A COMMAND AND DATA HANDLING UNIT FOR PICO-SATELLITE MISSIONS

D. Schor, J. Scowcroft, C. Nichols, and W. Kinsner

Department of Electrical and Computer Engineering
University of Manitoba, Winnipeg, Manitoba, Canada, R3T 5V6
email: {dschor|kinsner}@ee.umanitoba.ca, {janescowcroft|christopherjnichols}@gmail.com

ABSTRACT

This paper outlines the design and implementation of Phase Two of the Command and Data Handling (CDH) unit for the WinCube project. WinCube is a multi-year, multi-disciplinary project conducted at the University of Manitoba in which students from different departments participate in the design of a CubeSat. The purpose of the WinCube project is to provide an opportunity for engineering students to design and implement a space-ready satellite. The project also acts as a vessel for a high school science experiment, which will run an experiment while in space, and relay results down to a satellite base station. The CDH unit must manage the operations and internal communications of the various units inside the satellite. This unit includes a power management application, serial communications protocols to relay information, and a trustworthy timing system. The power management application must monitor the state of the battery, the output of the solar panels, as well as control the amount of power provided to each of the primary units. Serial communications are used to interact with other devices on the satellite. Finally, an external real time clock and watchdog timer were used to enhance the reliability of the system. Special measures were taken to protect the CDH unit from the harsh operating environment and to monitor the health of the system beyond temperature and battery readings by performing periodic tests of components.

Index Terms— Pico-satellite, CubeSat, command and data handling, complex system design process

1. INTRODUCTION

Satellites are orbiting spacecrafts used for scientific, communications and military purposes. Aside from the payload carried in the satellite, the architectural structure of most satellites consists of: a *Power Unit* including batteries and solar cells; *Communication Unit* to transmit and receive information with a ground station through an antenna; an *Attitude Control Unit* to align the satellite in space; and *Command and Data Handling (CDH) Unit* to manage data flow and the operations of the satellite.

Despite having similar architectures, satellites are classified by mass as either *small* (less than 500 kg), *medium* (500-1000 kg), or *large* (more than 1000 kg). Furthermore, small satellites are divided into five subcategories: mini (100-500 kg), micro (10-100 kg), nano (1-10 kg), pico (0.1-1 kg) and femto (less than 0.1 kg) [1].

1.1. CubeSat

Originally proposed by Robert Twiggs from Standford University and later developed by the California Polytechnique State University, CubeSat is a class of low-cost pico-satellites normally developed by university students to perform experiments in space [2, 3]. These satellites must be 10 cm by 10 cm by 10 cm with a maximum mass of 1 kg so that they could be deployed using a Poly Pico Orbital Deployer (P-POD) as described in [3].

1.2. WinCube - The University of Manitoba's CubeSat

WinCube is a multi-year, multi-disciplinary project conducted at the University of Manitoba in which students from different departments participate in the design of a CubeSat. The WinCube activities also include other projects such as a high-altitude balloon launching (BCube) by high school students and Near-Space Lab experiments at the University of Manitoba. One of the major elements of the WinCube project is the design and implementation of a CubeSat by various undergraduate and graduate students at the University of Manitoba, with the support of private industry. The design teams include students from several departments, including Electrical and Computer Engineering, Mechanical and Manufacturing Engineering, and School of Business. Some elements of the payload are being developed by high-school students. This is a multi-year project which began in 2006 and is expected to be completed by 2012. The goals of the project are to provide students with research opportunities where they can apply math, science and engineering skills to develop such a small satellite. The project is complex, and requires much coordination effort.

This paper outlines the work done by the CDH team of the WinCube project during the 2007/08 school year as part of the final year undergraduate capstone design project [4]. Although the research component of this paper is limited, its

978-1-4244-3508-1/09/$25.00 ©2009 IEEE

originality is relatively high as the CDH unit has been designed and evaluated from the ground up. The entire satellite must fit in a standard frame, but its components differ from one CubeSat to another. Since all the other units of the UofM CubeSat are also original, their integration is a formidable undertaking. Since the entire WinCube requires thousands of pages of documentation, this paper concentrates on the CDH unit only, in relation to the other units.

As of March 2009, there are two critical aspects of the mission that have not been finalized: the payload and orbit. A thorough description of the payload physical requirements and interface parameters was described in [5] and thus allowing for the payload to be designed in parallel to the rest of the system. The orbit will be established based on the available launch opportunities as the satellite approaches the assembly stage. Section 6 describes the measures taken in the design to account for the different levels of radiation expected for a worst case scenario.

2. ARCHITECTURE

2.1. Hardware Architecture

The CDH unit hardware consisted of a flight module with a built in micro-controller and three additional peripherals for a watchdog timer (WDT), a real-time clock (RTC), and secondary storage in the form of an SD Card. Figure 1 shows a block diagram of the hardware architecture for the satellite.

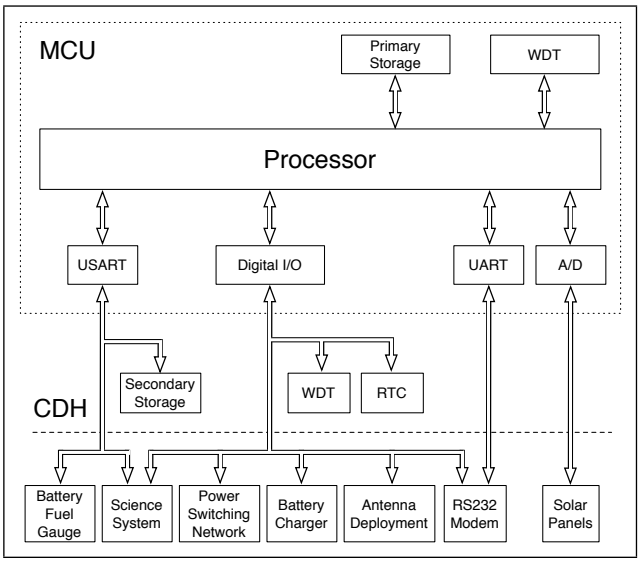

Fig. 1. Block diagram of WinCube hardware architecture.

The FM430 flight module was selected as it met all the interfacing and power requirements for the satellite such as input/output (IO) ports to support the other units on the satellite. This off-the-shelf device had worked successfully in other projects [6, 7, 8]. The core of the flight module consisted of the MSP430 - an ultra low power, 16 bit, reduced instruction-set computer (RISC) micro-controller with 2 KB of RAM and 60 KB flash memory [9]. In addition, two configurable universal synchronous asynchronous receiver transmitter (USART) units, two configurable 8 bit interrupt ports, and 12 and 10 bit analog-to-digital converters were also built in to the processor.

The design of a satellite requires both a reliable WDT and a timekeeping device. The WDT is intended to prolong the life of the mission by restarting the satellite in case of a failure, and the RTC is used to synchronize communications and log activities with respect to time. In the initial design, the WDT and RTC were to rely either on the built in capabilities of the FM430 or they would be added using a single off-the-shelf integrated-circuit (such as the bq4845) as suggested in [6] and [10]. However, both of those options implied that the WDT would have operated based on an oscillating crystal that would be susceptible to the wide temperature variations experienced in space.

The MAX6814 and DS12887 were selected as the WDT and RTC respectively. Although no documentation was found about the performance of the RTC selected, the WDT was successfully used in at least one project [11]. The MAX6814 was a low power integrated chip that did not use an oscillating crystal for operation. The DS12887 had a built-in oscillating crystal and back-up battery that ensured operation during battery charging orbits. The RTC used parallel loading through Port 5 on the MSP430. The RTC drifted by approximately 1 second per day under normal operating temperatures, and up to 13 seconds per day for extreme temperatures. In order to protect the clock from the low earth orbit temperature variations, it was determined to that the clock should be insulated with a NASA approved material for space such as Silver Teflon tape and that it would be placed near the center of the cube [12].

2.2. Software Architecture

The CDH unit needed to communicate with all external devices, control their operations and manage data flows. A cooperative real-time operating system (RTOS) was selected to accommodate the simultaneous operation of the other units and their different operating speeds. To enforce modularity and encapsulation in the design, a three layer architecture was selected (see Fig. 2). This allowed for the design of reusable drivers, independent applications, and the use of an off-the-shelf operating system. A similar architecture were successfully implemented and described in [13].

3. DRIVERS

3.1. Software Status Register

In order to keep track of non-volatile information about the mission such as whether or not the antenna was deployed, a

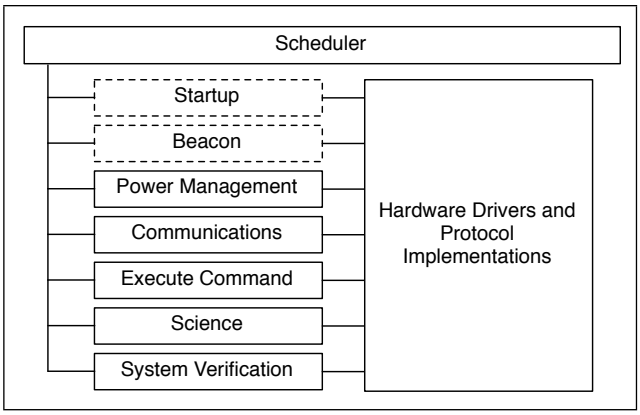

Fig. 2. Block diagram of WinCube software architecture.

software status register was implemented. This stored flags about the state of the system in various places in non-volatile memory and used a majority function to determine the values thus being able to recover from single bit errors.

3.2. Watchdog Timer

The WDT configured both the built-in timer in the FM430 and the external MAX6814. The use of two timers remained transparent to the operation of the CDH unit, however, this added an extra layer of protection.

3.3. Real Time Clock

The RTC software ensured both read/write locks to prevent corrupting the time or reading the clock as it was updating the time. In addition, the RTC served as an interface to schedule alarms that could be used to put the satellite to sleep during battery charging orbits.

3.4. Power Switching Network

The FDR8521L p-channel MOSFET with Gate Driver for Load Switch Application was selected as a physical switching network in the design of the Power Unit for the satellite [14]. However, the actual driver did not have interactions with this switcher phase. Instead, the driver simply controlled four pins to assign power to different devices.

3.5. Battery Fuel Gauge

The battery fuel gauge driver used the SMBus to communicate with the Texas Instrument's bqQ2060A Gas Gauge used in the satellite [14]. This driver provided read access to ten different functions describing the status of the battery such as remaining capacity, temperature, and cycle count. These readings were either used for telemetry logging purposes or to evaluate the power state of the satellite to determine whether to enter a low power mode and shut off selected devices.

3.6. Solar Panels

The solar panel drivers were partially implemented because the actual devices were not confirmed. However, a partial driver was written to check the voltage levels of the panels through analog-to-digital converters. This telemetry would serve to verify the operation of the solar panels from the ground.

3.7. Battery Charger

The battery charger used in the satellite was the Texas Instruments' bq24005 [14]. This device provided two status pins that determined whether the battery was in precharge, fast charge, fault, or done charging. An interrupt function was triggered in the event of a fault to immediately alert the CDH unit of a battery malfunction. In addition, the signal for done charging was used in conjunction with the battery fuel gauge to verify the operation of the sensors.

3.8. Built-in Flash Memory

Although not completely integrated in the design, a built-in flash driver was designed to provide read/write access to the flash memory. This provided the ability to expand the design in the future to allow software reconfiguration from a ground station as implemented in similar missions [13].

3.9. Log Recorder

The log recorder used a 512 byte buffer in built-in flash memory to minimize transfers to the external memory device (SD card). The driver received an error code (see Sec. 6) as input, attached a 6 byte timestamp from the RTC, and stored the information in memory.

3.10. Inter-Integrated Circuit Bus

The I^2C driver was designed to communicate with the Science Unit and the battery fuel gauge. This driver used the Sub-Main Clock built into the flight module in order to be able to manipulate the speeds by dividing the clock and stretching the high/low durations. This allowed for slower transfer rates in the event that lots of errors were detected in the transmission.

The driver implemented an interrupt to listen to requests on the line and also handle possible errors in transmissions such as arbitration lost or no-acknoledge received. DMA was considered as an option in the design, but was not used due to the small transfers (less than 64 bytes) expected. In addition, this provided with instant feedback in the event of failures as opposed to having to check whether the transfer finished successfully.

This driver contained two subdrivers that implemented special protocols for communications with selected devices.

3.10.1. System Management Bus Protocol

The SMBus protocol was added on top of the I^2C driver described in Sec. 3.10 to communicate with the battery fuel gauge. This utilized the functions already implemented and added packet error checking (PEC) in the form of a CRC-8 appended to each data transfer.

3.10.2. Inter-Integrated Circuit Bus Memory

I^2C was used to communicate with an EEPROM in the Science Unit. Since the payload was not available, testing was performed with the Microchip 24LC256 EEPROM that held 256 KB in blocks of 64 bytes. The MSP430 provided support for this protocol in USART0 only.

3.11. Serial Peripheral Interface Bus

The SPI bus was added to provide read/write access to the FM430's built-in SD card that acted as an external memory for the satellite. It was implemented using USART0 in the MSP430 because it is wired to the SD card slot on the FM430. However, I^2C was also implemented with the same port, thus a special state machine was designed to balance both protocols on the same port. Figure 3 shows the state machine used to operate both communication protocols over the same port. In this design, the unit remained in I^2C slave receive mode for the majority of the time until an alarm was received from the batteries, then it entered a slave received active mode. If the processor required information from the battery fuel gauge or the science unit, it enter master transmit mode to send the request to the corresponding device and then entered a master receive mode to wait for the data requested. In the event that a write to the SD card was required, the unit would enter an SPI master mode and return to an idle slave receive I^2C mode after the completion of the write routine.

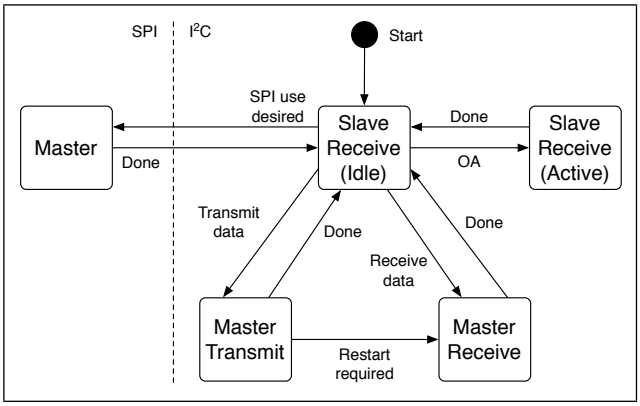

Fig. 3. State diagram of shared USART0.

3.11.1. Secure Digital Card Protocol

Read/write access to an SD Card was built on top of the SPI driver. This was used to log telemetry data to be sent to the ground station at a later time.

3.12. Antenna Deployment

This driver controlled the one-time deployment of the nylon wire under tension that served as the omni-directional antenna for the satellite. The driver applied a current to a small metal filament for a short period of time thus burning the filament and releasing the antenna. This is a similar deployment mechanism as that of the KySat1 mission even though in that mission tested other elements besides metal filaments [6].

3.13. Modem

To date, the details on the modem and communication units on the satellite have not been finalized. Consequently, development of those drivers was limited to conceptual ideas only. In particular, the recommendation was to develop downlink and uplink message queues. Thus, upon connecting with a ground station, commands received are stored in the uplink queue and telemetry sent would be taken from a downlink queue. The downlink queue would act as a buffer of information to be sent to the ground, while the uplink queue would store series of commands to be executed.

4. APPLICATIONS

The selected applications are constantly running on the satellite to control the operations of devices through their respective drivers. With the exception of the startup routine, all applications are designed to run constantly until the system is shutdown or restarted.

Only the Startup and Power Management applications were implemented during Phase Two of the project, however high level description of the remaining applications were developed to provide a holistic view of the satellite.

4.1. Startup

The startup application was responsible for initializing devices upon any startup condition. This application made use of the software status register described in Sec. 3.1 to determine whether it was a Restart or Deployment scenario. In the case of a restart, the devices would be initialized normally, and in the one-time deployment, an antenna would be deployed following a series of timing requirements. One alternative to this design was to launch the satellite with the minimum software required for deployment and then reconfiguring the software from the ground station. This approach described in [15] requires further research to determine the benefits and risks involved.

4.2. Power Management

The power management application established four threshold values that determined what components would be operating based on the percentage of battery power remaining. These thresholds were estimates based on the expected power consumption. These parameters would be adjusted to ideal values upon completing the implementation, and again after launch to adapt to the actual power consumption rates.

4.3. Execute Command

This application removed and executed command from the uplink priority queue. This approach relies on the scheduler to determine when the satellite is free to execute the next highest priority tasks as opposed to scheduling commands using a timestamp like in [6]. A list of predefined commands was designed that included setting the date and time of the satellite, resetting the satellite from the ground, requests for telemetry and possible expansion to reprogram satellite either from the ground station of from additional copies of the software stored in the SD Card.

4.4. System Verification Application

This application ran periodically to check the status of various devices through either built-in tests (such as the case with the DS12887 RTC) or through customized tests such as testing memories by writing and reading checkerboard patterns. The results from the tests were logged to be sent to the ground station. This idea expands the health monitoring techniques used in [6, 7, 8, 16] where only battery, attitude, temperature, and software crash information was relayed back ground.

4.5. Communications

The communication application was active whenever the beacon signal was inactive. Its purpose was to remove messages from the downlink message queue, and transmit them to the ground station.

4.6. Beacon Signal

This application transmitted a periodic identification signal through the communication drivers to the base station. Upon receipt of an acknowledge from the ground, the application initiated the Communication application and put itself to sleep. Other CubeSats, such as [6] and [13], use a beacon signal to transmit telemetry about the satellite on periodic intervals. In WinCube, all communications are handled in the Communications application and the Beacon is simply used until a communication link is established.

5. OPERATING SYSTEM

An operating system was used to manage the applications running on the satellite. Salvo OS was selected because it was bundled with the selected FM430 flight module used, thus facilitating the integration as outlined in [6]. Other options were considered such as running Linux like in the QuakeSat mission [16], however the integration would have delayed the mission progress. Salvo provided with a multitasking RTOS and message passing between applications. Most importantly, the RTOS used RAM and not the stack to keep track of applications. Estimates indicated that between 47 bytes and 110 bytes (2.3 to 5.4%) of RAM would be used for the satellite.

Each application received one of three priorities where the payload, power management and startup received the highest priority, followed by Communication and Beacon applications. Context switching was controlled by the application currently running on the CDH unit. Instead of using locks during critical tasks, an application would call a function whenever it had completed a critical application and was ready to allow other routines to execute.

6. SOFTWARE SYSTEM PROTECTION

The low earth orbit is a harsh operating environment for the satellite where high energy particles can cause single bit errors that could upset the operation of the satellite. Expected levels of radiation are described extensively in [17]. Four simulataneous approaches were taken into account in the design of the CDH unit to account for this phenomenon: transmission integrity, storage integrity, data redundancy and error reporting.

Cyclic Redundancy Codes (CRC) were added to transmissions over the satellite buses to detect errors. In addition, both the SPI and I^2C protocols allowed for customizable clock speeds generated by dividing SMCLK to adjust to the environment. Thus, if multiple errors were detected on the bus, the clock would be slowed down before attempting to re-transmit the data.

Memory devices were selected such that they would have built-in Error Correcting Codes (ECC) support such is the case of the payload memory and the SD Card. Also, to account for the possibility of a large block of memory being corrupted, multiple copies of selected components were stored. This level of data redundancy was only implemented for the Software Status Register and additional information could also be added later upon completing the actual memory map for the system.

System-wide Error Codes were designed in order to log both errors and telemetry. Two modes of error codes were used to store single errors and errors with telemetry. The 2 byte single error codes were made up of a single bit type identifier, a 7 bit module number, and a 1 byte error code. Extended errors had a variable length of 3-250 bytes. They

consisted of a single bit type identifier, a 7 bit module number, 1 byte describing the number of data bytes stored, and up to 248 bytes of data.

7. LIMITATIONS

The major limitations of the satellite was the lack of hardware redundancy due to the very small space available. The size and mass limitations prevented the addition of multiple batteries, clocks, storage devices and processors. As such, if any of the components fail, the mission could fail.

Although precautions were taken to account for possible software errors, additional software redundancy would be desired to be able to recover in the case that a CRC identifies a corrupted memory block. Also as previously suggested, the option to reprogram the satellite remotely was not fully implemented. Adding such feature would provide an additional means of recovering from a hardware failure by adapting the operation of the satellite to utilize the available resources.

8. FUTURE WORK

There are many issues that need to be addressed to complete the CDH unit. In particular, these include the development of communication drivers to work with the modem selected and the development of the remaining software applications that utilize said drivers. Finally, the integration of all these new components would need to undergo verification and testing to ensure it is working as intended.

9. CONCLUDING REMARKS

This paper summarized the design and implementation of the CDH unit for the WinCube mission. Phase Two successfully designed the drivers and setup the architecture for the CDH unit paving the road for the development of the applications during Phase Three of the project.

Acknowledgments

Special thanks go to Ron Britton, Associate Dean, Faculty of Engineering, University of Manitoba for his leadership and facilitation of the WinCube project. Thanks to Robert McLeod, Professor, Faculty of Engineering, University of Manitoba for lending us development equipment required to complete the project. Special recognition goes out to the participants from Magellan Bristol Aerospace who took time out of their busy schedule to challenge us to meet industry standards, and to evaluate the project. Thanks also goes to the rest of the WinCube team (both from Phase One and Two) for collaborating on a number of different design options.

References

[1] "TBS Internet: Satellite Encyclopedia Glossary," Available as of December 20, 2008 from http://www.tbs-satellite.com/tse/online/thema_glossary.html.

[2] B. Horais, R. Twiggs, and C. Byvik, "Innovative Methods for Placing Small Payloads in Space," in *Proc. IEEE Aerospace Conference, AERO2000 (Big Sky, MT; March 18-25, 2000)*, vol. 7, pp. 167 – 175.

[3] J. Puig-Suari, C. Turner, and W. Ahlgren, "Development of the Standard CubeSat Deployer and a CubeSat Class PicoSatellite," in *Proc. IEEE Aerospace Conference, AERO2001 (Big Sky, MT; March 10-17, 2001)*, vol. 1, pp. 347 – 353.

[4] C. Nichols, D. Schor, and J. P. Scowcroft, "A Command and Data Handling System for the WinCube Pico-Satellite Mission," April 2008, Undergraduate Capstone Design Project, Dept. of Electrical and Computer Eng., University of Manitoba.

[5] WinCube Team, "High School Science Specifications V2.0," April 2008, Dept. of Electrical and Computer Eng., University of Manitoba.

[6] G.D. Chandler, D.T. McClure, S.F. Hishmeh, J.E. Lumpp, J.B. Carter, B.K. Malphrus, D.M. Erb, W.C. Hutchison, G.R. Strickler, J.W. Cutler, and R.J. Twiggs, "Development of an Off-the-Shelf Bus for Small Satellites," in *Proc. IEEE Aerospace Conference, AERO2007 (Big Sky, MT; March 3-10, 2007)*, vol. 1, pp. 1 – 16.

[7] "Delfi-C^3," Delft University of Technology in the Netherlands. Available as of March 11, 2009 from http://www.delfic3.nl.

[8] "Primer Satelite Colombiano en Orbita (First Colombian Satellite In Orbit)," Universidad Sergio Arboleda. Available as of March 11, 2009 from http://www.usergioarboleda.edu.co/proyecto_espacial/.

[9] Pumpkin, Inc. (2007, Feb.). *"CubeSat Kit FM430 Flight Module"* [Online]. Available: http://www.cubesatkit.com/docs/datasheet/DS_CSK_FM430_710-00252-C.pdf [October 2, 2007].

[10] Win-Cube - Year One. *"Win-Cube Project: Preliminary Design Review,"* University of Manitoba, Winnipeg, MB, 2007.

[11] "CAPE1," University of Louisiana. Available as of March 11, 2009 from http://cape.louisiana.edu.

[12] H. Kozak, "WinCube: Thermal Control System Design," April 2008, Undergraduate Capstone Design Project, Dept. of Mechanical Eng., University of Manitoba.

[13] "University of Illinois - ION CubeSat Project," University of Illinois. Available as of March 11, 2009 from http://cubesat.ece.uiuc.edu.

[14] R. Chan, R. Benerjee, and A. Jani, "WinCube Project: Electrical Power System Phase Two Critical Design Review," April 2008, Undergraduate Capstone Design Project, Dept. of Electrical and Computer Eng., University of Manitoba.

[15] G. Smith (WA4SXM), "AMSAT O-51: Development, Operation and Specifications," Tech. Rep., 2004.

[16] T. Bleier, P. Clarke, J. Cutler, L. DeMartini, C. Dunson, S. Flagg, A. Lorenz, and E. Tapio, "QuakeSat Lessons Learned," Available as of March 11, 2009 from http://www.quakefinder.com/services/quakesat-ssite/.

[17] "Preferred Reliability Practices: Space Radiation Effects on Electronic Components in Low-Earth Orbit," NASA. Practice No. PD-ED-1258, April 2006.