FOCUSED INVESTIGATIONS OF RELATIVISTIC ELECTRON BURST INTENSITY, RANGE, AND DYNAMICS SPACE WEATHER MISSION GLOBAL POSITIONING SYSTEM

by

Mackenzie Charles Wilz

A professional paper submitted in partial fulfillment
of the requirements for the degree

of

Master of Science

in

Electrical Engineering

MONTANA STATE UNIVERSITY
Bozeman, Montana

January 2011

APPROVAL

of a professional paper submitted by

Mackenzie Charles Wilz

This professional paper has been read by each member of the professional paper committee and has been found to be satisfactory regarding content, English usage, format, citation, bibliographic style, and consistency and is ready for submission to the Division of Graduate Education.

Dr. Joseph Shaw

Approved for the Department of Electrical and Computer Engineering

Dr. Robert Maher

Approved for the Division of Graduate Education

Dr. Carl A. Fox

STATEMENT OF PERMISSION TO USE

In presenting this professional paper in partial fulfillment of the requirements for a master's degree at Montana State University, I agree that the Library shall make it available to borrowers under rules of the Library.

If I have indicated my intention to copyright this professional paper by including a copyright notice page, copying is allowable only for scholarly purposes, consistent with "fair use" as prescribed in the U.S. Copyright Law. Requests for permission for extended quotation from or reproduction of this professional paper in whole or in parts may be granted only by the copyright holder.

Mackenzie Charles Wilz

January 2011

ACKNOWLEGEMENTS

I would like to thank Dr. Joseph Shaw for serving as my committee chair and for guiding me throughout my graduate program. I would also like to thank Dr. Robert Maher for serving on my committee and teaching me so many valuable things in my college career and Dr. David Klumpar for serving on my committee, for his guidance, and for giving me so many opportunities while I worked at the Space Science and Engineering Laboratory at Montana State University, including working on this project.

I would like to acknowledge all my coworkers at SSEL, especially Ehson Mosleh. Thanks for your friendship and for all that you have contributed in to completion of this project.

I would also like to acknowledge the Electrical and Computer Engineering Department and all its faculty, staff, and professors for the excellent learning environment and for all the help that was provided to me during my years at Montana State University.

I would like to express gratitude to my family and friends for their support. I would not have made it this far without them, and they have all impacted my life for the better. They have all stood by my side in the hardest of times and have made me the person I am today.

DEDICATION

I would like to dedicate this paper to my parents, Guy and Sallie Wilz. Dad, you have taught me the strength of the human spirit and that with that strength all obstacles can be overcome. Mom, you have shown me the value of emotion and that if that emotion is true you can accomplish anything. You both have given me more than I could ever ask for and have always been there when I needed someone the most. Your unconditional support can never be repaid and the lessons you have taught me throughout my life are invaluable. It is because of you I have been able to accomplish all my goals.

TABLE OF CONTENTS

TABLE OF CONTENTS - CONTINUED

LIST OF TABLES

Table Page

LIST OF FIGURES

Figure Page

ABBREVIATIONS

CDH:	Command and Data Handler
FIREBIRD:	Focused Investigations of Relativistic Electron Burst Intensity, Range, and Dynamics
GPS:	Global Positioning System
IC:	Integrated Circuit
LNA:	Low Noise Amplifier
LVTTL:	Low Voltage Transistor-Transistor Logic
MIPS:	Millions of Instructions Per Second
PCB:	Printed Circuit Board
PLL:	Phase Lock Loop
RAM:	Random Access Memory
SPI:	Serial Peripheral Interface
SSEL:	Space Science and Engineering Laboratory
TTL:	Transistor-Transistor Logic
UART:	Universal Asynchronous Receiver Transmitter

ABSTRACT

The FIREBIRD mission (Focused Investigations of Relativistic Electron Burst Intensity, Range, and Dynamics) is a low earth orbit, space weather, CubeSat mission which is comprised of a two satellite constellation. This constellation is responsible for the measurement of relativistic electron microbursts with very fine spatial and temporal resolution. To achieve the spatial and temporal requirements of the mission, a global positioning system (GPS), for the purpose of navigation position and timing, is to be implemented on both satellites within the constellation. The integration and testing of this subsystem is integral to the mission's success. The GPS hardware must be capable of fulfilling the requirements of the mission in order for the science data to be interpreted reliably. This means that the GPS hardware must not only be accurate but precise as well. Also, a driver must be implemented in software in order for this data from the GPS hardware to be received, interpreted, and stored by the command and data handling subsystem.

INTRODUCTION

The SSEL FIREBIRD mission, sponsored by the National Science Foundation, requires an accurate way of finding on-orbit position and time due to the mission's statement. In the mission statement, it is stated that the mission will assess the spatial size and energy coherence of relativistic electron microbursts. These two statements yield two major mission requirements: The FIREBIRD mission shall 1) take two physically separate point measurements of relativistic electron microbursts over a variety of separations between 1.5 km and 150 km with an accuracy of ± 250 m and 2) measure microbursts at less than 200 ms resolution at each location. These two mission requirements, in turn, give way to two major flight system requirements: The FIREBIRD mission shall 1) have collection timing faster than 200 ms with 1% accuracy with a goal of 20 ms and 1% accuracy and be designed to allow accurate reconstruction of clock timing of observations and 2) be capable of deriving on-orbit absolute position of each spacecraft to ± 175 m accuracy. To satisfy these requirements, in-flight GPS systems are being implemented in both satellites within the FIREBIRD constellation. Other considerations had been made, but the GPS solution was chosen because of its overall precision and accuracy. The mission's spatial sampling will be resolved by a combination of the temporal sampling rate of the payload data, the dynamic satellite separation, and the GPS system. Due to the complexity of the GPS system, many considerations had to be made about the hardware and software of the system. Firstly, the hardware had to be chosen to be compatible with one another. Also, standard GPS hardware was not a solution due to restrictions which are implemented in the software of

most receivers. With these software restrictions in place, such as a ceiling on the altitude and velocity values, a GPS system solution was implausible, so a vendor that allowed these restrictions to be lifted had to be found. Finally, the software had to be written with multiple levels of error checking for any complications that may occur during testing or while in flight.

FIREBIRD MISSION STATEMENT

"The FIREBIRD mission (Focused Investigations of Relativistic Electron Burst Intensity, Range, and Dynamics) is a targeted, goal-directed, space weather CubeSat mission to resolve the spatial scale size and energy dependence of electron microbursts in the Van Allen radiation belts. Relativistic electron microbursts appear as short durations of intense electron precipitation measured by particle detectors on low altitude spacecraft, seen when their orbits cross magnetic field lines which thread the outer radiation belt. Previous spacecraft missions (e.g., SAMPEX) have quantified important aspects of microburst properties (e.g., occurrence probabilities), however, some crucial properties (i.e., spatial scale) remain elusive owing to the space-time ambiguity inherent to single spacecraft missions. While microbursts are thought to be a significant loss mechanism for relativistic electrons, they remain poorly understood, thus rendering space weather models of Earth's radiation belts incomplete. FIREBIRD's unique two-point, focused observations at low altitudes, that fully exploit the capabilities of the CubeSat platform, will answer three fundamental scientific questions with space weather implications: What is the spatial scale size of an individual microburst?; What is the energy dependence of an individual microburst?; and How much total electron loss from the radiation belts do microbursts produce globally?" [1]

FIREBIRD GPS SYSTEM HARDWARE

The FIREBIRD satellites are a complex system of a combination of hardware and software. The heart of the hardware is the command and data handler board. This board holds all of the in-house programmed flight software and is responsible for controlling all other hardware modules in the system. In the case of the GPS system, this would include communicating through a 4-line SPI interface to a SPI to UART converter and finally through UART to the GPS receiver card. Also within the hardware system, there are two switches. One switch is the switch for the low noise amplifier (LNA) for the GPS antenna. The other switch, which also acts as a step down regulator, is a power switch for the GPS receiver card. The GPS system can be a large sink when it comes to the power budget of the satellite. Because of this, the GPS system is only turned on when needed, thus the use of switches for the amplifier and the receiver card. The GPS systems power characteristics can be seen below in table 1. This table includes maximum power consumption, efficiency, and orbital duty factor for each piece of hardware within the system. The CDH board has been omitted due to the fact that it is not solely used for the GPS subsystem.

Table 1: GPS Power Budget

Load Characteristics:	*Current:*		*Current:*		Efficiency:	*Orbit Average*	***Peak*** *Sub-system*		***Average*** *Sub-system*	
Component:	*Direct from BAT(8.0V):*		*+3.3V Regulator:*		*+3.3V Reg.:*	*Duty Factor:*	*Power:*	*Units:*	*Power:*	*Units:*
GPS Card:	0	amps	0.333	amps	75%	20%	1.465	watts	0.293	watts
Low Noise Amplifier:	0.1	amps	0	amps	100%	20%	0.800	watts	0.160	watts
SPI to UART CONVERTER (MAX3100):	0	amps	0.0004	amps	80%	100%	0.002	watts	0.002	watts
System Load Characteristics:	*Orbit Average*		***Peak*** *Op-mode*		***Average*** *Op-mode*				***Average*** ***Single Orbit***	
Operational Mode:	*Duty Factor:*		*Power:*	*Units:*	*Power:*	*Units:*			*Power Consumption:*	
GPS ON/LNA ON:	20%		2.267	watts	0.453	watts				
GPS ON/LNA OFF:	0%		1.467	watts	0.000	watts			**0.455**	watts
GPS OFF/LNA OFF:	80%		0.002	watts	0.001	watts				

NovAtel OEMV-1 GPS Card

The NovAtel OEMV-1 is the off the shelf GPS receiver card that was chosen to be used on the FIREBIRD satellites for this mission. The OEMV-1 has limited capability for standard users, but for the FIREBIRD mission, the limiting software was pulled so that raw GPS data can be retrieved. The limiting software restricted the maximums for the altitude and velocity and limited the accuracy of all the measurements which are taken. The very nature of the mission, the high speeds of orbit, the high altitudes of orbit, and the high sampling rate for the science data, made the decision to remove the limiting software trivial. This along with the OEMV-1's many operational modes and communication interfaces made it an optimal candidate for the FIREBIRD mission's GPS system's receiver. The OEMV-1 is not part of the electronic hardware card stack because it does not comply with the CubeSat form factor. Because of this non-compliance issue, it is used in the system as a daughter card for the in-house designed SSEL GPS Board.

SSEL GPS Board

The SSEL GPS board is an in-house designed custom PCB specifically made for the FIREBIRD mission. This board was fashioned to work hand in hand with the form factor of the OEMV-1 and the CubeSat standard in order to ensure consistent mechanical and electrical connectivity. Also, this board's design contains a cutout for the GPS antenna to be attached to the OEMV-1. Also, upon population, the board contains three

other main components for the GPS system, the SPI to UART converter, the GPS power switch/buck converter, and the low noise amplifier power switch.

SPI to UART Converter

The MAX3100EEE+ is an SPI to UART converter from Maxim semiconductor. It includes a crystal oscillator with a software programmable clock divider to make all common baud rates possible. The MAX3100EEE+ also has an 8 word FIFO buffer. This part is used within the system to convert the SPI hardware communication protocol that is used by the PIC24 microcontroller to the UART hardware communication protocol that the OEMV-1 uses. The MAX3100EEE+ is connected as seen below in figure 1. Pins 1-4 of the integrated circuit are the lines that are dedicated to the SPI interface that is coming from the PIC24 microcontroller. The dedicated UART lines to talk to the OEMV-1 are found on pins 14 and 15, and pin 10 is a reference for UART 4. Pins 8 and 16 are for the ground and power signals respectively with a power save feature located on Pin 7 that allows for the converter to be shut down immediately without finishing the current transmission, if applicable. This in turn reduces supply current to the leakage current values.

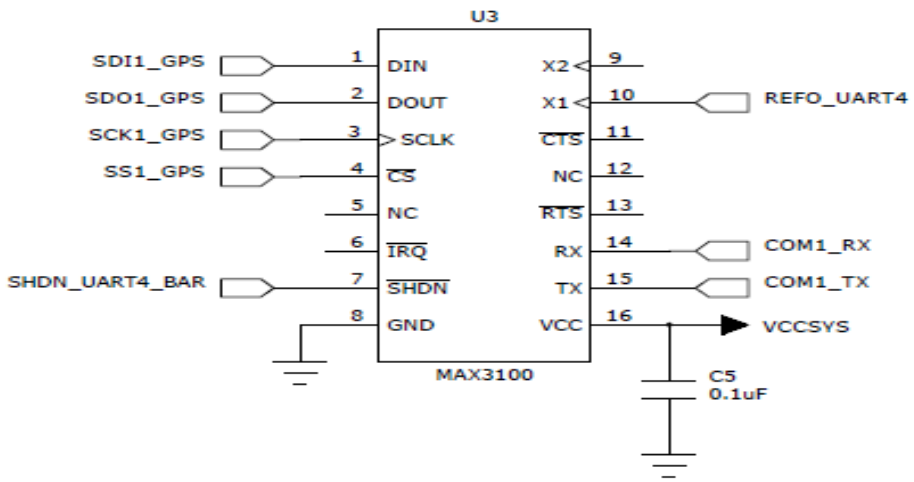

Figure 1: SPI to UART Converter Schematic

GPS Power Switch/Regulator

The MIC4680-3.3YM is the power switch and step down voltage regulator for the OEMV-1 GPS card. This IC takes in a TTL signal and a power source, in this case the 8.4 V unregulated power source, VBATT, and steps down the source through the switch if the TTL logic is low. The MIC4680-3.3YM is connected as seen below in figure 2. Pin 1 is shown to be the shutdown (SHDN) pin. This pin is an active low TTL logic input to the system which acts as the trigger to the switch. VBATT is inputted into pin 2. This line provides 8.4 volts to be stepped down to the 3.3 volts needed to power the GPS card. Pin 3, the feed forward switch line (SW), and pin 4, the feedback switch line, are used to regulate the voltage in the system along with the Schottky diode, the inductor, and the capacitor .

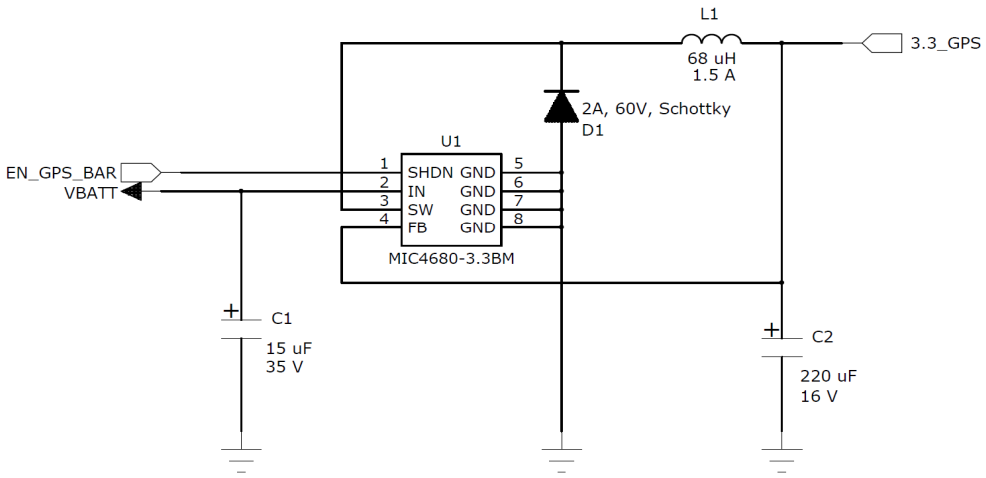

Figure 2: GPS Power Switch/Regulator Schematic

LNA Power Switch

The MIC2514YM5 is an integrated high side switch produced by Micrel. This switch has a TTL logic input which switches on the GPS antenna's low noise amplifier. The LNA uses the satellite's unregulated power signal, VBATT, which has a common value of 8.4 volts. This level of 8.4 volts is well within the specification of the MIC2514YM5 which can switch a voltage ranging from 3 volts to 13.5 volts. The LNA power circuit is located on the OEMV-1 card. The power is then transmitted to the antenna, where the LNA is located, and amplifies the signal received from the GPS constellation. The MIC2514YM5 is connected as follows in figure 3.

It can be seen from the figure that the power is inputted into pin 1 and outputted to pin 4 if pin 3 is TTL logic high. The ground pin for the device is pin 2, and pin 5 is a no connect.

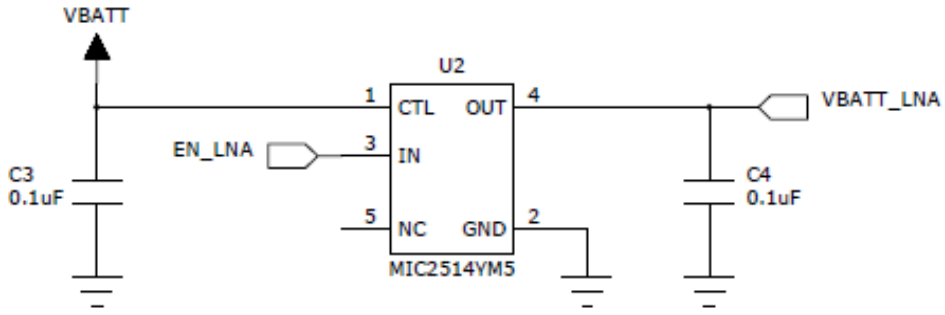

Figure 3: LNA Power Switch Schematic

Pumpkin CDH Board

The command and data handling (CDH) board which was chosen for the FIREBIRD mission is commercially available, CubeSat compliant, FM 430 board purchased from Pumpkin, Inc. This board contains a PIC24 microcontroller, a FLASH RAM chip, and an SD card reader. This allows the software to process and store data in multiple formats all on one board. Though the GPS data will be stored in the SD card memory, the details of the SD card system and interface will not be discussed, for they are not necessary for GPS operation, only for data storage.

PIC24 Microcontroller

The PIC24FJ256GA110 microcontroller is the brain of the FIREBIRD system. This small microcontroller runs at 2 mW/MIPS which is an extremely affordable when it comes to a satellite's flight computer. It is a 16-bit, modified Harvard architecture that can reach 16 MIPS when run at 32 MHz. This 32 MHz clock is accomplished by using a phase lock loop (PLL) on the internal 8 MHz oscillator. FIREBIRD uses this PLL to reach these higher speeds and to allow for faster inter-hardware communication. All of the SSEL written drivers lie within the program memory of this microcontroller. The GPS driver itself takes up 1845 bytes of program memory. This is equivalent 0.72% of the total memory of the system.

Antenna

A flight antenna has not yet been chosen for the FIREBIRD mission, but for testing purposes the NovAtel 2G15A-XT-1-N antenna is being used. This antenna is a 2.6” diameter, L1 band antenna. The L1 band specifications state the signal will have a frequency of 1575.42 ± 12 MHz. Within the antenna housing, there is a 33dB LNA for transmission amplification. Also, the antenna has a hemispherical pattern with right hand polarization and a beam width of 102 degrees in free space. This antenna uses a standard TNC connector in order to connect to the MCX connector of the OEMV-1 board. All testing that occurs with this GPS system will use this antenna until a flight antenna has been procured for the mission.

Communication

Communication is an integral part of the FIREBIRD GPS system. This includes both reception of signal on the OEMV-1 from the GPS constellation of satellites as well as hardware communication between the OEMV-1 and the PIC24 on the CDH board. The OEMV-1 receives data from the GPS constellation via an RF link through an antenna and communicates through UART to the MAX3100EEE+ which is then converted to SPI to interface with the FM 430 CDH board.

GPS RF Communication

The FIREBIRD GPS system communicates to the GPS constellation of satellites via an RF link. The GPS constellation has 6 different orbits at an altitude of 20200 km. Though the GPS system can receive a signal from any satellite within its field of view the GPS will not have a solution unless 4 or more of these satellites are transmitting data to the FIREBIRD GPS system.

Hardware Interface Communication

Though the OEMV-1 has many options for hardware communication, the default for its COM_1 port is a LVTTL UART interface. This interface is compatible with the PIC24 microcontroller, but due to having more UART interfaced peripherals than UART ports, not all of the peripherals could work with the UART standard. Therefore, it was decided to put the GPS interface on an SPI line. This was done by adding the MAX3100EEE+ SPI to UART converter. This integrated circuit converts the 4-line SPI interface that is coming from the PIC24 microcontroller to the necessary 2-line LVTTL

UART interface that the OEMV-1 card is expecting. To ensure that no data is lost due to differing baud rates, the MAX3100EEE+ has an internal FIFO. Other than the conversion that must take place, the hardware to hardware communication is very simple. The PIC24 asks the OEMV-1 for a specific log which contains that data of interest, either position, velocity, or time. The OEMV-1 then sends back a response that is an acknowledgment that echoes the original transmission back with an acknowledge byte and then sends the data log. This data log is then stored by the CDH system.

FIREBIRD GPS SYSTEM SOFTWARE

All of the FIREBIRD GPS software will be located in the program memory of the FM 430 board and will run on the PIC24 microcontroller. The CDH will be using the Salvo real time operating system (RTOS) from Pumpkin, Inc. The Salvo RTOS is a non-preemptive operating system which can use semaphores, messages, and message queues to context switch. In the FIREBIRD CDH code, message queues were chosen as the best option for context switching as multiple messages will need to be passed between tasks. These messages include, but are not limited to, data to be stored, commands to be executed, and data to be interpreted in order to execute conditional commands.

GPS Driver

The GPS driver is responsible for nearly all critical tasks on which the GPS system relies. The main function of the driver receives a message and then parses the message in order to execute a global function within the driver. Upon completion of one of these global functions, another task's message queue is signaled, if needed. The signaling of another task's message queue is dependent upon the function that is called, the status of the message, and the solution status of the GPS card. All of the tasks that communicate with the GPS card verify the message status and the solution status before signaling another task's message queue. The global functions that can be executed can be seen below in table 2. In order to communicate through SPI to the NovAtel OEMV-1, the GPS driver has an SPI driver embedded within it. The GPS driver code can be seen in Appendix A.

Table 2: GPS Driver Global Function List

Function Name	Function Description
GPS_GetAll	Gets both position and velocity logs, parses the logs, and concatenates the two logs' pertinent information into one structure to be sent to the data handler for storage
GPS_GetPos	Gets the position log, parses the log, and stores useful information to be sent to the data handler for storage
GPS_GetVel	Gets the velocity log, parses the log, and stores useful information to be sent to the data handler for storage
GPS_GetTime	Gets the UTC time log, parses the log, and stores useful information to be sent to the data handler for storage
GPS_GetOpMode	Gets the operational mode of the GPS card and stores the data to be sent to the requester
GPS_SetOpMode	Sets the operational mode of the GPS card

SPI Driver

The SPI driver is a hardware communication interface driver that is embedded within the GPS driver. The SPI driver itself is not a task and is solely responsible for communicating to the MAX3100EEE+. The driver takes in a length and a data pointer and uses it to create the four signals needed to communicate through the 4-line SPI interface standard. A new function was written within the baseline SPI driver that receives information from the GPS card and uses its terminating character to end the transmission rather than length. This function is called getsGPSSPI2. The baseline SPI driver was provided as a supplement to the FM 430 CDH board from Pumpkin, Inc. The SPI driver can be seen in appendix B.

TESTING

The GPS system is the most complex aspect to guarantee the spatial sampling that is required by the mission constraints. Because of its complexity, a series of tests was run in order to ensure the operational concept of the system. In orbit, the GPS system will be powered on when needed rather than staying on to remain within the power constraints of the system. The timing to powering on the system is critical as it will take up to 3 minutes for the GPS system to produce a lock on the GPS constellation and come up with a stable solution. Upon receiving a lock, a trigger pin on the GPS system will notify the system that a solution has been reached. Data acquisition from the GPS receiver can then begin. To ensure on-orbit operation of the system, a system of tests was devised. Firstly, each component of the system must be safe to mate tested. This test ensures that the hardware is in working order and is compatible to be run with all other pieces of hardware. Also, a functional test will be run on the OEMV-1 to ensure that it operates as expected. The software for the GPS driver must also be tested. This will ensure that communication can occur between the CDH board and the GPS receiver card through the MAX3100EEE+. This test also verifies the operation of the system as a whole in non-orbit conditions. Finally, the GPS system will be functionally tested in an on-orbit GPS simulator. This simulator mimics the GPS constellation and allows for orbital data to be entered as parameters for the simulation. This test will be used to guarantee that the operational concept of the system is sufficient for the GPS system to work while the FIREBIRD mission is in orbit.

NovAtel OEMV-1 Safe to Mate Test

The GPS card safe to mate was formed and completed to verify that the NovAtel OEMV-1 could successfully be implemented as a daughter card of the SSEL GPS Board without hardware mismatches or complications. This test consisted of three components: an unpowered GPS card 20 pin header test, an unpowered antenna port test, and a powered GPS card 20 pin header test. Each component of the test was crucial in seeing if the NovAtel was compatible with the SSEL GPS board and the test antenna. The first part of the test, the unpowered GPS card 20 pin header test, was used to verify that the impedances of each pin within the header matched the expected value. This along with the powered GPS card 20 pin header test verifies there are no flaws in the hardware so that the NovAtel card can be safely mated with the SSEL GPS board. The unpowered antenna port test is used to affirm the expected impedances of the antenna port. This proves that it safe to plug in an antenna, specifically the test antenna, to this port for functionality testing. Finally, to again ensure the NovAtel card can be safely mated with the SSEL GPS board, a powered GPS card 20 pin header test was completed. This test verified the voltage levels of the GPS card are compliant with the header on the SSEL GPS board. This ensures that no parts of the circuit are shorted or open so that, upon a successful mate, the GPS card does not destructively affect any of the circuitry on the SSEL GPS board. As the latest run of this test, all impedances and voltage levels were found to be within the tolerances of the system. The newest as run of the NovAtel

OEMV-1 safe to mate test, which was completed on November 13, 2010, can be found in appendix C.

SSEL GPS Board Safe to Mate Test

Much like the NovAtel OEMV-1 safe to mate, the SSEL GPS board safe to mate was created to verify that the SSEL GPS board was compliant with the CubeSat form factor board stack. This test consisted of two components, an unpowered test and a powered test. Three major components of the satellite are being connected to the GPS board: the GPS card, the payload board stack, and the electronics card stack. Therefore, each one of these tests had three subsections. Each subsection was meant to test the compatibility of the GPS board with one of the other three components of the satellite that will be connected to it. The GPS card, the payload board stack, and the electronics card stack are connected using a 20 pin header, a 21 pin micro D-sub connector, and a PC/104 connector respectively. During the unpowered test, the impedances of all three connectors were measured to verify if they matched the theoretical values. Next, power was applied to the GPS board, and voltage measurements were taken on all pins in all three headers. The combination of these two tests minimizes the chance of hardware failure. During the run of the powered test, it was found that there was a mistake in the schematic and layout of the GPS board. It was found that the analog and digital inputs of the LNA power switch were wired inversely. This problem was temporarily overcome by having an electronics technician cut and bridge the traces on the board correctly. The technicians work was then verified before further testing could be completed. This error can be seen in the newest as run of the SSEL GPS board safe to mate test, which was

completed on November 23, 2010, in appendix D. The GPS board will require a second rev in order to fix this error that was found. Also, the satellites' attitude control systems are planned to be added to the GPS board in the second rev of the build.

NovAtel OEMV-1 Functional Test

The NovAtel OEMV-1 functional test was devised to guarantee that the OEMV-1 GPS card was able to receive a signal from the GPS constellation, find a solution, and calculate the position of the card. This was done by placing an ESD mat, a power supply, a power strip, a laptop, and the NovAtel GPS card on a cart and taking it outdoors to receive a GPS signal. This was intended to be done as close to the SSEL K7MSU ground station antenna, which is located on the east side of the Cobleigh Hall roof, as possible, but due to inclement weather the measurement had to be done under the nearby EPS buildings awning. The EPS building is connected to the south side of Cobleigh Hall and the awning is located on the west side of the building. Therefore, the measurement was taken a few hundred feet to the southeast of the definitive ground station location. The OEMV-1 card was successful in receiving signal from the GPS constellation, getting a solution, and determining the coordinates of the GPS card. Table 3 below shows the coordinates that were determined for the EPS awning and the known coordinates of the K7MSU ground station.

Table 3: GPS Functional Test Coordinate Comparison

Location	Latitude	Longitude	Altitude
EPS Building Awning	45.666°	-111.047°	1500 m
K7MSU Ground Station	45.667°	-111.046°	1530 m

From the values in table 3, it can be seen that the GPS system is functioning as expected and is very accurate. The location is within one thousandth of a degree of the known location of the K7MSU ground station, but the variance is in the correct direction of the relative location of the test site from the K7MSU ground station. Also, the altitude values have a relatively large difference, but the K7MSU antenna sits on top of the roof of six story Cobleigh Hall building. Therefore, this difference in altitudes is to be expected. With the coordinates calculated from the GPS card, the NovAtel card was found to be fully functional.

GPS Driver Functional Test

The GPS driver, see code in appendix A, has been proven to be functional. All of the functions within the code perform as expected. To test this, the SPI lines have been scoped and upon function calls the driver can transmit and receive SPI structured communication signals. On the other hand, the GPS system in its current state is not fully functional. Errors in the GPS board's layout have been found and noted, and an engineering change order is in progress to document these issues. Because of this, there is an error in the functionality with the SPI to UART converter, so the data communication cannot be converted between protocols. All other aspects of the subsystem have been tested have been tested individually and have proven to work correctly.

FUTURE WORK

As of right now, the GPS subsystem is not fully functional and additional work is required for full operation. During testing and analysis of the first revision of the GPS subsystem, many functional problems arose in all but one major component of the subsystem: the NovAtel GPS card. The NovAtel GPS card was functionally tested and works as expected in first order stationary testing. The other three components of the subsystem have issues that need to be resolved before progress can continue.

GPS Antenna

A GPS flight antenna has yet to be chosen for the FIREBIRD mission. From STK analysis, it was found that a 60° FOV, ±100 MHz Doppler Shift, and -65 dB received power gave the GPS subsystem a 90% duty factor which is more than sufficient for the mission. These values should be used as a baseline for the decision of what flight antenna to use for the mission.

SSEL GPS Board

There are several issues on the first revision of the GPS board that need to be addressed. Firstly, the LNA power switch inputs are switched. This means that EN_LNA should go to the CTL pin, pin 1, of the MIC2514YM5 and that the VBATT line should be connected to the IN pin, pin 3, of the device. This can be easily fixed by swapping the lines in the schematic and following the change through the design. Next, the CSK headers, H1 and H2, are in the wrong order on the board which makes it unable to mate

with the CDH board. In other words, H1 is where H2 supposed to be and vice versa. Switching the placement of these to headers will then allow a mate to occur to the CDH board without the use of jumper wires. Also, it was discovered that the MAX3100 poses two problems, the baud clock for the device cannot be generated without a crystal and that it is not a straight through device so the buffer may not be large enough for the transmissions required. To solve both these problems, the MAX3107 should be used in place of the MAX3100. This device is slightly larger and consumes more power than the MAX3100, but it has an internal oscillator for baud rate generation and has a larger buffer, 128 bytes, which allows for entire transmissions to be stored within the buffer which will eliminate buffer overflow. One thing to make sure of when creating the schematic for the MAX3107 circuit is that the communication line pins are named in reference to the chip not the device to which they are going. For example, the RX pin for the UART on the MAX3107 must be connected to the TX line of the GPS card, not the RX line. Finally, the attitude control system needs to be added to the GPS board because of the limitations of space in the board stack.

GPS Driver

The GPS driver will have to be updated to fit the operation of the MAX3107. Firstly, a hardware trigger interrupt must be added in order to know when data is in the receive buffer on the MAX3107. Also, within this interrupt, a transmission sequence must be added to receive the data from the buffer. Finally, within the init function of the driver, a transmission sequence must be added to initialize the baud rate generator and

configuration options of the MAX3107. When a rework of the code is completed, full software functionality testing should be completed to make sure all cases are handled by the software. Also, the GPS driver may undergo on-orbit simulation testing. This testing process simulates on orbit characteristics of the GPS constellation through the orbit of the satellite and will verify that the subsystem is functional under orbital conditions.

SUMMARY

Although the system is not yet fully functional, substantial progress was made on the project, design obstacles were overcome, and design errors were found and documented. Many components of the subsystem have been proven functional. The function of the GPS driver and SPI driver was tested and verified by using an oscilloscope to view 4-lines of the SPI communication. Also, in its functional test, the NovAtel OEMV-1 GPS card received signals from the GPS constellation, a solution was reached, and the position, time, and velocity were calculated. Finally, errors were found within the design of the SSEL GPS board and were noted. This will assist the board designers in creating the next revision of the board and simplify the retesting process along the way. Therefore, the next revision of the design will be expedited, and in turn, the GPS subsystem's completion will be hastened as well.

REFERENCES CITED

[1] University of New Hampshire, and Montana State University. *Collaborative Research: CubeSat: Focused Investigations of Relativistic Electron Burst Intensity, Range, and Dynamics (FIREBIRD)*. Proposal. National Science Foundation, 6 May 2010. Web. 10 Nov. 2010. http://www.nsf.gov/awardsearch /showAward.do?AwardNumber=1035642&WT.z_pims_id=12808>.

[2] Maxim Integrated Products. *MAX3100*. Datasheet. Maxim-IC, Jan. 2009. Web. 28 Oct. 2010. <http://pdfserv.maxim-ic.com/en/ds/MAX3100.pdf>.

[3] Micrel Inc. *MIC2514*. Datasheet. Micrel, Aug. 2008. Web. 28 Oct. 2010. <http://www.micrel.com/_PDF/mic2514.pdf>.

[4] Micrel Inc. *MIC4680*. Datasheet. Micrel, March 2008. Web. 30 Oct. 2010. <http://www.micrel.com/_PDF/mic2514.pdf>.

[5] Microchip Technology Inc. *PIC24FJ256GA110 Family Data Sheet*. Datasheet. Microchip Technology Inc., 18 Nov. 2010. Web. 5 Dec. 2010. <http://ww1.microchip.com/downloads/en/DeviceDoc/39905e.pdf>.

[6] NovAtel Inc. *2G15A-XTB-1-N_D.pdf*. Datasheet. AntCom, 9 Aug. 2007. Web. 1 Nov. 2010. <http://antcom.com/productsheets/2G15A-XTB-1-N_D.pdf>.

[7] NovAtel Inc. *OEMV Family Installation and Operation Users Manual*. Datasheet. NovAtel Inc., 7 Oct. 2010. Web. 18 Oct. 2010. <http://www.novatel.com/assets/Documents/Manuals/om-20000093.pdf>.

[8] NovAtel Inc. *OEMV Family Firmware Reference Manual*. Datasheet. NovAtel Inc., 14 May 2010. Web. 6 Sept. 2010. <http://www.novatel.com/assets/Documents/Manuals/om-20000094.pdf>.

APPENDICES

APPENDIX A

GPS DRIVER CODE

GPS.h

```
/*****************************************************************************
*****************************************
*
*            Space Science and Engineering Laboratory
*                    Montana State University
*
*                          FIREBIRD
*
*
* Filename      : gps.h
* Source File(s) :
* Description   :
* Author(s)     : Mack Wilz
* Date          : 10/15/2010

*****************************************************************************
*****************************************/
#include "GenericTypeDefs.h"

#ifndef __GPS_H_
#define __GPS_H_

#define BESTPOSCOMMAND "log com1 bestpos once 0 0 hold\r\0"
#define BESTVELCOMMAND "log com1 bestvel once 0 0 hold\r\0"
#define TIMECOMMAND "log com1 time once 0 0 hold\r\0"
#define RESPONSETEST "\r<OK\r"

/*****************************************************************************
*****************************************
*                                  Typedefs
*****************************************************************************
*****************************************/
typedef struct {
        UINT8* position;
        UINT8* velocity;
} GPSDATA_TYPE;

typedef struct {
        UINT8* ptr;
        UINT8  length;
} DATA_TYPE;

/*****************************************************************************
*****************************************
*                                  Prototyptes
*****************************************************************************
*****************************************/

void                GPS_Main                ( void );
DATA_TYPE           GPS_GetAll              ();
```

```
DATA_TYPE               GPS_GetPos              ();
DATA_TYPE               GPS_GetVel              ();
DATA_TYPE               GPS_GetTime             ();
void                    GPS_Init                ();
DATA_TYPE               GPS_SetOpMode           ( UINT8 );
DATA_TYPE               GPS_GetOpMode           ();

#endif /* __GPS_H_ */
```

GPS.c

```
/*****************************************************************************
*******************************************
 *
 *          Space Science and Engineering Laboritory
 *              Montana State University
 *
 *                      FIREBIRD Software
 *
 * Filename     :  gps.c
 * Header(s)    :  architecture.h
 * Description  :
 * Function(s)  :
 * Authors(s)   :  Mack Wilz
 * Date         :  09/16/2010
*****************************************************************************
*******************************************/
#include "architecture.h"
#include "string.h"

/*****************************************************************************
*******************************************
* DESCRIPTION:      Main function for GPS
*
* ARGUMENTS  :      none
*
* RETURNS    :      none
*
* NOTES      :      none
*****************************************************************************
*******************************************/
void GPS_Main ( void )
{
        static PACKET_TYPE          recievedPacket;
        static PACKET_TYPE          replyPacket;
        static OStypeMsgP     msgP;
        static DATA_TYPE    data;

        while ( 1 )
        {
                /*Wait Message Queue for Full Testing and Flight*/
                //OS_WaitMsgQ(GPS_MSGQ, &msgP, OSNO_TIMEOUT);
```

```
		//recievedPacket = *(PACKET_TYPE *)msgP;

		/*Create receivedPacket for solo testing*/
		recievedPacket.SrcAddr = 0x0301;		/* Source Address
	*/
		recievedPacket.DestAddr = 0x0303;		/* Destination
Address  */
		recievedPacket.SrcPort = 0x01;			/* Source Port
		*/
		recievedPacket.DestPort = 0x01;			/* Destination
Port		*/
		recievedPacket.CRC = 0;				/*
CRC				*/
		recievedPacket.Seq = 0;				/*
Sequence			*/
		recievedPacket.Length = 0;			/* Length
			*/
		recievedPacket.Data = 0;			/* Data 0-245
bytes	*/

		switch ( recievedPacket.DestPort )
		{
			case 0x00:
				data = GPS_GetAll();
				break;
			case 0x01:
				data = GPS_GetPos();
				break;
			case 0x02:
				data = GPS_GetVel();
				break;
			case 0x03:
				data = GPS_GetTime();
				break;
			case 0x04:
				data = GPS_GetOpMode();
				break;
			case 0x05:
				GPS_SetOpMode(*recievedPacket.Data);
				break;
			default:
			;
		}

		if (recievedPacket.DestPort < 0x05)
		{
			replyPacket.DestAddr = recievedPacket.SrcAddr;
			replyPacket.SrcPort = recievedPacket.DestPort;
			replyPacket.CRC = recievedPacket.CRC;
			replyPacket.Seq = 0;
			replyPacket.Data = data.ptr;
			replyPacket.Length = data.length;
```

```
                        if(recievedPacket.DestPort < 0x04)
                        {
                                replyPacket.DestAddr = recievedPacket.SrcAddr &
0xFF00 | 0x0003;
                                replyPacket.DestPort = 0x03;
                        }
                        else
                        {
                                replyPacket.DestAddr = recievedPacket.SrcAddr;
                                replyPacket.DestPort = recievedPacket.SrcPort;
                        }


                        /*Signal Message Queue for Full Testing and Flight*/
                        //OS_SignalMsgQ(SWITCH_MSGQ, (OStypeMsgP)
&replyPacket);

                        /*Send out GSE UART for Solo Testing*/
                        UART2_Puts ( replyPacket.Data, replyPacket.Length );
                }
        }
}

DATA_TYPE GPS_GetAll ()
{
        GPSDATA_TYPE gpsData;
        DATA_TYPE pos = GPS_GetPos ();
        DATA_TYPE vel = GPS_GetVel ();
        DATA_TYPE    data;

        gpsData.position = pos.ptr;
        gpsData.velocity = vel.ptr;

        data.ptr = (UINT8*) &gpsData;
        data.length = pos.length + vel.length;


        return data;
}

DATA_TYPE GPS_GetPos ()
{
        UINT8 crcount;
        UINT8 resplength;
        UINT8 poslength;
        UINT8* response;
        UINT8* position;
        DATA_TYPE    data;

        //create message for GPS
        UINT8 *command = BESTPOSCOMMAND;
        UINT8 length = strlen(BESTPOSCOMMAND);
```

```
//Test with loop
//UINT8 *command = RESPONSETEST;
//UINT8 length = strlen(RESPONSETEST);

/*SPI2 code for when issues with SPI to UART converter are resolved*/
//Transmit message
/*gps_open();
putsSPI2(length, command);
gps_close();

//Receive BestPOS log
resplength = getsGPSSPI2(response,100);
poslength = getsGPSSPI2(position,100);*/

/*UART version of GPS use while SPI to UART is not working
 *Note: Uses Comm UART line will not work with Comm implemented*/
UART1_Puts ( command, length );

crcount = 2;
resplength = 0;
while(crcount)
{
        *(response+resplength)=UART1_GetChar();
        if (*(response+resplength)==15)
        {
                crcount--;
        }
        else if (!*(response+resplength)==0)
        {
                resplength++;
        }
}

crcount = 2;
poslength = 0;
while(crcount)
{
        *(position+poslength)=UART1_GetChar();
        if (*(position+poslength)==13)
        {
                crcount--;
        }
        if (*(position+poslength)==0)
        {
                poslength++;
        }

}

UINT8 resbool = strcmp(response,"\r<OK\r");

if(resplength < 3)
{
```

```
                data.ptr = "Error: The response transmission timed out\r";
                data.length = 43;
        }
        else if(poslength < 3)
        {
                data.ptr = "Error: The log transmission timed out\r";
                data.length = 38;
        }
        else if(resbool != 0)
        {
                data.ptr = "Error: The message and/or response was not correct\r";
                data.length = 51;
        }
        else
        {
                data.ptr = position;
                data.length = poslength;
        }

        return data;
}

DATA_TYPE  GPS_GetVel ()
{
        UINT8 resplength;
        UINT8 vellength;
        UINT8 *response;
        UINT8 *velocity;
        DATA_TYPE    data;

        //Build message to get BestVEL log
        UINT8 *command = BESTVELCOMMAND;
        UINT8 length = strlen(BESTVELCOMMAND);

        //Test with loop
        //UINT8 *command = RESPONSETEST;
        //UINT8 length = strlen(RESPONSETEST);

        /*SPI2 code for when issues with SPI to UART converter are resolved*/
        //Transmit message
        /*gps_open();
        putsSPI2(length, command);
        gps_close();

        //Receive BestPOS log
        resplength = getsGPSSPI2(response,100);
        vellength = getsGPSSPI2(velocity,100);*/

        /*UART version of GPS use while SPI to UART is not working
         *Note: Uses Comm UART line will not work with Comm implemented*/
        UART1_Puts ( command, length );

        crcount = 2;
```

```
resplength = 0;
while(crcount)
{
	*(response+resplength)=UART1_GetChar();
	if (*(response+resplength)==15)
	{
		crcount--;
	}
	else if (!*(response+resplength)==0)
	{
		resplength++;
	}
}

crcount = 2;
vellength = 0;
while(crcount)
{
	*(velocity+vellength)=UART1_GetChar();
	if (*(velocity+vellength)==13)
	{
		crcount--;
	}
	if (*(velocity+vellength)==0)
	{
		vellength++;
	}

}

UINT8 resbool = strcmp(response,"\r<OK\r");

if(resplength < 3)
{
	data.ptr = "Error: The response transmission timed out\r";
	data.length = 43;
}
else if(vellength < 3)
{
	data.ptr = "Error: The log transmission timed out\r";
	data.length = 38;
}
else if(resbool != 0)
{
	data.ptr = "Error: The message and/or response was not correct\r";
	data.length = 51;
}
else
{
	data.ptr = velocity;
	data.length = vellength;
}
```

```
	return data;
}

DATA_TYPE  GPS_GetTime ()
{
	UINT8 resplength;
	UINT8 timelength;
	UINT8 *response;
	UINT8 *time;
	DATA_TYPE    data;

	//Build message to get TIME log
	UINT8 *command = TIMECOMMAND;
	UINT8 length = strlen(TIMECOMMAND);

	//Test with loop
	//UINT8 *command = RESPONSETEST;
	//UINT8 length = strlen(RESPONSETEST);

	/*SPI2 code for when issues with SPI to UART converter are resolved*/
	//Transmit message
	/*gps_open();
	putsSPI2(length, command);
	gps_close();

	//Receive BestPOS log
	resplength = getsGPSSPI2(response,100);
	timelength = getsGPSSPI2(time,100);*/

	/*UART version of GPS use while SPI to UART is not working
	 *Note: Uses Comm UART line will not work with Comm implemented*/
	UART1_Puts ( command, length );

	crcount = 2;
	resplength = 0;
	while(crcount)
	{
		*(response+resplength)=UART1_GetChar();
		if (*(response+resplength)==15)
		{
			crcount--;
		}
		else if (!*(response+resplength)==0)
		{
			resplength++;
		}
	}

	crcount = 2;
	timelength = 0;
	while(crcount)
	{
		*(time+timelength)=UART1_GetChar();
```

```
                if (*(time+timelength)==13)
                {
                        crcount--;
                }
                if (*(time+timelength)==0)
                {
                        timelength++;
                }

        }

        UINT8 resbool = strcmp(response,"\r<OK\r");

        if(resplength < 3)
        {
                data.ptr = "Error: The response transmission timed out\r";
                data.length = 43;
        }
        else if(timelength < 3)
        {
                data.ptr = "Error: The log transmission timed out\r";
                data.length = 38;
        }
        else if(resbool != 0)
        {
                data.ptr = "Error: The message and/or response was not correct\r";
                data.length = 51;
        }
        else
        {
                data.ptr = time;
                data.length = timelength;
        }

        return data;
}

void GPS_Init ()
{
        SPI2_Init();
        REFOCON = 0xBA00;
}


DATA_TYPE GPS_SetOpMode (UINT8 opMode)
{
        //if needed
        DATA_TYPE    data;
        data.ptr = &opMode;
        data.length = 1;
        return data;
}
```

```
DATA_TYPE GPS_GetOpMode ()
{
        //if needed
        DATA_TYPE    data;
        UINT8* opMode;
        *opMode = 2;

        data.ptr = opMode;
        data.length = 1;

        return data;
}
```

APPENDIX B

SPI DRIVER CODE

```
/*****************************************************************************
*******************************************
 *
 *            Space Science and Engineering Laboritory
 *                 Montana State University
 *
 *                           FIREBIRD Software
 *
 * Filename      : spi1_driver.c
 * Header(s)     : spi_driver.h
 * Description   : This driver is for the SPI1 on the processor which coorisponds to the
SPI2 on the software side
 * Function(s)   : CloseSPI2, ConfigIntSPI2, DataRdySPI2, getsSPI2, getsGPSSPI2,
OpenSPI2, putsSPI2, ReadSPI2, WriteSPI2, gps_open,
 *               gps_close
 * Authors(s)    : Seth Berardinelli, Mack Wilz
 * Date          : December 04, 2010

*****************************************************************************
*******************************************/
#include "spi_driver.h"

/*****************************************************************************
*******************************************
* DESCRIPTION:      Init SPI peripheral to talk to GPS Module. requires general config
of I/O
*                                 pins, as well as config of the SPI1 peripheral, and the
related PPS pins.
*
* ARGUMENTS  :      none.
*
* RETURNS    :      Returns 0 when intilization is complete.
*
* NOTES      :      This function CANNOT be changed.
*****************************************************************************
*******************************************/
void SPI2_Init ( void )
{
        /* Enable -CS_SD (IO.0) control, keep it high / inactive.                  */
        init_gps_open();
        spi_cs_hi2();

        /*
         * Configure I/O pins.
         * SDO2:    RG8  / IO.1(SDO0)
         * SDI2:    RG7  / IO.2(SDI0)
         * SCK2OUT: RG9  / IO.3(SCK0)
   */
        PORTG |=  (SDO1); /* SDO0 is an output, idle low                            */
        TRISG &= ~(SDO1); /*  ""                                          */
        PORTG &= ~(SDI1); /* SDO1 is an input                                  */
        TRISG |=  (SDI1); /* SDI0 is an input                                */
        PORTG &= ~(SCK1); /* SCK0 is an output, idle low                             */
```

```
        TRISG &= ~(SCK1); /*  ""                                          */


        /*
    * Disable SPI2 module if previously enabled. Required because SPIxCONx
         * registers cannot be written to while SPIx modules are enabled.
    */
        CloseSPI2();


        /*
         * 8 MHz (4MHz Fcy) oscillator with 32:1 prescalar yields 125MHz initial
         * clock speed (must be <400kHz during card identification stage).
    */
        ConfigureSPI2(0);


        /* Set baudrate placeholder                                        */
        /* baudrate = SPI_INIT_BAUD_RATE; */


        /*
    * Configure I/O for SPI2 via PPS:
         * SDI2:    RP19  / IO.1
         * SDI2:    RP24  / IO.2
         * SCK2OUT: RP21  / IO.3
    */
        iPPSOutput(OUT_PIN_PPS_RP19,OUT_FN_PPS_SDO2);
        iPPSInput(IN_FN_PPS_SDI2,   IN_PIN_PPS_RP24);
        iPPSOutput(OUT_PIN_PPS_RP21, OUT_FN_PPS_SCK2OUT);
}

/*****************************************************************************
*****************************************
* Function          : void CloseSPI2(void)
*
* Description       : This function turns off the SPI module
*
* Arguments         : None
*
* Returns           : None
*
* Remarks           : This function disables the SPI interrupt and then turns
*                   off the module.The Interrupt Flag bit is also cleared
*****************************************************************************
*****************************************/
void  CloseSPI2 ( void )
{
        IEC2bits.SPI2IE = 0;      /* Disable the Interrupt bit in the  Interrupt Enable
Control Register     */
   SPI2STATbits.SPIEN = 0;   /* Disable the module. All pins controlled  by PORT
Functions            */
```

```
   IFS2bits.SPI2IF = 0;      /* Disable the Interrupt flag bit in the  Interrupt Flag Control
Register  */
}

/*********************************************************************
*****************************************
* Function          : void ConfigIntSPI2(unsigned int config)
*
* Description       : This function configures the SPI Interrupt
*
* Arguments         : config - SPI interrupt priority and enable/disable
*                   information as defined below
*
*                    Interrupt enable/disable
*                           *       SPI_INT_EN
*                           *       SPI_INT_DIS
*                    Interrupt Priority
*                           *       SPI_INT_PRI_0
*                           *       SPI_INT_PRI_1
*                           *       SPI_INT_PRI_2
*                           *       SPI_INT_PRI_3
*                           *       SPI_INT_PRI_4
*                           *       SPI_INT_PRI_5
*                           *       SPI_INT_PRI_6
*                           *       SPI_INT_PRI_7
*
* Returns           : None
*
* Remarks           : This function clears the Interrupt Flag bit, sets the
*                   interrupt priority and enables/disables the interrupt
*********************************************************************
*****************************************/
void  ConfigIntSPI2 ( unsigned int config )
{
        IFS2bits.SPI2IF = 0;              /* Clear IF bit              */
   IPC8bits.SPI2IP=(config &0x0007);      /* Assign interrupt priority      */
   IEC2bits.SPI2IE=(config &0x0008) >> 3; /* Interrupt Enable/Disable bit   */
}

/*********************************************************************
*****************************************
* Function          : char DataRdySPI2(void)
*
* Description       : This function determines if the SPI buffer contains
*                   any data to be read.
*
* Arguments         : None
*
* Returns           : If '1' is returned, it indicates that the data has been
*                    received in the receive buffer and is to be read.
*                    If '0' is returned, it indicates that the receive is not
*                   complete and the receive buffer is empty.
*
```

```
* Remarks          : This function returns the status of SPI receive buffer.
*                  This indicates if the SPI receive buffer contains any new
*                  data that is yet to be read as indicated by the
*                  SPIxSTAT<SPIRBF> bit. This bit is cleared by hardware
*                  when the data is read from the buffer.
*********************************************************************
*********************************************/
char  DataRdySPI2 ( void )
{
   return SPI2STATbits.SPIRBF; /* return RBF bit status */
}

/*********************************************************************
*********************************************
* Function          : unsigned int getsSPI2(unsigned int length,unsigned int *rdptr,
*                                   unsigned int spi_data_wait);
*
* Description       : This function reads a string of data of specified length and
*                  stores it into the location specified
*
* Arguments         : length - This is the length of the string to be received.
*                  rdptr - This is the pointer to the location where the data
*                  received have to be stored.
*                  spi_data_wait - This is the time-out count for which the module
*                  has to wait before return. If the time-out count is 'N', the
*
* Returns           : This function returns the number of bytes yet to be received.
*                  If the return value is a '0', it indicates that the complete
*                  string has been received.
*                  If the return value is a non-zero, it indicates that the
*                  complete string has not been received
*
* Remarks           : None
*********************************************************************
*********************************************/
unsigned int  getsSPI2 ( unsigned int length, unsigned int *rdptr, unsigned int
spi_data_wait )
{
   int wait = 0;
   char *temp_ptr = (char *)rdptr;
   while ( length )                    /* stay in loop until length = 0                    */
   {
      while( !DataRdySPI2() )
      {
         if ( wait < spi_data_wait )
                        {
                  wait++ ;                  /* wait for time out operation                    */
                        }
         else
                        {
           return(length);            /* Time out, return number of byte/word read for a given
length  */
         }
```

```
        }
                        wait = 0;

                        if ( SPI2CON1bits.MODE16 )
                        {
          //*rdptr++ = getcSPI2();        /* read a single word                                  */
                                *rdptr++ = ReadSPI2();
                        }
        else
                        {
          //*temp_ptr++ = getcSPI2();     /* read a single byte                                  */
                                *temp_ptr++ = ReadSPI2();
                        }
                        length--;                       /* reduce string length count by 1
*/
     }

    return (length);                    /* return number of byte/word to be read i.e, 0
*/
}

/*********************************************************************
*********************************************
* Function          : unsigned int getsGPSSPI2(unsigned int *rdptr, unsigned int
spi_data_wait);
*
* Description       : This function reads a string of data recieved from the GPS card and
*                     stores it into the location specified
*
* Arguments         : rdptr - This is the pointer to the location where the data
*                     received have to be stored.
*                     spi_data_wait - This is the time-out count for which the module
*                     has to wait before return. If the time-out count is 'N', the
*
* Returns           : This function returns the number of carriage return characters yet to be
received.
*                     If the return value is a '0', it indicates that the complete
*                     string has been received.
*                     If the return value is a non-zero, it indicates that the
*                     complete string has not been received
*
* Remarks           : None
*********************************************************************
*********************************************/
unsigned int  getsGPSSPI2 ( unsigned int *rdptr, unsigned int spi_data_wait )
{
          int length = 0;
          int crcount = 2;
    int wait = 0;
    char *temp_ptr = (char *)rdptr;
    while ( crcount )                   /* stay in loop until crcount = 0                      */
    {
```

```
      while( !DataRdySPI2() )
      {
         if ( wait < spi_data_wait )
                        {
                wait++ ;                /* wait for time out operation                          */
                        }
         else
                        {
           return(crcount);             /* Time out, return number of byte/word read for a
given length  */
         }
      }

                wait = 0;

                if ( SPI2CON1bits.MODE16 )
                {
         //*rdptr++ = getcSPI2();        /* read a single word                                   */
                        *rdptr++ = ReadSPI2();
                }
      else
                {
         //*temp_ptr++ = getcSPI2();     /* read a single byte                                   */
                        *temp_ptr++ = ReadSPI2();
                }

                if ( *temp_ptr == 13 )
                {
                        crcount--;                      /* reduce carriage rutrn count by 1
*/
                }
                length++;
   }

   return (length);                     /* return length of the transmission
                                        */
}

/*********************************************************************************************
* Function          : void OpenSPI2(unsigned int config1,unsigned int config2,
*                       unsigned int config3)
*
* Description       : This function configures the SPI module
*
* Arguments         : config1 - This contains the parameters to be configured in the
*                   SPIxCON1 register as defined below

*
*                               SCK Pin Control bit
*                                       *       DISABLE_SCK_PIN
*                                       *       ENABLE_SCK_PIN
*                               SDO Pin Control bit
```

```
*                   *       DISABLE_SDO_PIN
*                   *       ENABLE_SDO_PIN
*               Word/Byte Communication mode
*                   *       SPI_MODE16_ON
*                   *       SPI_MODE8_ON
*               SPI Data Input Sample phase
*                   *       SPI_SMP_ON
*                   *       SPI_SMP_OFF
*               SPI Clock Edge Select
*                   *       SPI_CKE_ON
*                   *       SPI_CKE_OFF
*               SPI slave select enable
*                   *       SLAVE_ENABLE_ON
*                   *       SLAVE_ENABLE_OFF
*               SPI Clock polarity select
*                   *       CLK_POL_ACTIVE_LOW
*                   *       CLK_POL_ACTIVE_HIGH
*               SPI Mode Select bit
*                   *       MASTER_ENABLE_ON
*                   *       MASTER_ENABLE_OFF
*               Secondary Prescale select
*                   *       SEC_PRESCAL_1_1
*                   *       SEC_PRESCAL_2_1
*                   *       SEC_PRESCAL_3_1
*                   *       SEC_PRESCAL_4_1
*                   *       SEC_PRESCAL_5_1
*                   *       SEC_PRESCAL_6_1
*                   *       SEC_PRESCAL_7_1
*                   *       SEC_PRESCAL_8_1
*               Primary Prescale select
*                   *       PRI_PRESCAL_1_1
```

```
*                                   *       PRI_PRESCAL_4_1
*                                   *       PRI_PRESCAL_16_1
*                                   *       PRI_PRESCAL_64_1
*
*               config2 - This contains the parameters to be configured in the
*               SPIxCON2 register as defined below
*                                   Frame SPI support Enable/Disable
*                                   *       FRAME_ENABLE_ON
*                                   *       FRAME_ENABLE_OFF
*                                   Frame Sync Pulse direction control
*                                   *       FRAME_SYNC_INPUT
*                                   *       FRAME_SYNC_OUTPUT
*                                   Frame Sync Polarity bit
*                                   *       FRAME_SYNC_ACTIVE_HIGH
*                                   *       FRAME_SYNC_ACTIVE_LOW
*                                   Frame Delay bit
*                                   *       SPI_FRM_PULSE_FIRST_CLK
*                                   *       SPI_FRM_PULSE_PREV_CLK
*                                   Enhance Buffer Enable\Disable
*                                   *       SPI_ENH_BUFF_ENABLE
*                                   *       SPI_ENH_BUFF_DISABLE
*
*               config3 - This contains the parameters to be configured in the
*               SPIxSTAT register as defined below
*
*                                   SPI Enable/Disable
*                                   *       SPI_ENABLE
*                                   *       SPI_DISABLE
*                                   SPI Idle mode operation
*                                   *       SPI_IDLE_CON
*                                   *       SPI_IDLE_STOP
*                                   Clear Receive Overflow Flag bit
*                                   *       SPI_RX_OVFLOW
*                                   *       SPI_RX_OVFLOW_CLR
*
* Returns       : None
*
* Remarks       : This functions initializes the SPI module and sets the Idle mode
*               Operation.
```

```
*************************************************************************
*****************************************/
void  OpenSPI2 ( unsigned int config1, unsigned int config2, unsigned int config3 )
{
        SPI2CON1 = config1;   /* Initalizes the spi module     */
   SPI2CON2 = config2;
   SPI2STAT = config3;   /* Enable/Disable the spi module */
}

/*************************************************************************
*********************************************
* Function          : void putsSPI2(unsigned int length,unsigned int *wrptr)
*
* Description       : This function writes the data to be transmitted into the
*                   Transmit Buffer (SPIxBUF) register.
*
* Arguments         : length - This is the number of data words/bytes to be transmitted.
*                   wrptr - This is the pointer to the string of data to be transmitted
*
* Returns           : None
*
* Remarks           : This function writes the specified length of data words/bytes to be
*                   transmitted into the transmit buffer.Once the transmit buffer is full,
*                   it waits until the data gets transmitted and then writes the next data
*                   into the Transmit register.The control remains in this function if SPI
*                   module is disabled while SPITBF bit is set.
*************************************************************************
*********************************************/
void  putsSPI2 ( unsigned int length, unsigned int *wrptr )
{
   char *temp_ptr = (char *) wrptr;
   while ( length )                  /* write byte/word until length is 0      */
   {
      if ( SPI2CON1bits.MODE16 )
                  {
         SPI2BUF = *wrptr++;        /* initiate SPI bus cycle by word write    */
                  }
                  else
                  {
         SPI2BUF = *temp_ptr++;     /* initiate SPI bus cycle by byte write    */
                  }

                  while ( SPI2STATbits.SPITBF );  /* wait until 'SPITBF' bit is cleared
*/

                  length--;                  /* decrement length                    */
   }
}

/*************************************************************************
*********************************************
* Function          : unsigned int ReadSPI2(void)
*
```

```
* Description       : This function reads the content of the SPI Receive Buffer
*                   (SPIxBUF) register.
*
* Precondition      : Before ReadSPI call SPI2_Rx_Buf_Full macro to check whether
*                   data is reaceived.
*
* Arguments         : None
*
* Returns           : This function returns the content of Receive Buffer
*                   (SPIxBUF) register.
*
* Remarks           : This function returns the content of the Receive Buffer register.
*                   If 16-bit communication is enabled, the data in the SPIxBUF
*                   register is returned.
*                   If 8-bit communication is enabled, then the lower byte of SPIxBUF
*                   is returned.
*******************************************************************************
*******************************************/
unsigned int  ReadSPI2 ( void )
{
        SPI2STATbits.SPIROV = 0;

        if ( SPI2CON1bits.MODE16 )
        {
           return ( SPI2BUF );       /* return word read */
        }
        else
        {
        return (SPI2BUF & 0xff);   /* return byte read */
        }
}

/*******************************************************************************
*******************************************
* Function          : void WriteSPI2(unsigned int data_out)
*
* Description       : This function writes the data to be transmitted
*                   into the Transmit Buffer (SPIxBUF) register.
*
* Arguments         : data_out - This is the data to be transmitted which
*                   will be stored in SPI buffer.
*
* Returns           : None
*
* Remarks           : This function writes the data (byte/word) to be transmitted
*                   into the transmit buffer.
*                   If 16-bit communication is enabled, the 16-bit value is
*                   written to the transmit buffer.
*                   If 8-bit communication is enabled, then upper byte is masked
*                   and then written to the transmit buffer.
*******************************************************************************
*******************************************/
void WriteSPI2(unsigned int data_out)
```

```
{
   if ( SPI2CON1bits.MODE16 )          /* word write */
         {
                SPI2BUF = data_out;
         }
   else
         {
      SPI2BUF = data_out & 0xff;       /*  byte write  */
         }
}

/*********************************************************************
*********************************************
*
* FUNCTION    :    void ConfigureSPI2(unsigned int prescalar_bits)
*
* DESCRIPTION :    Configure the SPI2 peripheral. Uses Microchip library functions.
*
* ARGUMENTS   :    prescalar_bits - bit values for primary and secondary prescalars.
*
* RETURNS     :       none
*
* NOTES       :
*********************************************************************
*********************************************/
void  ConfigureSPI2 ( unsigned int prescalar_bits )
{
         unsigned int SPICON1Value;
   unsigned int SPICON2Value;
   unsigned int SPISTATValue;


   /*
    *   SPI2 configuration.
    *
    *   * SPI2 communication is BYTE (8 bits) wide.
    *   * At GPS (SPI slave), MISO data is latched on the rising
    *     edge of the clock. Therefore, at the SPI Master, MOSI data
    *     must be present and stable before this, so "transmit happens
    *     from active clock state to idle clock state."
    *   * SPI2 in MASTER mode.
    *   * Primary and secondary prescalar bits are passed in as paramaters
    *   * Frame SPI support Disable.
    *   * SPI2 enabled.
    */
   SPICON1Value = ENABLE_SCK_PIN        | ENABLE_SDO_PIN     |
SPI_MODE8_ON
             | MASTER_ENABLE_ON      | SPI_SMP_OFF        | SPI_CKE_ON
             | SLAVE_ENABLE_OFF      | CLK_POL_ACTIVE_HIGH |
MASTER_ENABLE_ON
             | (prescalar_bits & (SEC_PRESCAL_1_1 | PRI_PRESCAL_1_1));
   SPICON2Value = FRAME_ENABLE_OFF     | FRAME_SYNC_OUTPUT   |
FRAME_SYNC_ACTIVE_LOW
```

```
            | SPI_FRM_PULSE_PREV_CLK | SPI_ENH_BUFF_DISABLE;
   SPISTATValue = SPI_ENABLE          | SPI_IDLE_CON      |
SPI_RX_OVFLOW_CLR;

        OpenSPI2(SPICON1Value,SPICON2Value,SPISTATValue);
}

/*********************************************************************
*********************************************
* DESCRIPTION:      Transmit 1 byte
*
* ARGUMENTS  :      Input is an unsigned char (8-bit) value.
*
* RETURNS    :      none
*
* NOTES      :      Microchip library functions are not used here, for speed and code
size.
*********************************************************************
*********************************************/
void  spi2_tx1 ( unsigned char data8 )
{
        volatile int dummy;

        dummy = SPI2BUF;                /* Ensure buffer full flag is cleared before xfer
starts. */

        SPI2BUF = (unsigned int) data8;    /* Start transfer.                          */

   /*
    * Checking the receive bit is done (instead of the IFG bit -- there are
    * errata and observed problems with that) is done appears to be a highly
    * reliable method to ensure the transfer is complete. Must do this so
    * that e.g. subsequent spi_cs_hi() does not occur before transfer is
    * truly complete.
    */
   while ( !SPI2_Rcv_Complete );
} /* spi_tx1() */

/*********************************************************************
*********************************************
* DESCRIPTION:      Transmit 512 bytes
*
* ARGUMENTS  :      Input is an unsigned char pointer (8-bit) value.
*
* RETURNS    :      none
*
* NOTES      :      Microchip library functions are not used here, for speed and code
size.
*********************************************************************
*********************************************/
void  spi2_tx512 ( unsigned char *buf )
{
        unsigned short i;
```

```
        for ( i = 0; i < 512; i++ )
        {
        spi2_tx1( *buf );
        ++buf;
        }
} /* spi_tx512() */

/*********************************************************************
*********************************************
* DESCRIPTION:     Recieves 1 byte
*
* ARGUMENTS  :     none.
*
* RETURNS    :     Returns the value in the receive buffer on the master device
*
* NOTES      :     Microchip library functions are not used here, for speed and code
size.
*********************************************************************
*********************************************/
unsigned char   spi2_rx1 ( void )
{
  spi2_tx1( 0xFF );        /* Receive is the same as transmit, but we pick up what came in
... */

  return (unsigned char) SPI2BUF & 0xFF;

} /* spi_rx1() */

/*********************************************************************
*********************************************
* DESCRIPTION:     Recieves 512 byte
*
* ARGUMENTS  :     none.
*
* RETURNS    :     Returns the value in the receive buffer on the master device
*
* NOTES      :     Microchip library functions are not used here, for speed and code
size.
*********************************************************************
*********************************************/
void  spi2_rx512 ( unsigned char *buf )
{
        unsigned short i;
        for ( i = 0; i < 512; i++ )
        {
                *buf++;
                *buf = spi2_rx1();
        }
} /* spi2_rx512() */

/*********************************************************************
*********************************************
```

```
* DESCRIPTION:  Open the GPS for use. Just makes -CS_GPS_BAR an output, leaves
it inactive.
*
* ARGUMENTS  :     none
*
* RETURNS    :         none
*
* NOTES      :  none
*****************************************************************************
*********************************************/
void init_gps_open ( void )
{
   /* Configure -CS_GPS_BAR as an output, and make it inactive (i.e., HIGH) */
   PORTG |=  CS_GPS_BAR;
   TRISG &= ~CS_GPS_BAR;
}

/*****************************************************************************
*********************************************
* Function          : void gps_open ( void );
*
* Description       : Open the SD Card for use. Just makes -CS_SD an output, leaves it
inactive.
*
* Arguments         : None
*
* Returns           : None
*
* Remarks           : Requires that SD Card be powered prior to use.
*****************************************************************************
*********************************************/
void  gps_open ( void )
{
        /* Configure -CS_SD as an output, and make it inactive (i.e., HIGH) */
   PORTG |=  CS_GPS_BAR;
   TRISG &= ~CS_GPS_BAR;
}

/*****************************************************************************
*********************************************
* Function          : void sd_close ( void );
*
* Description       : Close the SD Card.
*
* Arguments         : None
*
* Returns           : None
*
* Remarks           : None
*****************************************************************************
*********************************************/
void  gps_close ( void )
{
```

```
        /* Restore -CS_GPS_BAR to an input. It's therefore pulled HIGH by its pull-up
resistor. */
    PORTG |= _CS_GPS_BAR;
    TRISG |= _CS_GPS_BAR;
}
```

APPENDIX C

GPS CARD SAFE TO MATE TESTING PROCEDURE

1.0 SCOPE

This document outlines the Global Positioning System (GPS) board safe to mate procedure for the FIREBIRD mission.

2.0 Test Readiness Review (TRR)

A TRR shall be held prior to execution of this procedure. This review shall include a description of the test article, the levels, inspections, and evaluation criteria.

3.0 Applicable Documents and Standards

The following documents and standards shown form a part of this document to the extent specified. If a revision number is not shown, then it is the issue in effect on the date of this document. In the event of a conflict between this document and the contents of one of the documents or standards listed below, this document shall take precedence.

Table 1 Applicable Documents

Document Number	Document Title
FB-R-0510	BIRD Master Pin out

4.0 Test Documentation and Reporting

4.1 Test Data

Test data will be recorded directly on this procedure or on a computer printout attached to this procedure in Appendix A. Mark front page of this procedure "As-Run Test Report". Submit completed test report to SSEL Document Control.

4.2 Quality Assurance

An SSEL QA representative will be present and shall monitor portions of the testing. Prior to the start of the test, SSEL QA shall verify that the test setup and equipment meets the requirements of this document.

4.3 Nonconformances and Failures

Nonconformances, including test failures encountered during the execution of this test shall be documented using the SSEL Nonconformance Reporting System. The reason for any test failure shall be identified. **The hardware under test or the test setup shall not be disturbed in any way that prohibits duplication of the test sequence and configuration that resulted in the failure.** Any corrective actions identified shall be evaluated by the engineering team prior to execution.

4.4 Changes to Test Procedure

Changes to this test procedure may be made in response to last minute changes in test requirements or to correct errors in the test procedure identified after the test has started. Changes are defined as minor or major changes depending upon how much the procedure is changed.

Minor Changes

Minor changes are defined as those changes that will not significantly change the actual test procedure or affect the results of the test. Such changes as equipment model number changes or corrections of procedural errors are minor changes. Minor changes may be made by the test conductor, with the concurrence of the responsible QA Technician. Changes must be initialed and dated by the test conductor and receive a QA stamp or initial and date after approval.

Major Changes

Major changes are defined as those changes that will significantly change the actual test procedure or affect the test results. Changes to procedure to reduce schedule impact, changes as the result of test failure closeouts, or changes in the scope of the test are defined as major changes and shall be initiated only by the project manager and must have the concurrence of the QA representative.

Test Configuration Control

Any changes to the test setup after the start of the test shall be documented and permanently attached to this procedure. This should include modifications to the unit under test, GSE, wiring, jumpers, etc. In addition, each time this procedure is performed, a log shall be maintained of any test configuration changes that may have occurred.

4.5 Run Time Log

Throughout the execution of this test procedure, be sure to maintain an accurate log of the run time of the system(s) under test. Enter the start and stop times in the margins of the procedure or another suitable location within this test procedure.

5.0 Constraints

5.1 Cleanliness and Contamination Control

Only qualified personnel wearing appropriate protective garments shall be allowed to handle the FIREBIRD flight hardware. Appropriate garments include an ESD smock with wrist strap and Nitrile gloves.

5.2 Lifting and Handling

Care should be exercised when handling the assembled circuit boards. The boards shall be transported between test benches only when placed within an ESD bag and suitable protective container.

5.3 ESD Constraints

Personnel handling FIREBIRDS PCBs, shall be strap grounded to a suitable grounding point. The SSEL ESD Policy (00A0000) outlines the basics of ESD control and shall be referenced prior to the execution of this test.

5.4 Safety

Unit Under Test Safety

Utmost care must be taken to avoid physical damage to FIREBIRD PCBs. The use of proper lifting techniques is required. As stated previously, only qualified SSEL personnel shall be allowed to handle the flight article. During work on the flight hardware, two persons are required for all operations.

Facility and Equipment Safety

Appropriate barriers should be used to separate the area in which flight hardware testing is being conducted.

6.0 Equipment Calibration Requirements

All equipment used to make measurements during the execution of this procedure shall have current calibration certifications. Measuring equipment shall have precision nominally an order of magnitude greater than that required for the measurement data, and not less than a factor of two greater than that required. Record all pertinent calibrated test equipment in Table 1 of this procedure.

Required Equipment:	Calibration Required?
Triple Output DC Power Supply	YES
Digital Multimeter	YES
PCB Test Stand	NO

Table 2 Test Equipment Identification

Equipment Identification	Serial Number	Last Cal. Date
Fluke 189 Multimeter	85160315	12-1-2003
Agilent E3631A DC Power Supply	MY40021584	as received

Table 3 Units Under Test (Hardware and Software)

Name	Serial Number	STM run #
Novatel OEMV-1 GPS Card	DZV093500B 01017488	2

7.0 Test Preparation

The following steps shall be completed before performing the interface test:

Initial step when completed

7.1 Pre-test Operations

MW A) Setup relevant test GSE in the test area. For testing, place the board on a PCB test stand. For GPS unpowered STM and GPS powered STM, the board should be mounted header down. For the antenna port test, the board should be mounted header up.

MW B) Verify that cables are out of high traffic areas

MW C) Verify that cables are in good condition with no pinches or frays

MW D) Verify that the Unit Under Test is mounted in a mechanically safe way

MW E) Take numerous photos of entire test set up prior to start of test

F) Have QA verify the test setup prior to the start of the test

8.0 Safe to Mate Test

8.1 GPS Unpowered STM Test

MW A) Ensure all power is off prior to running test.

MW B) Ensure that the GPS board is grounded and safe from any ESD.

MW C) Use a multimeter to test for open/shorts from the given header pin to the ground pin 18.

Table 4 GPS Unpowered

Pin	Name	Type	Resistance Expected	Resistance Measured
1	LNA_PWR	Power	Open	4.80 MΩ
2	V_{IN}	Power	Open	1.43 MΩ
3	USB D (-)	Logic	Open	1.75 MΩ
4	USB D (+) / COM3_Rx	Logic	Open	2.95 kΩ
5	RESETIN	Logic	Open	21.17 kΩ
6	VARF / CAN1_Rx	Logic	Open	1.00 MΩ
7	Event2 / CAN1_Tx	Logic	Open	1.08 MΩ
8	CAN2_Rx	Logic	Open	1.27 MΩ
9	Event1 / COM3_Tx	Logic	Open	1.09 MΩ
10	GND	Power	Short	0.13 Ω
11	COM1_Tx	Logic	Open	2.38 MΩ
12	COM1_Rx	Logic	Open	2.23 MΩ
13	GND	Power	Short	0.14 Ω
14	COM2_Tx	Logic	Open	2.38 MΩ
15	COM2_Rx	Logic	Open	39.15 kΩ
16	GND	Power	Short	0.13 Ω
17	PV	Logic	Open	10.19 kΩ
18	GND	Power	Short	0.14 Ω
19	PPS	Logic	Open	190.28 kΩ
20	CAN2_Tx	Logic	Open	1.27 MΩ

Note: The pass criteria are ensured when Open/Short is met. Resistance values will vary slightly between tests but should remain within a predictable range.

Were all resistances within limits? (PASS / FAIL)

Test Conductor Initials: MW Date: 11-13-2010 Time: 3:48 pm

8.2 GPS Antenna Port Unpowered STM Test

MW A) Using a multimeter, verify the following resistances between the antenna appropriate connectors and the pins as listed

Table 5 Antenna Port Connector

Measured Pin	Reference Pin	Resistance Expected	Resistance Measured
Connector Center Conductor	Connector Outer Shell	Open	121.3 kΩ
Connector Center Conductor	Pin 18	Open	185.1 kΩ
Connector Outer Shell	Pin 18	Short	0.3 Ω

Results of test: (PASS / FAIL)

Test Conductor Initials: MW Date: 11-13-2010 Time: 3:51 pm

8.3 GPS Powered STM Test

MW A) Ensure all power is off prior to running the test.

MW B) Make all connections from power supply to board as necessary. A diagram of the necessary connections can be found in Appendix B of this doc.
a) Connect ground line from power supply to pin 18.
b) Connect the 8.4V voltage line from power supply to pin 1.
c) Connect the 3.3V voltage line from power supply to pin 2.

MW C) Apply power to GPS board by setting the V_{IN} voltage to 3.3V with 500mA limit and the LNA_PWR voltage to 8.4V with 500mA limit.

MW D) Test each pin for the expected voltage by placing one end on the given pin and the other to ground pin 16.

Table 6 GPS Powered

Pin	Name	Type	Voltage Expected	Voltage Measured
1	LNA_PWR	Power	8.4	8.33 V
2	V_{IN}	Power	3.3	3.24 V
3	USB D (-)	Logic	0 or 3.3	0.06 V
4	USB D (+) / COM3_Rx	Logic	0 or 3.3	0.57 V
5	RESETIN	Logic	0 or 3.3	3.14 V
6	VARF / CAN1_Rx	Logic	0 or 3.3	3.22 V
7	Event2 / CAN1_Tx	Logic	0 or 3.3	3.15 V
8	CAN2_Rx	Logic	0 or 3.3	3.20 V
9	Event1 / COM3_Tx	Logic	0 or 3.3	3.22 V
10	GND	Power	0	0.01 V
11	COM1_Tx	Logic	0 or 3.3	3.30 V
12	COM1_Rx	Logic	0 or 3.3	3.40 V
13	GND	Power	0	0.01 V
14	COM2_Tx	Logic	0 or 3.3	3.13 V
15	COM2_Rx	Logic	0 or 3.3	3.20 V
16	GND	Power	0	0.01 V
17	PV	Logic	0 or 3.3	0.07 V
18	GND	Power	0	0 V

19	PPS	Logic	0 or 3.3	0.005 V
20	CAN2_Tx	Logic	0 or 3.3	3.12 V

Voltages matched expected values? (PASS / FAIL)

Test Conductor Initials: MW Date: 11-13-2010 Time: 4:03 pm

9.0 Test Closeout

Perform the following steps in closing out the board level interface test.

MW A) Ensure all test documentation are properly annotated and permanently attached to this procedure and that all witness signatures are in place.

Performed / Reviewed by:

Test Engineer Mackenzie W Date 11-13-2010

QA [signature] Date: 11-13-2010

APPENDIX D

GPS BOARD SAFE TO MATE TESTING PROCEDURE

1.0 SCOPE

This document outlines the Global Positioning System (GPS) board safe to mate procedure for the FIREBIRD mission.

2.0 Test Readiness Review (TRR)

A TRR shall be held prior to execution of this procedure. This review shall include a description of the test article, the levels, inspections, and evaluation criteria.

3.0 Applicable Documents and Standards

The following documents and standards shown form a part of this document to the extent specified. If a revision number is not shown, then it is the issue in effect on the date of this document. In the event of a conflict between this document and the contents of one of the documents or standards listed below, this document shall take precedence.

Table 1 Applicable Documents

Document Number	Document Title
FB-R-0510	BIRD Master Pin out

4.0 Test Documentation and Reporting

4.1 Test Data

Test data will be recorded directly on this procedure or on a computer printout attached to this procedure in Appendix A. Mark front page of this procedure "As-Run Test Report". Submit completed test report to SSEL Document Control.

4.2 Quality Assurance

An SSEL QA representative will be present and shall monitor portions of the testing. Prior to the start of the test, SSEL QA shall verify that the test setup and equipment meets the requirements of this document.

4.3 Nonconformances and Failures

Nonconformances, including test failures encountered during the execution of this test shall be documented using the SSEL Nonconformance Reporting System. The reason for any test failure shall be identified. **The hardware under test or the test setup shall not be disturbed in any way that prohibits duplication of the test sequence and configuration that resulted in the failure.** Any corrective actions identified shall be evaluated by the engineering team prior to execution.

4.4 Changes to Test Procedure

Changes to this test procedure may be made in response to last minute changes in test requirements or to correct errors in the test procedure identified after the test has started. Changes are defined as minor or major changes depending upon how much the procedure is changed.

Minor Changes

Minor changes are defined as those changes that will not significantly change the actual test procedure or affect the results of the test. Such changes as equipment model number changes or corrections of procedural errors are minor changes. Minor changes may be made by the test conductor, with the concurrence of the responsible QA Technician. Changes must be initialed and dated by the test conductor and receive a QA stamp or initial and date after approval.

Major Changes

Major changes are defined as those changes that will significantly change the actual test procedure or affect the test results. Changes to procedure to reduce schedule impact, changes as the result of test failure closeouts, or changes in the scope of the test are defined as major changes and shall be initiated only by the project manager and must have the concurrence of the QA representative.

Test Configuration Control

Any changes to the test setup after the start of the test shall be documented and permanently attached to this procedure. This should include modifications to the unit under test, GSE, wiring, jumpers, etc. In addition, each time this procedure is performed, a log shall be maintained of any test configuration changes that may have occurred.

4.5 Run Time Log

Throughout the execution of this test procedure, be sure to maintain an accurate log of the run time of the system(s) under test. Enter the start and stop times in the margins of the procedure or another suitable location within this test procedure.

5.0 Constraints

5.1 Cleanliness and Contamination Control

Only qualified personnel wearing appropriate protective garments shall be allowed to handle the FIREBIRD flight hardware. Appropriate garments include an ESD smock with wrist strap and Nitrile gloves.

5.2 Lifting and Handling

Care should be exercised when handling the assembled circuit boards. The boards shall be transported between test benches only when placed within an ESD bag and suitable protective container.

5.3 ESD Constraints

Personnel handling FIREBIRDS PCBs, shall be strap grounded to a suitable grounding point. The SSEL ESD Policy (00A0000) outlines the basics of ESD control and shall be referenced prior to the execution of this test.

5.4 Safety

Unit Under Test Safety

Utmost care must be taken to avoid physical damage to FIREBIRD PCBs. The use of proper lifting techniques is required. As stated previously, only qualified SSEL personnel shall be allowed to handle the flight article. During work on the flight hardware, two persons are required for all operations.

Facility and Equipment Safety

Appropriate barriers should be used to separate the area in which flight hardware testing is being conducted.

6.0 Equipment Calibration Requirements

All equipment used to make measurements during the execution of this procedure shall have current calibration certifications. Measuring equipment shall have precision nominally an order of magnitude greater than that required for the measurement data, and not less than a factor of two greater than that required. Record all pertinent calibrated test equipment in Table 1 of this procedure.

Required Equipment:	Calibration Required?
Triple Output DC Power Supply	YES
Digital Multimeter	YES
PCB Test Stand	NO

Table 2 Test Equipment Identification

Equipment Identification	Serial Number	Last Cal. Date
Agilent E3631A Triple Output DC PWRSupply	MY40021589	As received
Agilent E3631A Triple Output DC PWRSupply	MY40021584	As received
Fluke 189 Multimeter	85160315	12/1/2003

Table 3 Units Under Test (Hardware and Software)

Name	Serial Number	STM run #
FIREBIRD SSEL GPS BOARD	GPS# A	1

7.0 Test Preparation

The following steps shall be completed before performing the interface test:

Initial step when completed

7.1 Pre-test Operations

MW A) Setup relevant test GSE in the test area. For testing, place the board on a PCB test stand.

MW B) Verify that cables are out of high traffic areas

MW C) Verify that cables are in good condition with no pinches or frays

MW D) Verify that the Unit Under Test is mounted in a mechanically safe way

MW E) Take numerous photos of entire test set up prior to start of test

_____ F) Have QA verify the test setup prior to the start of the test

8.0 Safe to Mate Test

8.1 GPS Board Unpowered STM Test

MW A) Ensure all power is off prior to running test.

MW B) Ensure that the GPS board is grounded and safe from any ESD.

MW C) Use a multimeter to test for open/shorts from the given header pin to the ground pin 81.

Table 4 GPS Board to GPS Card Header Unpowered

Pin	Name	Type	Resistance Expected	Resistance Measured
1	LNA_PWR	Power	Open	14.9 MΩ
2	V_{IN}	Power	Open	2.6 KΩ
3	USB D (-)	Logic	Open	Open
4	USB D (+) / COM3_Rx	Logic	Open	Open
5	RESETIN	Logic	Open	Open
6	VARF / CAN1_Rx	Logic	Open	Open
7	Event2 / CAN1_Tx	Logic	Open	Open
8	CAN2_Rx	Logic	Open	Open
9	Event1 / COM3_Tx	Logic	Open	Open
10	GND	Power	Short	0.25 Ω
11	COM1_Tx	Logic	Open	8.29 MΩ
12	COM1_Rx	Logic	Open	Open
13	GND	Power	Short	0.24 Ω
14	COM2_Tx	Logic	Open	Open
15	COM2_Rx	Logic	Open	Open
16	GND	Power	Short	0.25 Ω
17	PV	Logic	Open	Open
18	GND	Power	Short	0.23 Ω
19	PPS	Logic	Open	Open
20	CAN2_Tx	Logic	Open	Open

Table 5 GPS Board PC104 Unpowered

Pin	Name	Type	Resistance Expected	Resistance Measured
1	NC	No Connect	Open	OPEN
2	NC	No Connect	Open	Open
3	NC	No Connect	Open	Open
4	NC	No Connect	Open	Open
5	UTX3_PL	Logic	Open	5.06 MΩ
6	URX3_PL	Logic	Open	Open
7	NC	No Connect	Open	Open
8	NC	No Connect	Open	Open
9	EN_LNA	Logic	Open	Open
10	EN_GPS_BAR	Logic	Open	Open
11	NC	No Connect	Open	Open
12	NC	No Connect	Open	Open
13	SCK1_GPS	Logic	Open	Open
14	SDI1_GPS	Logic	Open	Open
15	SDO1_GPS	Logic	Open	8.4 MΩ
16	SS1_GPS	Logic	Open	Open
17	NC	No Connect	Open	Open
18	NC	No Connect	Open	Open
19	NC	No Connect	Open	Open
20	NC	No Connect	Open	open
21	NC	No Connect	Open	Open
22	NC	No Connect	Open	Open
23	NC	No Connect	Open	Open
24	NC	No Connect	Open	Open
25	NC	No Connect	Open	Open
26	NC	No Connect	Open	Open
27	NC	No Connect	Open	open
28	NC	No Connect	Open	Open
29	NC	No Connect	Open	Open
30	NC	No Connect	Open	Open
31	NC	No Connect	Open	Open
32	NC	No Connect	Open	Open
33	NC	No Connect	Open	Open
34	NC	No Connect	Open	Open
35	NC	No Connect	Open	Open
36	NC	No Connect	Open	Open
37	NC	No Connect	Open	Open
38	NC	No Connect	Open	Open
39	NC	No Connect	Open	Open
40	NC	No Connect	Open	Open
41	NC	No Connect	Open	Open
42	NC	No Connect	Open	Open
43	NC	No Connect	Open	Open
44	NC	No Connect	Open	Open
45	NC	No Connect	Open	Open

46	NC	No Connect	Open	Open
47	NC	No Connect	Open	Open
48	NC	No Connect	Open	
49	NC	No Connect	Open	
50	NC	No Connect	Open	
51	NC	No Connect	Open	
52	NC	No Connect	Open	
53	D1_LKG	Analog	Open	
54	NC	No Connect	Open	
55	D2_LKG	Analog	Open	
56	NC	No Connect	Open	
57	FIRE_TEMP	Analog	Open	
58	NC	No Connect	Open	
59	NC	No Connect	Open	
60	NC	No Connect	Open	
61	REFO_UART4	Logic	Open	9,5 MΩ
62	NC	No Connect	Open	open
63	NC	No Connect	Open	
64	NC	No Connect	Open	
65	NC	No Connect	Open	
66	NC	No Connect	Open	
67	NC	No Connect	Open	
68	NC	No Connect	Open	
69	NC	No Connect	Open	
70	NC	No Connect	Open	
71	PPS_GPS	Logic	Open	
72	EVENT1_GPS	Logic	Open	
73	RST_GPS	Logic	Open	
74	EVENT2_GPS	Logic	Open	
75	SHDN_UART4_BAR	Logic	Open	
76	PV_GPS	Logic	Open	
77	+5VSYS	Power	Open	
78	+5VSYS	Power	Open	
79	VCCSYS	Power	Open	4,97 MΩ
80	VCCSYS	Power	Open	4,97 MΩ
81	GND	Power	Short	0,3 Ω
82	GND	Power	Short	0,3 Ω
83	GND	Power	Short	0,3 Ω
84	GND	Power	Short	0,3 Ω
85	NC	No Connect	Open	open
86	NC	No Connect	Open	
87	NC	No Connect	Open	
88	NC	No Connect	Open	
89	NC	No Connect	Open	
90	NC	No Connect	Open	
91	NC	No Connect	Open	
92	NC	No Connect	Open	
93	NC	No Connect	Open	
94	NC	No Connect	Open	
95	NC	No Connect	Open	

96	NC	No Connect	Open	Open
97	VBATT	Power	Open	19.7 MΩ
98	NC	No Connect	Open	Open
99	VBATT_PL	Power	Open	Open
100	VBATT_PL	Power	Open	Open
101	NC	No Connect	Open	Open
102	NC	No Connect	Open	Open
103	NC	No Connect	Open	Open
104	NC	No Connect	Open	Open

Table 6 GPS Board Micro-D Unpowered

Pin	Name	Type	Resistance Expected	Resistance Measured
1	NC	No Connect	Open	
2	NC	No Connect	Open	
3	GND	Power	Short	
4	GND	Power	Short	
5	NC	No Connect	Open	
6	FIRE_DAT_P	Analog	Open	
7	FIRE_CMD_P	Analog	Open	
8	FIRE_DAT_N	Analog	Open	
9	FIRE_CMD_N	Analog	Open	
10	D1LKG	Analog	Open	
11	FIRE_TEMP	Analog	Open	
12	D2LKG	Analog	Open	
13	GND	Power	Short	
14	GND	Power	Short	
15	NC	No Connect	Open	
16	NC	No Connect	Open	
17	NC	No Connect	Open	
18	NC	No Connect	Open	
19	VBATT_PL	Power	Open	
20	VBATT_PL	Power	Open	
21	NC	No Connect	Open	

Note: The pass criteria are ensured when Open/Short is met. Resistance values will vary slightly between tests but should remain within a predictable range.

Were all resistances within limits? (PASS / FAIL)

Test Conductor Initials: MW Date: 11-23-10 Time: 5:15 pm

8.2 GPS Board Powered STM Test

MW A) Ensure all power is off prior to running the test.

MW B) Make all connections from power supply to board as necessary.

- ✓ a) Connect ground line from power supply to pins 10, ~~81, 82, 83~~, & ~~84~~ of the PC104 connector.
- ✓ b) Connect the 3.3V voltage line from power supply to pins 9, ~~78~~, & ~~80~~ of the PC104 connector.
- ✓ c) Connect the 5V voltage line from the power supply to pins 77 ~~& 78~~ of the PC104 connector.
- ✓ d) Connect the 8.4V voltage line from power supply to pins 97, ~~98~~ & ~~100~~ of the PC104 connector.

MW C) Apply power to GPS board by setting the current limits as follows:

a) 3.3V - 100mA limit
b) 5V - 100mA limit
c) 8.4V - 800mA limit

MW D) Test each pin for the expected voltage by placing one end on the given pin and the other to ground pin 81.

Table 7 GPS Board to GPS Card Header Powered

Pin	Name	Type	Voltage Expected	Voltage Measured
1	LNA_PWR	Power	8.4	3.3 V
2	V_{IN}	Power	3.3	3.3 V
3	USB D (-)	Logic	0 or 3.3	0 V
4	USB D (+) / COM3_Rx	Logic	0 or 3.3	0 V
5	RESETIN	Logic	0 or 3.3	0 V
6	VARF / CAN1_Rx	Logic	0 or 3.3	0 V
7	Event2 / CAN1_Tx	Logic	0 or 3.3	0 V
8	CAN2_Rx	Logic	0 or 3.3	0 V
9	Event1 / COM3_Tx	Logic	0 or 3.3	0 V
10	GND	Power	0	0 V
11	COM1_Tx	Logic	0 or 3.3	3.3 V
12	COM1_Rx	Logic	0 or 3.3	0 V
13	GND	Power	0	0 V
14	COM2_Tx	Logic	0 or 3.3	0 V
15	COM2_Rx	Logic	0 or 3.3	0 V
16	GND	Power	0	0 V

0.8 V w/LA disabled
3.3 V w/LA enab ?

17	PV	Logic	0 or 3.3	0 V
18	GND	Power	0	0 V
19	PPS	Logic	0 or 3.3	0 V
20	CAN2_Tx	Logic	0 or 3.3	0 V

Table 8 GPS Board PC104 Powered

Pin	Name	Type	Voltage Expected	Voltage Measured
1	NC	No Connect	0	0 V
2	NC	No Connect	0	0 V
3	NC	No Connect	0	0 V
4	NC	No Connect	0	0 V
5	UTX3_PL	Logic	0 or 3.3	3.3 V
6	URX3_PL	Logic	0 or 3.3	0 V
7	NC	No Connect	0	0 V
8	NC	No Connect	0	0 V
9	EN_LNA	Logic	3.3	3.3 V
10	EN_GPS_BAR	Logic	0	0 V
11	NC	No Connect	0	0 V
12	NC	No Connect	0	0 V
13	SCK1_GPS	Logic	0 or 3.3	0 V
14	SDI1_GPS	Logic	0 or 3.3	0 V
15	SDO1_GPS	Logic	0 or 3.3	0 V
16	SS1_GPS	Logic	0 or 3.3	0 V
17	NC	No Connect	0	0 V
18	NC	No Connect	0	0 V
19	NC	No Connect	0	0 V
20	NC	No Connect	0	0 V
21	NC	No Connect	0	0 V
22	NC	No Connect	0	0 V
23	NC	No Connect	0	0 V
24	NC	No Connect	0	0 V
25	NC	No Connect	0	0 V
26	NC	No Connect	0	0 V
27	NC	No Connect	0	0 V
28	NC	No Connect	0	0 V
29	NC	No Connect	0	0 V
30	NC	No Connect	0	0 V
31	NC	No Connect	0	0 V
32	NC	No Connect	0	0 V
33	NC	No Connect	0	0 V
34	NC	No Connect	0	0 V
35	NC	No Connect	0	0 V
36	NC	No Connect	0	0 V
37	NC	No Connect	0	0 V
38	NC	No Connect	0	0 V
39	NC	No Connect	0	0 V

40	NC	No Connect	0	0 V
41	NC	No Connect	0	0 V
42	NC	No Connect	0	0 V
43	NC	No Connect	0	0 V
44	NC	No Connect	0	0 V
45	NC	No Connect	0	0 V
46	NC	No Connect	0	0 V
47	NC	No Connect	0	0 V
48	NC	No Connect	0	0 V
49	NC	No Connect	0	0 V
50	NC	No Connect	0	0 V
51	NC	No Connect	0	0 V
52	NC	No Connect	0	0 V
53	D1_LKG	Analog	0	0 V
54	NC	No Connect	0	0 V
55	D2_LKG	Analog	0	0 V
56	NC	No Connect	0	0 V
57	FIRE_TEMP	Analog	0	0 V
58	NC	No Connect	0	0 V
59	NC	No Connect	0	0 V
60	NC	No Connect	0	0 V
61	REFO_UART4	Logic	0 or 3.3	0 V
62	NC	No Connect	0	0 V
63	NC	No Connect	0	0 V
64	NC	No Connect	0	0 V
65	NC	No Connect	0	0 V
66	NC	No Connect	0	0 V
67	NC	No Connect	0	0 V
68	NC	No Connect	0	0 V
69	NC	No Connect	0	0 V
70	NC	No Connect	0	0 V
71	PPS_GPS	Logic	0 or 3.3	0 V
72	EVENT1_GPS	Logic	0 or 3.3	0 V
73	RST_GPS	Logic	0 or 3.3	0 V
74	EVENT2_GPS	Logic	0 or 3.3	0 V
75	SHDN_UART4_BAR	Logic	0 or 3.3	0 V
76	PV_GPS	Logic	0 or 3.3	0 V
77	+5VSYS	Power	5	5 V
78	+5VSYS	Power	5	5 V
79	VCCSYS	Power	3.3	3.3 V
80	VCCSYS	Power	3.3	3.3 V
81	GND	Power	0	0 V
82	GND	Power	0	0 V
83	GND	Power	0	0 V
84	GND	Power	0	0 V
85	NC	No Connect	0	0 V
86	NC	No Connect	0	0 V
87	NC	No Connect	0	0 V
88	NC	No Connect	0	0 V

89	NC	No Connect	0	0 V
90	NC	No Connect	0	0 V
91	NC	No Connect	0	0 V
92	NC	No Connect	0	0 V
93	NC	No Connect	0	0 V
94	NC	No Connect	0	0 V
95	NC	No Connect	0	0 V
96	NC	No Connect	0	0 V
97	VBATT	Power	8.4	8.4 V
98	NC	No Connect	0	0 V
99	VBATT_PL	Power	8.4	8.4 V
100	VBATT_PL	Power	8.4	8.4 V
101	NC	No Connect	0	0 V
102	NC	No Connect	0	0 V
103	NC	No Connect	0	0 V
104	NC	No Connect	0	0 V

Table 9 GPS Board Micro-D Powered

Pin	Name	Type	Voltage Expected	Voltage Measured
1	NC	No Connect	0	
2	NC	No Connect	0	
3	GND	Power	0	
4	GND	Power	0	
5	NC	No Connect	0	
6	FIRE_DAT_P	Analog	0	
7	FIRE_CMD_P	Analog	0	
8	FIRE_DAT_N	Analog	0	
9	FIRE_CMD_N	Analog	0	
10	D1LKG	Analog	0	
11	FIRE_TEMP	Analog	0	
12	D2LKG	Analog	0	
13	GND	Power	0	
14	GND	Power	0	
15	NC	No Connect	0	
16	NC	No Connect	0	
17	NC	No Connect	0	
18	NC	No Connect	0	
19	VBATT_PL	Power	8.4	
20	VBATT_PL	Power	8.4	
21	NC	No Connect	0	

Voltages matched expected values? (PASS) / FAIL

Test Conductor Initials: MW Date: 11-23-10 Time: 6:00pm

LNA power for the GPS antenna does not have correct voltage @ active or

9.0 Test Closeout

Perform the following steps in closing out the board level interface test.

MW A) Ensure all test documentation are properly annotated and permanently attached to this procedure and that all witness signatures are in place.

Performed / Reviewed by:

Test Engineer Mackenzie W Date 11-23-10

QA Keith Maas Date: 12/2/10

Appendix A: Test Data

All values matched expected values except the LNA_PWR pin on the 20-pin GPS Header (J1). This value should be 0 V when idle (Pin 9 = ~~GND~~ 0V) & 8.4 V when active (EN_LNA (Pin 9) = 3.3 V). It currently has values of 0V when idle & 3.3 V when Active. Upon Looking through schematics, The schematic has CTL and IN pins for the MIC2514YMS used for the opposing value. This meaning, the circuit connected to CTL should be connected to ~~IN~~ and the circuit connected to IN should be connected to CTL. This is reflected on the GPS Board.