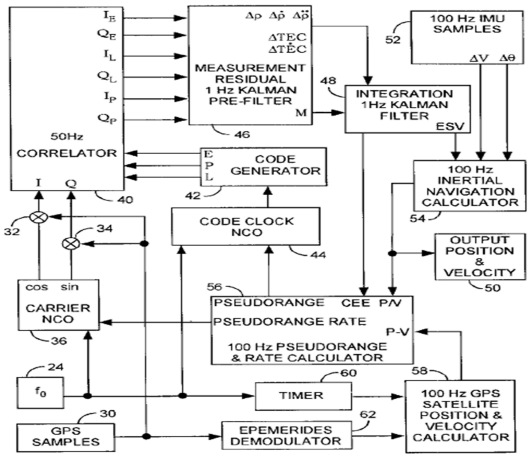

Figure 2.12: The Aerospace Corporation (from [5])

2.4 Summary

This chapter has presented an overview of GPS and the levels of integration at which GPS and INS are combined. It has been shown using cited examples that a Deeply Integrated GPS/INS system (DI) is currently the most desirable due to its anti-jamming capabilities and overall robustness in comparison to former systems. Furthermore, in order to begin the design and testing of DI, it has been shown what parameters are necessary and what specific areas of GPS must be thoroughly researched. The remainder of this paper will discuss the hardware and software requirements of a advanced algorithm test bed, error sources that must be contended with, and the acquisition of necessary data samples to be used in a advanced algorithm.

Chapter 3

GPS Intermediate Frequency and IMU Data Acquisition System Design

3.1 Motivation for a GPS IF and IMU Data Acquisition System

The primary contribution of this thesis is the design, implementation, and validation of a test bed built for a Global Positioning System and Inertial Navigation System (GPS/INS) integrated architecture. Without such a test bed only simulated data could be used. Currently it is a formidable task to create an accurate simulation of the GPS signal that includes all of the error sources. An accurate modeling of these error sources is still being discussed and debated [53, 54, 55]. These error sources are discussed in detail in Chapter 5.

In order to truly validate a GPS/INS system, actual GPS and IMU data must inevitably be used. A GPS "truth" and an IMU "truth" must be available. Without this truth data, no benchmark can be made concerning the accuracy, dependability, and robustness of the levels of integration tested. The importance of these benchmarks was discussed in Chapter 1.

As covered in Chapter 1, the data needed for the analysis of a Deeply Integrated system begins with the acquisition of In-phase and Quadrature samples, or I and Q. Acquiring these I and Q samples is done in a software correlator, as well as the aided tracking of the code phase and carrier frequency essential to advanced integration algorithms, which also includes Tightly Coupled Integration. In addition to the I and Q samples, the Signal to Noise Ratio, or SNR, is needed and is also obtained in software. The acquisition of this data is discussed in Chapter 4.

The GPS RF signal is down converted and sampled resulting in a digitized intermediate frequency, covered later in the chapter. The digitized intermediate frequency (IF) is the first step of the software correlation process, thereby making it the first step of the digital process. Design of a data acquisition system to obtain this digitized IF is one of the focal points of this thesis.

Various COTS (commercial off the shelf) units available that produce the digitized IF. Two of the more popular are the GPS Signal Tap by Accord Software and Systems, Inc and the NordNav R25 by NordNav Technologies. These systems are currently priced at $3,400US and $2,500US; beyond the price limit of a low cost test-bed. Furthermore, neither of these systems record synchronized IMU data (needed for DI development and analysis) and therefore would not be applicable in the long-term goal of developing a real-time, multi-frequency, low-cost, software-receiver-based GPS/INS integrated platform.

Furthermore these systems and similar systems are closed systems, that is, while fully functional and having user-defined settings available, the user is limited in understanding and determining some of the details and set up of the system. It is therefore advantageous to develop such a system for research purposes. Several universities and research groups have built their own RF Front-End boards for developing and testing software receivers [56, 57]. Developing an RF Front-End board would eventually be a desired scenario, but the system in this thesis is constrained such that this is not a practical path.

The aim of this research is to develop a test-bed that must be based on a system that is relatively inexpensive, configurable, and can be designed, built and tested in a limited period of time. The system must include a GPS RF Front-End, a down conversion of RF to IF, and an Analog-to-Digital sampler that produces a digitized IF signal, along with

IMU data. This would allow the system to couple an IMU and a GPS based software receiver for validation and testing. In this thesis the system is validated using techniques discussed in Chapter 5 that do not require integrated IMU-aided tracking loops.

3.2 GPS Intermediate Frequency Acquisition System Hardware

In this section the hardware used in this thesis is discussed in detail. The path of the GPS received signal is followed as it would travel through the system. The section begins with component selection and an overview of the system and concludes with a more in depth discussion of the individual components.

A block diagram of the IF data collection system developed in this thesis is shown Figure 3.1. As can be seen in the diagram the input is the GPS Radio-Frequency (RF) signal from the satellites. The outputs are the 2-bit Sign and Magnitude digitized intermediate frequency and the ADC sample clock.

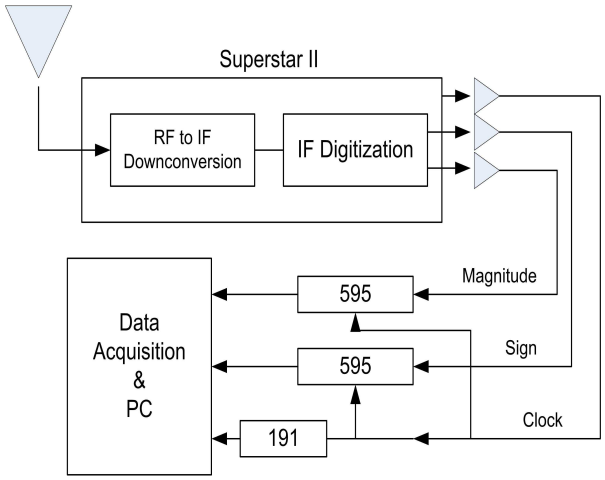

Figure 3.1: GPS Intermediate Frequency Acquisition System

3.2.1 Chip Set Selection

At the heart of the GPS IF Acquisition System (GIAS) is the need to receive, filter, down convert and sample the GPS RF signal. This down-converted and sampled RF signal is known as the digitized intermediate frequency (IF). As mentioned before, it was within the constraints of this thesis that a low-cost front-end receiver chip would be used to produce the digitized IF. Therefore, in order to build the GIAS, one of several available chip sets meeting the criteria would need to be chosen. The chip sets investigated were the Maxim 2745 [58], the Nemerix NJ1006A [59], and the Zarlink GP2000/GP2015 [6, 60]. Each chip set is available with the necessary peripherals for operation.

The Maxim 2745 is available with an evaluation kit containing all necessary peripherals. The kit has one RF input port, three IF output ports, one clock output port, and one PC control port. The IF frequency is 4.092 MHz and the reference clock is 16.367 MHz.

The Nemerix NJ1006A is also available with an evaluation kit containing all necessary peripherals. The kit has one RF input port and a 10-pin connector, J2, which provides the digital output available in 2-bit Sign and Magnitude format. The oscillator clock is also available on J2. The corresponding GPS IF option has an IF Output frequency of 4.092 MHz and a reference clock of 16.367 MHz.

The Zarlink GP2000 chip series is available on several different boards: the SigTec Navigation MG5001, the SigTec Navigation MG5003, and the Novatel Superstar II. Each of these boards had its advantages and additional boards not listed here are also available. The SigTec Navigation MG5001 was considered given that it is the official board of the GPL-GPS Project and there is a large amount of available resources specifically for this

board [61]. The MG5003 is a similar board to the MG5001, only smaller. For the requirements of the system the two boards provided the same purpose. However, the Novatel Superstar II provided a low price option with wide recognition and use among the GPS community.

The chip set and complementary board chosen was the Zarlink GP2000/GP2015 chip set on the Novatel Superstar II board. This chip set was chosen due to the ease of availability of a digitized IF at a lower intermediate frequency, 1.405 MHz. The direct benefit of a lower intermediate frequency is that the sampling frequency is correspondingly lower. The lower sampling frequency eases the requirement of the acquisition system. Furthermore, a great deal of research and documentation has been done on the chip set and is publically available [6, 60, 62]. The Superstar II (SSII) was also chosen because it possessed all of the requirements for selection at the lowest price. Therefore the SSII provided all of the requirements including a digitized IF in an available path to the user and no further peripherals for operation were required beyond the RF Input and power.

3.2.2 System Overview

The first component of the system is the active antenna, followed by the antenna splitter/amplifier, the GP2015/4020 chip set, the voltage followers, the binary counter, the shift registers, the data acquisition block and board, and finally the PC. The evolution of signal attributes such as frequency, power level, and logic is discussed when such changes occur in the process.

Although not required for operation, an active antenna was chosen for the system based on manufacture's suggestions and performance advantages [63]. The antenna

chosen was the Mighty Mouse II active antenna. The Mighty Mouse II (MMII) has a 29 db active gain. The major distinction between an active antenna and a non-active antenna is that the active antenna has a current running through it to improve reception. The MMII has a recommended voltage of 2.5-5.5V with a current limitation of 40mA. In order to safely power the antenna a voltage splitter and amplifier (NALDCBS1X4-S by GPS Networking, INC.) is used.

The splitter option was chosen so that the system may be compared to a COTS system using the same antenna and therefore the same satellite data for validation purposes and performance evaluation. The amplifier powers the antenna with a current limited 5V source. The GPS receiver ports are DC blocked to prevent damage and interference in the system.

From here, the GPS signal is sent to the GP2015 chip located on the Superstar II board. At this point, the GPS signal is in its rawest form. As discussed in Chapter 2, the GPS signal is at the antenna is about -130 dBm and spread over a 2.046 MHz bandwidth; well below the noise floor. In particular, the L1 band is a spread spectrum signal at a carrier frequency of 1575.42 MHz with 1.023 Mbps binary phase-shift keying (BPSK) modulation.

For the GP2015 to be used, the Superstar II board must be properly powered. The board is available in a 3.3V option or a 5V option. A 5V board was chosen for consistency with the rest of the hardware requirements. The Superstar II board is powered through Jumper 1, a 0.1-inch 20-pin connector. A schematic for the board can be found in [63].

The GP2015's IF section is tuned to down convert the L1 band only. This is an allowable limitation as this system is not concerned with any frequencies other than L1. Other common GPS frequencies are discussed in Chapter 2, such as L2 and L5.

The RF-to-IF portion of the GP2015 essentially down converts the GPS signal at 1575.42 MHz down to 4.309 MHz through three mixing and filtering steps, as seen in Figure 3.2. An advantage of this triple-down-conversion is that a majority of potential unintentional jamming frequencies are filtered out.

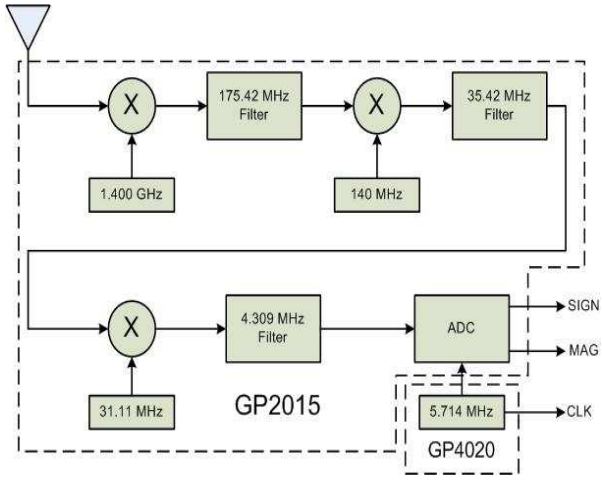

Figure 3.2: GP2015 Down Conversion Process

The first stage of the mixing/filtering process mixes the GPS RF signal with a 1.400 GHz signal and then filters the result at 175.42 MHz. The second stage mixes the resulting signal from the first stage with a 140 MHz signal and filters this at 35.42 MHz. The third stage mixes the 35.42 MHz signal with a 31.11 MHz signal and puts this new signal through a final filter at 4.309 MHz. At this point the signal is at an Intermediate Frequency (IF) and is available in a raw analog form for testing purposes.

Once the signal has been brought down to its IF, it is sampled so that it may be put into its digital form, the Digitized IF. The IF is sent to a 2-bit quantizer creating a Sign bit and a Magnitude (Mag) bit. The Sign and Magnitude bits are latched using the

clock sent from the GP4020 chip. This clock is at 5.714 MHz on pin 63. The sampled (digitized) IF is centered at 1.405 MHz.

The Magnitude bit is extracted at pin 14 and the Sign bit is extracted at pin 15. The extraction requires very precise soldering due to the spacing of the leads, or pitch, on the printed circuit board (PCB) [64]. The Superstar II board locates an inline resistor for each Sign and Magnitude signal. The board was tapped immediately following each of these resistors on the side away from the GP2015 chip for noise cancelling and signal integrity.

As previously mentioned, the GP4020 chip produces the 5.714 MHz sample clock with a 4:3 mark to space ratio used by the GP2015 ADC to latch the digital samples. Therefore the Sign and Magnitude digitized IF bit stream is operating at 5.714 Mbps. This sample clock will be used later in order to spread this bit stream so that the data may be acquired at a lower frequency. A 1 pulse-per-second (1PPS) signal is also available from the GP4020 at pin 69 and is discussed in Chapter 6. Other than the sample clock and the PPS, the GP4020 serves no other purpose for this thesis. The data sheets for the GP2015 and the GP4020 are included in [60, 62].

Follwing the A/D sampling the digitized IF is represented by a 2-bit Sign and Magnitude Transistor-Transistor Logic (TTL) bit stream at 5.714 Mbps. The bit stream must be conditioned to avoid signal data loss. The common solution is a buffer, or an operational amplifier acting as a voltage follower. A diagram of a basic voltage follower can be seen in Figure 3.3.

The behavior of the operational amplifier configured as a buffer is very simple, and allows the output voltage (V_{out}) to follow the input voltage (V_{in}), i.e. $V_{out} = V_{in}$. Although simple, this is very useful because the input impedance of the operational

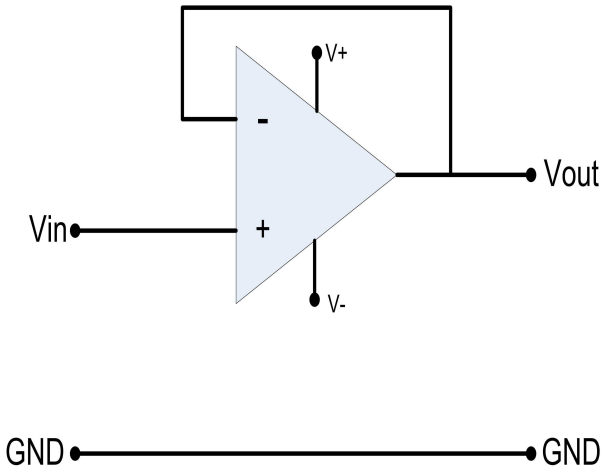

Figure 3.3: Voltage Follower

amplifier is very high, giving effective isolation of the output from the signal source. Furthermore, very little power is drawn from the signal source which avoids loading effects [65].

The data collection of the system used in this thesis directly acquires the Sign and Magnitude bit streams at the 5.714 MHz sampling rate in parallel using the National Instruments 6251 PCI Data Acqusition Card, capable of digital acquisition at speeds up to 10 MHz. The data collection of a second method intended for future real-time algorithm development occurs in two parallel operations using shift registers and a counter. As the digitized IF bit stream is operating at 5.714 MHz, it would be advantageous for real-time operation to slow this rate down in order to use a high-speed data acquisition card (DAQ) coupled with a real-time capable software receiver [57]. The high-speed DAQ used for the system is the National Instruments PCI-DIO-6533, capable of sampling multiple digital inputs at speeds up to 2 MHz. The 6251 and the 6533 will both be discussed in greater detail in Chapter 5.

In order to slow the digitized IF down to a lower frequency, shift registers are used. The chip used for the shift registers is the Texas Instruments SN74HC595. The TI SN74HC595 (595) is an 8-bit high-speed shift register capable of operating at speeds up to 25 MHz. The 595 has a serial digital input, a serial output, eight parallel outputs, and a serial clock input. The 595 operates on 4.5-5.5V.

The digitized 2-bit Sign and Magnitude IF are each sent to an identical set of 595's. Each signal, the Sign and the Magnitude, is treated identically from this point on. Each signal is fed to the serial input of the first of two 595's. The serial output of the first 595 is fed to the serial input of the second 595. The serial clock input of each of the four 595's is driven with the 5.714 MHz sample clock created by the GP4020. The 16 parallel outputs of each of the two chip sets are fed to the DAQ breakout block. A diagram of the set up can be seen in Figure 3.4.

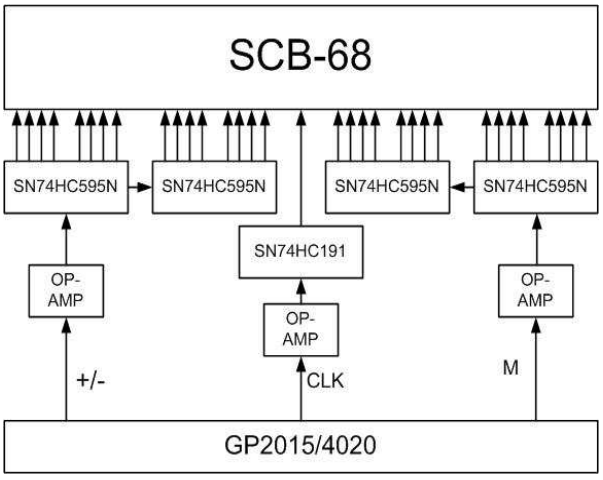

Figure 3.4: RFFE and DAQ Diagram

In order for the DAQ to know when each 16-bit shift register set (data buffer) is full, a clock signal is needed that is 1/16th as fast as the 5.714 MHz sample clock. The

chip used to get this "buffer-full" update signal is the Texas Instruments SN74HC191. The TI SN74HC191 (191) is a 4-bit synchronous up/down binary counter. The 191 has a clock input, several logic setting inputs, and four counter outputs. In order for the 191 to count down from 15 and then send an output pulse each time it hits zero, Table 2.1 is followed.

$CT\bar{E}N$	LOW	4
$D\bar{/}U$	LOW	5
$R\bar{C}O$	HIGH/LOW	13
$LO\bar{A}D$	HIGH	11

Table 3.1: 191 Pin Settings

The input clock used is the 5.714 MHz clock created by the GP4020. Pin 7, Q_D, gives a pulse when 16 pulses are received by the input clock. Thus, the resulting frequency at Pin 7 is 357.142 kHz. This clock is fed to the DAQ breakout board along with the 32 buffered TTL Sign and Magnitude digitized IF samples.

The Data Acquisition Card is the final step of the hardware process. As previously mentioned, the cards used are the National Instruments PCI-DIO-32HS and the 6251 Data Acquisition Cards. The 6533 has 32 digital input/output ports, and it is a high-speed (2 MHz) DAQ. All of the 32 ports are used for digital input of the buffered Sign and Magnitude.

In order to acquire the 32 digital inputs in parallel at each time update from the DAQ, a Labview^{TM} program was used. It was important that all 32 signals were sampled simultaneously by the program and properly arranged for post processing the data. Using the DAQ Assistant utility and a for loop set for 32 iterations, the signal is acquired and saved to a spreadsheet for rearrangement.

As a result of the digitized IF signal being sent through a series of shift registers the data for both the Sign and Magnitude signals is rearranged in a manner seen in Table 2.2. The number indicates the placement of that bit in time as it is sampled from the GP2015.

Pin 1:	1	17	33	49...
Pin 2:	2	18	34	50...
Pin 3:	3	19	35	51...
Pin 4:	4	20	36	52...
Pin 5:	5	21	37	53...
Pin 6:	6	22	38	54...
Pin 7:	7	23	39	55...
Pin 8:	8	24	40	56...
Pin 9:	9	25	41	57...
Pin 10:	10	26	42	58...
Pin 11:	11	27	43	59...
Pin 12:	12	28	44	60...
Pin 13:	13	29	45	61...
Pin 14:	14	30	46	62...
Pin 15:	15	31	47	63...
Pin 16:	16	32	48	64...

Table 3.2: Data Bit Arrangement

In order to rearrange the data, a Matlab program was used to take the sequential rows of the data from each of the 16 columns and concatenate them in order to have two continuous bit streams; one for Sign and one for Magnitude. Then using Matlab and Table 2.3 the two bit data of 0's and 1's (Sign and Magnitude) is translated into a single bit stream of +/- 1's and 3's (Value) for the final digital IF.

Sign	**Magnitude**	**Value**
0	0	-1
0	1	-3
1	0	1
1	1	3

Table 3.3: Sign and Magnitude to Value

The total price for the system, not including the DAQ or PC, is roughly $175US. The active antenna was approximately $40US, the SS II was approximately $125US, and the voltage follower, counter, and shift register chips were all purchased for under $20US. Including the antenna splitter/amplifier and the DAQ, the total system price is roughly $1950US.

3.2.3 Initial Validation

Once the hardware portion on the system was completed, it was checked for correct operation. Unfortunately in the digitized IF state the relevant information is still buried in noise. That is, the GPS navigation data that is sent in by the satellites is still modulated by a BPSK code as well as modulated with an intermediate frequency.

Therefore, in order to perform an initial validation, the correct operation of the counter, the shift registers, and the data acquisition configuration, a simulated signal is used in place of an actual modulated GPS signal. A BK Precision 4011A 5MHz function generator is used to create multiple sinusoidal signals that are derivatives of the GP2015 analog-to-digital converter (ADC) latch rate, i.e. 5.7 MHz / 2, 5.7 MHz / 3, 5.7 MHz / 4, etc. The signals are subsequently input into the system at the voltage followers, using the clock provided by the GP4020. As the generated signals are predictable continuous wave (CW) inputs, the data from the acquisition and rearrangement programs should exhibit the expected behavior, i.e. repeating regular multiples of 1's and 0's. This test was implemented and the operation of the bit acquisition system was validated. A more detailed validation using GPS acquisition software is provided in Chapter 5.

In order to validate the correct operation of the GPS Intermediate Frequency Acquisition System (GIAS) at this point, the behavior of the GP2015 integrated circuit

must be looked at in more depth. The signal sampled by the on board ADC can be seen as a continuous wave (CW). The Sign signal indicates the polarity of the signal being sampled. If a CW signal is properly adjusted such that its amplitude is at midpoint, the duty cycle should be 50%. That is, the value of Sign should be '1' 50% of the time and '0' 50% of the time [6].

Additionally, the Magnitude signal indicates the magnitude (or amplitude) of the signal being sampled. If a CW signal is properly scaled, statistically the Magnitude signal should have a duty cycle of 30%. That is, the value of Mag should be '1' 30% of the time and '0' 70% of the time [6]. Based on data given in the Zarlink GP2000 Application Note, the magnitude threshold, that is the voltage at which the magnitude bit becomes '1' is +/- 115mV [6]. This can be seen in Figure 3.5.

Ideally, if a CW signal is created and then sampled by the GP2015, the resulting Sign and Magnitude signals should follow this statistical pattern. However, the modulation scheme used for GPS is binary phase-shift keying, as is discussed in Chapter 2, will not leave a perfect CW to be sampled.

The Gold code used for C/A modulation has a period of 1023 bits and a clock rate of 1.023 Mbps. This gives the C/A code used on the L1 GPS frequency a period of 1 ms [24]. If a sample of 160,000 bits is acquired at 5.714 MHz, roughly 28 ms of data is taken. Thus a possible 29 C/A code periods, or phase shifts, have elapsed.

Although there are 29 possible C/A code periods, and a minimum of 27, the resulting signal is still considered a CW due to the small effect of the modulation although it is not by definition ideal. For the purposes of validation, the principle of the duty cycle's statistical behavior still holds. Therefore, a histogram can be taken of the sampled Sign and Magnitude samples to ensure proper operation of the RF Front End Data Acquisition

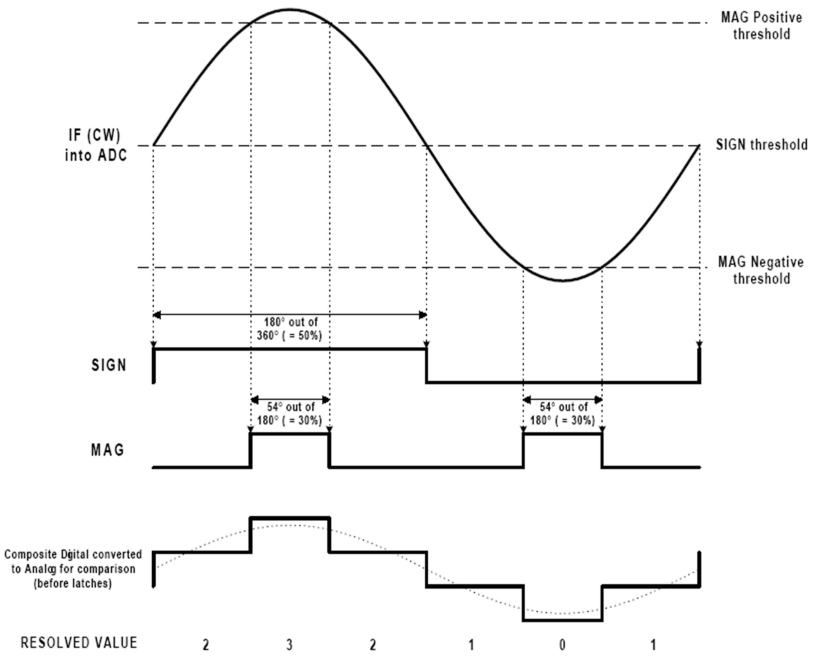

Figure 3.5: Sign/Mag Duty Cycle (from [6])

System. The histogram of the resulting digitized IF can be seen in Figure 3.6. When given a true GPS signal the GIAS closely mimics the 50/50-70/30 histogram of the true continuous wave, as can be seen in Figure 3.7.

3.3 Acquisition of GPS and INS Measurements

The data from the GPS IF Acquisition System (GIAS) must be simultaneously sampled with the data from the IMU such that proper analysis of the desired integration algorithm may be performed. This requires that the GPS and IMU data be acquired and stored together in time, or synchronized. Three possible methods are presented here. The third method was used in this thesis.

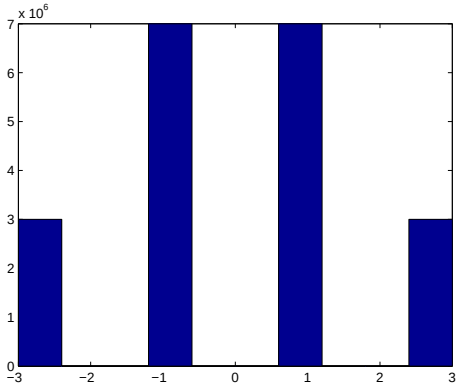

Figure 3.6: Histogram of continuous wave signal

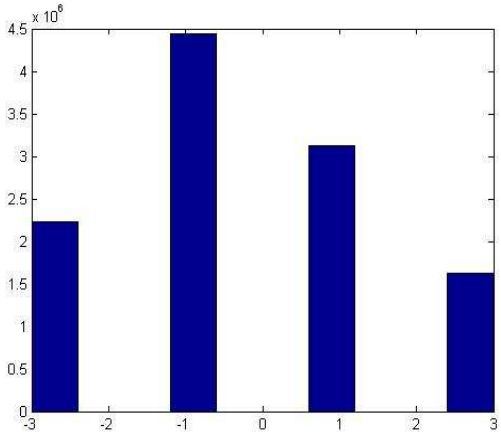

Figure 3.7: Histogram of actual GPS data

In one method the IMU data may be sampled using other available acquisition systems and integrated in post-processing, while the GPS data is acquired using the 32 spread signal method or direct 2 signal method. A shared clock could be used between the two systems to ensure synchronization. A clock from the GPS chip set could be used. This method could be applied when a digital IMU is used and the data must be logged using a separate acquisition system.

A second method proposes the acquisition of both GPS digitized IF Sign and Magnitude samples and raw IMU data using the National Instruments PCI-DIO-32HS-6533 Data Acquisition Board. The NI DAQ has 32 digital inputs. In the design as discussed, these 32 digital inputs are reserved for the acquisition of the 16 Sign bits and the 16 Magnitude bits. A simple reconfiguration consisting of assigning 8 Sign bits and 8 Magnitude bits to 16 of the digital inputs could be used. Furthermore, a sampling clock of twice the previous frequency would be required. The remaining 8 digital inputs could be used for IMU data collection. A diagram of this configuration is given in Figure 3.8.

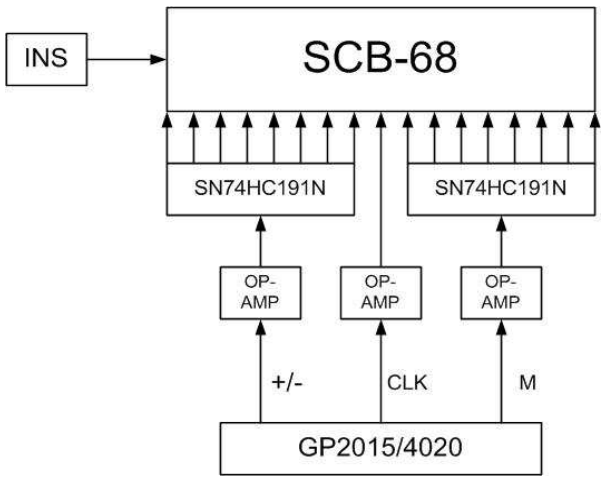

Figure 3.8: 16 Bit DAQ

A third method is to use an acquisition system capable of at least two digital inputs at 5.714 MHz and an additional six single-ended (or 12 for differential) analog inputs for the IMU. This method was used for this thesis. The board selected was the National Instruments 6251 Data Acquisition Board. By connecting the 5.714 MHz Sign and Magnitude signals from the GP2015 to the two digital inputs on the 6251, the 5.714 MHz clock from the GP4020 can be used for the digital input sample clock.

The 6251 has 16 analog inputs for IMU data sampling. The Bosch IMU signals are differential, requiring 12 analog input channels. In order to synchronize the GPS IF and IMU samples, a shared clock is used. Since the 5.714 MHz clock from the GP4020 is already used for the GPS IF samples, it would be ideal to use this clock for the IMU sampling as well. This method is discussed further in the following section.

3.4 Inertial Measurement Unit Data Acquisition and Validation

This section will discuss the inertial measurement units intended to provide the measurements necessary for the advanced integration algorithms. IMU's of varying grades will be reviewed, followed by a discussion of how each will be synchronized to GPS sampled data.

3.4.1 Inertial Measurement Unit Selection

In Chapter 2, two grades of IMU's were listed as being available for use: automotive and tactical, due to performance requirements of advanced integration architectures as discussed in [66]. Future efforts may include additional IMU's.

The IMU used in this thesis is the Bosch automotive grade IMU. The Bosch consists of three accelerometers and three rate gyros that provide an analog output for each of

its six sensors. The analog output can be sampled using an analog-to-digital converter (ADC) clocked with an external trigger. A 100 Hz clock signal from the GPS Intermediate Frequency Acquisition System (GIAS) is used for this thesis, the generation of which will be discussed later in the section.

The second IMU is the Honeywell HG1700 tactical grade IMU, consisting of Honeywell GG1308 Ring Laser Gyros and Honeywell RBA500 accelerometers. This model HG1700 samples delta velocities and delta angles at 600 Hz, internally, and has the capability to synchronize its measurements to an external 100 Hz clock. According to the HG1700 specifications sheet, the latency, or time from when the sensor data is read to when the inertial data is transmitted, is 1.712 0.005 milliseconds. Furthermore, the latency between the rising edge of the 100 Hz clock input and the transmission of the measured data is within 2 milliseconds. Use of the HG1700 with the GIIAS is intended for the near future.

Another IMU inteded for future use is the Litton LN200 Fyber Optic/Silicon tactical grade IMU. The LN200 provides linear acceleration and angular rate measurements by an RS485 data bus at 400 Hz using the SDLC protocol. According to [66], the LN200 can be provided with a 1 pulse-per-second (1PPS) clock and a 100 Hz clock. However, other investigations have suggested that this external clock input is not possible and the sampling is done internally [67]. Regardless, both methods of synchronization will be covered in the remainder of the section.

3.4.2 IMU and GPS Clock Synchronization

As discussed, the Bosch and the HG1700 can utilize a 100 Hz external clock. In order to synchronize these IMU's with the GPS measurements being taken with the

GIAS, the same clock used to sample the GPS IF can be used for the 100 Hz IMU clock. Earlier in the chapter, the Superstar II board used for the GIAS was covered in detail, including the 5.714 MHz clock used by the GP2015 chip in its ADC. This 5.714 MHz clock is produced in the GP4020 chip by dividing a 40 MHz master clock by 7 using an on-chip counter, thus making the 5.714 MHz clock, the 40 MHz and any derivative of the two synchronized.

The 40 MHz differential master clock is produced on the GP2015 and sent to the GP4020 through a bias-shift circuit. Because it is a differential signal, there is an outgoing signal and a return sinal path. Therefore it is sent through two lines between the boards. The connection is made between pins 16 and 17 on the GP2015 to pins 58 and 59 on the GP4020. Pins 17 and 58 connect the true phase signal; pins 16 and 59 carry the inverse phase signal.

The 40 MHz master clock can be made available by picking off the signal in a similar manner to the Sign, Magnitude, and 5.7 MHz clock signals as discussed earlier in the chapter. As seen in Figure 3.9 the 40 MHz clock can be reduced to a 100 Hz clock signal by using a divide-by-4 counter and a series of five decade counters. The chips intended for use in the thesis were the SN74HC191 for the divide-by-4 and the DM74LS90 for the decade counters.

Similarly, a UTC aligned 1PPS clock signal is produced by the GP4020 chip and can be extracted at pin 69. This provides the 1PPS clock signal necessary for the synchronization method proposed in [66]. This approach suggests that the 1PPS signal, once retrieved, can be sent to a timing controller that produces a 100 Hz clock used in the LN200 to control the INS data sampling and a 1PPS signal to reset the unit's

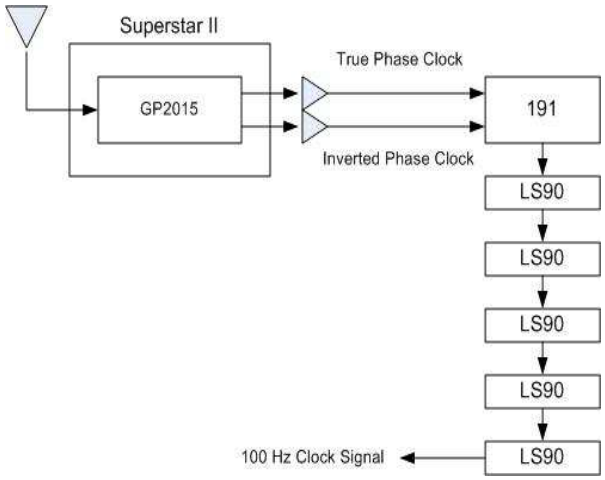

Figure 3.9: 40 MHz to 100 Hz Clock Deduction

internal time tag counter. However, according to the paper, this method has yet to be implemented.

According to the approach outlined in [67], an automotive grade IMU, such as the Bosch, is synchronized with the 100 Hz clock signal produced from a GPS system, along with a 1PPS. This synchronized automotive IMU data is then post-process correlated with data taken during the same time frame with the LN200. The LN200 data is then post-process time stamped to appear synchronized with the GPS data.

While all three of these methods may be viable, the design developed in this thesis is more direct. The method makes use of the 5.714 MHz clock signal used by the internal ADC to produce digitized IF samples. By sending the 5.714 MHz clock through a decade counter and then through a divide-by-N counter system preloaded to count down from 5,714, a 100.00500025 Hz clock signal is produced. A detailed discussion of the divide-by-N counter system used is discussed in Appendix B. The 100 Hz clock deduced using this method provides synchronized data acquisition between the GIAS and the

INS, producing the hardwired synchronized GPS Intermediate Frequency and Inertial Measurement Unit Acquisition System (GIIAS) seen in Figure 3.10.

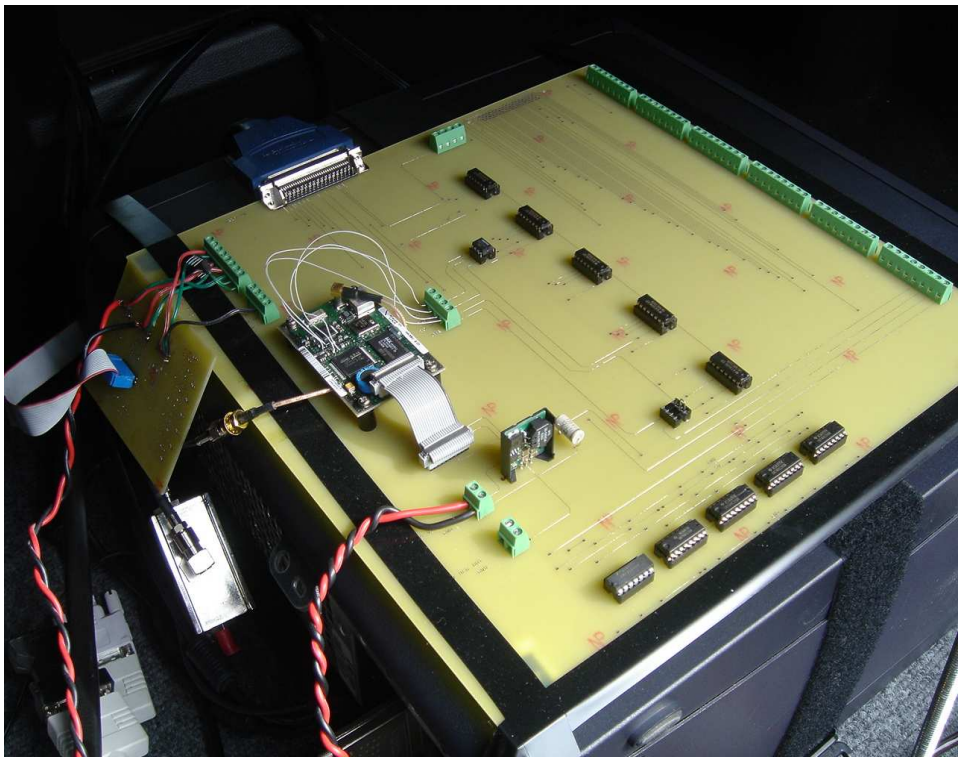

Figure 3.10: GIIAS

3.5 Conclusion

This chapter has provided the detailed design of the GPS IF Acquisition System developed in this thesis. The motivation for a GPS IF and IMU Data Aquisition System (GIIAS) was given along with the requirements of the system. The hardware used was discussed in detail, including the GPS chip set and its necessary peripherals. An overview was given of IMU's commonly used, their classification, and their role in each

level of GPS/INS integration. Finally, methods for sampling and synchronizing IMU measurements with the GIAS were given.

Chapter 4

GPS Software Receiver

4.1 Introduction

This chapter provides an overview of the software receiver used to process the recorded GPS IF data. The necessity for a software receiver is discussed, followed by the techniques of signal processing and GPS navigation algorithms implemented in software. A typical software receiver can be divided into three components: the acquisition phase, code and carrier tracking, and the navigation algorithms. Each of these is covered in this chapter.

4.1.1 Motivation for Software Correlator

The aim of this chapter is to provide the background and necessary information for the design of a system that provides GPS data samples necessary for the development and testing of advanced integrated GPS/INS algorithms, that is, Deeply Integrated GPS/INS as well as former levels of integration. In Chapter 1, it was shown that the necessary samples consisted of the in-phase and quadrature (I and Q) samples provided by a correlator. When inertial measurement unit (IMU) samples are employed, the generation of these samples involves corrections sent from an inertial navigation system (INS). A traditional hardware receiver is incapable of performing the mixing of data samples; therefore the integration of data at this level requires the use of a software receiver, i.e., a software correlator, as seen in Figure 4.1.

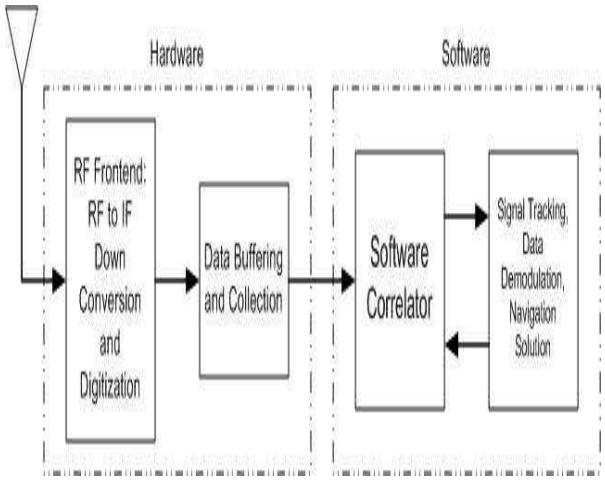

Figure 4.1: GPS Software Receiver

4.1.2 Software Receiver Advantages

The majority of GPS units in service today use a hardware receiver, that is, the GPS correlations are done in hardware, while the acquisition, code and carrier tracking, and navigation algorithms are performed on a microprocessor. A block diagram of this configuration can be seen in Figure 4.2.

As an example, and of particular interest to this writing, Zarlink Semiconductor's GP2021 is a 12 parallel-channel direct-sequence spread-spectrum signal correlator for GPS C/A code signals. It is capable of processing data from 12 satellites simultaneously with the aiding of a microprocessor. The GP2021 is typically used as a supplement to the GP2015 RF Front-End chip, discussed in Chapter 3, combined with a microprocessor to provide a user position solution. This configuration is similar to that of Figure 4.2. A continued discussion of this system can be found in [6].

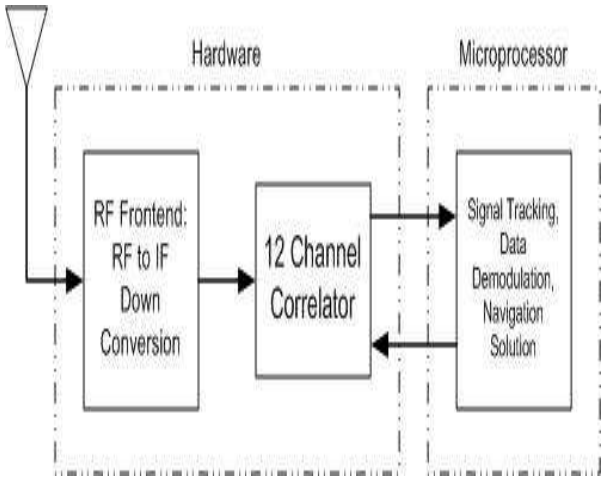

Figure 4.2: GPS Hardware Receiver

Although the hardware correlator is capable of operating on several GPS satellite channels in parallel, it is heavily limited in its interoperability. A requirement of this thesis is the ability to control a GPS correlator such that it can incorporate its corrections with those from an INS. Historically, computer platforms have not possessed the processing power necessary to compete with hardware correlators. With advancements in computer computation speed, the past several years have seen a number of GPS software receiver approaches, making the integration of data at this level realizable. There is now available a large amount of literature on software receivers [68].

4.2 Software Receiver Theory

4.2.1 Acquisition

This section covers the calculations and equations used for acquisition in a typical software correlator. It should be noted that Deeply Integrated and Ultra-tightly Coupled algorithms may employ different methods than the ones covered in this generalized overview. Nevertheless, it is important to understand the fundamental operation of a software receiver in order to give background to the methods used in higher levels of tracking.

The purpose of the acquisition phase is to determine which satellites, if any, are currently visible to the user and, if so, to determine coarse measurements for the carrier frequency and the code phase of the signal. Since there are roughly 32 possible codes, the correct code as well as the code's phase are necessary to generate a replica code signal that is perfectly aligned with the incoming code signal. Additionally, the incoming signal's carrier frequency must be replicated. Although the transmitted carrier frequency is known at 1575.42 MHz on L1, during transit from the satellite to the user, an effect known as Doppler shift takes place on the signal which is dependent on the line-of-sight velocity of the satellite. This causes the expected signal to be off by as much as 10 kHz in either direction. Therefore this deviation must be estimated in order to generate a local carrier signal.

In a hardware receiver, these acquisitions take place in integrated circuits, while in a software receiver, the acquisition of code phase and carrier frequency are done in software using a method known as parallel code phase search acquisition. Parallel code phase search acquisition is advantageous to other methods by parallelizing the search

for the code phase as opposed to the carrier frequency. In order to search through all possible code phases, each chip of the 1023 chip long C/A code must be considered. In order to search through all possible carrier frequencies, the 20 kHz of probable Doppler shift deviation are broken into 41 500 Hz sections. Therefore, by parallelizing the code phase search, only 41 search iterations are necessary, as opposed to 1023 using parallel carrier frequency search acquisition.

In order for parallel code phase search acquisition to search through all 1023 code phase possibilities, a method based on circular correlation through Fourier transforms is employed [68]. The discrete Fourier transform of a finite length sequence x(n) with length N is defined as

$$X(k) = \sum_{n=0}^{N-1} x(n) e^{-j2\pi kn/N} \tag{4.1}$$

The cross-correlation between two finite length sequences x(n) and y(n) both with length N is defined as

$$z(n) = \sum_{m=0}^{N-1} x(m) y(n+m) \tag{4.2}$$

The convolution between the two finite length sequences x(n) and y(n) is defined as

$$z(n) = x(n) * y(n) = \sum_{m=0}^{N-1} x(m) y(n-m) \tag{4.3}$$

Equations (4.2) and (4.3) show that the only difference between cross-correlation and convolution between two finite length sequences is the sign in y(n,m). The convolution can be transferred to the frequency domain through Fourier transform. The

combination of Equations (4.1) and (4.3) defines the discrete Fourier transform (DFT) of the convolution between x and y as

$$\begin{aligned} Z(k) &= \sum_{n=0}^{N-1}\sum_{m=0}^{N-1} x(m)y(n-m)e^{-j2\pi kn/N} && (4.4)\\ &= \sum_{m=0}^{N-1} x(m)e^{-j2\pi km/N}\sum_{n=0}^{N-1} y(n-m)e^{-j2\pi k(n-m)/N} && (4.5)\\ &= X(k)Y(k) && (4.6) \end{aligned}$$

Here the convolution-multiplication property of the Fourier transform is seen, which states that convolution in time corresponds to multiplication in frequency, or

$$z(n) = x(n) * y(n) \tag{4.7}$$

$$Z(k) = X(k)Y(k) \tag{4.8}$$

The combination of Equations (4.1) and (4.2) defines the DFT of the cross-correlation between x and y as

$$\begin{aligned} Z(k) &= \sum_{n=0}^{N-1}\sum_{m=0}^{N-1} x(m)y(n+m)e^{-j2\pi kn/N} && (4.9)\\ &= \sum_{m=0}^{N-1} x(m)e^{-j2\pi km/N}\sum_{n=0}^{N-1} y(n+m)e^{-j2\pi k(n+m)/N} && (4.10)\\ &= X^*(k)Y(k) && (4.11) \end{aligned}$$

In other words, the connection between a time-domain correlation and a frequency domain representation is similar to the connection between convolution and its frequency domain representation. However the Fourier transform of one of the two input sequences is complex conjugated before multiplication.

When the frequency domain representation of the correlation is determined, the time-domain representation can be found through the inverse discrete Fourier transform (IDFT) [22].

$$x(n) = \frac{1}{N} \sum_{k=0}^{N-1} X(k) e^{j2\pi kn/N} \tag{4.12}$$

As seen in the block diagram in Figure 4.3, the in-phase and quadrature (I and Q) signals are created by multiplying the incoming signal by the locally generated carrier signal and the 90 degrees phase shifted locally generated carrier signal, respectively. The signals are combined and transformed to the frequency domain using a DFT. The locally generated code replica is likewise transformed using a DFT, complex conjugated, and then multiplied by the manipulated incoming signal. This product is sent through an inverse Fourier transform (IFT), thus transforming it back to the time domain. The absolute value of this time domain signal denotes the correlation between the incoming signal and the locally generated code. In the case where there is correlation, a peak is observed and the location of this peak indicates the code phase of the incoming signal. In the case where there is no correlation, no peak will be observed.

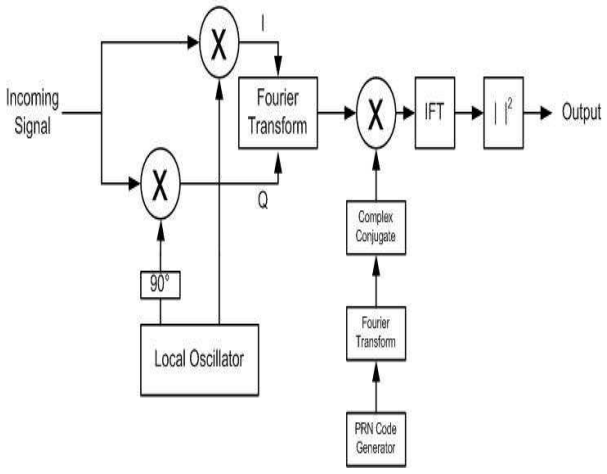

Figure 4.3: Satellite Acquisition

4.2.2 Tracking

It is necessary to track the incoming signal in order to maintain accurate estimates of the signal's carrier frequency and code phase. These estimates are used to generate carrier and code signals that are multiplied by the incoming signal in order to demodulate the navigation data. The locally generated carrier and code signals are continuously produced using two feedback loops known as a carrier tracking loop and a code tracking loop, respectively. A block diagram of this process can be seen in Figure 4.4, which shows that the code tracking loop and the carrier tracking loop are behaving somewhat independently. These two loops are discussed in the following two sections, followed by a discussion of demodulating the incoming signal.

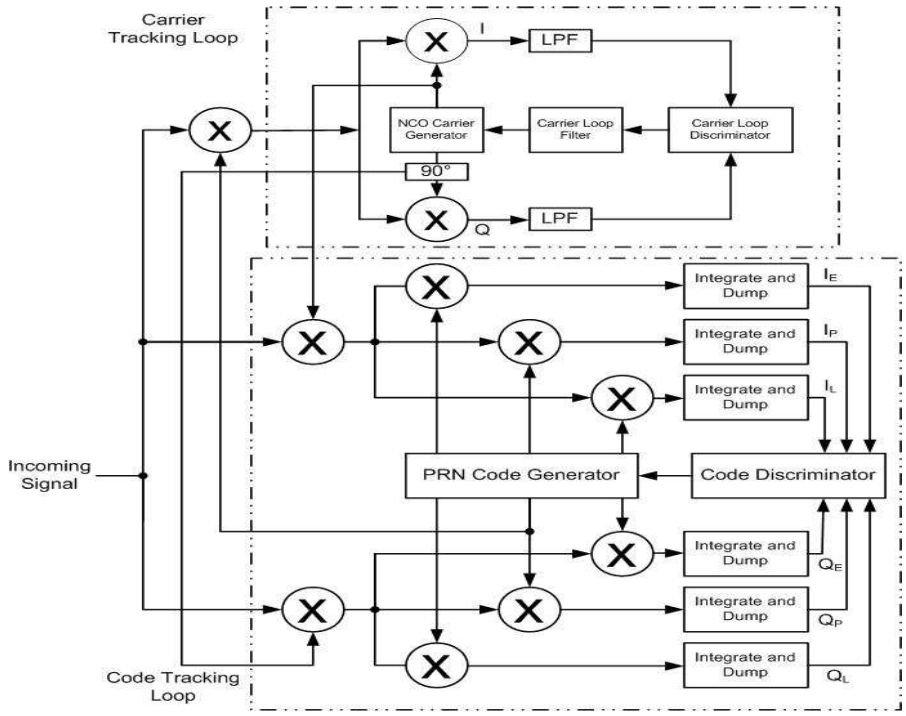

Figure 4.4: Parallel Tracking Algorithm

4.2.3 Code Tracking

The ideal code tracking loop generates a code signal that is a perfect replica of the incoming signals code, i.e. it is the correct code and is perfectly aligned in phase with the incoming signal. The method commonly used in a GPS code tracking loop is the delay lock loop (DLL). The DLL produces multiple versions of the incoming signal by propagating the locally generated signal forwards or backwards in phase and multiplying these by the incoming signal. In GPS applications, the DLL produces three signals: early (E), prompt (P), and late (L), typically with early being a half-chip (or 4.8876 x 10^{-4} seconds) ahead of the prompt signal and late being a half-chip behind.

As seen in Figure 4.5, the incoming signal is first multiplied by in-phase and quadrature components on a locally generated carrier in order to remove this signal from the code and navigation data. It is necessary to use both in-phase and quadrature in case the generated carrier is not perfectly in-phase with the incoming signal. In the case where the two signals are in-phase, all signal power will remain in the in-phase portion of the loop. In the case where the two signals are perfectly out of phase, all signal power will remain in the quadrature portion of the loop.

Once the carrier is removed from the signal, the DLL is implemented, creating six new signals which are integrated and dumped over each millisecond, i.e. each millisecond the signal is integrated and then fed forward. The resulting outputs are I_E, I_P, I_L, Q_E, Q_P, and Q_L. These values are evaluated using a discriminator to determine error in the phase of the locally generated code signal. Four common discriminators are Early-Late, Early-Late Power, Normalized Early-Late Power, and Dot Product, and are defined in Equations (4.13) - (4.16), respectively [69].

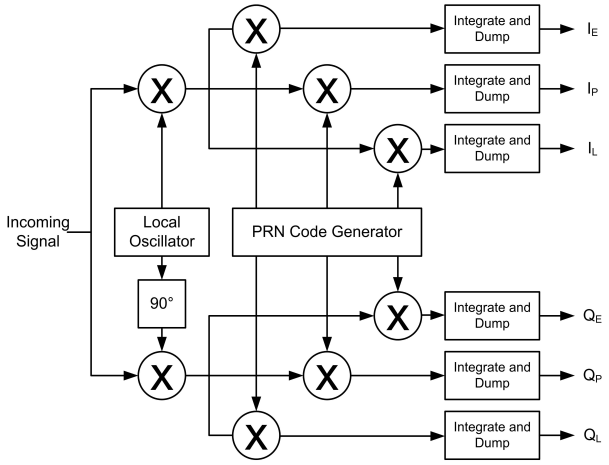

Figure 4.5: Code Tracking Loop

$$D = I_E - I_L \tag{4.13}$$

$$D = (I_E^2 + Q_E^2) - (I_L^2 + Q_L^2) \tag{4.14}$$

$$D = \frac{(I_E^2 + Q_E^2) - (I_L^2 + Q_L^2)}{(I_E^2 + Q_E^2) + (I_L^2 + Q_L^2)} \tag{4.15}$$

$$D = I_P(I_E - I_L) + Q_P(Q_E - Q_L) \tag{4.16}$$

4.2.4 Carrier Tracking

A carrier tracking loop is necessary to remove the carrier frequency modulation from the incoming signal, therefore the incoming carrier signal's frequency and phase must be accurately replicated. In a stand-alone (unaided) GPS receiver, the carrier tracking loop is typically the weakest link thus its ability to track (or stay locked) characterizes the overall ability of the GPS receiver. Therefore much care must be given to the

architecture of the carrier tracking loop. The three considered possibilities for tracking a carrier frequency are a phase lock loop (PLL), a Costas loop, and a frequency lock loop (FLL).

Immediately it can be determined that a Costas loop is desired over a PLL as the signal being tracked is binary phase-shift keying modulated, that is, there are 180 degree phase transitions at each data bit. A typical PLL will lose lock at a 180 degree phase shift, however, a Costas loop is a PLL modified to track during such a shift. While an FLL alone does not provide a high enough level of accuracy, it is often used as a back-up to a PLL because of the FLL's tolerance of dynamic stress (which compliment's the PLL's accuracy) [70]. FLL-assisted-PLL's are discussed further in Chapter 5. This chapter is limited to the Costas loop because of its insensitivity to BPSK modulation and its commonality.

In Figure 4.6, a block diagram of a Costas loop is shown. From the diagram it can be seen that the incoming signal is first multiplied by the locally generated code signal, leaving only the carrier modulated with the navigation data. This signal is then multiplied by the in-phase (I) and 90 degree out of phase quadrature (Q) components of the locally generated carrier signal. The I and Q branches are symmetrically filtered to remove the higher frequency terms resulting from the second set of multiplications. The I and Q terms before and after filtering are defined in Equations (4.17) - (4.20), respectively, where ϕ is the phase of the signal, ω_{IF} is the intermediate frequency, and n is the iteration.

$$D(n)\cos(\omega_{IF}n)\cos(\omega_{IF}n+\phi) = \frac{1}{2}D(n)\cos(\phi)+\frac{1}{2}D(n)\cos(2\omega_{IF}n+\phi) \quad (4.17)$$

$$D(n)\cos(\omega_{IF}n)\sin(\omega_{IF}n+\phi) = \frac{1}{2}D(n)\sin(\phi)+\frac{1}{2}D(n)\sin(2\omega_{IF}n+\phi) \quad (4.18)$$

$$I = \frac{1}{2}D(n)\cos(\phi) \quad (4.19)$$

$$Q = \frac{1}{2}D(n)\sin(\phi) \quad (4.20)$$

A carrier loop discriminator is used to determine the phase error between the locally generated carrier signal and the incoming signal. The most effective, but computationally taxing, method is the Arc-Tan discriminator derived in Equations (4.21) - (4.23).

$$\frac{Q}{I} = \frac{\frac{1}{2}D(n)\sin(\phi)}{\frac{1}{2}D(n)\cos(\phi)} \quad (4.21)$$

$$= \tan\phi \quad (4.22)$$

$$\phi = \tan^{-1}\frac{Q}{I} \quad (4.23)$$

The output of the discriminator is then fed back to the numerically controlled oscillator (NCO), which generates the carrier signal, closing the loop.

4.2.5 Data Demodulation

It has been shown how replica signals for the code and carrier are generated and refined. This section discusses the process of demodulating the buried data from the incoming signal. The signal, as transmitted by a single satellite, can be modeled as

$$s(t) = \sqrt{2P_C}(C(t)D(t))\cos(2\pi f_{L1}t) + \sqrt{2P_{PL1}}(P(t)D(t))\sin(2\pi f_{L1}t) + \sqrt{2P_{PL2}}(P(t)D(t))\cos(2\pi f_{L2}t) \quad (4.24)$$

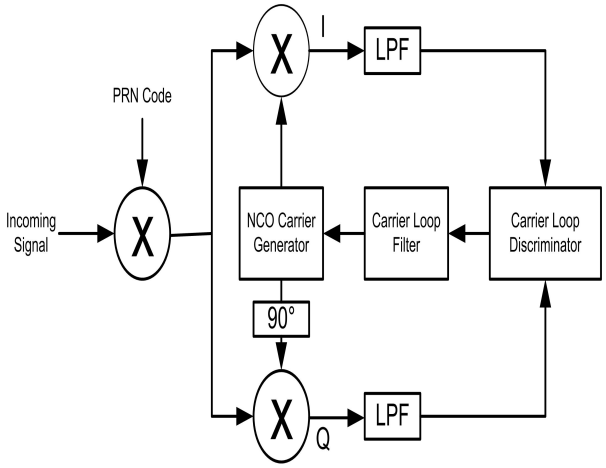

Figure 4.6: Carrier Tracking Loop

In Chapter 3, the process of an RF Front End receiving a signal and down-sampling the data was discussed. Furthermore, it is desired to use only the C/A code modulation scheme and to ignore the P-code modulation.

Because of the P-code's larger bandwidth, the filtering effects in the front-end corrupt the P-code signal such that the C/A signal, as it appears after its down conversion to an intermediate frequency and sampling (discretization denoted by 'n'), can be modeled as

$$s(n) = (C(n)D(n)) \cos(\omega_{IF} n) \tag{4.25}$$

Therefore, the incoming signal can then be simply multiplied by the locally generated carrier signal produced by the carrier tracking loop discussed earlier in the chapter. This signal is fed through a low-pass filter (LPF) to remove the higher frequency term resulting from the multiplication. The signal can now be defined as

$$s(n) = \frac{1}{2}(C(n)D(n)) \tag{4.26}$$

Likewise, the signal is multiplied by the locally generated code signal produced by the code tracking loop. Ideally, the replica code perfectly matches the incoming code. This results in the desired navigation data, D(n), as seen in Equation (4.27), which is ready to be sent to the navigation algorithm processor.

$$\frac{1}{N}\sum_{n=0}^{N-1} C(n)(C(n)D(n)) = D(n) \tag{4.27}$$

4.3 In-phase and Quadrature Signals in GPS/INS Integration

It is generally accepted that a system that aids the tracking of and integrates GPS correlator measurements with an INS is termed an Ultra-Tightly Coupled or Deeply Integrated system as was discussed in Chapter 2. It has been shown earlier in this chapter that the GPS in-phase (I) and quadrature (Q) samples can be created using signal processing techniques in a software correlator. This section gives a brief discussion of why and how these I and Q samples are essential for Ultra-Tightly Coupled (UTC) and Deeply Integrated (DI) GPS/INS.

The previous sections on code and carrier tracking show why accurate tracking loops must be continuously provided in order to produce an accurate position solution. In cases of high dynamics, these loops can lose track. In order to maintain tracking, a wider tracking loop bandwidth is necessary. Consequently, as the tracking loop bandwidth increases, the accuracy degrades due to infiltration of more noise into the signal. This dilemma can be solved using the aiding architecture of UTC or DI integration.

In traditional tracking loops, I and Q measurements are used for power calculations to switch between different loops and discriminator corrections. In [71] it is shown that these measurements can be used to estimate errors in other systems such as the INS. Their work shows that GPS correlator measurements and the INS measurements are related through phase and frequency errors. The relationship between I and Q measurements and phase and frequency errors is developed, followed by the relationship between phase and frequency errors and INS position and velocity states.

The latter of these two relationships is generally known: carrier frequency is used to approximate velocity and phase estimates can be used to approximate position once their initial integer ambiguities are determined [30]. The equations used in [71] define the dependencies as

$$w' = w(1 - \frac{v_r}{c}) \tag{4.28}$$

$$\phi' = -\frac{w}{c}(|X_s - X_u(t_0)| - v_r t_0) \tag{4.29}$$

where w' and ϕ' are the received carrier frequency and phase of the GPS signal, typically tracked using a frequency lock loop (FLL) and a Costas phase lock loop (PLL), respectively. The variable w is angular frequency, 'c' is the velocity of light, and X_s and X_u are the satellite and user positions, respectively, with v_r being the differential of their range with respect to time.

The former relationship, between I and Q and the frequency and phase errors, can be seen in [71] through a derivation of the quadrature signals, which leads to a direct relationship between I and Q measurements and position and velocity. Using a Kalman

filter, the I and Q measurements can be combined with INS measurements to develop an Ultra-tight GPS/INS system, using the following measurement model:

$$z = \{\text{INS predicted measurements}\} - \{\text{GPS measurements}\} \tag{4.30}$$

$$z = \{I + dI, Q + dQ\}_i - \{I - \eta_I Q - \eta_Q\}_i \tag{4.31}$$

$$z = \{dI + \eta_I, dQ + \eta_Q\}_i \tag{4.32}$$

"...where dI and dQ are the deviations in the INS predicted I and Q measurements caused by the inertial sensor errors, and η_I and η_Q are the quadrature noise components in the GPS I and Q measurements," [71].

4.4 Navigation Algorithms

An overview of the theory behind using the received satellite data to determine receiver position was discussed in Chapter 2. The satellite's orbital parameters, the transit time, and the initial distance approximation known as the pseudorange were defined there. This section introduces the equations necessary for the completion of the navigation algorithm.

4.4.1 Satellite Position

The initial step in solving for user location is calculating a satellite's position in time in an applicable reference frame. The position of the satellite with respect to time is calculated using the satellite ephemerides to correct its Keplerian orbit. First, the Kepler orbit must be defined, along with its parameters. Second, the satellite ephemerides are

used to define the satellite's true orbit. And third, the orbit must be translated to the applicable reference frame. In GPS, the Earth-Center Earth-Fixed (ECEF) reference frame is used.

Figure 4.7 illustrates a satellite's orbit defined by the three Keplerian orbital elements: the semimajor axis, the eccentricity of the ellipse, and the time of perigee passage, or a, e, and τ, respectively. The ephemeris data transmitted by a GPS satellite are listed in Table 4.1 [72]. The values for these parameters are found in the navigation data subframes two and three, discussed in Chapter 2.

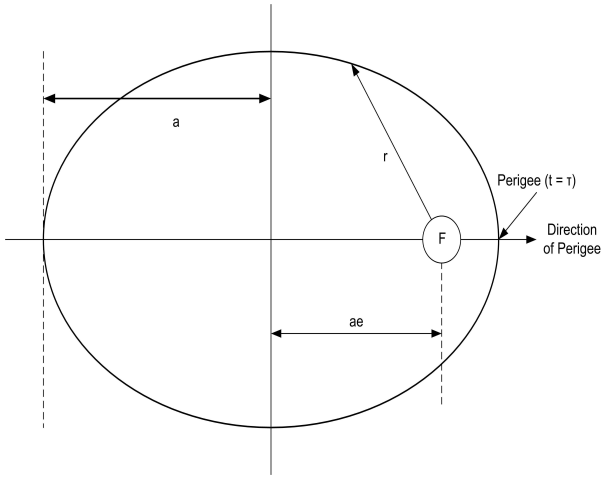

Figure 4.7: ECEF in terms of Keplerian elements

These parameters are then used to calculate the satellite's position vector in the ECEF coordinate system. The input to the calculation is t, the time of satellite transmission, and is used to calculate the time from ephemeris epoch,

$$t_k = t - t_{0e} \tag{4.33}$$

t_0e	Reference time of ephemeris
$\sqrt{a}$	Square root of semimajor axis
e	Eccentricity
i_0	Inclination angle (at time t_{0e})
Ω_0	Longitude of the ascending node (at weekly epoch)
ω	Argument of perigee (at time t_{0e})
$_0$	Mean anomaly (at time t_{0e})
di/dt	Rate of change of inclination angle
$\dot{\Omega}$	Rate of change of longitude of the ascending node
δn	Mean motion correction
C_{uc}	Amplitude of cosine correction to argument of latitude
C_{us}	Amplitude of sine correction to argument of latitude
C_{rc}	Amplitude of cosine correction to orbital radius
C_{rs}	Amplitude of sine correction to orbital radius
C_{ic}	Amplitude of cosine correction to inclination angle
C_{is}	Amplitude of sine correction to inclination angle

Table 4.1: GPS Ephemeris Data

From this point on in the calculation, a subscript k denotes that the parameter was calculated at time t_k.

The semimajor axis (a), corrected mean time (n), and mean anomaly (M_k) are calculated as shown in Equations (4.34) through (4.36).

$$a = \left(\sqrt{a}\right)^2 \tag{4.34}$$

$$n = \sqrt{\frac{\mu}{a^3}} + \Delta n \tag{4.35}$$

$$M_k = M_0 + n(t_k) \tag{4.36}$$

where μ = 398,600.5 x 10^8 m^3 / s^2.

Once the mean anomaly has been calculated, eccentric anomaly must be solved for iteratively, using

$$M_k = E_k - e \sin E_k \tag{4.37}$$

The true anomaly is defined as the angle in the orbital plane measured counter-clockwise from the direction of perigee to the satellite. This parameter cannot be solved for linearly in time and therefore must be solved for using the following two equations

$$\sin \nu_k = \frac{\sqrt{1-e^2} \sin E_k}{1 - e \cos E_k} \tag{4.38}$$

$$\cos \nu_k = \frac{\cos E_k - e}{1 - e \cos E_k} \tag{4.39}$$

Alternatively, a smart arcsine function can be used in order to determine the correct quadrant. Figure 4.8 illustrates the relationship between the eccentric anomaly and the true anomaly. Finally, once the true anomaly has been calculated, the following parameters are calculated to determine the satellite position.

Argument of latitude:

$$\phi_k = \nu_k + \omega \tag{4.40}$$

Argument of latitude correction:

$$\delta\phi_k = C_{us} \sin(2\phi_k) + C_{uc} \cos(2\phi_k) \tag{4.41}$$

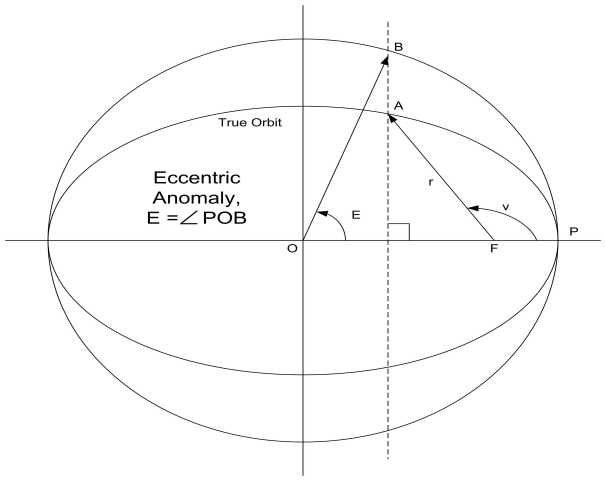

Figure 4.8: Eccentric vs. True Anomaly

Radius correction:

$$\delta r_k = C_{rs} \sin(2\phi_k) + C_{rc} \cos(2\phi_k) \tag{4.42}$$

Inclination correction:

$$\delta i_k = C_{is} \sin(2\phi_k) + C_{ic} \cos(2\phi_k) \tag{4.43}$$

Corrected argument of latitude:

$$u_k = \phi_k + \delta\phi_k \tag{4.44}$$

Corrected radius:

$$r_k = a(1 - e \cos E_k) + \delta r_k \tag{4.45}$$

Corrected inclination:

$$i_k = i_0 + (di/dt)t_k + \delta i_k \tag{4.46}$$

Corrected longitude of node:

$$\Omega_k = \Omega_0 + (\dot{\Omega} - \dot{\Omega}_e)(t_k) - \dot{\Omega}_e t_{0e} \tag{4.47}$$

where the Earth's rotation rate, $\dot{\Omega}_\epsilon$, is 7.2921151467 x 10^{-5} rad/s [25].

In-plane x position:

$$x_p = r_k \cos u_k \tag{4.48}$$

In-plane y position:

$$y_p = r_k \sin u_k \tag{4.49}$$

The satellite position can then be transformed to the ECEF coordinated system using Equations (4.50) through (4.52).

$$x_s = x_p \cos \Omega_k - y_p \cos i_k \sin \Omega_k \tag{4.50}$$

$$y_s = x_p \sin \Omega_k + y_p \cos i_k \cos \Omega_k \tag{4.51}$$

$$z_s = y_p \sin i_k \tag{4.52}$$

4.4.2 Receiver Position

Now that the satellite's position has been defined in an ECEF coordinate system, the receiver position can be calculated. The initial step in receiver position calculation is determining the satellite-to-user vector, r. Once this is determined, the satellite ECEF position vector, s, is used to determine the user ECEF position vector, u, where r = s - u. This concept is illustrated in Figure 4.9.

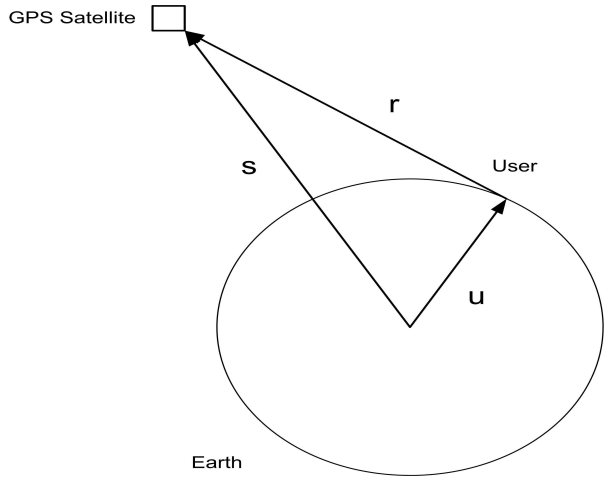

Figure 4.9: Position Vectors

The satellite-to-user vector, r, is defined as the signal transit time from satellite to user multiplied by the signal's speed, the velocity of light, c (3 x 10^8 m/s). However, the transit time from satellite to user is not perfectly known due to satellite-system clock offset and user-system clock offset, t_u. The satellite-system clock offset corrections are determined by the control segment and sent with the satellite navigation data message. The user-system clock offset is initially unknown. Therefore r is referred to as a pseudorange and can be initially expressed as

$$\rho_j = ||\mathbf{s}_j - \mathbf{u}|| + c\, t_u \tag{4.53}$$

where the subscript j ranges from 1 to N, where N is the number of possible satellites. Expanding Equation (4.53), the pseudorange of satellite j can be expressed as

$$\rho_j = \sqrt{(x_j - x_u)^2 + (y_j - y_u)^2 + (z_j - z_u)^2} + c\, t_u \tag{4.54}$$

In a navigation algorithm, the unknown user position and receiver clock offset consists of an approximation and offset, as seen in Equations (4.55) through (4.58).

$$x_u = \hat{x}_u + \Delta x_u \tag{4.55}$$

$$y_u = \hat{y}_u + \Delta y_u \tag{4.56}$$

$$z_u = \hat{z}_u + \Delta z_u \tag{4.57}$$

$$t_u = \hat{t}_u + \Delta t_u \tag{4.58}$$

This approximation can be substituted into Equation (4.54), which can be rewritten as

$$\hat{\rho}_j = \sqrt{(x_j - \hat{x}_u)^2 + (y_j - \hat{y}_u)^2 + (z_j - \hat{z}_u)^2} + c\, \hat{t}_u \tag{4.59}$$

Using a Taylor series expansion about the approximated position and clock offset and neglecting higher-order non-linear terms, the linearization of Equation (4.59) is defined as

$$\hat{\rho}_j - \rho_j = \frac{x_j - \hat{x}_u}{\hat{r}_j}\Delta x_u + \frac{y_j - \hat{y}_u}{\hat{r}_j}\Delta y_u + \frac{z_j - \hat{z}_u}{\hat{r}_j}\Delta z_u \tag{4.60}$$

This equation is simplified in order to rearrange into a matrix form using the following variables:

$$\Delta\rho = \hat{\rho}_j - \rho_j \tag{4.61}$$

$$a_{xj} = \frac{x_j - \hat{x}_u}{\hat{r}_j} \tag{4.62}$$

$$a_{yj} = \frac{y_j - \hat{y}_u}{\hat{r}_j} \tag{4.63}$$

$$a_{zj} = \frac{z_j - \hat{z}_u}{\hat{r}_j} \tag{4.64}$$

Rearranging Equation (4.60) into four linear equations (denoting four sets of satellite measurements), they are represented in matrix form using the following definitions:

$$\Delta\rho = \begin{bmatrix} \Delta\rho_1 \\ \Delta\rho_2 \\ \Delta\rho_3 \\ \Delta\rho_4 \end{bmatrix} \mathbf{H} = \begin{bmatrix} a_{x1} & a_{y1} & a_{z1} & 1 \\ a_{x2} & a_{y2} & a_{z2} & 1 \\ a_{x3} & a_{y3} & a_{z3} & 1 \\ a_{x4} & a_{y4} & a_{z4} & 1 \end{bmatrix} \Delta\mathbf{x} = \begin{bmatrix} \Delta x_u \\ \Delta y_u \\ \Delta z_u \\ -c\Delta t_u \end{bmatrix} \tag{4.65}$$

The least squares solution to this matrix representation is

$$\Delta\mathbf{x} = \mathbf{H}^{-1}\Delta\rho \tag{4.66}$$

The vector $\Delta\mathbf{x}$ contains the incremental updates for user position and receiver clock offset and are substituted back into Equations (4.55) - (4.58) to yield an updated user position and clock offset solution.

Typically more than four satellites are desired and least squares techniques are used that involve redundant measurements. In addition to least squares techniques, the Kalman Filter can also be used for determining user position and clock offset solutions. Furthermore, determining user position and clock offset using the Kalman Filter is the basis of GPS/INS integrated systems, discussed in Chapter 2. An overview of the Kalman Filter can be found in [12], among several other sources.

4.5 Summary

This chapter have provided a discussion of a software based approach to a GPS receiver. The necessity of a software based receiver and as well as its contrast to a conventional hardware based receiver was presented. Specifically, the software correlator was provided, covering the methods of signal acquisition, code and carrier tracking, and navigation data demodulation. The discussion on acquiring I and Q samples from the software is of particular interest to this thesis as these samples are vital to advanced integration architectures such as Ultra-Tightly Coupled and Deeply Integrated GPS/INS solutions. The chapter concluded by presenting the equations necessary for constructing a navigation algorithm capable of determining satellite and user position.

Chapter 5

GPS IF and IMU Data Acquisition System: Collection and Analysis

5.1 Introduction

It is the intention of this thesis to provide a test bed that is capable of providing the data needed for the testing and analysis of advanced levels of GPS/INS integration, i.e. Tightly Coupled, Ultra-Tightly Coupled, and Deeply Integrated algorithms. These algorithms were discussed in detail in Chapter 2. Chapter 3 discussed the GPS and IMU hardware required for this system, while Chapter 4 discussed the theory behind the software used for GPS satellite acquisition and tracking.

This chapter demonstrates the testing and analysis of the GPS Intermediate Frequency Acquisition System (GIAS) capable of producing a digitized intermediate frequency (IF) that can be validated using a GPS software correlator, i.e. a satellite acquisition and tracking algorithm capable of producing In-phase and Quadrature samples. It will be shown that the system described in Chapter 3 is capable of receiving, sampling, buffering, and acquiring this digitized GPS IF signal. Furthermore, methods of acquiring IMU data will be discussed which provide a means of testing and analyzing the synchronization of GPS digitized IF data and IMU measurements.

5.2 GPS IF Data and Satellite Acquisition

As discussed in Chapter 4, there are two stages of a software receiver that make it distinct from a hardware receiver: the software based satellite acquisition and the satellite tracking. In the satellite acquisition stage, parallel code phase search acquisition is used

to determine the code phase and the carrier frequency. Each satellite is searched, in series, using this search method resulting in an initial code phase and carrier frequency for each satellite that is then sent to the tracking algorithm. In the tracking algorithm, the incoming data is demodulated using the local generated code and carrier signals.

A discriminator is used to determine the accuracy of the locally generated estimates. Several types of discriminators were discussed in Chapter 4. In the tracking algorithm used in this thesis an Arc-Tan discriminator is used in the carrier tracking loop as shown in Equations (5.1) and (5.3).

$$\frac{Q}{I} = \frac{\frac{1}{2}D(n)\sin(\phi)}{\frac{1}{2}D(n)\cos(\phi)} \tag{5.1}$$

$$= \tan\phi \tag{5.2}$$

$$\phi = \tan^{-1}\frac{Q}{I} \tag{5.3}$$

If the estimates are in-phase within a predetermined acceptable range, the "energy" of the tracking loop will be redirected from an even distribution between both the prompt In-phase and Quadrature channels to only the In-phase channel, leaving a minimal amount of "energy" in the Quadrature channel. Likewise, if the estimates are 90 out of phase, the "energy" will be redirected to the Quadrature channel.

It is through this method of analyzing the carrier loop discriminator output values that the tracking of a satellite can be determined. Simply stated, if the magnitude of the I and Q channels stays equal, no satellite is being tracked. If the magnitude of the I channel becomes significantly larger than that of the Q channel, a satellite is being tracked. Furthermore, the Doppler frequency can be determined by plotting the

difference of the initial intermediate carrier frequency and the numerically controlled oscillator aided intermediate carrier frequency. The Doppler frequency along with the output of the code tracking loop discriminator is used to demodulate the incoming signal so that the navigation data signal can be extracted.

In order to adequately determine that the digitized IF samples produced by the GIAS were valid, several runs were completed in which all 32 satellites were searched with an attempt to track each one so that a distinction could be made between viewable and non-viewable satellite data. Only four runs are included here, although their results are consistent with several runs not included in this writing. Of the four runs, two were 28 milliseconds of sample data and two were one second samples.

Data sample length was determined based on the sampling frequency of the analog-to-digital converter and the number of samples acquired by the data acquisition system. As covered in Chapter 3, the digitized IF from the GP2015 chip set was sampled at 5.714 MHz and then spread across a 16-line buffer, such that the bit rate as seen by the data acquisition system is roughly 357 kHz. One full second of data at this speed requires 5,714,000 total bits of data from the Sign and Magnitude signals, or 357,000 bits from each the 16 buffered lines. The sub-second samples acquired 160,000 total bits, or 10,000 samples at each line, providing approximately 28 milliseconds of data, or nearly one and a half navigation data bits.

The first GPS data sample was taken June 26, 2006. The sample lasted roughly 28 milliseconds. As can be seen in Figure 5.1, Satellite Vehicle Number (SVN) 18 has a correlation spike at 1.397797 MHz with a code phase of 5,458. These two measurements are sent to the tracking program, which returns I and Q values, along with a Doppler shift as seen in Figure 5.2. As seen in the figure, the I channel maintains a steady-state

magnitude of roughly +/- 200 while the Q channel drops to a magnitude of roughly +/- 100. Furthermore, the Doppler frequency oscillates over a 2-3 Hz bandwidth centered on a 10 Hz shift.

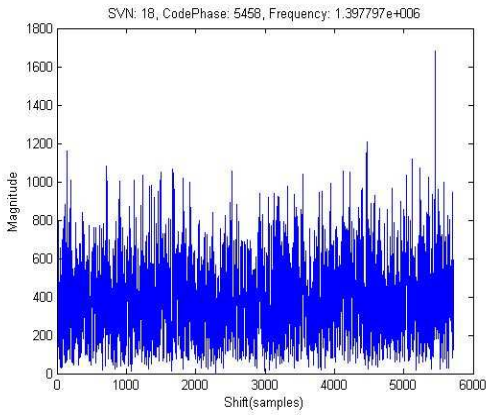

Figure 5.1: SVN 18 Acquisition on 060626

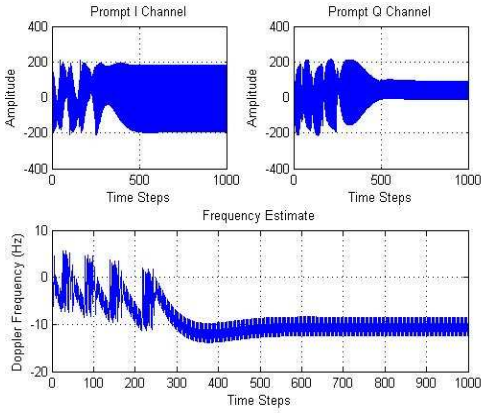

Figure 5.2: SVN 18 Tracking on 060626

To contrast SVN 18's acquisition and tracking results with a satellite known to not be viewable at the time, SVN 4's acquisition and tracking results can be seen in

Figures 5.3 and 5.4. As can be seen in the first figure, SVN 4's highest correlation spike is at 1.413947 MHz at code phase 1,027, although there are several other spikes of considerable magnitude. As seen in the second figure, the I and Q channels both maintain equal magnitude at +/- 200, with the Doppler oscillating over a 10 Hz spectrum. By contrasting the two satellites it is clear that SVN 18 is successfully acquired while SVN 4, as expected, is not.

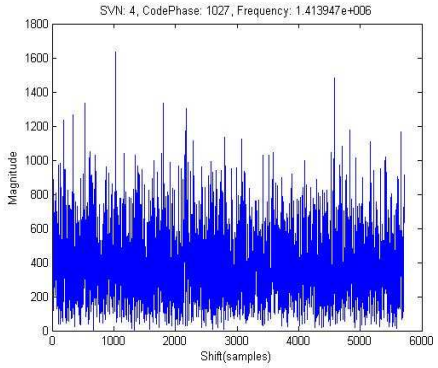

Figure 5.3: SVN 4 Acquisition on 060626

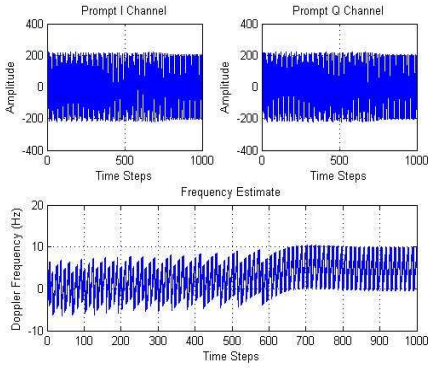

Figure 5.4: SVN 4 Not Tracking on 060626

On July 5, 2006, another 28 millisecond sample of data was taken with the GIAS. While all 32 satellites were searched for with an attempt at tracking each one, four satellites, SVN's 8, 15, 26 and 30 were located, as seen in Figures 5.5 through 5.8. This is consistent with the notion that, generally speaking, at any given time four or more satellites should be viewable.

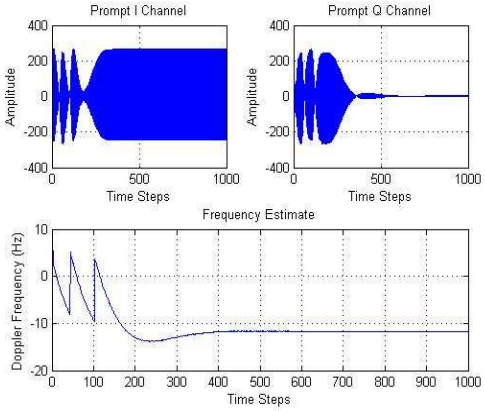

Figure 5.5: SVN 8 Tracking on 060705

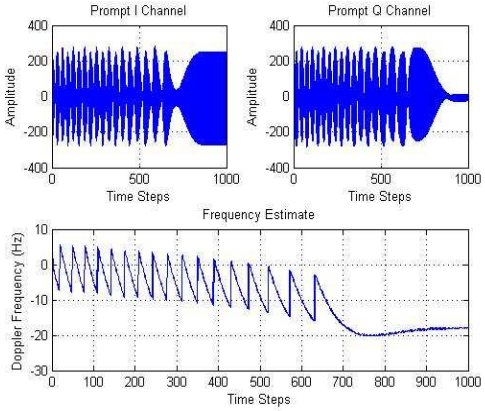

Figure 5.6: SVN 15 Tracking on 060705

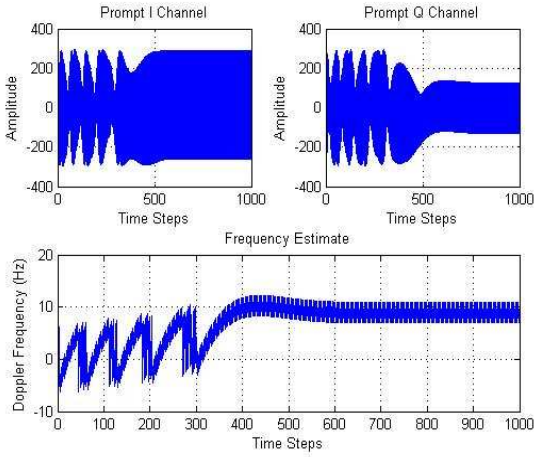

Figure 5.7: SVN 26 Tracking on 060705

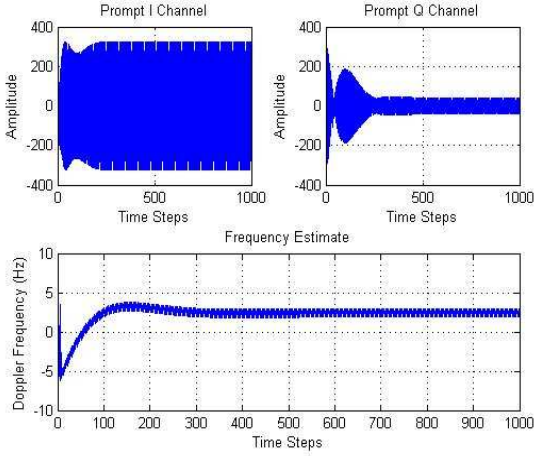

Figure 5.8: SVN 30 Tracking on 060705

Again, in contrast to the satellites successfully tracked, note that using the same 28 milliseconds of data taken on July 5, 2006, SVN 19 is seen to have not been viewable, as seen in Figure 5.9.

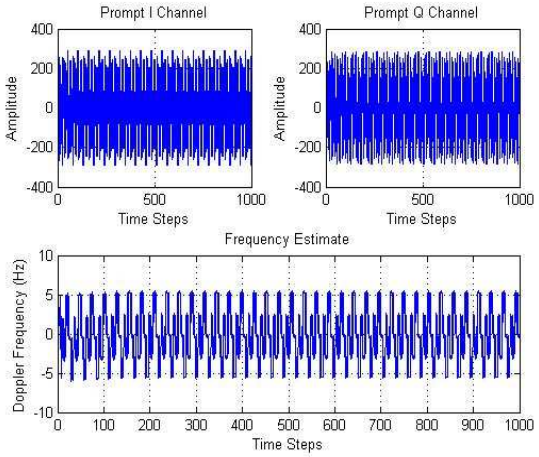

Figure 5.9: SVN 19 Not Tracking on 060705

On July 13, 2006, at 2:20 PM CST, a full second of data was collected using the GIAS. Using a hand-held Meridian GPS Magellan Unit to confirm the satellite availability, SVN 20 was acquired and tracked as seen in Figures 5.10 and 5.11. According to the acquisition algorithm, the satellite's correlation peak was located at 1.412347 MHz at code phase 4572. According to the tracking algorithm's carrier tracking loop, the Doppler shift settles to roughly a 12 Hz lag offset.

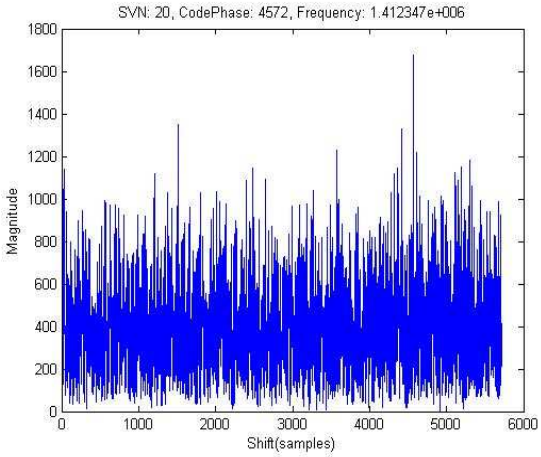

Figure 5.10: SVN 20 Acquisition on 060713

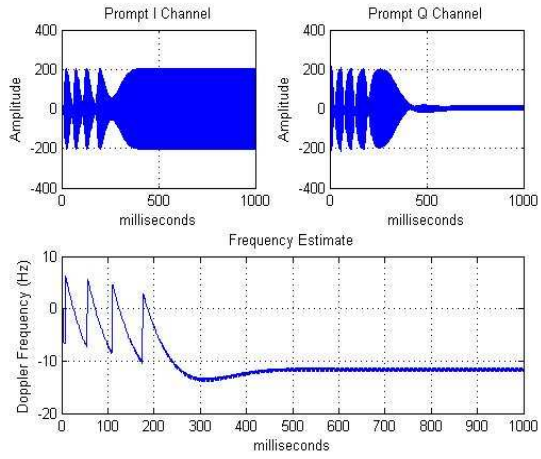

Figure 5.11: SVN 20 Tracking on 060713

A full second of data was again collected on August 10, 2006 at 10:28 AM CST. Using the Meridian GPS Magellan Unit, it was confirmed that SVN 14 would be in clear view of the GIAS's antenna. Figures 5.12 and 5.13 below show the results of the acquisition and tracking algorithms for SVN 14. According to the acquisition algorithm, the satellite's correlation peak was located at 1.414347 MHz at code phase 2258. According to the tracking algorithm's carrier tracking loop, the Doppler shift settles to roughly a 10 Hz lag offset.

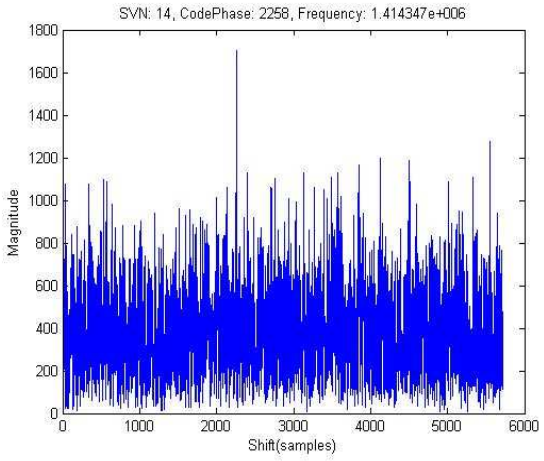

Figure 5.12: SVN 14 Acquisition on 060810

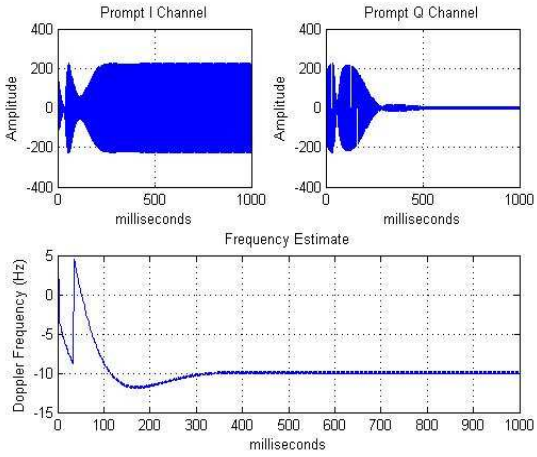

Figure 5.13: SVN 14 Tracking on 060810

5.3 IMU Data Validation

On October 13, 2006, the GIIAS collected digitized GPS IF along with IMU data using the 5.714 MHz GPS A/D sampling clock and the 100 Hz derivative clock. The test vehicle was initially accelerated to 40 MPH. Once a steady velocity was reached, a hard deceleration was induced on the vehicle at which point data from the GIIAS was collected. Results from this GIIAS data collection experiment can be seen in Figures 5.14 and 5.15.

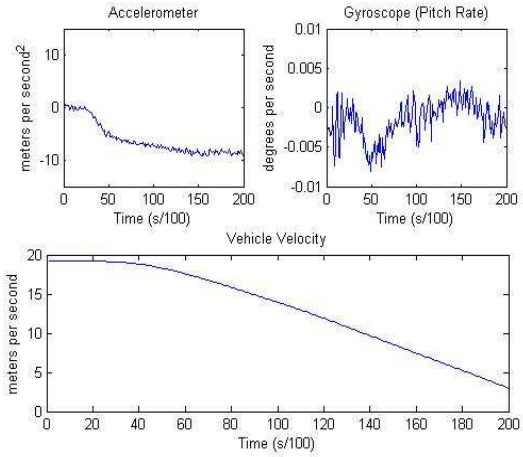

Figure 5.14: Synchronized IMU Data Collection

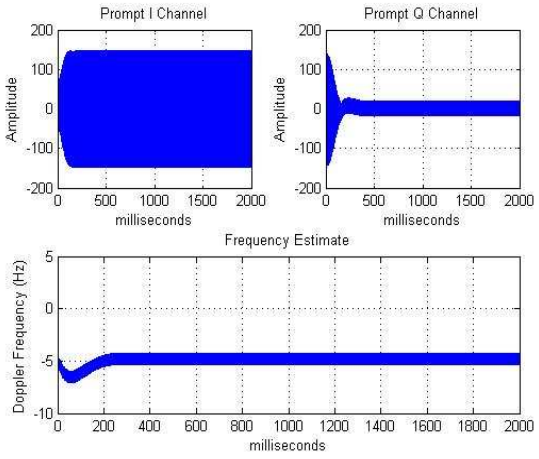

Figure 5.15: GPS Software Receiver I and Q output using GIIAS GPS Data

In Figure 5.14 the accelerometer and gyroscope data seen in the upper half of the figure were integrated to produce the vehicle velocity in the lower half of the figure, using Equations (5.4) through (5.6).

$$\Theta = \int rho_{gyro} \tag{5.4}$$

$$Accel = a_{FOV} + g * \sin\Theta \tag{5.5}$$

$$Velocity = \int Accel \tag{5.6}$$

Inferred by the validated GPS IF data collected using the 5.714 MHz A/D sampling clock and the validated IMU data collected using the 100 Hz clock derived from the 5.714 MHz clock to sample the Bosch analog IMU measurements, the GIIAS has successfully collected IMU measurements synchronized with GPS IF data. This data can now be used to test and validate the proper operation of advanced integration algorithms. Subsequently, an HG-1700 digital IMU's measurements will be logged using the same 100 Hz clock to drive the unit's A/D.

Future development of the integrated architectures presented in this thesis will allow for direct analysis and validation of the synchronization of the data collected with the GIIAS. At present, the software receiver used was limited in its capabilities particular to the tracking of the Doppler shift, preventing any direct validation based on dynamics in vehicle velocity corresponding to relative satellite velocity.

5.4 Conclusion

This chapter has covered the post processing and validation of digitized GPS IF samples through the use of the GIAS discussed in Chapter 3 and a portion of the GPS software receiver discussed in Chapter 4. It was shown that the IF samples were properly acquired and conditioned for future processing in an advanced GPS/INS integration algorithm. The varying grades of IMU's available for data collection were shown in Chapter 3 as well as methods in which these sensors' data could be properly synchronized with GPS data using a clock produced by the GIAS. The methods of acquiring the proper clock signals were discussed, i.e. the GPS sampling aligned 100 Hz clock signal.

Furthermore, results of the implementation and data collection of the GPS Intermediate Frequency and IMU Data Acquisition System (GIIAS) have been shown. Thus validating the design and development of a system for producing digitized GPS Intermediate Frequency samples and Inertial Measurement Unit samples that are synchronized in time for post-processing in advanced GPS/INS integration algorithms. Additionally, error sources and methods for modeling them were presented with the intention of both highlighting the advantages of the integrated architectures and providing some overview for error sources to be expected despite the level of integration. Future work and improvements for further system development can be found in the following chapter.

Chapter 6

Conclusions and Future Work

6.1 Introduction

Deeply Integrated GPS/INS Integration (DI) is at the forefront of modern day navigation. This thesis presented the steps to develop a test bed for the implementation and testing of DI algorithms, as well as Tightly Coupled algorithms requiring the same development hardware. Chapter 2 presented the necessary background of GPS, INS, and their integration. The design of the system was given in Chapter 3, highlighting the hardware and methods used. An overview of the software receiver used to validate the GPS IF data was presented in Chapter 4. The results of the testing and validation of the GIIAS were shown in Chapter 5. There is a great deal of research in regards to this approach beyond the scope of this writing. This chapter provides a brief discussion of future work to be done in addition to an overall summary and conclusion of this thesis.

6.2 GIIAS Cost

The components used for the GPS Intermediate Frequency and IMU data acquisition system designed and developed in this thesis have a total cost of $1500US to $1750US. This cost includes the Superstar II GPS Receiver, the printed circuit board, the integrated circuits, the Mighty Mouse 2 L1 GPS antenna, the GPS antenna amplifier/splitter, and the NI-PCI 6251 DAQ. Negating the cost of the standard GPS components, i.e. the antenna, the amplifier/splitter, and the NI-PCI 6251 DAQ, the digitized

GPS IF and IMU synchronization system costs $300US. This is a significant reduction in the cost of a commercial of the shelf (COTS) GPS IF logger.

6.3 Real-Time Acquisition

In order for any navigation algorithm to be thoroughly subjected to real-world requirements, the system must be capable of real-time data acquisition, processing, and application. Much research has been done regarding real-time GPS applications such as real-time data acquisition, real-time software correlators, and real-time navigation processors [57]. Future enhancements of the design presented in this thesis would be to develop similar systems that are capable of processing real-time GPS integration algorithms. In preparation for real-time implementation, RT-Linux, a real-time Linux operating system was investigated for use as a platform for real-time integration algorithm development [73]. It is run as a parallel operating system along with a standard Linux distribution such as Debian or Fedora. Furthermore, Comedi is a driver suite available to RT-Linux that includes a driver for the PCI-DIO-32HS data acquisition card used for this system. Appendix A discusses in detail initialization and set-up of Debian, RT-Linux, and Comedi.

6.4 Remaining Methods of Synchronizing IMU and GPS Data

There are two paths by which the GPS and IMU data can be synchronized in time. As discussed in Chapter 3, investigating the effects and benefits of using the 5.714 MHz clock provided by the GP4020 for both the GPS front-end and the INS data sampling is the first option. This shared clock system is simple to implement using a combination of

decade and binary counters, as discussed in Chapters 3, which down convert the 5.714 MHz clock to a 100 Hz clock signal.

The second path samples the analog GPS IF signal using an ADC supplied with the same clock signal used to sample the IMU data. This path is generally not recommended as the sampling of the GPS IF is already done in the GP2015 integrated circuit and sampling the signal externally only complicates the process. However, it may be beneficial to investigate this method if it becomes desirable to create a GPS IF sampler in future research. Regardless of the path chosen, synchronization of GPS and IMU's is necessary for the development of a fully integrated system.

6.5 UT/DI Algorithm Development and Testing

As referenced throughout this writing, there are currently several sources providing some insight into the requirements, development, and implementation of integrated architectures using advanced techniques for code and carrier signal tracker aiding. Although there are no explicitly defined or widely agreed upon methods for Ultra-Tightly Coupled or Deeply Integrated algorithms, one could deduce them from current literature regarding advanced levels of integration.

However, before any serious expense is invested in such developments, it should be noted once again that these methods require proficient knowledge of Kalman Filtering techniques, signal processing, and receiver design. Two significant technical difficulties in implementing a tightly coupled/deeply integrated design are presented in [15]. They are (1) the modeling of the code loop nonlinearity by the Kalman Filter and (2) loss of lock detection. These issues, among many others, must be thoroughly researched before initiating any serious design efforts.

6.6 Sensitivity Analysis using the GIIAS Platform

Once a reliable DI/UTC algorithm has been developed, it would be desirable to test several variables that are common to all levels on navigation integration to have a better understanding of the direct effects of these variables. Such variables include the quality of the IMU's used, i.e. determining the performance behaviors and limitations of an advanced architecture using consumer, tactical, automotive, and navigational grade IMU's. Additionally, oscillator quality should be investigated using several different types of clocks. The adaptability of a platform is necessary to quantize the effects of system components' susceptibility to varying level of RF interference and platform dynamics.

6.7 GPS/INS Platform Advancements

As has been done at other research institutions and universities, a printed circuit board (PCB) based on the GP2015 chip set capable of providing the Sign, Magnitude, and clock signals used in this thesis should be developed. Although such a board would add no immediate benefit, the process of designing and using such a platform would be a resource that could be used for the continuation of integration algorithm development. Furthermore, it would be beneficial to combine all of the afore mentioned attributes together into one system. This initial stages of developing this full system were discussed and tested in Chapter 3.

By developing a board that incorporates the GP2015/4020 chip set, a strap-down INS(s), a digital signal processing board (DSP) capable of processing data at very high speeds, and all necessary peripherals, integration algorithms of nearly all levels may be possible on one platform. As mentioned in a previous section, the GPS and INS systems

will need to be synchronized from the same clock source for real-time operation. Once an algorithm has been validated in post processing using data available from the current platform and an associated INS, it can be loaded onto the DSP for real-time performance analysis. Furthermore, it would be beneficial for the RF front-end portion of the board to have the capability of receiving and filtering the new L2C and L5 frequency bands.

The requirements of the DSP board must be defined before a board suitable for the system can be selected. As said before, the board must be capable of receiving a minimum of two digital inputs at 5.7 MHz, along with a corresponding clock signal. Furthermore, the board must be capable of receiving and outputting up to 32 digital inputs/outputs, with each line being capable of data acquisition up to 2 MHz. Lines 1 through 16 will be necessary for the GPS raw IF signal. Up to 6 lines may be necessary for IMU measurements and an additional two lines will be dedicated to the GPS and IMU clocks. The remaining 8 lines will be reserved for data output, i.e. the user position solution as well as other desired data.

The DSP must also be large enough to hold the files necessary to process the data, including GPS software correlator program files, GPS tracking and navigation software, IMU measurement acquisition and processing, and the advanced GPS/IMU integrated algorithms being tested.

One board meeting these preliminary requirements is the TMS320C6455-1000 Fixed-Point Digital Signal Processor, which can be found at the Texas Instruments Website [74]. The board has up 64 independent channels using an EDMA3 controller, up to 1 GHz clock rate, and over 2 megabytes of SRAM. Although this board may appear capable based on the given specifications, it should be further investigated as to what

signal characteristics are presented by the GPS and IMU measurement devices and their compatibility with the DSP board chosen.

6.8 Development of a GPS Simulator

Once an integrated platform is available, the only remaining piece of equipment to a complete UTC/DI algorithm test bed is a GPS satellite signal simulator. Currently, a commercial off the shelf (COTS) GPS simulator capable of simulating a GPS satellite constellation and accurately providing realistic signals including accurate error prediction from all "viewable" satellites is available in the $25,000US to $50,000US price range. The possession of such a resource would be very beneficial to GPS research. However, the development of such a system is a formidable task.

As GPS research advances and computer processing power escalates, it is inevitable that a highly accurate GPS satellite signal generator may become more reasonably available. At the time of this writing a first generation open source GPS simulator is available at [75]. This program produces a navigation message that can be sent to a software receiver (also available as open source software) to produce a limited position solution. However, this program is severely limited in capabilities, e.g. only 30 seconds of navigation data is available and, therefore, no almanac data, among others. Revisions are intended for future software versions.

6.9 Summary

Clearly there is a great deal of work that can be continued and improved upon from this thesis. As more advanced signal processing techniques are developed and processor computational power continues to grow, advanced methods for GPS signal tracking and

integration algorithms are inevitable. It will be important to have an adaptable real-time multi-frequency advanced GPS/INS algorithm based system in order to take advantage of these progressions. This thesis provides a foundation for the design of an improved GPS/INS system based on the system designed and developed in this work.

Bibliography

[1] FAA, "Federal aviation administration gps basics," *http://gps.faa.gov/GPSbasics/index.htm*, 2006.

[2] C. Hamm, "Analysis of simulated performance of integrated vector tracking and navigationloops for gps," Master's thesis, Auburn University, 2005.

[3] D. E. Gustafson, J. R. Dowdle, and J. John M. Elwell, "Deeply-integrated adaptive GPS-based navigator with extended-range code tracking," U. S. Patent 6,331835, The Charles Stark Draper Laboratory, Inc., Cambridge, MA, December 2001.

[4] J. M. Horslund and J. R. Hooker, "Increase jamming immunity by optimizing processing gain for gps/ins systems," U. S. Patent 5,983,160, Raytheon Company, Lexington, Massachusetts, 1999.

[5] A. S. Abbott and W. E. Lillo, "Global positioning systems and inertial measuring unit ultratight coupling method," U. S. Patent 6,516,021, The Aerospace Corporation, Segundo, CA, Feburary 2003.

[6] ZarlinkSemiconductor, "Gp2000 gps hardware design application note," *www.zarlink.com*, 2005.

[7] J. J. J. Spilker and B. W. Parkinson, "Overview of gps operation and design," *In Parkinson, B. W., editor, Global Positioning System: Theory and Applications,Volume 1, volume 163 of Progress in Astronautics and Aeronautics, Chapter2*, 1996.

[8] USNO, "Us naval observatory current gps constellation," *tycho.usno.navy.mil/gpscurr.html*, 2006.

[9] D. B. Cox, "Integration of gps with inertial navigation systems," *reprinted in Collected GPS Papers*, vol. 1, pp. 144–153, 1980.

[10] I. Warren S. Flenniken, "Modeling inertial measurement units and analtyzing the effect of their errorsin navigation applications," Master's thesis, Auburn University, 2005.

[11] R. L. Greenspan, "Gps and inertial integration," *In Parkinson, B. W., editor, Global Positioning System: Theory and Applications,Volume II, volume 164 of Progress in Astronautics and Aeronautics, Chapter7.*, 1996.

[12] R. E. Kalman, "A new approach to linear filtering and prediction problems," *Journal of Basic Engineering (ASME)*, vol. 82D, pp. 35–37, March 1960.

[13] C. Z. H. Lewantowicz, "Architectures and gps/ins integration: Impact on mission accomplishment," *Position Location and Navigation Symposium*, 1992.

[14] S. Rounds, "Jamming protection of gps receivers: Part i: Receiver enhancements," *GPS World*, 2004.

[15] J. B. Bullock, M. Foss, G. Geier, and M. King, "Integration of gps with other sensors and network assistance," *In Kaplan, E. D., editor, Understanding GPS: Principles and Applications,Second Edition, Mobile Communication Series*, pp. 459–558, 2006.

[16] D. Gustafson, J. Dowdle, and K. Flueckiger, "A high anti-jam GPS-based navigator," in *Proceedings of the Institute of Navigation National Technical Meeting*, (Anaheim, CA), pp. 495–503, Institute of Navigation, January 2000.

[17] D. Gustafson, J. Dowdle, and K. Flueckiger, "A deeply integrated GPS-based navigator with extended range code tracking," in *Proceedings of the IEEE Position, Location, and Navigation Symposium*, pp. 118–124, IEEE, 2000.

[18] D. Gustafson and J. Dowdle, "Deeply integrated code tracking: Comparative performance analysis," in *Proceedings of Institute of Navigation GPS/GNSS Conference*, (Portland, OR), pp. 2553–2561, Institute of Navigation, September 2003.

[19] C. Kreye, B. Eissfeller, and G. Ameres, "Architectures of gnss/ins integrations: Theoretical approach and practical tests," in *Symposium on Gyro Technology*, pp. 14.0–14.16, 2004.

[20] A. Soloviev, F. van Graas, and S. Gunawardena, "Implementation of deeply integrated gps/low-cost imu for reacquisition and tracking of low cnr gps signals," in *In Proceedings of the Institute of Navigation National Technical Meeting*, (San Diego, CA), pp. 923–935, Institute of Navigation, 2004.

[21] NEMCE, "Mems program: Precision guided weapons aim for increased war impact," *National Electronics Manufacturing Center of Excellence, www.empf.org/empfasis/oct03/603memsimu.htm*, 2006.

[22] B. Porat, *A Course in Digital Signal Processing.* New York: John Wiley & Sons, 1997.

[23] FAA, "Federal aviation administration gps modernization," *gps.faa.gov/gpsbasics/GPSmodernization-text.htm*, 2006.

[24] J. J. J. Spilker, "Gps signal structure and theoretical performance," *In Parkinson, B. W., editor, Global Positioning System: Theory and Applications,Volume 1, volume 163 of Progress in Astronautics and Aeronautics, Chapter3*, 1996.

[25] USDOD, "Navstar gps space segment/navigation user interfaces. iterface control documentno. icd-gps-200c," *Department of Defense*, 2000.

[26] B. C. Barker, J. W. Betz, J. E. Clark, J. T. Correia, J. T. Gillis, S. Lazar, L. K. A. Rehborn, and J. R. S. III, "Overview of the gps m code signal," *In Proceedings of the ION 2000 National Technical Meeting, Anaheim, CA*, pp. 542–549, 2000.

[27] P. Ward, J. W. Betz, and C. J. Hegarty, "Gps satellite signal characteristics," *In Kaplan, E. D., editor, Understanding GPS: Principles and Applications,Second Edition, Mobile Communication Series, Chapter 4*, pp. 113 – 151, 2006.

[28] B. Sklar, *Digital Communications: Fundamentals and Applications 2nd edition*. Prentice Hall PTR, 2001.

[29] J. J. J. Spilker, "Gps navigation data," *In Parkinson, B. W., editor, Global Positioning System: Theory and Applications,Volume 1, volume 163 of Progress in Astronautics and Aeronautics, Chapter4*, 1996.

[30] P. Misra and P. Enge, *Global Positioning System: Signals, Measurements, and Performance*. Ganga-Jamuna Press, 2001.

[31] ARINC, "Navstar gps space segment/navigation user interfaces, interface specification, is-gps-200d (public release version)," *ARINC Research Corporation, Fountain Valley, CA*, 2004.

[32] J. Taylor and E. Barnes, "Gps current signal-in-space navigation performance," in *In Proceedings of the ION 2005 National Technical Meeting*, (San Diego, CA), 2005.

[33] N. Ashby and J. J. J. Spilker, "Introduction to relativistic effects on the global positioning system.," *Parkinson, B. W., editor, Global Positioning System: Theory and Applications,Volume II, volume 164 of Progress in Astronautics and Aeronautics, Chapter18.*, 1996.

[34] G. Seeber, *Satellite Geodesy*. Berlin, Germany: Walter de Gruyter, second ed., 2003.

[35] N. Ashby and M. Weiss, "Global positioning receivers and relativity," *NIST Technical Note 1385*, March 1999.

[36] C. Rocken, "Analysis and validation of gps/met data in the neutral atmosphere," *J. Geophys. Res., 102*, pp. 29849–29866, 1997.

[37] USDOD, "Global positioning system standard positioning service performance standard," *U.S. Department of Defense*, October 2001.

[38] P. Ward, J. W. Betz, and C. J. Hegarty, "Satellite signal acquisition, tracking, and data modulation," *In Kaplan, E. D., editor, Understanding GPS: Principles and Applications,Second Edition, Mobile Communication Series, Chapter 5*, pp. 153 – 241, 2006.

[39] C. J. Hegarty, "Multipath performance of the new gnss signals," in *Proceeding of The Institute of Navigation National Technical Meeting*, (San Diego, CA), The Institute of Navigation, 2004.

[40] L. Wanninger and M. May, "Carrier phase multipath calibration of gps reference stations," in *In Proceedings of The Institute of Navigation ION-GPS-2000*, (Salt Lake City,UT), 2000.

[41] R. E. Phelts and P. Enge, "The multipath invariance approach for code multipath mitigation," in *In Proceedings of the Institute of Navigation ION-GPS-2000*, (Portland, OR), 2002.

[42] R. L. Fante and J. J. Vaccaro, "Multipath and reduction of multipath-induced bias on gpa time-of-arrival," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 39, July 2003.

[43] B. Townsend, "Performance evaluation of the multipath estimating delay lock loop," in *In Proceedings of the Institute of Navigation ION GPS-94*, (Salt Lake City,UT), 1994.

[44] C. Rizos, M. Higgins, and S. Hewitson, "New gnss developments and their impact on cors service providers," in *4th Int. Symp. & Exhibition on Geoinformation , CD-ROM Proceedings*, (Penang, Malaysia), September 2005.

[45] H. Kim, S. Bu, G. Jee, and C. Park, "An ultra-tightly coupled gps/ins integration using federated kalman filter," in *In Proceedings of The Institute of Navigation GPS/GNSS Conference*, (Portland,OR), 2003.

[46] R. J. Cosentine, D. W. Diggle, M. U. Haag, C. J. Hegarty, D. Milbert, and J. Nagle, "Differential gps," *In Kaplan, E. D., editor, Understanding GPS: Principles and Applications,Second Edition, Mobile Communication Series*, pp. 379–458, 2006.

[47] WAASPAN, "Waas performance analysis reports," *www.nstb.tc.faa.gov/ArchiveList.html*, 2005.

[48] G.-E. Demoz, *Design and Performance Analysis of a Low-Cost Aided Dead Reckoning Navigator*. PhD thesis, Stanford University, 2003.

[49] J. Nielson, "Gps aided inertial naviation," in *Proceedings of IEEE NAECON*, (Dayton, Ohio), p. 20, 1986.

[50] W. Travis, "Methods for minimizing navigation errors induced by ground vehicle dynamics," Master's thesis, Auburn University, 2006.

[51] E. M. Copps, "Optimal processing of gps signals," *NAVIGATION: Journal of The Institute of Navigation*, Fall 1980.

[52] R. Babu and J. Wang, "Performance of code tracking loop in ultra-tight gps/ins integration," *ENC-GNSS*, 2005.

[53] D. Lawrence, R. Langley, and D. Kim, "Decorrelation of troposphere across short baselines," in *In Proceedings of the IEEE and Institute of Navigation Position Locationand Navigation Symposium*, (San Diego, CA), Institute of Navigation, 2006.

[54] L. Sparks, A. Komjathy, and A. J. Mannucci, "Estimating ionospheric slant delay without resorting to the thin-shell approximation," in *In Proceedings of the IEEE and Institute of Navigation Position Locationand Navigation Symposium*, (San Diego, CA), Institute of Navigation, 2006.

[55] P. Joosten, "The effect of misspecifications in the mathematical model on carrier phaseambiguity resolution," in *Proceedings of the IEEE and Institute of Navigation Position Locationand Navigation Symposium*, (San Diego, CA), Institute of Navigation, 2006.

[56] J. B. Y. Tsui and D. M. Akos, "Comparison of direct and downconverted digitizatoin in gps receiver frontend designs," in *In Microwave Symposium Digest, IEEE MTT-S International*, vol. 3, (San Francisco,CA), pp. 1343–1346, 1996.

[57] B. Ledvina, M. Psiak, S. Powell, and P. Kintner, "Bit-wise parallel algorithms for efficient software correlation applied to a gps software receiver," *IEEE Transactions on Wireless Communications*, vol. 3, September 2004.

[58] Maxim, "2745 single-chip global positioning system receiver front-end evaluation kit data sheet," *www.maxim-ic.com*, 2006.

[59] Nemerix, "Eb1006a evaluation board for nj1006a data sheet, rev. 1.0, march," *www.nemerix.com*, 2005.

[60] ZarlinkSemiconductor, "Gp2015 gps receiver rf front end data sheet," *www.zarlink.com*, 2005.

[61] GPL-GPS, "Gpl-gps website," *gps.psas.pdx.edu/GpsHardware*, 2006.

[62] ZarlinkSemiconductor, "Gp4020 gps receiver baseband processor data sheet," *www.zarlink.com*, 2005.

[63] Novatel, "Superstar ii gps receiver user manual," *M-20000077, Rev 6.*, September 2005.

[64] J. Haggerty, "Personnal communication," 2005.

[65] R. C. Jaeger, *Microelectronics Circuit Design*. Irwin/McGraw-Hill, 1997.

[66] C. Kreye, B. Eissfeller, and T. Luck, "Improvements of gnss receiver performance using tightly coupled ins measurements," in *In Proceedings of Intl Symposium on Kinematic Systems in Geodesy, Geomatics,and Navigation*, (Banff, Alberta, Canada), pp. 15–25, 2001.

[67] T.-Y. Chiou, S. Alban, S. Atwater, J. Gautier, S. Pullen, P. Enge, D. Akos, D. Gebre-Egziabher, and B. S. Pervan, "Performance analysis and experimental validation of a doppler-aided gps/ins receiver for jpals applications," *ION GNSS 17th International Technical Meeting of the Satellite Division*, September 2004.

[68] J. B. Y. Tsui, *Fundamentals of Global Positioning System Receivers: A Software Approach*. New York: John Wiley & Sons, 2000.

[69] M. S. Braasch and A. van Dierendonck, "Gps receiver architectures and measurements," *In Proceedings of the IEEE*, vol. 87, no. 1, pp. 48–64, 1999.

[70] J. J. J. Spilker, "Satellite constellation and geometric dilution of precision," *In Parkinson, B. W., editor, Global Positioning System: Theory and Applications,Volume 1, volume 163 of Progress in Astronautics and Aeronautics, Chapter5*, 1996.

[71] R. Babu and J. Wang, "Ultra-tight gps/ins/pl integration: A system concept and performance analysis," *Submitted to GPS Solutions.*, 2005.

[72] E. D. Kaplan, J. L. Leva, D. Milbert, and M. S. Pavloff, "Fundamentals of satellite navigation," *In Kaplan, E. D., editor, Understanding GPS: Principles and Applications,Second Edition, Mobile Communication Series, chapter 2*, pp. 21–65, 2006.

[73] FSM, "Fsm labs rtlinux website," *www.fsmlabs.com*, 2006.

[74] TexasInstruments, "Tms320c6455-1000 fixed point digital signal processor," *http://focus.ti.com/docs/prod/folders/print/tms320c6455.html*, 2006.

[75] OSGPS, "An open source gps simulator," *Open Source GPS, gfz-potsdam.de/pb1/staff/gbeyerle/opengps*, 2006.

[76] K. Borre, D. M. Akos, and N. Bertelsen, *A Software-Defined GPS and Galileo Receiver: A Single-Frequency Approach*. Springer Verlag, 2006.

Appendices

Appendix A

Linux, RTLinux, Comedi

The purpose of this document is to give a brief overview, installation instructions, and operating instructions particular to the system defined in this thesis for the Debian Linux operating system, the Real-Time Linux operating system, and the Comedi driver library. The combination of these three resources provides a means for a real time data acquisition system when combined with an appropriate data acquisition board such as the one described in the thesis. Each of these resources is available free of charge.

A.1 Debian Linux

A.1.1 Debian Linux Overview

From a non-official Linux website, www.linux.org, "Linux is a free Unix-type operating system originally created by Linus Torvalds with the assistance of developers around the world. Developed under the GNU General Public License, the source code for Linux is freely available to everyone." Being that the source code of Linux is freely available, a large number of distributions are available, most of them are free of charge.

For the purposes of this thesis and its compatibility with running Real-Time Linux, Debian Linux 3.1, or Debian Sarge, was chosen. From the Debian Linux website, www.debian.org, "Debian is a free operating system (OS) for your computer. An operating system is the set of basic programs and utilities that make your computer run. Debian uses the Linux kernel (the core of an operating system), but most of the basic OS tools come from the GNU project; hence the name GNU/Linux. Debian GNU/Linux

provides more than a pure OS: it comes with over 15490 packages, precompiled software bundled up in a nice format for easy installation on your machine."

A.1.2 Debian Linux Install

Before Real Time Linux can be installed, Debian Linux must be installed first. Preferably a computer will be dedicated to a Linux OS, however it is possible to install Linux along with another operating system once the computer's hard drive has been properly partitioned.

To begin installation, the Debian Linux installation discs must be made. There are 14 install discs for Debian 3.0 and can be found at http://cdimage.debian.org/debian-cd/3.1_r0a/i386/iso-cd/. Only the first four are required for basic install. The remaining discs will only be needed for supplementary programs as desired. Save all 14 .iso files to your desktop and then burn them each to separate discs.

Detailed installation "how to" file may be found at http://www.us.debian.org/ releases/sarge/i386/apa. During the install process, be sure to select the Desktop Environment Install and create a 'Root' user and an additional user for general use.

A.1.3 Debian Linux Operation

In following is a list of some basic Linux terminal commands (Comments are made after the '#', just like Linux.):

```
shutdown -r now # reboots computer
```

```
shutdown now # shuts down computer
```

```
nano <filename># Text editor. ctrl+X closes text editor
              # ex.: nano lilo.conf

cd # moves up/back to user folder
# (as high as you can go in the hierarchy)

cd .. # moves back one folder

ls # Lists contents of current folder

cd <folder># opens <folder>
# ex.: cd linux; opens 'linux' folder
# ex.: cd /usr; opens 'usr' folder
# '/' is used when 'usr' is not in current folder

mv <file> /<folder>
# moves <file> to <folder>
# '/' is needed to designate folder not in current folder
# ex.: mv myfile /usr/src; moves 'myfile' to the 'src' folder,
# which is in the 'usr' folder

mv <file> <newname># renames 'file'
# ex.: mv file output; renames 'file'
# to 'output'

Ctrl+Alt+F1
# switches from GUI login to console login
# (switches system to run-level 1; F2 will switch to
# run-level 2, etc.)
```

For questions and support about using Debian Linux, a FAQ is available at http://www.debian.org/doc/manuals/debian-faq/. Once the Debian Linux OS has been installed the next step is to install Real-Time Linux, or RTLinux.

A.2 Real Time Linux

A.2.1 RTLinux Overview

A free distribution of Real Time Linux (RTLinux) is available from FSM Labs called RTLinux Free. From the FSM Labs RTLinux Free website, http://www.fsmlabs.com/rtlinuxfree.html, "RTLinux grew out of real-time research and open source development begun at New Mexico Tech almost a decade ago. Today, rock-solid RTLinux technology is deployed in thousands projects across a wide range of applications, including industrial control, instrumentation, communications and networking, aerospace and defense."

RTLinux must be installed "on top of" another Linux Operating System. Assuming Debian Linux from the previous section has been properly installed, RTLinux is now ready to be installed.

A.2.2 RTLinux Free Install

The installation files for RT-LinuxFree may be found at www.rtlinuxfree.com or http://www.rtlinuxfree.com/index.php?option=com_remository&Itemid=41& func=selectcat&cat=1. Download (or transfer by CD, USB, etc.) the following files to your Debian Desktop:

prepatched_linux_kernel-2.6.9-rtl.tgz

rtlinux-3.1.tgz

The first file, "prepatched_linux_kernel", is the basic Linux kernel OS that RTLinux will run "on top of". This kernel will actually run in the background in the lowest priority.

The second file, "rtlinux-3.1", is the RTLinux OS. As said before, this will be loaded on top of the Linux kernel.

Open the Terminal, usually found under Applications/Debian/XShells/ Gnome_Terminal, all depending on your Desktop Environment.

Now type the following commands:

```
# Unpack RT-Linux Install files

cd Desktop   # You are now in your Desktop Folder
# to view files saved on your Desktop.

ls # Shows files on Desktop

mv prepatched_linux_kernel-2.6.9-rtl.tgz /usr/src
mv rtlinux-3.1.1.tgz /usr/src
# Moves the two rt-linux files to /usr/src folder

cd /usr/src # Moves you to the /usr/src folder

tar zxvf prepatched_linux_kernel-2.6.9-rtl.tgz
tar zxvf rtlinux-3.1.1.tgz
# Decompresses the files

ls # Shows you the files in your current folder,
```

```
   # i.e. the new files you made

mv linux-2.6.9-rtl linux
# Renames this folder to linux, needed later
cd linux

# Configure Linux Kernel

make config
# or
make menuconfig
# or
make xconfig

# Whichever works
# Be sure to leave everything how it is unless
# you really know what you're doing.

make dep

# Compile the Linux Kernel and Modules
make bzImage
make modules

# Install Linux Kernel and Modules
make modules_install

cp arch/i386/boot/bzImage /boot/rtzImage
# Adds RT-Linux to system boot menu
```

```
# Configure LILO

cd

cd /etc

nano lilo.conf

# Opens text editor for editing lilo (Linux Loader) file


# Add the following lines to the end of the lilo.conf file


image=/boot/rtzImage

label=rtlinux

read-only

root=/dev/hda1


# WARNING:  replace root=/dev/hda1 in the above with

# your root filesystem.

# The easiest way to find out which filesystem it should be,

# take a look at the existing entry in your /etc/lilo.conf

# for "root=".

# Alternatively, type "df" at the prompt, and look for the

# line for "/"

# in the "mounted on" column.  The corresponding entry in

# the "Filesystem"

# column is your root filesystem.


cd /sbin

lilo # Configures lilo


# Restart your computer (leaving out the '-r' makes it shutdown):
```

```
/sbin/shutdown -r now
```

In order to run RTLinux, load the RTLinux kernel. In order to do this, at the 'LILO:' prompt, press "Shift" or "Tab". This will give you a listing of the available kernels. Select "rtlinux".

Continuing with the install:

```
# Compile RTLinux

cd /usr/src/rtlinux-3.1
ln -sf /usr/src/linux linux

# Configure RTLinux
make config
# or
make menuconfig
# or
make xconfig
# Whichever works.

make dep

# Compile RTLinux
make
make devices

# Load RTLinux Modules
cd
```

```
cd /usr/src/rtlinux-3.1/scripts
insrtl
```

A majority of the installation instructions are taken from a number of different websites, such as
http://www.rtlab.org/HOWTO-INSTALL.txt, http://midas.psi.ch/rtlinux/install.html, http://www.artemio.net/projects/linuxdoc/rtlinux/RTLinux-Installation-HOWTO.html. However, several additions and revisions were made to improve clarity and ease of installation.

A.2.3 RTLinux Free Operation

Now that RTLinux is installed on your PC, a good way of testing the operating system is to run a quick example.

In order to see the examples, reboot your computer and select RTLinux. When you are at the GUI Debian Login, hit Crtl+Alt+F1 to go to the Text-based Login. Log in as 'root'.

Enter the following commands into the command prompt:

```
cd
cd /usr/src/rtlinux-3.1/examples
cd hello # hello world example
nano readme # Read the help file
make hello.c
make test
```

If the example runs successfully, you are now running RTLinux with Debian Linux running in the background. If not, go through the detailed instructions referred to at the beginning of this section.

A.3 Comedi

A.3.1 Comedi Overview

Comedi gets its name from 'Linux Control and Measurement Device Interface'. According to its website, www.comedi.org, "The Comedi project develops open-source drivers, tools, and libraries for data acquisition. Comedi is a collection of drivers for a variety of common data acquisition plug-in boards. The drivers are implemented as a core Linux kernel module providing common functionality and individual low-level driver modules."

A.3.2 Comedi Install

At the time of this writing, the most recent version of Comedi is version 7.73, released on August, 16, 2006. The installation files, comedi-0.7.73.tar.gz and comedilib-0.7.22.tar.gz, can be downloaded from http://www.comedi.org/download/.

Once the file is saved to the desktop, open the command terminal and type the following commands:

```
cd /usr/src
tar xvzf comedi-0.7.73.tar.gz
tar xvzf /path/to/comedilib-0.7.22.tar.gz

cd /usr/src/comedi-0.7.73
```

```
./configure

# Your Linux source tree should be auto-detected,
# so hit enter on that question.

# Tell it your RTL Tree is:
/usr/src/rtlinux-3.1

# IMPORTANT: Say 'Y' to Real-time support (CONFIG_COMEDI_RTL).

# Next, just answer the prompts, saying 'M' for the correct driver for
your board: ni_pcidio.o

# Note:
# Comedi does not work well with emulated command timers.  Say
# NO to "RT Timer Emulation" or whatever they call it.  It should be
# the last or next-to-last question.



# Once you have finished configuring Comedi,
# compile & install it (as root):

make
make dev # (to create device nodes)
chmod 666 /dev/comedi? # to make sure user programs can open /dev/comediX
make install

# Next, compile and install comedilib (as root):
```

```
cd /usr/src/comedilib-0.7.22
make
make install
```

From http://www.rtlab.org/HOWTO-INSTALL.txt,

"At this point it may be necessary to install Qt 3.x. Go to http://www.trolltech.com and get the latest Qt 3.x version. Read the instructions on INSTALLing, in case they tell you to do something different, but essentially you do the following:"

```
# Unpack it into /usr/src (as root):

cd /usr/src
tar xvzf /path/to/the/qt/archive.tar.gz

# Configure the Qt Sources for building, in bash (as root):

cd /usr/src/qt-x11-free-x.x.x
export PATH=‘pwd‘/bin:$PATH
export QTDIR=‘pwd‘
./configure

# Now compile Qt. This takes FOREVER.

gmake
gmake install

# Now do some maintenance:
# Edit /etc/profile using the nano command and add:
```

```
export QTDIR=/usr/local/qt
```

A.3.3 Comedi Operation

On the Comedi website is a list of drivers available in the package. Included in the list is the National Instruments PCI-6533, the same data acquisition card discussed in Chapter 2 of this thesis. The driver name is 'ni_pcidio.o' and the config name is 'ni_pcidio'. Further instructions for setting up the Comedi driver and writing Comedi programs can be found at http://www.comedi.org/doc/index.html.

A.4 Conclusion

It is the intention of this paper's future work to include a real time GPS IF data acquisition system based on the RTLinux OS and the Comedi driver using the National Instruments PCI6533DIO DAQ.

Appendix B

Divide-by-N using the SN74HC191N

This provides an explanation for how one or more SN74HC191N 4-Bit Synchronous Up/Down Binary Counters can be used alone or in series to produce a divide-by-N. A single 191 alone can divide by any integer up to 16, or 2 to the power of 4. Two 191's can divide by any integer up to 256, or 2 to the power of 8 (4 + 4). Three 191's can divide by any integer up to 4096, or 2 to the power of 12 (4 + 4 + 4), et cetera.

Figure B.1 is a diagram that shows the pin connections for two 191's. As can be seen the CLK's, pin 14, are both connected to the input signal. The LOAD pins, pin 11, are all connected together along with second 191's Ripple Clock, RCO: pin 13. The first 191's count-enable, CTEN: pin 4 along with all of the down/up, D/U: pin 5, pins are both LOW, that is, connected to ground. The remaining CTEN pins are connected to the previous 191's RCO.

The only remaining connections are A, B, C, and D, pins 15, 1, 10, and 9, respectively. These are the pre-load pins that determine N, or the integer to which the network of 191's will count. To explain, an example will be used.

If the desired N is 99, then 99 is taken from 256 (2 to the power of 8 for two 191's), such that the Preload number is 157, as seen in Equation (B.1), where m is the number of 191's used.

$$2^{4*m} - N \quad = \quad Preload \tag{B.1}$$

$$256 - 99 \quad = \quad 157$$

The Preload number, 157, is then converted to its binary equivalent, 1001 1101. This number is loaded into pins A, B, C, and D, of each of the two 191's, beginning with the least significant bit at the A pin of the first 191. A '1' designates a High/Vcc input and a '0' designates a Low/GND input. For this example, the pin inputs would be as follows.

First 191:

$$\begin{array}{ll} A & 1 \\ B & 0 \\ C & 1 \\ D & 1 \end{array} \tag{B.2}$$

Second 191:

$$\begin{array}{ll} A & 1 \\ B & 0 \\ C & 0 \\ D & 1 \end{array} \tag{B.3}$$

This particular preload configuration is shown in Figure B.1.

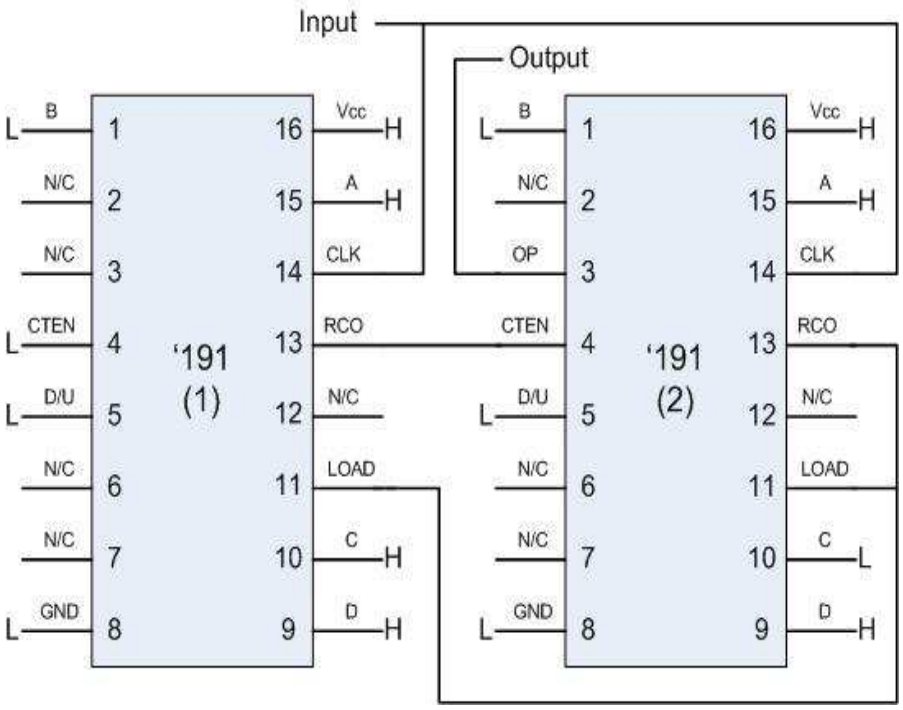

Figure B.1: Divide-by-99 using the SN74HC191N

Appendix C

Software Receiver

This document will provide the MATLAB code used to validate the GPS IF Acquisition System. An original version of this code was provided by Dr. Demoz Gebre-Egziabher and is being published near the time of this writing. The published work can be found in "A Software-Defined GPS and Galileo Receiver: A Single-Frequency Approach" by Kai Borre, Dennis M. Akos, and Nicolaj Bertelsen [76].

In following is the source code of the partial software receiver used in this thesis. The code, written by Dennis M. Akos, was based on work presented by AJRM Coenen and DJR Vannee, and has been edited from its original format by Alireza Razavi, J.T. Lee, and the author.

```
SWRCR.m


close all
clear all
clc


plot_acq    = 1; % Plot Acquisition Results:plot:1,no plot:0
plot_track  = 1; % Plot Tracking Results:plot:1,no plot:0


load 'C:\GIIAS_data\GIIAS_061003t0930_2sec_GPS.mat';


F_s             = 5.71426e6;  % Sampling frequency of GP2015,
% created by GP4020
```

```
F_c             = 1.023e6;  % Code Frequency

numofcodeperiod = 5;  % Number of 1ms Data Code Periods being
% searched over
SampleSize      = floor(F_s/1000)*6; % Size of Sample

IF              = 1.4053968254e6; % Intermediate Frequency of GP2015

RawIFData = value(1:SampleSize,1); % Data Sampled

clear value

% Acquisition Algorithm
for SVN = 1:32

[CorrFcn(SVN,:),CodePhase(SVN),Frequency(SVN),T_Elapse(SVN)]...
 = Acquisition(SVN,F_c,F_s,RawIFData,...
 numofcodeperiod,IF,plot_acq);

hgsave(cat(2,'C:\GIIAS\acq_sat_',num2str(SVN)))
SVN
close
end
save(['Acquisition_Results'])
clear CorrFcn CodePhase Frequency t_elapse RawIFData

% Tracking Algorithm

load Acquisition_Results.mat
```

```
for SVN = 1:32 % GPS_Sat
Central_Frequency = Frequency(SVN)-20;
Initial_CodePhase = CodePhase(SVN);
Time = 2000;      % in milliseconds
BW = 10;          % PLL bandwidth
N = 1;            % Coherent averaging time in milliseconds
[CarrFreqOut,P_I,P_Q,Er_phase_PLL] = CarrierTracking(SVN,...
Central_Frequency,Time,Initial_CodePhase,...
BW,N,F_s,F_c,plot_track);

hgsave(cat(2,'C:\GIIAS\track_sat_',num2str(SVN)))
close
end

%Acquisition.m

function [CorrFcn, CodePhase, Frequency, T_Elapse] = ...
Acquisition(SVN,F_c,F_s,RawIFData,numofcodeperiod,IF,plot_acq)

%   Inputs:
%   SVN - Satellite Vehicle Number
%   F_c - Code Frequency
%   F_s - Sampling Frequency
%   RawIFData - IF Data File
%   numofcodeperiod - Number of 1ms Code Periods being searched
%   IF - Intermediate or Baseband Frequency
%   plot_acq - to plot or not to plot

%   Outputs:
```

```
%   CorrFcn - Correlation Function
%   CodePhase - Phase of SVN Code being searched
%   Frequency - Intermediate Frequency with
%   Estimated Doppler Shift
%   T_Elapse - Elapsed Time to Run

format compact

% Carrier Frequency bin for searching
% for the matching Carrier Frequency.
% Contains 121 carrier frequencies (+/- 15KHz Search
% Bandwidth with 250Hz Steps in 1ms data).

StepSize        = 250;
SearchBandwidth = 15e3;
dfreq           = StepSize/numofcodeperiod;
bin             = [-SearchBandwidth:dfreq:SearchBandwidth];
CarrFreqVec     = ones([1,length(bin)])*IF + bin;
% CarrFreqVec - vector of all carrier freq's: baseband
                + all bin possibilities

N = floor(1023*F_s/F_c)*numofcodeperiod;
% N - number of samples during (1ms code period)*numofcodeperiod

% GPS satellite's C/A code generation for a codeperiod
prncode = cacode(N,F_s,F_c,SVN,0);

% A discrete fourier transform after
% taking its complex conjugate
```

```
temprn  = fft(prncode + i .* prncode);

% Load gps data for numofcodeperiod
N       = length(temprn);
data    = RawIFData(1:N)';

% Find peak correlation for the testing
% carrier frequency range by using
% circular convolution techniques

for k = 1:length(bin)
% Generate the I and Q channel components for
% 1ms*numofcodeperiod for the signal being
% translated to baseband (no carrier freq) and
% then multiply them with incoming gps signal

tempdata = fft(data .* exp(i .* 2*pi*...
CarrFreqVec(k)*[0:N-1]/F_s));

% Correlation Operation
% Store the peak correlation value in
% each carrier frequency searching

temp    = abs(ifft(tempdata .* conj(temprn)));
corr    = temp(1:N/numofcodeperiod);
peak(k) = max(corr);
end

% Post Processing
```

```
% Based on the above searching process,
% find the codephase and carrier frequency.
% Do post-correlation FFT to see whether
% or not the maximal PSD exists for the
% carrier frequency found in the search process.
% If not, increase number of code period.

[pval,pos]  = max(peak);
Frequency   = CarrFreqVec(pos);
% Maximum Correlation Peak Carrier Frequency

tempdata = fft(data.*exp(i.*2*pi*...
CarrFreqVec(pos)*[0:N-1]/F_s));

temp     = abs(ifft(tempdata .* conj(temprn)));
CorrFcn  = temp(1:N/numofcodeperiod); % Correlation Function
[pcorr,CodePhase] = max(CorrFcn);

% Maximum Correlation Peak Code Phase

if plot_acq ~=1
   return
end

str = sprintf('SVN: %d, CodePhase: %d,...
Frequency: %d',SVN,CodePhase,Frequency);
figure
plot(CorrFcn), xlabel('Shift(samples)'), ...
```

```
ylabel('Magnitude'), title(str)
return
```

CarrierTracking.m

```
function [CarrFreqOut,P_I,P_Q,Er_phase_PLL] = ...
CarrierTracking(SVN, Central_Frequency,Time,Initial_CodePhase,
BW,N,F_s,F_c,plot_track);

% Inputs:
% SVN - Satellite Vehicle Number
% Central_Frequency - Frequency Estimate from Acquisition
% Time - Run Time in milliseconds
% Initial_CodePhase - Code Phase Estimate from Acquisition
% BW - PLL Bandwidth
% N - Coherent averaging time to 1 millisecond
% F_s - sampling frequency
% plot_track - to plot or not to plot
%
% Outputs:
% CarrFreqOut - Estimated Carrier frequency
% P_I - Real part of signal after mixer
% P_Q - Imaginary part of signal after mixer
% Er_phase_PLL - Phase error

pi = atan(1)*4;
j = sqrt(-1);
codeperiods = floor(Time);
```

```
% Perform Various Initializations
% loop counter for times through the loop
loopcnt = 1;
% define carrier frequency which is used over tracking period

carrfreq = Central_Frequency;
carrfreq_basis = Central_Frequency;

% define how much carrier phase is "left over" at the end of
% each code period to reset trig argument

remcarrphase = 0.0;

% insert necessary carrier/costas loop initializations
% carrier/costas loop parameters
% initial loop

oldcarr_nco = 0.0;
olderrort = 0.0;
PDI = 0.001*N;

% Loop bandwidth in Hz
LBWcarr = BW;

% damping coefficient for 2nd order loop
zetacarr = 0.7071;
Wncarr = 2.0*LBWcarr/(zetacarr +(1/(4*zetacarr)));
% filter coefficient values
t2_div_t1 = 2.0 * zetacarr * Wncarr;
```

```
deltat_div_t1 = (Wncarr * Wncarr * PDI);


P_I = zeros(1,codeperiods);

P_Q = zeros(1,codeperiods);


CarrFreqOut = zeros(1,codeperiods);

EstimatedPhase = 0;

numofcodeperiod = 1;

searchbin = 8;


BlockLength = round(F_s/F_c*1023)*numofcodeperiod;

blksize = round(F_s/1000)*(numofcodeperiod+1);


load 'C:\GIIAS_data\GIIAS_061003t0930_2sec_GPS.mat';


GPSdata = value(1:blksize,1);

scount = length(GPSdata);

tstart = clock;

beginposition = 0;

BlockReadLength = blksize;

trigarg = zeros(1,blksize);

prn_code = prn_code_gen(SVN);


while (loopcnt <= codeperiods)

% generate the carrier frequency to mix the signal to baseband

time = (0:blksize-1) ./ F_s;

% get the argument to sin/cos functions

trigarg = ((carrfreq*2.0*pi).*time') + remcarrphase - 0*pi/180;
```

```
% Code Tracking

[uncoded_GPS,codeposition,promptcode,fine_code_time]...
 = CodeTracking(Initial_CodePhase,GPSdata.*...
 exp(j*trigarg),F_s,F_c,numofcodeperiod,SVN,searchbin,prn_code);

promptcode = promptcode';
rawdata = GPSdata(codeposition:codeposition+BlockLength-1);
GPSdata = GPSdata(codeposition+BlockLength-searchbin:end);
remcarrphase = rem(trigarg(codeposition+BlockLength-...
searchbin),(2 * pi));

beginposition = codeposition+beginposition+...
((blksize-scount)*(loopcnt~=2)+
BlockLength*(loopcnt==2)-searchbin-1)*(loopcnt~=1);

BlockReadLength = blksize-length(GPSdata);
newdata = value(1:BlockReadLength,1);
scount = length(newdata);

% EOF
if (scount ~= BlockReadLength)
disp('Not able to read the specified number of...
 data samples, exiting...')
p_i = 0.0;
p_q = 0.0;
fclose all;
return
end
```

```
GPSdata = [GPSdata;newdata];
Initial_CodePhase = searchbin + 1;

% Phase Tracking

% compute the signal to mix the collected data to baseband
carrcos = cos(trigarg(codeposition:codeposition+BlockLength-1));
carrsin = sin(trigarg(codeposition:codeposition+BlockLength-1));

% generate the two prompt standard accumulated values
% first mix to baseband
tempdatacos = carrcos .* rawdata;
tempdatasin = carrsin .* rawdata;

% now get early, late, and prompt values for each
P_I(loopcnt) = sum(promptcode .* tempdatasin);
P_Q(loopcnt) = sum(promptcode .* tempdatacos);

% implement carrier loop discriminator
% (phase detector) (arc-tan discriminator)
errort(loopcnt) = atan(P_Q(loopcnt)./P_I(loopcnt))/(2.0 * pi);

% implement carrier loop filter and generate NCO command
carr_nco = oldcarr_nco + (t2_div_t1  *...
(errort(loopcnt) - olderrort))...
+ errort(loopcnt) * deltat_div_t1;
oldcarr_nco = carr_nco;
olderrort = errort(loopcnt);
```

```
% modify carrier freq based on NCO command
carrfreq = carrfreq_basis + carr_nco;
carrfreqout(loopcnt) = carrfreq;
loopcnt = loopcnt+1;
end

fclose all;
Er_phase_PLL = atan(sin(errort*2*pi)./cos(errort*2*pi));

if (plot_track == 1)
figure
subplot(2,2,1),plot(P_I)
grid
hold on
xlabel('milliseconds')
ylabel('Amplitude')
title('Prompt I Channel')
subplot(2,2,2),plot(P_Q)
grid
hold on
xlabel('milliseconds')
ylabel('Amplitude')
title('Prompt Q Channel')
subplot(2,1,2),plot(carrfreqout-Central_Frequency);
% Subtract out the nominal IF to get doppler freq plotted
% Central_Frequency is the IF freq from the acquisition
% of the sat carrfreqout is the carrfreq_basis + the
% carr_nco measurement, i.e. Central_Frequency + carr_nco;
```

```
% plot(carr_nco)
grid
xlabel('milliseconds')
ylabel('Doppler Frequency (Hz)')
title('Frequency Estimate')
end

CodeTracking.m

function [uncoded_GPS,codeposition,code,fine_code_time]...
 = CodeTracking(ICP,GPSdataIn,fs,cr,numofcodeperiod,...
 svno,searchbin,prn_code)

% Inputs:
% ICP - Initial Code Position
% svno - satellite prn number
% cr   - code frequency
% fs   - sampling frequency
% GPSdataIn - gps raw datafile
% numofcodeperiod - number of 1ms data code period,
% should be an integer
% carrier_ferquency - carrier ferquency
% searchbin -
% prn_code - SV prn code
%
% Outputs:
% uncoded_GPS uncoded GPS Signal
% codeposition - code phase in samples
% code - GPS C/A code
```

```
% fine_code_time - fine code time
%


% number of samples during
% (1ms code period)*numofcodeperiod
N = round(1023*fs/cr)*numofcodeperiod;
GPSdata = GPSdataIn(ICP-searchbin:ICP+N+searchbin-1);
% generate C/A code
code = code_sample(prn_code,N,fs,cr,0);
code = code';
% Compute correlation
for bin = -searchbin:searchbin
Corr(bin+searchbin+1) = abs(sum(GPSdata(bin+searchbin+1:...
bin+searchbin+N).*code));
end;
% Find maxumum of Corroleation Function
[maxvalue,codeposition] = max(Corr);
codeposition = codeposition - searchbin - 1;
% Check initial position
if abs(codeposition) == searchbin
warning('wrong initial position')
fine_code_time = ICP+codeposition;
uncoded_GPS = GPSdata(codeposition+searchbin+1:N+...
codeposition+searchbin).*code;
codeposition = ICP+codeposition;
code = code';
return
end
```

```
%DLL implements
earlyI = real(sum(GPSdata(codeposition+searchbin:...
codeposition+searchbin+N-1).*code));
earlyQ = imag(sum(GPSdata(codeposition+searchbin:...
codeposition+searchbin+N-1).*code));
promptI = real(sum(GPSdata(codeposition+searchbin+1:...
codeposition+searchbin+N).*code));
promptQ = imag(sum(GPSdata(codeposition+searchbin+1:...
codeposition+searchbin+N).*code));
promptI = real(sum(GPSdata(codeposition+searchbin+2:...
codeposition+searchbin+N+1).*code));
promptQ = imag(sum(GPSdata(codeposition+searchbin+2:...
codeposition+searchbin+N+1).*code));
t = 1/fs;
fine_time_model_coefs = inv([t^2,-t,1;0,0,1;t^2,t,1])...
*Corr(codeposition+searchbin:codeposition+searchbin+2)';
fine_time = -fine_time_model_coefs(2)/2/...
fine_time_model_coefs(1);
fine_code_time = fine_time/t+codeposition;
codeposition = round(fine_code_time);

%change for fine time position
uncoded_GPS = GPSdata(codeposition+searchbin+1:...
searchbin+N+codeposition).*code;
codeposition = round(fine_code_time) + ICP;
code = code_sample(prn_code,N,fs,cr,(fine_code_time...
-round(fine_code_time))*t);

cacode.m
```

```
function sample_ca = cacode(N, F_s, F_c, SVN, offset)
% Generates a GPS C/A code with an offset.
% Used by Acquisition.m

% Inputs:
% N - Number of samples
% SVN - the Satellite's PRN number
% F_c - code rate
% F_s - sampling frequency
% offset - offset

% Output:
% sample_ca - sampled C/A code

% The g2s vector holds the appropriate shift
% to generate the C/A code
g2s = [5;6;7;8;17;18;139;140;141;251;252;254;...
       255;256;257;258;469;470;471;472;473;...
       474;509;512;513;514;515;516;859;860;861;862];

g2shift = g2s(SVN,1);

% Generate G1 code
% load shift register
reg = -1*ones(1,10);
for i = 1:1023
g1(i) = reg(10);
save1 = reg(3)*reg(10);
```

```
reg(1,2:10) = reg(1:1:9);
reg(1) = save1;
end

% Generate G2 code
%  Load shift register
reg = -1*ones(1,10);
for i = 1:1023
    g2(i) = reg(10);
    save2 = reg(2)*reg(3)*reg(6)*reg(8)*reg(9)*reg(10);
    reg(1,2:10) = reg(1:1:9);
    reg(1) = save2;
end

% Shift G2 code
g2tmp(1,1:g2shift) = g2(1,1023-g2shift+1:1023);
g2tmp(1,g2shift+1:1023) = g2(1,1:1023-g2shift);
g2 = g2tmp;

% Form single sample C/A code by multiplying
% G1 and G2 point by point
ca = g1.*g2;

% Apply code rate, sampling frequency, & offset
b = [1:N];
c = mod(ceil((b/F_s+offset)*F_c),1023);
c(find(c==0)) = 1023;
ca = ca(c);
sample_ca = ca;
```

prn_code_gen.m

```
function sample_ca = prn_code_gen(svnum)
% Generates a GPS C/A code; no offset.
% Used by CarrierTracking.m

% Input Arguments:
% svnum - the Satellite's PRN number

% Output Argument:
% sample_ca - sampled ca code

% The g2s vector holds the appropriate shift
% of the g2 code to generate

g2s = [5;6;7;8;17;18;139;140;141;251;252;...
254;255;256;257;258;469;470;471;472;473;...
474;509;512;513;514;515;516;859;860;861;862];

g2shift = g2s(svnum,1);

% Generate G1 code
reg = -1*ones(1,10);
for i = 1:1023,
    g1(i) = reg(10);
    save1 = reg(3)*reg(10);
    reg(1,2:10) = reg(1:1:9);
    reg(1) = save1;
```

```
end

% Generate G2 code
reg = -1*ones(1,10);
for i = 1:1023,
    g2(i) = reg(10);
    save2 = reg(2)*reg(3)*reg(6)*reg(8)*reg(9)*reg(10);
    reg(1,2:10) = reg(1:1:9);
    reg(1) = save2;
end

% Shift G2 code
g2tmp(1,1:g2shift) = g2(1,1023-g2shift+1:1023);
g2tmp(1,g2shift+1:1023) = g2(1,1:1023-g2shift);
g2 = g2tmp;

% Form single sample C/A code by multiplying
% G1 and G2 point by point

ca = g1.*g2;
sample_ca = ca;
```

code_sample.m

```
function sample_ca = code_sample(ca,n,fs,c_rate,offset)
% Generates offset GPS C/A code.  Used by CodeTracking.m

% Input Arguments:
% ca - prn code
```

```
% n - Number of samples
% fs - sampling frequency
% c_rate - code rate
% offset - prn code offset

% Output Argument
% sample_ca - sampled ca code

b = [1:n];
c = mod(ceil((b/fs+offset)*c_rate),1023);
c(find(c==0)) = 1023;
code = ca(c);
ca = code;
sample_ca = ca;
```