



**NAVAL
POSTGRADUATE
SCHOOL**

MONTEREY, CALIFORNIA

THESIS

**DEVELOPMENT OF LOW-COST SATELLITE CONTROL
SOFTWARE**

by

Bryan D. Waterman

June 2005

Thesis Advisor:
Second Reader:

Jim A. Horning
I. M. Ross

Approved for public release; distribution is unlimited.

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE			Form Approved OMB No. 0704-0188
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington DC 20503.			
1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE June 2005	3. REPORT TYPE AND DATES COVERED Master's Thesis	
4. TITLE AND SUBTITLE: Development of Low-Cost Satellite Control Software		5. FUNDING NUMBERS	
6. AUTHOR(S) Waterman, Bryan		8. PERFORMING ORGANIZATION REPORT NUMBER	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000		10. SPONSORING/MONITORING AGENCY REPORT NUMBER	
9. SPONSORING /MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A		11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.	
12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution is unlimited.		12b. DISTRIBUTION CODE	
13. ABSTRACT (maximum 200 words) This thesis details the development and validation procedures of the experiment control software to be used on the NPSAT1. The software developed for a standard Linux kernel instead of an expensive proprietary space computer system includes functions for satellite orbit prediction, precise satellite location, and adaptive experiment scheduling using inputs from the electrical power system, sub-satellite position, and sub-satellite local time.			
14. SUBJECT TERMS satellite, control, software, Linux, Command & Data Handling, C&DH, orbit propagation.		15. NUMBER OF PAGES 61	16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT UL

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release; distribution is unlimited.

DEVELOPMENT OF LOW-COST SATELLITE CONTROL SOFTWARE

Bryan D. Waterman
Lieutenant, United States Navy
B.A., Auburn University, 1995

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN ASTRONAUTICAL ENGINEERING

from the

**NAVAL POSTGRADUATE SCHOOL
June 2005**

Author: Bryan Waterman

Approved by: Jim A. Horning
Thesis Advisor

I. M. Ross
Second Reader

Anthony Healey
Chairman, Department of Mechanical and
Astronautical Engineering

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

This thesis details the development and validation procedures for the experiment control software to be used on the NPSAT1 spacecraft. The software was developed for a Linux kernel instead of an expensive proprietary space computer operating system, and includes functions for satellite orbit prediction, precise satellite location, and adaptive experiment scheduling using inputs from the electrical power system, sub-satellite position, and sub-satellite local time.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

I.	INTRODUCTION	1
A.	BACKGROUND	1
II.	NPSAT1 OVERVIEW	3
III.	EQUIPMENT	5
A.	C&DH OPERATING SYSTEM	5
B.	HARDWARE	6
1.	Command and Data Handling	6
2.	Electrical Power System	9
3.	Attitude Control System	9
4.	Communication System	10
5.	Onboard Experiments	10
a.	<i>Coherent Electromagnetic Radio Tomography (CERTO)/Langmuir Probe</i>	<i>11</i>
b.	<i>Visible Wavelength Imager (VISIM)</i>	<i>12</i>
c.	<i>Configurable Fault Tolerent Processor (CFTP)</i>	<i>12</i>
d.	<i>Solar Cell Measurement System (SMS)</i>	<i>13</i>
IV.	PROGRAM DESIGN CONSIDERATIONS	15
A.	PROGRAM ENVIRONMENT	15
B.	HARDWARE/SOFTWARE CONSTRAINTS	16
C.	ON-ORBIT PROCESSES	16
V.	SOFTWARE DESIGN	19
A.	PROGRAM OVERVIEW	19
1.	DATA STRUCTURES	19
a.	<i>List Structures</i>	<i>19</i>
b.	<i>SatData Structure</i>	<i>21</i>
c.	<i>Other Structures</i>	<i>21</i>
B.	PROGRAM MODULES	22
1.	Orbit Propagation	22
2.	Event Scheduling	25
3.	Power Estimation	28
4.	Idle CPU Usage	29
5.	Data Compression	29
6.	Communications Scheduling	29
8.	DATA ENCRYPTION/DECRYPTION	30
VI.	ORBIT PROPAGATOR VERIFICATION PROCEDURES	33
VII.	FUTURE WORK AND CONCLUSIONS	39
A.	CONCLUSIONS	39
B.	FUTURE WORK	40

LIST OF REFERENCES	41
APPENDIX I	43
1. Vector Structures	43
2. List Structures	43
3. SatData Structures	43
4. Miscellaneous Structures	44
APPENDIX II	45
INITIAL DISTRIBUTION LIST	47

LIST OF FIGURES

Figure 1. NPSAT1	3
Figure 2. NPSAT Block Diagram (Sakoda & Horning, 2002)	6
Figure 3. C&DH overview (Horning, 2002)	7
Figure 4. Generic List Structures	20
Figure 5. Combined List Structure	21
Figure 6. SPG4.c logic	24
Figure 7. Overhead Pass Geometry	27
Figure 8. Earth Centered, Earth Fixed Coordinate System	34
Figure 9. Error Analysis for Individual Axis	35
Figure 10. Combined Error	35
Figure 11. Long Term Error Analysis	36
Figure 12. Latitude and Longitude Errors	37

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGMENTS

The author wishes to give special acknowledgement to the NPSAT1 design team for all their assistance with the programming work and preparation that went into this thesis.

Dan Sakoda
Jim Horning
Ron Phelps

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

A. BACKGROUND

In 2006, the Naval Postgraduate School (NPS) will launch a low-earth orbiting satellite, NPSAT1. This satellite will have a mixture of experiments from the Naval Research Laboratory and NPS students. This thesis describes the development and validation of portions of the software that will control NPSAT1. The software will be used on-board the satellite, as well as on the ground station.

In general, software is one of the more complex systems on board a modern satellite (Wertz and Larson). Software is the least reliable portion of a satellite and most likely to cause delay and/or actual failure. Most satellite software packages are custom written with little or no re-use from previous efforts. NPSAT1 will use standard PC components such as the PC/104 bus and the Linux operating system. This allows programmers to develop code in high level languages on PCs using common and low cost development tools. Thus, re-use of this non-specialized code is simpler and less coding intensive.

From its initial concept, NPSAT1 was designed with commercial off-the-shelf (COTS) hardware and software, including the main processor. The controlling software will be designed and written to run on the Open Source operating system GNU/Linux. The software will be written using commonly available tools which do not require proprietary license or expensive custom-built operating systems. Much of the software will be written as independent, re-usable modules. Some of these modules will

be released under the GPL (General Public License), and available for use by other satellite programs, or other students projects. The portions of the software written for this thesis are available in electronic format from the attached CDROM.

II. NPSAT1 OVERVIEW

NPSAT1 is a 81.6 kilogram [180 lb] satellite manifested on the Department of Defense (DOD) Space Test Program (STP) MLV-05 Delta IV mission due to launch in October 2006 into a 560 kilometer orbit with an inclination of 35.4 degrees. The spacecraft is a 12-sided cylinder with body-mounted solar cells on all twelve sides with the top and bottom used for antennas and experiment interfaces. See Figure 1.

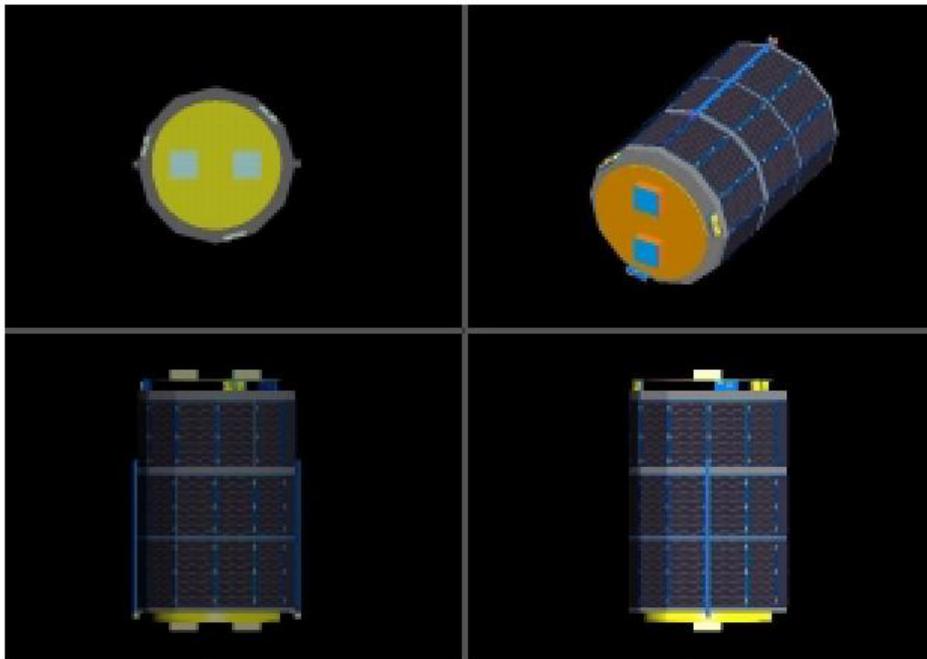


Figure 1. NPSAT1

NPSAT1's primary goal is providing educational opportunities for NPS students. NPSAT1 provides a basis for class projects for the Space Systems curricula and is a unique platform for hands-on thesis work. NPSAT1's flight mission has two parts. It provides a platform for several space-based experiments and it provides a test flight for

the solar cells. There are five space-based experiments: two from the Naval Research Lab (NRL) and three from NPS.

The NRL experiments are a coherent electromagnetic radio tomography (CERTO) experiment (a three-frequency radio beacon), and a Langmuir probe that measures plasma properties such as ion density.

The NPS experiments are a configurable fault tolerant processor (CFTP), a three-axis micro electro-mechanical (MEMS) rotation rate sensor system, and a visible light imaging camera (VISIM).

NPSAT1 will serve as a flight test of advanced, triple-junction solar cells with a measurement system that profiles battery properties and operational characteristics while in an orbital environment.

III. EQUIPMENT

NPSAT1 is composed of several subsystems:

- Command and Data Handler (C&DH)
- Attitude Control Subsystem (ACS)
- Electrical Power Subsystem (EPS)
- Radio Frequency Subsystem (RFS)
- Mechanical Subsystems
- NPS and NRL experiments

The C&DH system provides overall control of each of the systems, directing the function of the spacecraft subsystems and experiments, as well as controlling the communications to the ground station. See Figure 2. A description of the hardware and experiments is provided as background information. The selection of hardware greatly influenced the software design process.

A. C&DH OPERATING SYSTEM

The C&DH system is designed to run the GNU/Linux operating system. Current software development is using the 2.4.14 Linux kernel. The kernel will be stripped down as much as possible. It will only support those tools and features needed for the C&DH system to minimize the size of the kernel, maximizing the available computing resources.

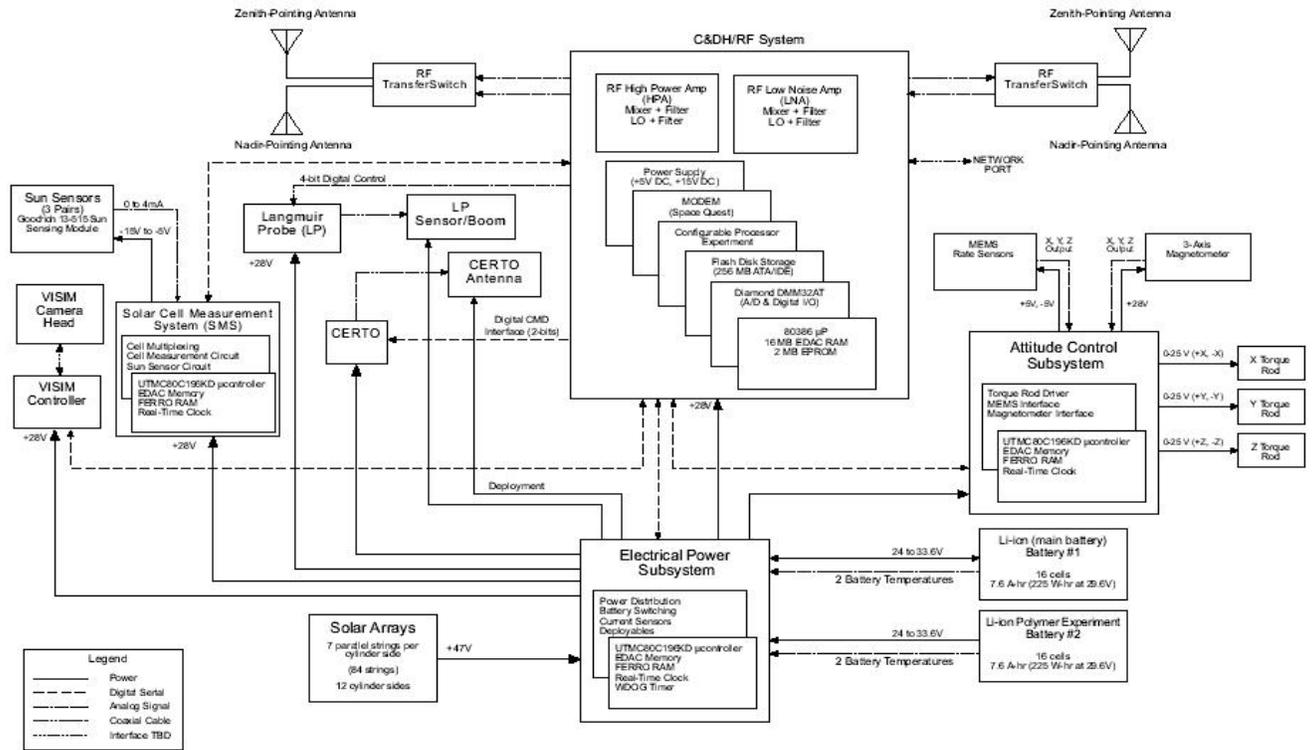


Figure 2. NPSAT Block Diagram (Sakoda & Horning, 2002)

B. HARDWARE

1. Command and Data Handling

In the conceptual sense the C&DH system spans both on-orbit hardware/software and equipment at the ground station since both play an important part in controlling the satellite operations. See Figure 3. This discussion deals only with the onboard portion of the C&DH system.

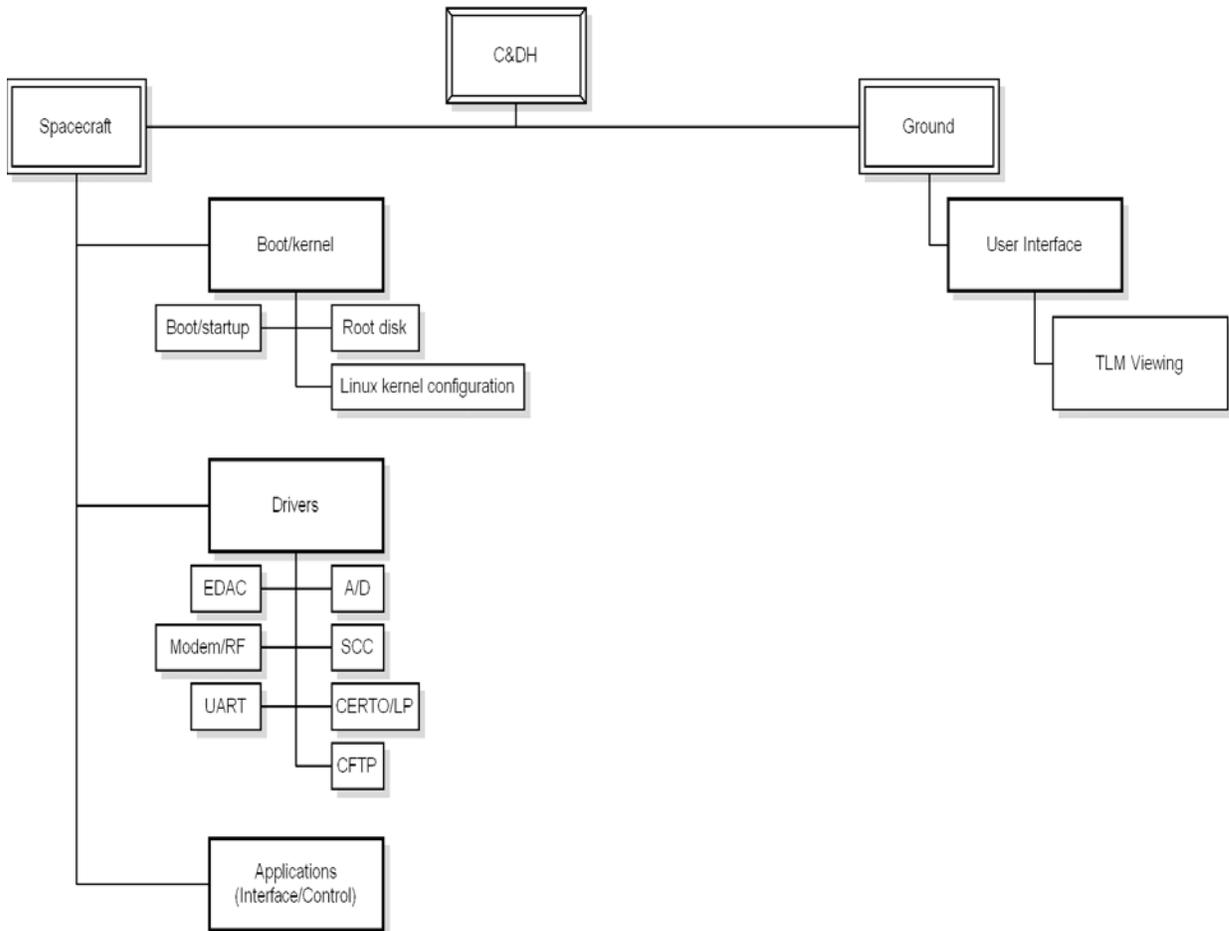


Figure 3. C&DH overview (Horning, 2002)

The main elements of the C&DH system are: the CPU; a 2 megabyte ROM image containing the boot loader, kernel, and root directory; 16 megabytes of error corrected RAM; and a 256 megabyte solid state disk for mass storage of data.

The CPU is an LH79520 System-on-Chip made by Sharp Electronics with a 32-bit ARM7TDMI RISC Core, running at speeds up to 77.4 MHz. This chip is well suited for low power variable duty systems as it consumes only 55 milli-Amps (mA) in active mood, 35 mA in standby mode, and 5.5 mA in sleep mode.

Read-Only Memory (ROM) is a non-volatile type of memory used for permanent storage of data. It can neither be erased nor altered except by malfunction or environmentally caused defects. The ROM has 512 kilobytes dedicated to the BIOS, loader, and kernel. The remaining 1.5 megabytes will hold the root directory of the Linux OS. The boot loader is responsible for initial startup of the C&DH system, including any subsequent reboots. The kernel is the essential center of a computer operating system, the core that provides basic services for all other parts of the operating system.

Random Access Memory (RAM) is the place in a computer where the operating system, application programs, and data in current use are kept so that they can be quickly reached by the computer's processor. RAM is a volatile memory. The 16 megabytes of RAM is error detected and corrected (1 bit error correction and 2 bit error detection). Approximately 1 megabyte will be for the (uncompressed) kernel and device drivers. Approximately 3 megabytes are for the (uncompressed) root disk. The remaining, approximately 12 megabytes, are for the application programs. For NPSAT1, the applications include Linux tools such as BZIP2 (a compression program), as well as possible updates to C&DH software.

Solid-state disks are collections of non-volatile Flash memory organized to appear as a hard disk to the system. They are commonly used in place of spinning platter type disks for space-based systems because they have no moving parts, are more reliable than spinning disk storage, and are more resistant to the space environment.

2. Electrical Power System

The electrical power subsystem (EPS) consists of triple-junction solar cells mounted on the satellite, and a lithium-ion (Li-ion) battery for energy storage. The control electronics are composed of a processor board for digital logic functions and an analog/switching board for power switching and telemetry gathering. The solar cells are from Spectrolab and are part of a flight demonstration which includes the solar cell measurement system (SMS) experiment. (Sakoda & Horning, 2002)

The interface between the EPS and the C&DH uses 16550A compatible Universal Asynchronous Receiver Transmitters (UARTs) but the data exchange is minimal, as the battery charge/discharge is controlled by the EPS system, rather than the C&DH. The EPS provides two important inputs to the C&DH system: current battery state (expressed as a percentage of full capacity) and current electrical production of the solar cells (expressed in milli-Amps).

3. Attitude Control System

The attitude control subsystem (ACS) consists of three magnetic torque rods, a three-axis magnetometer, and the ACS controller. The ACS controller uses onboard orbit information from the C&DH system, and uses a table lookup to determine the expected magnetic field vector for that location so that attitude can be calculated. Attitude time rate of change is measured by three MEMS rate gyros. (Sakoda & Horning, 2002)

The interface between the ACS and the C&DH system also uses UARTs. Data packets between the ACS and C&DH will

include, at least: position information (latitude, longitude, and altitude), system time information (GMT), and high level commands. Of these data types, the position information will be passed most frequently since the ACS controller expects updated position data approximately every 2 seconds.

4. Communication System

The NPSAT1 radio frequency subsystem (RFS) is a full-duplex system with 100 kilobits per second transmission rate on both the uplink and downlink channels. NPSAT1 will operate at 1767.565 MHz in the forward, or uplink channel, with a down link at 2207.3 MHz. The RFS uses a single-conversion to baseband with 70 MHz as the intermediate frequency (IF). (Sakoda & Horning, 2002)

The RFS acts as a link between the ground station portions of the C&DH and the on-orbit equipment, transmitting the internet protocol (IP) packets generated by the satellite to the ground station. Experimental data and telemetry will be transmitted as compressed files to maximize the amount of data that can be transmitted to the ground.

5. Onboard Experiments

NPSAT1 will host several experiments that will be under the control of the C&DH system. The experiments have time, location, and/or power limitations so scheduling the experiments is one of the major elements of the C&DH system.

a. Coherent Electromagnetic Radio Tomography (CERTO)/Langmuir Probe

The CERTO experiment is a radio beacon that transmits at three frequencies; 150, 400 and 1067 MHz. The space-based beacon, in conjunction with a network of ground receivers in the United States, South America, and India, will be used to measure the integrated electron density of the ionosphere in the plane of observation. CERTO will also be used to develop and test tomographic algorithms for reconstruction of ionospheric irregularities; to provide a database for global models of the ionosphere; to characterize the ionosphere for geolocation; and to perform scintillation studies of the ionosphere. (Sakoda & Horning, 2002)

CERTO and Langmuir Probe has four modes of expected operation. The first mode is for system checkout. The second mode is the mode used in normal day-to-day operations where the objective is to collect total-electron content and scintillation data from a chain of ground receivers. The 150 and 400 MHz frequencies transmit when NPSAT1 is overhead a ground receiver. Additionally the 1067 MHz beacon transmits when the sub-satellite point has a Latitude within $\pm 15^\circ$ of the equator and local time is between 6 PM and midnight. During normal operations, the Langmuir probe will be enabled collecting four separate 12-bit A/D channels at samples rates between 1 and 1000 samples per second. The last two modes will occur several times a year for a one to two day period. These special modes are called CERTO/CITRIS Tandem modes. Another satellite launched on the same mission will be placed into a similar orbit with NPSAT1 and is hosting CITRIS, the

space-based receiver counterpart to CERTO so conjunctions between that satellite and NPSAT1 will occur. The objective will be to perform tandem operation for Ionospheric monitoring of small scale irregularities. One of the Tandem modes will operate for about eight minutes collecting Langmuir Probe data. The other mode will operate for up to two days but not operate the Langmuir probe.

b. Visible Wavelength Imager (VISIM)

The Visible Wavelength Imager (VISIM) is a COTS digital camera operated by a COTS single board computer. The CCD controller is a PC/104 board. The single-board computer is a 486 processor running at 66 MHz supporting the PC/104 bus. This system will normally be powered off and turned on only for a small period of time when passing over an area of interest to take images and process the data.

The CCD controller produces raw image data in a format known as Bayer pattern. This raw data will be lossless-compressed using bzip2. A lossy representation using JPEG at very high compression will be used as a preview of the image. The VISIM computer sends the lossless and lossy versions of the raw data back to the C&DH for temporary storage before a future download to the NPS ground station.

c. Configurable Fault Tolerant Processor (CFTP)

The Configurable Processor (CFTP) is an NPS designed and built project. This experiment consists as a single electronic circuit board and housed within the C&DH. It interfaces with the C&DH motherboard via the PC/104 bus.

The CFTP design is a low power re-configurable processor. It will allow in-flight modifications to the processor configuration. One configuration will implement a triple redundant microprocessor using voting logic to study single event transient mitigation. Another configuration will implement image compression capable of producing JPEG representations of the image data taken by the Visible Wavelength Imager experiment.

d. Solar Cell Measurement System (SMS)

The purpose of the Solar Cell Measurement System is to gather on-orbit operational data for the triple junction solar cells provided by Spectrolab. The twenty four test cells will be placed around the body of the spacecraft, two per side, in order to collect data about the performance of the solar cells. The Solar Cell Measurement System will perform IV curve traces twice per orbit for each illuminated test cell. The data collected will be used to demonstrate the suitability of this type of cell in solar powered space based systems and document their specific performances parameters.

THIS PAGE INTENTIONALLY LEFT BLANK

IV. PROGRAM DESIGN CONSIDERATIONS

A. PROGRAM ENVIRONMENT

The program environment utilizes common Linux tools, kernel Application Program Interfaces (APIs), and programming languages. However there are important limitations imposed by the limited resources available on orbit.

Utilizing a Linux environment as much as possible contributes to the project in several ways. GNU/Linux brings with it a common framework which is particularly important for projects that span many months and involve several contributors. The C&DH software for NPSAT1 will likely be written by several authors, so the Linux programming environment is just one of the many benefits of the design. Algorithm development can be performed on a desktop PC running Linux without the need for the ARM development board or the on-orbit hardware.

All of the existing modules are written in ANSI compliant C so that the written code is platform independent as much as possible. The primary advantages of C are execution speed, easy interface with the operating system, and the compactness of the resulting programs. C was also chosen for its flexibility, particularly with regard to transition of string data to numeric and the reverse; for ease of programming; as the programming language of choice of the lead programmer; and to simplify the required set of developer's tools.

B. **HARDWARE/SOFTWARE CONSTRAINTS**

The C&DH system has several constraints that are imposed on it by remote location and limited availability of power. These constraints strongly affect the design of the system.

The remote location (onboard an orbiting satellite) imposes two constraints: the satellite must be able to perform an unattended boot/reboot of the C&DH system when commanded from the ground station, and essential services such as satellite operation must be self starting. Both of these constraints are accounted for by the design of the boot-up sequence.

The limited power constrains the C&DH system in three ways. The component hardware of the system is chosen with power usage as one of the major design factors. The C&DH system must be operated to minimize power usage; and since the C&DH system is responsible for directing the operation of the other systems and experiments, it is responsible for scheduling and using them within the available power limits. The first of these constraints was dealt with during hardware selection as the power consumption of the Sharp LH79520 was one of the primary reasons for selecting that component. The low power states of the CPU reduce the operational power requirements of the C&DH system. It is the last constraint that affects the software development most strongly. This constraint must be dealt with by the scheduling software.

C. **ON-ORBIT PROCESSES**

The actual on-orbit processes are to be made as compact and efficient as possible. This is sensible both

from the power reduction point of view and because onboard computing resources are scarce. Hence most the software is written to be called from other processes. Once a process is started, it actively runs for just the length of time needed to produce the required result and then exits or returns to an idle state using the sleep() function. The results of the process are left behind as a file. These result files can then be read by other processes as needed. With traditional disk storage, this strategy would incur a performance penalty from the comparatively long access time associated with disk drive reads. For this system, the fast access times for the solid state disk minimizes that performance cost.

THIS PAGE INTENTIONALLY LEFT BLANK

V. SOFTWARE DESIGN

A. PROGRAM OVERVIEW

The current form of the project is a series of independent modules that are designed to perform discrete tasks. The general format of the modules is read input from a file, passed in as a file name (if input is required), and write the output of the operation to another file before exiting or sleeping. This design conserves computing resources, minimizing power usage by minimizing CPU usage, and taking advantage of the relatively abundant storage of the solid state disk as compared to memory. This is particularly important with regard to orbit determination. It is one of the major computing tasks for the system, and the resulting data is used by several other programs.

1. DATA STRUCTURES

There are several important data structures that are embedded in the code that will shape further programming efforts. Some data structures, such as those used by the orbit propagator, are rigid and can not be modified greatly because of their specialized nature, however some structures like lists of events are more general.

a. List Structures

The 'list' concept is well understood in computer science and lends itself to a high degree of abstraction. In this vein, a data structure called ListType has been created to implement a singly linked list. Since all the anticipated uses of the list data type are either unidirectional (a list of satellite events in time sequence) or order independent (processing a list of locations to

determine the best time to activate the VISIM), a singly linked list was the chosen over a doubly linked list. ListType and the associated nodeType are shown graphically in Figure 4. The pointer to data is what allows the listType to be abstracted. It allows the list to contain any type of data that may be needed.

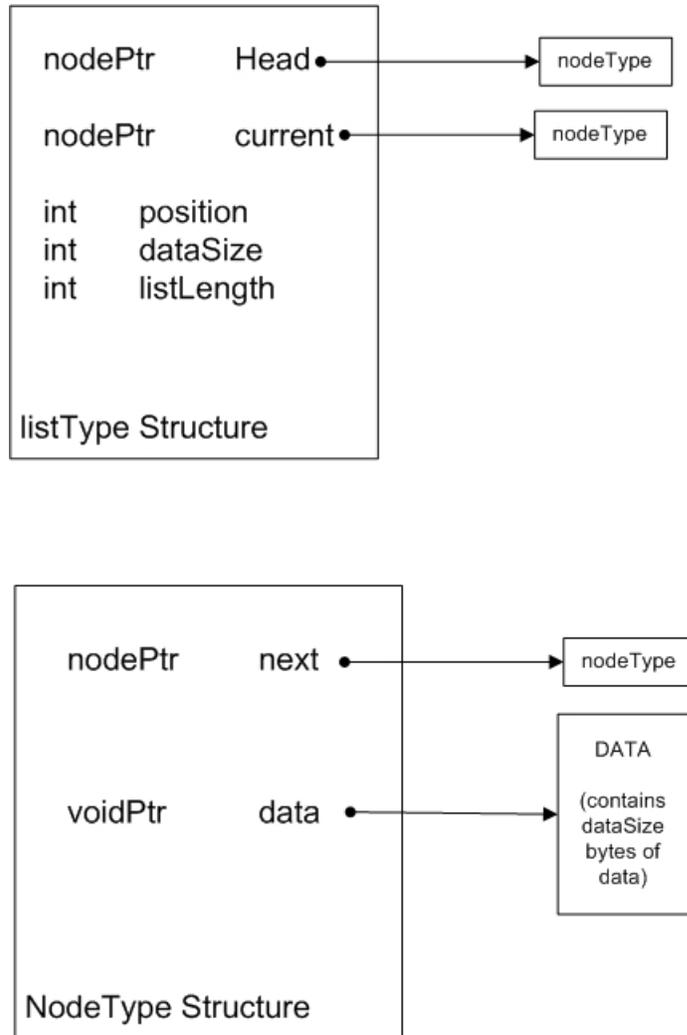


Figure 4. Generic List Structures

The combination of a listType with a nodeType allows the list to contain any sort of data since a generic pointer is used, instead of a pointer to a specific data type. The combined data structure is shown in Figure 5.

The C code specification for both structures is given in appendix II.

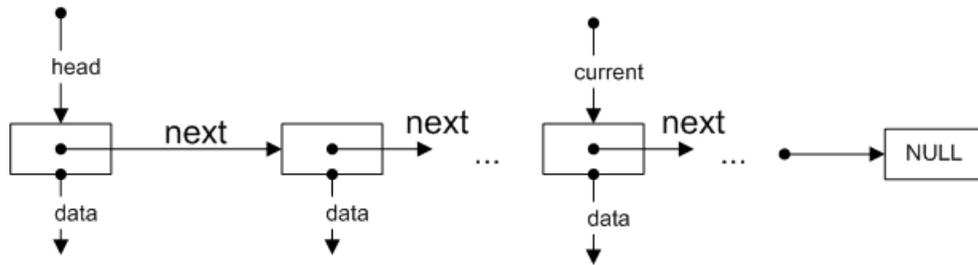


Figure 5. Combined List Structure

b. SatData Structure

The largest data structure is the SatData type, as this combines two other defined types. SatData is the union of SatElem, and sgp4_data. SatElem, short for satellite elements, refers to the six Keplerian orbital elements plus the rates of change parameters as defined by NORAD for satellite tracking purposes. SatData consists of the results of many preliminary calculations, some of which are time dependent, some of which are not. The propagation algorithm is written so that the non-time dependent quantities only need to be calculated once. appendix I appendix I shows the formal declaration of these types.

c. Other Structures

Most other data types are designed to be used with the listType, and contain just a few discrete fields containing predefined aspects of that type of data. For example, siteType has a latitude, longitude, height above sea level, name, and a minimum elevation above which line of site can be established with NPSAT1. Later development can add more fields to these data types since the listType functions are generic in nature. The disadvantage to this style of programming is that fields can not be removed

without careful editing of the other modules to ensure that they no longer reference the deleted field.

B. PROGRAM MODULES

1. Orbit Propagation

Orbit propagation is the process of determining the exact path a satellite follows as it orbits the earth. The process begins with initial data; position, velocity and the associated time, and then applies the laws of orbital mechanics to predict the satellites position and velocity at any other time. A typical process is stepwise, proceeding from the start time to the end time by small increments, producing a position and velocity at each step. Each successive position is separated by a difference in time, referred to as the 'time step' for that propagation. Small time steps mean that the satellite has not moved very far since the previous position, and larger time steps require less computing resources. The goal is balancing the trade off to meet the requirements of the different experiments while minimizing CPU usage.

Orbit propagation is a well studied problem, and a variety of codes already exist, such as the commercial package STK, or the open source software version of SATRACK. However, none of the available packages adequately met the specific needs of the NPSAT1 project. NPSAT1 requires a specific type of orbit propagation, with a premium on ease of calculation and relative paucity of computing resources. The problem was solved by the location of a C language source code package written by B. Magnus Bäckström which implements the SPG4 algorithm originally published by NORAD in 1998 (Hoots, Roehrich, and

Kelso). This algorithm forms the basis of most medium precision orbit determination programs, allowing accuracy in the range of 1-5 kilometers.

The Bäckström code was used as a starting point for the current orbit propagator. It was trimmed extensively and modified to fit the restricted nature of the orbit that NPSAT1 will occupy.

Orbit propagation is an essential function of the C&DH system, as experiment scheduling and communication with the ground station depend on knowing where the satellite will be in relation to particular points on the earth's surface. The problem is further complicated by the need to express the satellite's position in several different coordinate systems and by the requirement of knowing the local time of the sub-satellite point.

NPSAT1's required attitude is nadir pointing so that VISIM points toward the earth and CERTO is oriented perpendicular to the velocity vector of the satellite. The attitude control system needs almost constant knowledge of the satellite's position so that it keeps the satellite in the correct orientation. For this reason the orbit propagator will be kept as a running process, but sleeping until called upon. The ACS will be asking for current position on a frequent basis. Other programs will call on the propagator to produce results for various durations and using different time steps, but these will occur less frequently. The interface of the orbit propagator has been designed to handle both types of requests.

The basic logic of the orbit propagation is:

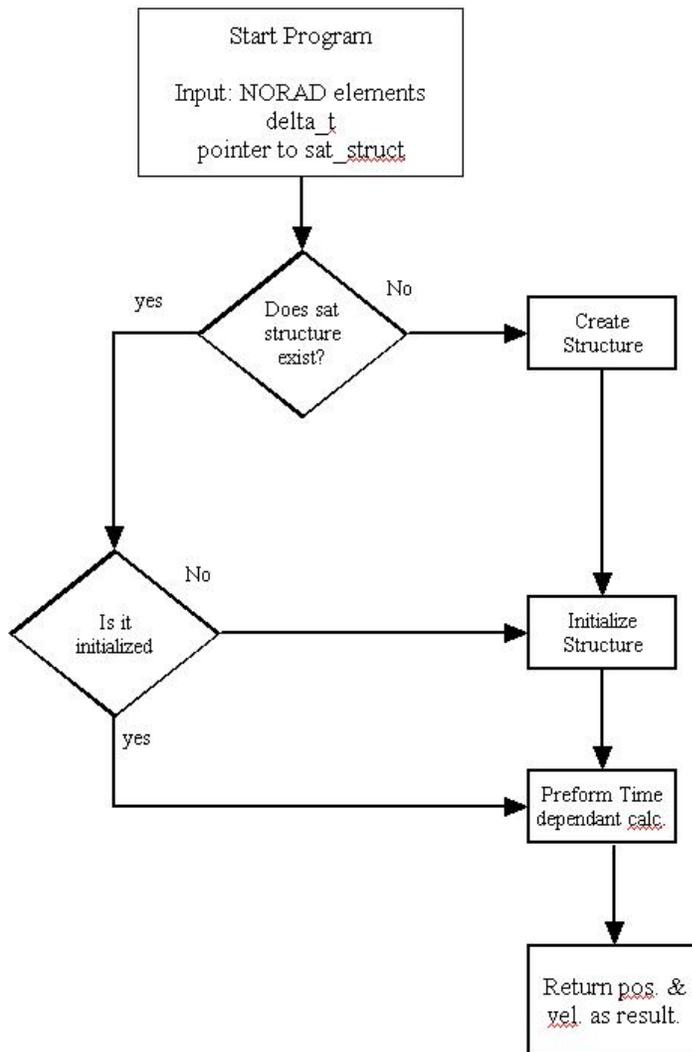


Figure 6. SPG4.c logic

Figure 6. shows one of the design efficiencies of the spg4.c program. It only does the initial calculations once for a given set of orbital predictions. The calculation of the time dependent quantities only comprises about one hundred lines of code, making it possible to do many sequential propagation calculations with good efficiency. This is important since it is desirable to use as small a time step between orbit positions as possible.

Current design uses a five minute time step between position calculations. This corresponds to steps of about 20 degrees of longitude and 7 degrees of latitude. Further testing on the actual satellite CPU will determine the feasibility of moving to a finer time step. A finer time step gives more accurate scheduling of events but it does come at a cost of CPU usage. The best balance between the two will have to be determined by testing.

2. Event Scheduling

Event scheduling works using a series of modules that calculates a schedule for each different experiment. Each module is coded with the specific rules for that experiment, with all the modules producing a result file. The master scheduler reads all the result files and integrates them as needed to produce a single comprehensive schedule for the satellite.

Most of the experiments such as CERTO and VISIM have specific rules that can be easily coded but some experiments do not have (at this time) adequately defined requirements to allow operation to be determined by software. These experiments require operator input to control precisely. The CFTP is one of these experiments.

The most complex schedule module is the VISIM, since it requires significant additional computation to determine the time at which the satellite is overhead the target with sufficient precision. The sub-satellite point moves about 6.97 km/sec and VISIM has a view area of 147 x 194 km, so the target could move from center of frame to completely out of the field of view in as little as 10.5 seconds, assuming 73.5 km of travel at 6.97 km/sec. Pointing errors

in the satellite attitude could decrease this time even further if the line of sight of the camera is not pointed at the sub-satellite point with the worst case being that the camera line of sight is forward of the sub-satellite point. The goal is to control the shutter activation of VISIM with a precision of one second. Any greater precision would be meaningless as the best accuracy of the location from the propagation code is approximately the same as one second of satellite travel.

The VISIM scheduling application is designed to run on the ground for two reasons. The first reason is that a planned web application will allow outside individuals a chance to request images of a given location. Secondly, to allow these people and the ground station operations a good selection of overhead passes to choose from, orbit propagation will be done for 14 days, instead of the shorter time frame that the satellite will use on orbit. This will minimize use of on orbit computing resource (where they are scarce) by using ground station computers (plentiful).

VISIM scheduling is done with the PictureSchedule.c module. The goal is to find optimal camera activation time by determining the precise time at which the elevation angle from the target to the satellite is the greatest.

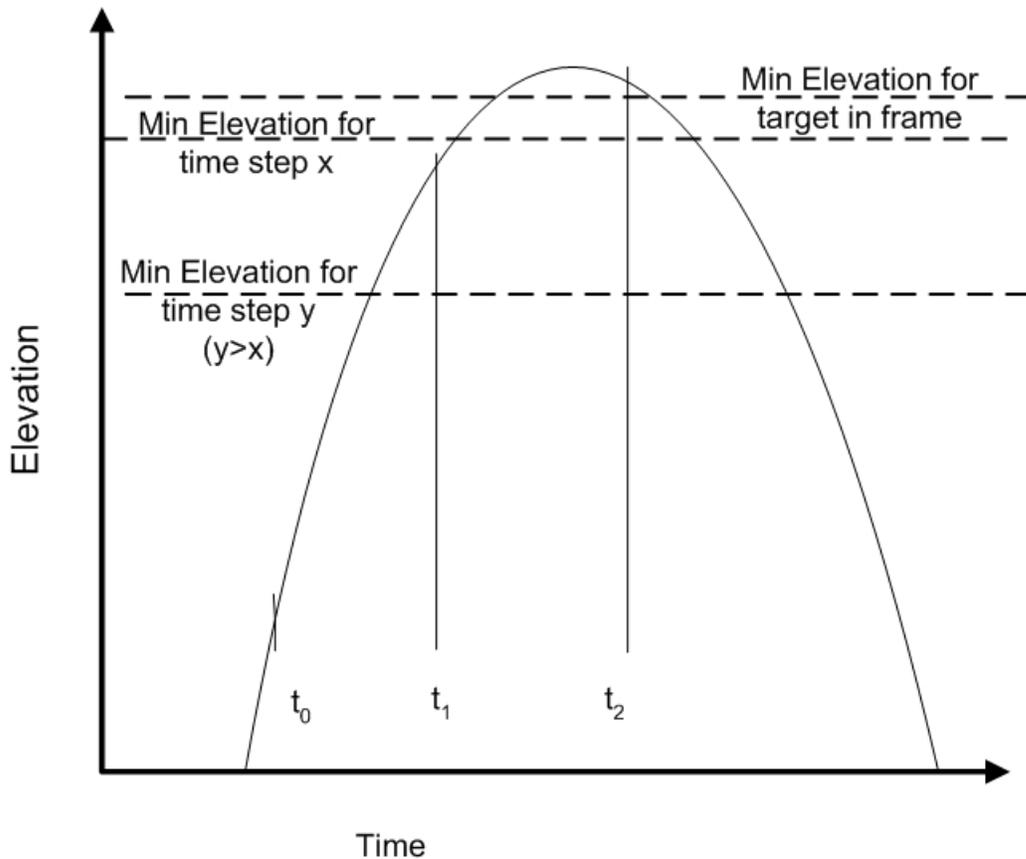


Figure 7. Overhead Pass Geometry

A typical overhead pass is shown in Figure 7. It shows the satellite passing over head, starting at a low elevation, climbing to a peak and then descending till it is over the local horizon. The optics of the VISIM camera and the satellite altitude give a required elevation of 83 degrees for the target to be in the field of view of the camera. Because the maximum elevation for a given pass will occur between two time steps, it is useful to know the minimum elevation the satellite needs to have and still reach the 83 degree elevation required to put the target in the field of view before the end of the next time step. As the time step between positions increases, that minimum becomes lower.

The program searches for the best overhead time using a two step process. First, pictureSchedule.c runs a propagation with a large time step. This first step determines the ten best time windows to take the picture. The program handles cases in which there are fewer than ten, or even zero. In the second step, pictureSchedule.c calls the propagator again, using a shorter time interval but only for searches around the time windows identified by the first step.

The current code uses 5 minutes for the large time interval, and the second step uses 1 second. Increasing the large time step beyond 5 minutes will result in missing overhead times. Decreasing the time step means using more resources. The large time step should be made large enough so that only one solution per orbit is considered. The small time step was chosen based on the anticipated precision in scheduling events.

The results from the first step (large time interval) is dynamically replaced with more accurate solutions from the second step. The resulting list can either be saved to a file, or used to with the interactive web application.

3. Power Estimation

Power estimation is not implemented on the satellite, but has been modeled in detail by Ron Phelps and Andrew Miller of the NPS Space Systems Academic Group. This allows the operator to run what-if scenarios to maximize the satellite's experiment schedule.

On board the satellite, the scheduling program checks the list of scheduled tasks. This list is a time

sequential list of actions to execute; such as turn on CERTO, or take a picture using VISIM. This list contains information on the calculated power usage that execution will use. If the current power state (inputs supplied by the EPS system) is not sufficient to complete the task, it will be aborted and a log entry detailing the specific circumstance is created.

4. Idle CPU Usage

The C&DH system expects times of relative idleness. The current design includes plans to implement a list of tasks such as data consolidation and compression, system checks, and telemetry tasks which can be completed as resources allow. Linux includes native tools to allow the automatic execution of these tasks when the CPU is idle and suspending them if it becomes busy. The developed code only needs to identify what tasks should be added to the list.

5. Data Compression

Linux includes standard compression tools. Bzip2 will be used to compress most data before transmission to the ground station to maximize the amount of data that can be collected. No other tools need to be coded for this task. Bzip2 will also be used by the ground station to compress data sent up to the satellite.

6. Communications Scheduling

Communications scheduling is a good example of code re-use stemming from the generic way the modules are designed. It is done using the pictureSchedule.c module,

using Monterey's latitude and longitude, but lowering the minimum elevation to 10 degrees above the horizon for line of site radio communications and setting the time step to one minute. Past experience with PANSAT has shown that communications can be established with the satellite once it is above the horizon but this is not allowed due to possible interference by surrounding buildings.

8. DATA ENCRYPTION/DECRYPTION

NPSAT1 has been granted a partial waiver to standard NASA policy of encrypted communications in both the uplink and downlink. As none of the data collected is classified, all down link communications will be unencrypted.

Nearly all uplink communications will be ciphered with a symmetric encryption algorithm, using one of five different keys stored in the satellite's C&DH system. The extra keys are provided in the event of corruption of the first key. To allow for possibility of key corruption, there is a subset of satellite commands that will be accepted unencrypted. This list of commands includes

1. ChangeKey(n) - change to decryption key n
2. ReportHash(n) - transmit hash of key n in unencrypted format

A hash is a mapping algorithm that produces a fixed length result string based on the contents of another string. It has the property that different strings produce different hashes (except in rare instances) and the hash function is one way. That is to say that there is no way to reconstruct the original string from knowing the hash.

There is a slight possibility of a malicious attack on the satellite by using the unencrypted ChangeKey command. Should an attacker issue a ChangeKey command changing the decryption key to m while the ground station is using key n , the initial decryption of the uplink will be random garbage, since applying decryption key n to a block enciphered with key m will not produce a meaningful text.

If this type of attack is suspected, ground station communications can be initiated with a ReportHash request. This will tell the ground station which encryption key is in use, as the ground station will know the hash of all of the keys. Once the correct key is known, it will be used for subsequent uplink communications. Transmitting the hash of the key will not compromise the security of the satellite.

One secure method of detecting meaningful text is to first transmit the hash of the file. Even though hashes are not necessarily unique the chance of decrypting a file with the wrong key and having a matching hash is sufficiently remote very miniscule.

As with all symmetric crypto systems, the security of this system will depend on the security of the key storage at the ground station. For symmetric systems, the encryption and decryption are exactly the same so that if keys are compromised at the ground station, on-orbit security is compromised. Key security at the satellite end is assumed to be negligible since the effort and resources that would be needed to obtain the keys from an orbiting satellite vastly outweighs any value of the keys. The satellite keys will be vulnerable during the time between final program load and launch. This risk is judged to be

about the same as the vulnerability of the ground station
key storage.

VI. ORBIT PROPAGATOR VERIFICATION PROCEDURES

Of all the code, the most crucial and most technical is the orbit propagator. It is the only one that requires expertise beyond general software testing to verify proper operation.

The orbit propagator was verified using the STK commercial package. The module's ability to take Two Line Element (TLE) files as input made for straight forward analysis of the results. See APPENDIX II for a complete description of the TLE format. For the purposes of validating the propagator, the results from STK are considered to be correct and differences from the STK results are considered to be errors.

The first verification procedure used archived TLE sets from the PANSAT satellite over a 24 hour time period. The same values were used in SPG4.c as were given to STK, and the outputs in the earth centered, earth fixed coordinate system were compared. The coordinate axis are typically referred to as I,J, and K as shown in Figure 8. Figure 9. shows the errors for each individual axis and Figure 10. shows the vector magnitude of the individual axis errors.

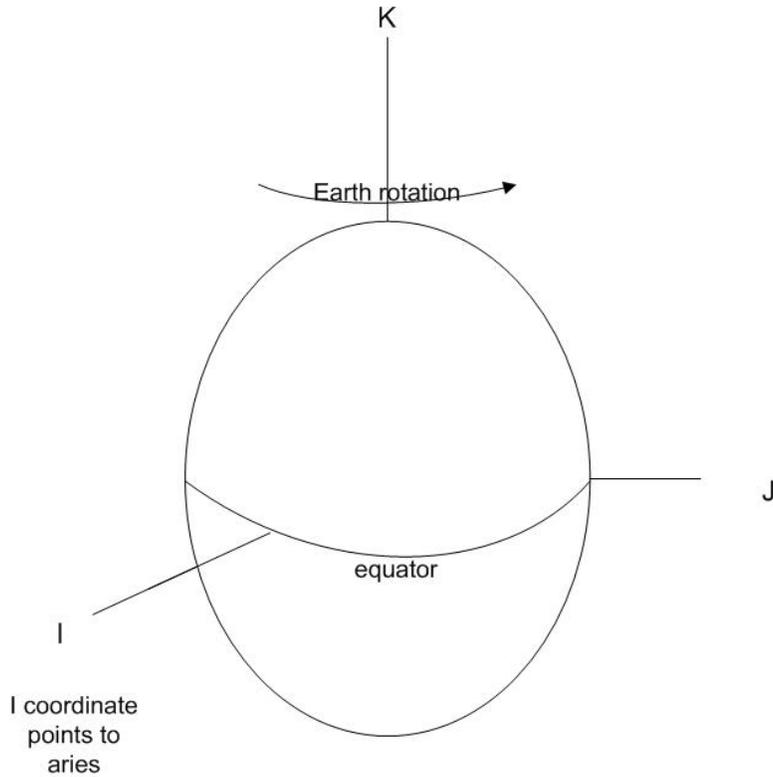


Figure 8. Earth Centered, Earth Fixed Coordinate System

Any TLE set could have been used as long as the satellite is in a low earth orbit with an orbit period less than 220 minutes, since STK uses a different algorithm to predict the orbits of those satellites. PANSAT TLE's were used since they were available, they fit the orbital requirements, and a two week long series was available (a requirement for the 2nd validation procedure).

Short Term Propagator Error

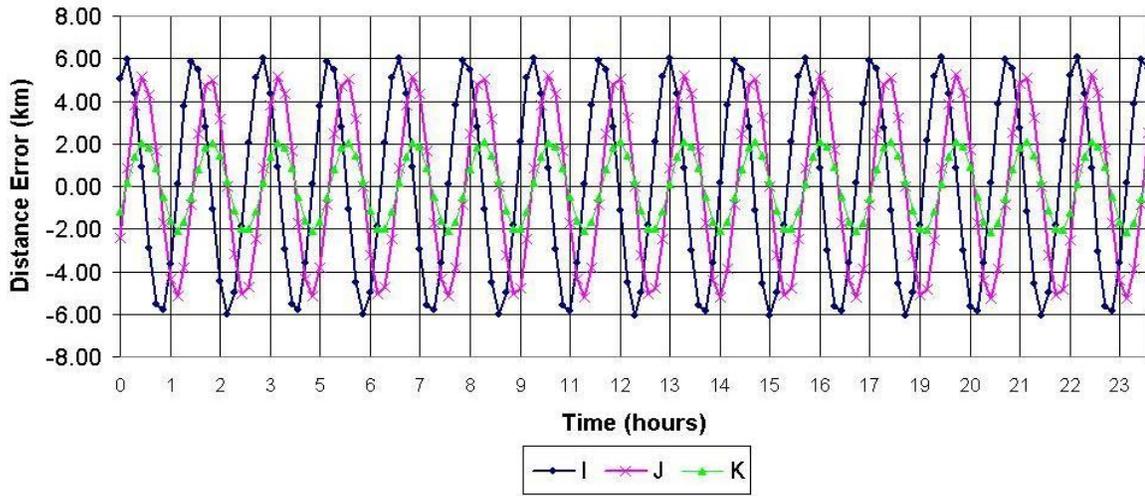


Figure 9. Error Analysis for Individual Axis

Short Term Propagator Error

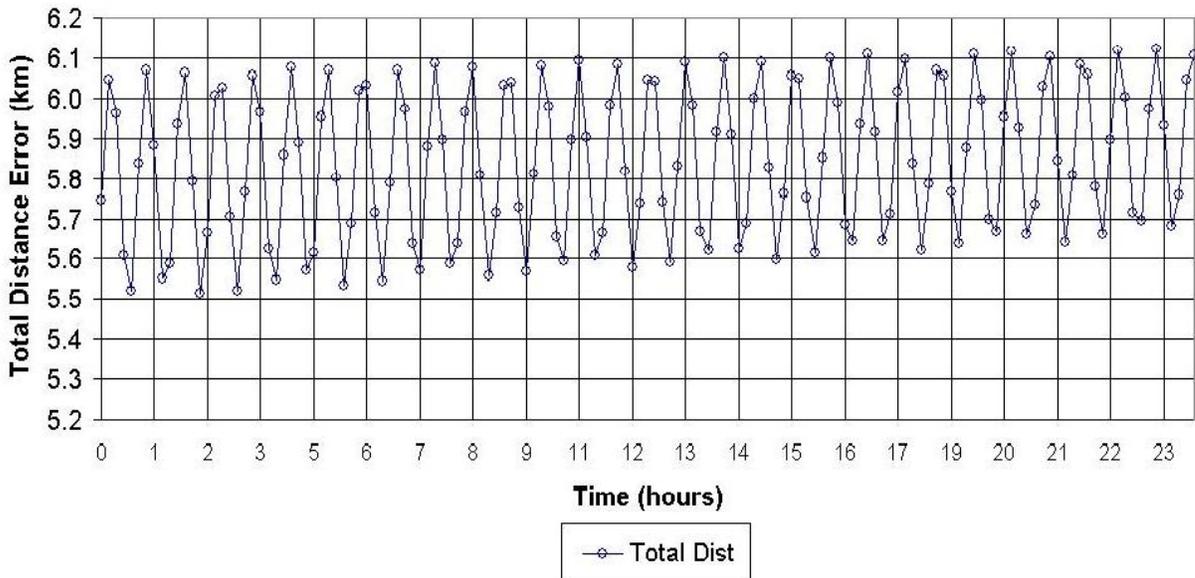


Figure 10. Combined Error

The 2nd validation involves the accuracy of propagation over a long time period. While TLE sets are generally available every three days, there is the possibility that

new sets will not be available, or there could be difficulties transmitting the new TLE data to the satellite.

To evaluate the consequences, a series of TLEs for PANSAT were used. To test the long term accuracy of the orbit propagator, it was used to generate two weeks of orbit positions from the initial TLE. The positions on the 14th day were compared to the STK predictions with STK using TLE from that day. Figure 11. shows the magnitude of error that can be expected from not receiving updated TLE over a two week period. Figure 12. shows this error in terms of the latitude and longitude of the sub-satellite point.

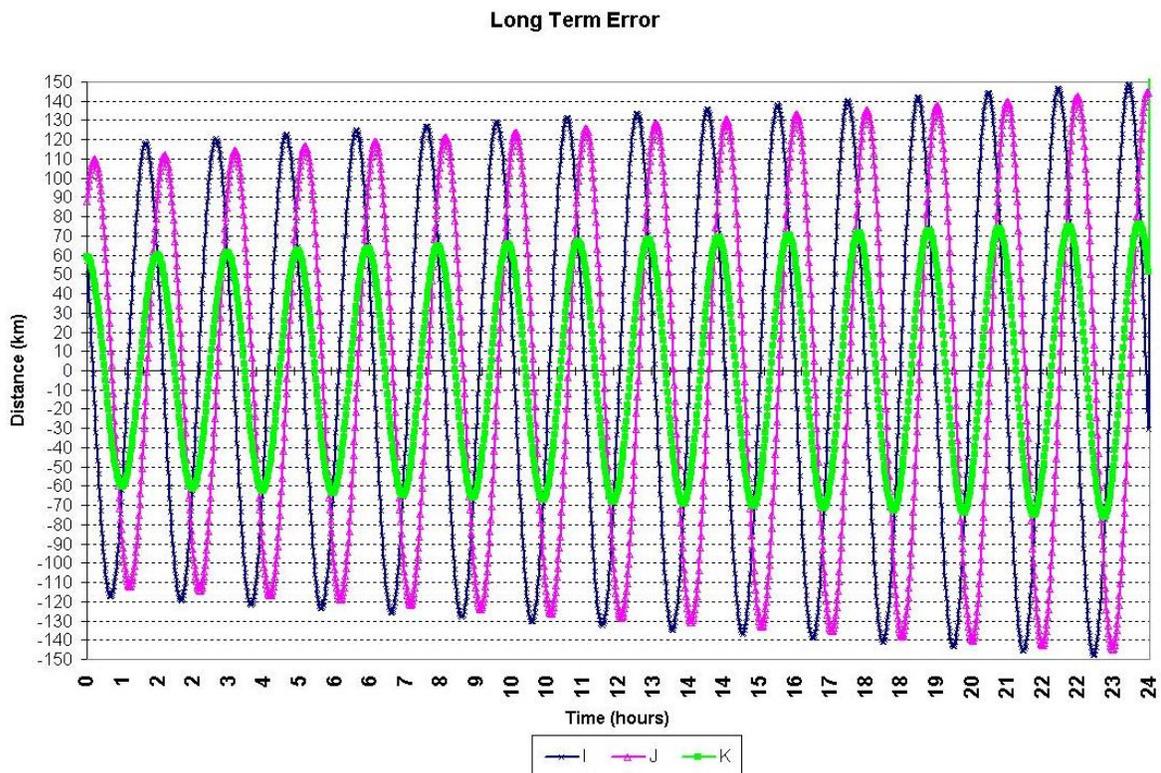


Figure 11. Long Term Error Analysis

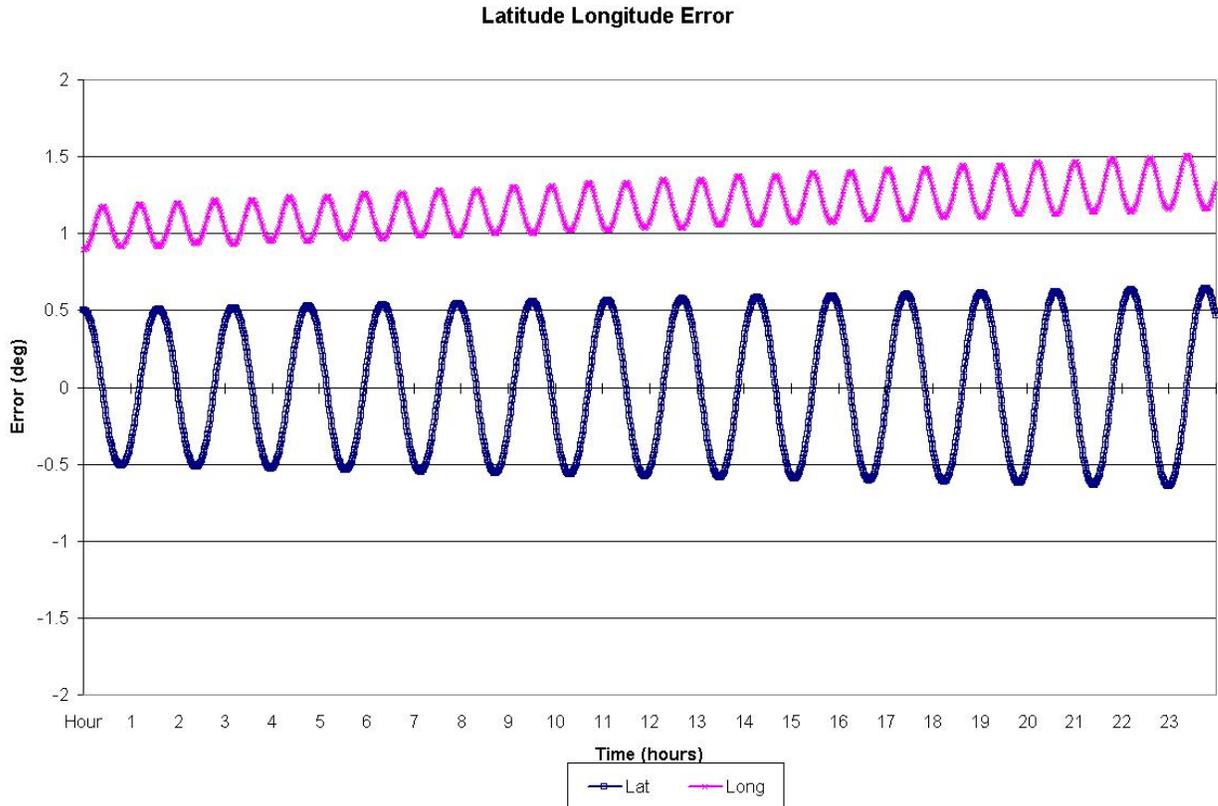


Figure 12. Latitude and Longitude Errors

The errors for each axis during the 15th day are shown in Figure 11. The magnitude of error has grown to about 90 km over the 14 days, mostly due to slight changes in orbital period of the satellite over that time frame. Ignoring the short term periodic errors, the graphs show a slow linear growth in absolute magnitude. The linear growth in the magnitude of the errors is expected and is the result of the slight changes in orbital period of the satellite.

A crude analogy is the distance between the tip of the second hand and minute hand of a clock. If the hands start out at the same place and rotate at the same rate, the distance between the tips never changes. If the rotation rate of one hand changes, the tips will start to move

apart. If the rates are only slightly different the distance grows slowly compared to the time it takes for the hands to rotate. For example if one hand takes one minute to rotate, and the other takes 61 seconds, it will take 30 minutes for the hands to be the maximum distance apart, after which the distance will start to decrease. The error in position from the propagator exhibits this long term, slow growth

VII. FUTURE WORK AND CONCLUSIONS

A. CONCLUSIONS

The design, construction, and operation of a satellite present unique challenges, particularly in the area of software design and implementation. The limited resources and unique environment alter the standard methods of approaching problems slightly. The software design and implementation for this thesis are a small part of the overall system but there are still some distinct conclusions to be drawn from this work.

Software for a satellite does not necessarily have to be written on the hardware that will run the final software. The Linux OS, by virtue of running on a large variety of hardware is a valuable tool in software implementation for satellites since it allows the development of software on an entirely separate platform, and not be restricted to flight hardware.

The construction of general methods may be more complex to implement at first, but they pay dividends in later development as previously coded and tested methods are re-used. This was demonstrated quite clearly in the implementation of `listType`, which is currently used in three other modules, and will likely be used in several others in the future. The initial implementation was concrete; instead of using a void pointer to reference the data, it used a pointer to a specific data type. Later development on the picture scheduling code required the same functionality, except with a different data type. A quick rework of `listType` into a generalized method

demonstrated just why general methods are preferred. This allowed it to be re-used in other situations as well.

A general truism of programming is that modules should be written to accept the simplest input possible. The orbit propagator uses this concept to good advantage. The orbit propagator module is used by the ACS system, the VISIM scheduler, and the CERTO scheduler. It also facilitates the two step search method for VISIM scheduling, making that program more efficient.

B. FUTURE WORK

NPSAT1 as a whole, and the software for the C&DH system in particular, are very much a work in progress, and will be so even after the launch of the satellite. Specific future work includes consolidating the scheduling modules, collecting the output of all the subroutines written into one cohesive schedule of all system events.

The cryptographic design of the communication subsystem needs to be coded. Many aspects of data security have been considered, and these aspects have been taken into account, the design needs a concrete implementation.

Telemetry also needs to be worked with extensively. All the subsystems are designed with various measured parameters to gauge the health and functionality of the satellite. The telemetry code will collect these measurements and organize it into a useful form. None of the work to date deals with that aspect of the C&DH system.

LIST OF REFERENCES

1. D. Sakoda, & Horning, J.A.(Aug, 2002) Overview of the NPS Spacecraft Architecture and Technology Demonstration Satellite, NPSAT1, Proceedings of the Sixteenth Annual AIAA/Utah State University Conference on Small Satellites, Paper SSC02-I-4, Logan, Utah.
2. Hoots, F.R., Roehrich, R.L., Kelso, TS (Dec, 1988) Spacetrack Report No. 3 - Models for Propagation of NORAD Element Sets.
3. <http://www.gnu.org/copyleft/gpl.html> accessed Dec 2003.
4. Wertz, James R., Larson, Wiley j. (1999) Space Mission Analysis and Design, Microcosm Press, Third Ed.
5. Kochan, Stephen G, (1988) Programming in ANSI C, Hayden Books, First Ed.

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX I

DATA STRUCTURES

1. Vector Structures

```
typedef struct _Vec3 {
    double x, y, z;
} Vec3;
```

2. List Structures

```
typedef struct nodeStruct *nodePtr;
typedef struct nodeStruct {
    void *data;
    nodePtr next;
} nodeType;
```

```
typedef struct listStruct {
    nodePtr head;
    nodePtr current;
    int position;
    int dataSize;
    int listLength;
} listType;
```

3. SatData Structures

```
typedef struct _SatData {
    struct _SatElem *elem;
    union {
        struct sgp4_data *sgp4;
    } prop;
} SatData;
```

```
typedef struct _SatElem {
    double se_XMO;
    double se_XNODEO;
    double se_OMEGAO;
    double se_EO;
    double se_XINCL;
    double se_XNDD60;
    double se_BSTAR;
    double pad1;
    double se_XNO;
    double se_XNDT20;
    double se_EPOCH;
} SatElem;
```

```
struct sgp4_data {
    unsigned int sgp4_flags;
    unsigned int pad;
    double sgp4_AODP;
    double sgp4_AYCOF;
```

```

double sgp4_C1;
double sgp4_C4;
double sgp4_C5;
double sgp4_COSIO;
double sgp4_D2;
double sgp4_D3;
double sgp4_D4;
double sgp4_DELMO;
double sgp4_ETA;
double sgp4_OMGCOF;
double sgp4_OMGDOT;
double sgp4_SINIO;
double sgp4_SINMO;
double sgp4_T2COF;
double sgp4_T3COF;
double sgp4_T4COF;
double sgp4_T5COF;
double sgp4_X1MTH2;
double sgp4_X3THM1;
double sgp4_X7THM1;
double sgp4_XLCOF;
double sgp4_XMCOF;
double sgp4_XMDOT;
double sgp4_XNODCF;
double sgp4_XNODOT;
double sgp4_XNODP;
};

```

4. Miscellaneous Structures

```

struct _site {
    float lat; // latitude, in degrees
    float lon; // longitued in deg
    float h; // height above the elliptic, in km
    float min_el; // degrees above the horizon for detection
    char name[80]; // name of site
};

typedef struct _site siteType;

struct picStruct {
    double JD; //Julian Date of event
    double local_time; //local_time of event, as a Unix time
    number
    double distance; //distance, site to sub satellite point, in
    km
    double npsat_lat;
    double npsat_lon;
    double range; // total range
    double azimuth; // in degrees
    double elevation; // in degrees
};

```

APPENDIX II

TLE FORMAT

Two Line Element (TLE) sets are ways of describing a satellites position in both space and time. These sets of data are standard ways of communication this information in a concise way, if not exactly easy to read and translate. The TLE format is show below.

Data for each satellite consist of three lines in the following format:

XXXXXXXXXXXX

```
1 AAAAAU 00 0 0 BBBB.BBBBBBBB +.CCCCCCCC +DDDDD-D +EEEE-E F GGGZ
2 AAAAA HHH.HHHH III.IIII JJJJJJJ KKK.KKKK LLL.LLLL MM.MMMMMMMMMNNNNNZ
```

X = Satellite name
A = Catalog number
B = Epoch time
C = One half of first derivative of mean motion (decay rate)
D = One sixth of second derivative of mean motion
E = BSTAR drag coefficient
F = Ephemeris type
G = Number of the element set
H = Inclination
I = RAAN
J = Eccentricity
K = Argument of perigee
L = Mean anomaly
M = Mean motion
N = Orbit number
Z = Check sum (modulo 10)

Line 0 is an eleven-character name. Lines 1 and 2 are the standard two-line orbital element set format identical to that used by NASA and NORAD. The format description is as follows:

Line 0:

Column	Description
1-11	Satellite name

Line 1:

1- 1	Line number of element set
3- 7	Satellite number
8- 8	Classification (U = unclassified)
10-11	International designator (last two digits of launch year)
12-14	International designator (launch number of the year)
15-17	International designator (piece of launch)
19-20	Epoch year (last two digits of year)
21-32	Epoch (Julian day and fractional portion of the day)
34-43	One half of first time derivative of mean motion (decay rate)
	or ballistic coefficient (depending on ephemeris type)
45-52	One sixth of second time derivative of mean motion (decimal point assumed; blank if n/a)
54-61	BSTAR drag coefficient if SGP4/SGP8 general perturbation theory used; otherwise, radiation pressure coefficient (decimal point assumed)
63-63	Ephemeris type
66-68	Element set number
69-69	Check sum (modulo 10) (letters, blanks, periods, plus sign = 0; minus sign = 1)

Line 2:

1- 1	Line number of element set
3- 7	Satellite number
9-16	Inclination [deg]
18-25	Right Ascension of the ascending node [deg]
27-33	Eccentricity (decimal point assumed)
35-42	Argument of perigee [deg]
44-51	Mean anomaly [deg]
53-63	Mean motion [rev/d]
64-68	Orbit (revolution number) at epoch [rev]
69-69	Check sum (modulo 10)

All other columns are blank or fixed.

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
Ft. Belvoir, VA
2. Dudley Knox Library
Naval Postgraduate School
Monterey, CA 93943
3. Chairman, Space Systems Academic Group
Code SP/Pz
Naval Postgraduate School
Monterey, CA 93943
4. Chairman, Mechanical and Astronautical Engineering
Code ME/Hy
Naval Postgraduate School
Monterey, CA 93943
5. Jim A. Horning
Code SP/Jh
Naval Postgraduate School
Monterey, CA 93943
6. I. Michael Ross
Code MERO
Naval Postgraduate School
Monterey, CA 93943