Design of On Board Computers for a Nanosatellite

A Thesis Submitted

in Partial Fulfillment of the Requirements

for the Degree of

Master of Technology

by

Ankush Pankaj Desai

Y8111008

to the

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY KANPUR

May, 2010

CERTIFICATE

This is to certify that the work contained in the thesis entitled "*Design of On Board Computers for Nanosatellite*" by *Ankush Pankaj Desai* has been carried out under my supervision and that this work has not been submitted elsewhere for a degree.

Arnab Bhattacharya

May, 2010

Dr. Arnab Bhattacharya
Department of Computer Science and Engineering
Indian Institute of Technology, Kanpur
Kanpur 208016.

Contents

List of Figures

List of Tables

Acknowledgements

I offer my sincerest gratitude to my guide Dr. Arnab Bhattacharya who has always given me opportunity to learn more and more, supported me throughout my thesis with constant encouragement, being patient.

I am deeply indebted to Dr. N. S. Vyas for his stimulating support. I further extend my gratitude to Dr. S. G. Dhande for his motivational and encouraging words. I would like to extend my sincere thanks to my colleagues, Shantanu Agrawal, Ankesh Garg, Amritsagar and my juniors, Shashank Chintalagiri, Vijay S., Arpit Mathur and to all the other members of JUGNU Team for their support, to all my friends for their inspiring words and for making my stay at IIT, Kanpur a memorable one.

I would also like to thank the faculty members of Computer Science and Engineering Department for imparting me with invaluable knowledge and adding a lot of value-oriented growth to my career.

Above all, I am blessed with such caring parents who have given me full freedom to do whatever I wish and have always supported me in my decisions. I extend my deepest gratitude to my parents, my sister for their invaluable love, affection, encouragement and support. Ankush Pankaj Desai

Abstract

On-board computers are key elements in most embedded applications. In space systems, the requirements from on-board computers become much more stringent. These added requirements are largely due to the need for reliability, robustness, and autonomous survival. Of the different important components of a space program, nanosatellites are fast becoming a useful platform for the design and development of such high-reliability on-board computer systems. Nanosatellites are ultra low cost test beds for testing newer technology in space; this ultra low cost is achieved by using commercial-off-the-shelf devices and designing reliable software with redundancy and fault tolerance. This thesis describes the design and development of the two on-board computers used in the nanosatellite JUGNU developed at the Indian Institute of Technology, Kanpur. An overview of JUGNU is first presented, along with the requirements of the on-board computers. The design of the on-board computer software is then discussed. The communication protocol for high-reliability operation of the command and data handling functions of the on-board computers in the presence of error prone communication links is next described in detail. Finally, the various schemes put in place to confer fault tolerance to the system, which enable its recovery from radiation induced errors or power brownouts, are described.

Chapter 1

Introduction

In the recent past, nanosatellites have opened new avenues of technology. This has given impetus to the growth of satellite technology in many countries. Academic institutions along with the corporate world are involved in collaborative research and development work. Most of the nanosatellite projects are born in universities. The main benefit of these low cost nanosatellites is that they serve as a test bed for latest miniaturized technologies.

Typical nano/micro satellite applications include terrain mapping, weather tracking, disaster management, low gravity experiments, astronomical studies, global positioning and geological studies. They are also very useful in scientific research. Further, they are extensively used in diverse experiments related to satellite communication, atmospheric studies, low gravity behaviour, earth's magnetism etc. Various missions are already under progress equipped with spectrometers, telescopes, magnetometers, and a variety of other instrumentation designed by several universities and companies.

JUGNU is an attempt by Indian Institute of Technology, Kanpur at implement-

ing miniaturized space technologies and creating opportunities for learning in the academic context. The Indian Space Research Organization (ISRO) has helped the team in meeting with the necessary requirements of the project. JUGNU should serve as a test bed for the next generation semiconductor and other micro- and nano-technologies for space. It would also serve as a platform for exploring possibilities of cost-effective space missions. JUGNU is proposed to be placed in low-earth polar orbit at an altitude of about 700 km by ISRO's Polar Satellite Launch Vehicle (PSLV).

Nanosatellite or nanosat name is applied to satellites whose mass lies in between 1kg - 10kg. It has nothing to do with the term nanotechnology. It has almost all the features of a small satellite but components used in nanosats are in miniaturized form.

1.1 Mission Objectives Of JUGNU Nanosatellite

The Mission objectives for the JUGNU nanosatellite are:

1. Demonstrate the viability of the nanosatellite platform for future experiments
2. Design and develop a near IR imaging system and test its compression algorithm
3. Develop and evaluate a small form factor GPS receiver for satellite navigation
4. Augment, test, and qualify an indigenously developed MEMS (Micro Electromechanical Systems) based Inertial Measurement Unit (IMU) for space applications.

5. Transmit a beacon signal on the amateur band that can be received from anywhere in the world.

1.2 JUGNU: As a Satellite

The high-level design of JUGNU was based on the objectives initially proposed. JUGNU comprises of nine subsystems - six making up the nanosatellite bus and three payloads (GPS, IR-Imaging and IMU). Additionally, another system, Ejection Mechanism, is being designed for the separation of the satellite from the launch vehicle. These sub-systems work in coordination with each other. The brief description about these subsystems is given below:

Attitude Determination and Control System (ADCS): Attitude Determination and Control System deals with the position and orientation of the satellite in space, which is required for maintaining stability and maneuvering for imaging and communications.

On Board Data Handling Computer: On board computer will be the Command and data handling part of the satellite. It also schedules and controls the payload operations on the satellite. It acts as the heart and brain of the satellite for its survival in space environment.

Communication system: Communication system consist of the transmitter, receiver and beacon on the satellite. For receiving the telemetry data successfully on ground station and for sending telecommands to the satellite to control its operation in space the communication system of the satellite should be extremely reliable.

Power System: Power system consists of solar arrays or solar panels joined to a power management system which takes care of power regulation and transmission to all parts of satellite.

Thermal Control: We are using passive temperature control on the satellite, to maintain the temperature of the satellite within specified temperature limits 273K to 313K.

Imaging System - Payload: Perform space based observations of vegetation, by virtue of its high reflectance in the Near-IR region. This band also helps us to differentiate between crops and water bodies.

GPS Receiver - Payload: Custom made algorithm to be programmed into the ARM7 based processor which allows for corrections and improvements to be made according to our requirements. This will also help in testing of newly developed correction algorithms on commercially available GPS receivers.

IMU - Payload: An inertial measurement unit uses accelerometers and gyroscopes to measure the inertial components of a system resulting from motion or vibration. The data can be used to calculate the position and orientation of the vehicle at any later moment and also to study the vibration characteristics of the system.

Ejection System: Ejection System is the most crucial system in the sense it is the interface between the satellite and the rocket. It will eject the satellite in space on getting signal from the rocket.

The Ground Station for control and monitoring of the satellite has been set up at Indian Institute Technology, Kanpur. The telemetry data (for health monitoring)

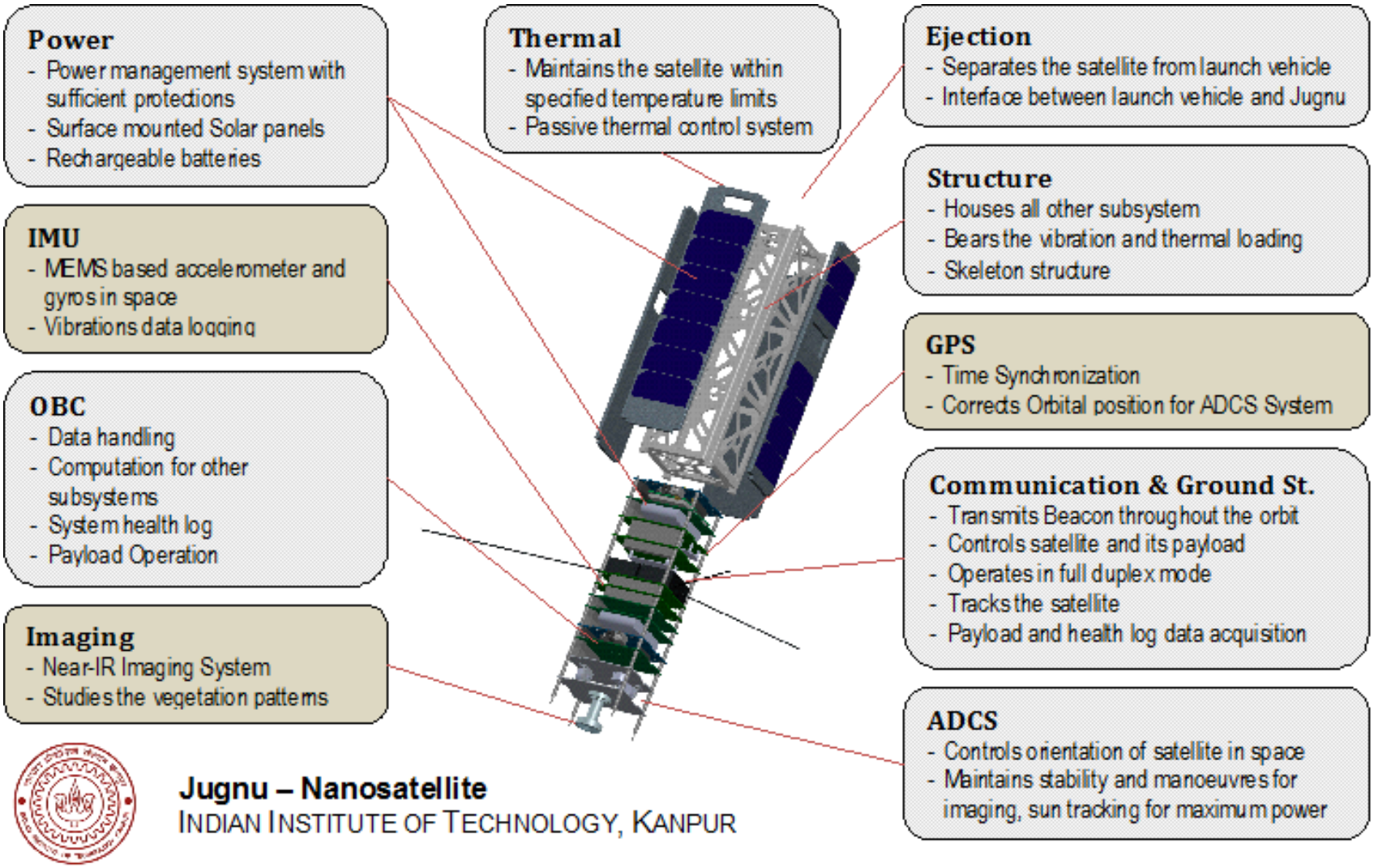

Figure 1.1: *Subsystems of JUGNU nanosatellite*

collected for all the subsystems at ground station will validate the performance of the sensors and the miniaturized technology in the satellite subsystems. Experiments related to payloads: GPS, Imaging and IMU, will demonstrate the performance of the hardware and algorithms used in their design. Images collected will also be useful in studying the vegetation and the water bodies. Other image related applications are also being envisaged.

The satellite design is mostly indigenous. For the most part, non-space grade, commercial-off-the-shelf (COTS) components are used to keep the cost low. In a nano-satellite, there are very stringent specifications for the weight and size of the individual subsystems. Hence, the main technical challenge in nano-satellite development is to design the subsystems fitting the specified dimensions without compromising on their performance.

The design of JUGNU is compatible with the CubeSat standards proposed by California Polytechnic State University, San Luis Obispo [1] and Stanford University's Space Systems Development Lab [2]. This allows the use of COTS (Commercial-off-the-shelf) components with space heritage, and ensures continued compatibility of the hardware designed with international standards, for use in future missions. The satellite consists of various Printed Circuit Boards (PCBs). All the components are mounted on these PCBs in a manner that they are compatible with the 3U-Cubesat structure procured from Pumpkin Inc [3].

1.3 On Board Computers for JUGNU

JUGNU is capable of autonomous operation, which implies that it does not require intervention from the ground station for its basic survival. Based on the requirements

with respect to the operation of the payloads, the ground station can send commands to the satellite. In order to achieve this level of autonomy, on board computers has been programmed to operate under various pre-defined modes. Each mode of operation represents satellite's subsystems in a specific state.

In addition to the autonomous survival of the satellite, the software and hardware designs of on board computers allow considerable flexibility to the system configuration. This configuration can be controlled from the ground station. Several payload operations can be controlled from the ground station. Sub-systems can be selectively disabled and the processors can also be reprogrammed from the ground station. As was mentioned initially, commercial-off-the-shelf components are used on the satellite its essential that design of on board computers is robust and tolerant to single point failures.

1.4 Problem Description

In this thesis we tried to incorporate as many features on OBC, to make it efficient, reliable and tolerant. We used open source softwares like μCOS [4, 5] and EFSL filesystem [6] which are very widely used in embedded industry, on our indigenously develop board. Commercially available space proven software like Salvo Pro [3, 7] and EFFS-THIN [8] are also used. We also implemented software fault tolerance techniques like triple modular redundancy, check point based recovery for increasing the reliability of software of on board computers in space. Communication protocol used on JUGNU nanosatellite is suitable for Low Earth Orbits (LEO) and has salient features like variable packet length and variable baud rate. Downlink protocol is designed so as to provide high channel utilization.

Unlike conventional satellites which have a large number of redundancy for data handling and control system, on board computers of JUGNU has minimal redundancies at component level. Some redundancy is maintained at the functional level and efforts are made to achieve single point failure tolerant design using on board computer of the satellite.

1.5 Layout of the thesis

This thesis is organized as follows. In Chapter 2, we describe the software and hardware specifications of on board computers. In Chapter 3, we present the software requirements and design of an on board computer for nanosatellite. Chapter 4 describes the communication protocol designed for JUGNU, various features implemented for reliability and high throughput are described. Chapter 5 presents the software fault tolerance techniques implemented to increase the reliability of on board software. Chapter 6 concludes the thesis giving in a nutshell the contribution of this thesis.

Chapter 2

On Board Computers For JUGNU

Nanosatellites are typically examples of highly integrated and miniaturized embedded systems, and pack a wide variety of hardware and software into an extremely small package. The associated complexity in the design and implementation of such systems is well represented in Figure 2.1, which shows the high level schematic, or the functional block diagram of the JUGNU nanosatellite.

The on-board computers, specifically OBC-1 and OBC-2 in the case of JUGNU, perform functions in satellites similar to those of the central nervous system in humans. They interface with the various other subsystems and sensors to ensure the satellite's survival. In space parlance, the on-board computers are at the very core of what is called the 'Satellite Bus', which is essentially a minimal system which can be reused in multiple spacecraft with different payloads. The capability of selectively reconfiguring the on-board computers in subsequent spacecraft without adversely affecting the performance of the satellite bus is one of the most important factors defining the viability of a given satellite bus. With the use of current computing and embedded technology, this can be achieved in large spacecraft by the use of a

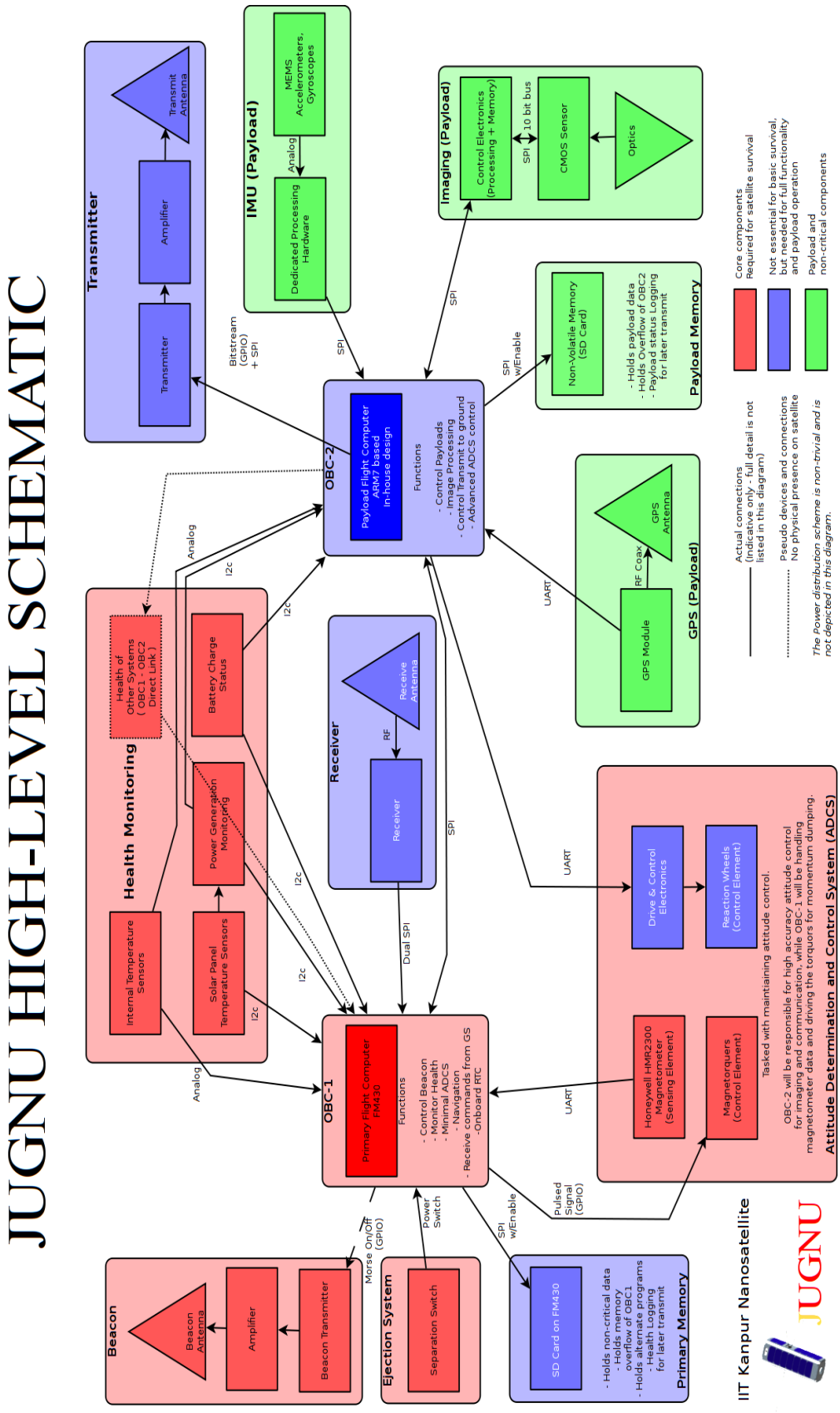

Figure 2.1: *Block diagram Of JUGNU nanosatellite*

number of adapters and translators, The ability to meet this requirement is greatly hindered by the highly integrated nature of nanosatellites, where power, space, and processor time is at a premium.

The capabilities of the on-board computers dictate the configuration of the rest of the satellite to a very large extent. It is common in embedded systems to use a variety of industry-standard communication protocols for interfacing with off-chip peripherals. In the case of a nanosatellite with a number of complex systems functioning in tandem, there exists a need to interface with the various systems via a protocol which they support. It is also necessary and to ensure correct data processing and flow in a real-time or near-real-time basis. Further, the computational requirements of various critical satellite bus algorithms, such as those for attitude determination and control are quite high. They require optimization and careful scheduling to ensure that they do not interfere with other real-time response requirements on the on-board computer.

The two On-Board Computers perform command and data handling functions, critical housekeeping and maintenance functions, and a vast amount of bus arbitration. Due to this, there exists a very real requirement for high reliability and fault tolerance in the software running on these computers. The use of commercial grade electronics requires the on-board computers to tolerate errors caused by SEUs (Single Event Upsets) and other forms of radiation damage, power brownout, and malfunctioning peripherals. In order to evaluate the risks and optimize the chances of satellite survival, Figure 2.1 was developed. The various blocks shown in the diagram represent the various subsystems of the satellite. Most of these blocks are themselves complex units comprising of electronics, software, and multiple sensing

and actuating elements. An intricate network of SPI, I2C, and UART connections are used to make the connections with the various subsystems.

In the figure, the blocks shown in red represent those which form the core of the nanosatellite bus developed for JUGNU. These systems are involved in essential housekeeping and maintenance functions, and must be able to recover from seemingly fatal error. OBC-1, the primary flight computer, is tasked with maintaining the satellite systems at acceptable power and temperature levels. It is based on a flight computer which has space heritage, and in the absence of more information is believed to have a lower chance of failure in the space environment.

OBC-2, or the 'Payload Flight Computer', is the workhorse of the nanosatellite bus. It provides the more advanced features of the JUGNU nanosatellite bus, such as 3-axis stabilization, ground and object tracking, higher bandwidth data handling and transmission, and in some cases reprogrammability of the payloads. OBC-2 and the other elements shown in blue are essential for proper operation of the payloads. Due to this, these elements are also treated with the same kind of reliability analysis as the red sections. However, slightly greater risk is taken in using previously unproven technology in space so as to make the nanosatellite bus viable despite the constraints on it.

The green elements in the diagram are the payloads. These are vastly more experimental in nature. The failure of any of the green blocks would not effect any other system in the satellite. It is taking into account the apparent reliability of the various sections that the on-board computer and its software is designed.

2.1 On Board Computer - 1

We are using MSP430 [9, 10] as our master on board computer, the flight module for OBC-1 is procured from `Pumpkin, Inc` [3, 11], its a Cubesat Kit flight module FM430 (REV C) with Pluggable Processor Module A3 (PPM A3) [12]. Till date, 5 CubeSat Kit FM430 flight modules have flown in to space and are functioning successfully [13]. FM430 is space-proven with high reliability, hence it is used as OBC-1 which has to perform all the critical health monitoring task on the satellite.

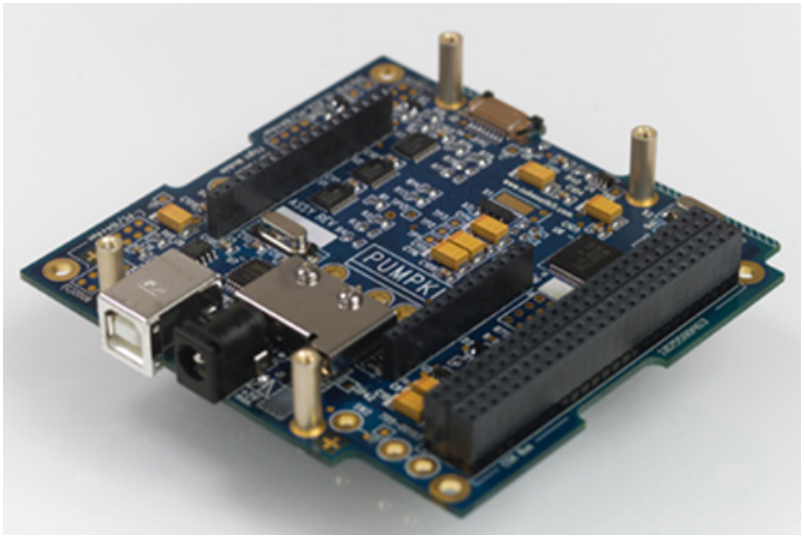

Figure 2.2: *CubeSat Kit flight module FM430 (REV C)*

2.1.1 Hardware Specifications

Table 2.1: *Hardware specification of OBC - 1*

Mother Board	CubeSat Kit FM430 (Rev B) [14]
Daughter Board	Pluggable Processor Module A3 (PPM A3) with TI's MSP430F2618 (Rev D) [12]

Table 2.1 continues on next page

Table 2.1 continued from previous page

Microcontroller	TI's MSP430F2618 (16 bit, RISC) [9]
Internal RAM	8 KB
Internal Program Flash	116 KB + 256 B
Processor Frequency	1 - 8 Mhz
External flash memory	Industrial grade SDCARD of 1 GB size
Internal Modules	Watchdog timer - 16 bit 2 - UART modules 4 - SPI modules 2 - I2C modules 1 - ADC(12-Bit) 1 - DAC(Dual 12-Bit) 2 - 16 bit Timer Modules DMA(3 channels internal DMA)
External RTC	M41T81S [15]

2.1.2 Software Specifications

SALVO PRO - RTOS

For real time operation and for scheduling the tasks on OBC-1, we planned to use a RTOS with minimal memory requirement and high compatibility with FM430. We obtained licensed version of SALVO PRO from `Pumpkin, Inc`[3, 7]. Salvo is the first Real-Time Operating System (RTOS) designed expressly for very-low-cost embedded systems with severely limited program and data memory. 4+ applications

built with Pumpkin's Salvo Pro RTOS have been launched into space [13].

Salvo is a purely event-driven cooperative multitasking RTOS. Some of the main features of Salvo PRO are minimal RAM requirement(small stack size), fast context switching, fast response to interrupts, etc. Since the tasks running on OBC-1 are mainly periodic tasks with soft deadlines other than the health monitoring task, cooperative round robin scheduler of Salvo PRO is best suited for OBC-1.

EFFS-THIN Filesystem

File System is a method for storing and organizing files and data to make it easy to find and access them. External SDCARD is connected to OBC-1 for logging health monitoring data, telecommand received from ground station, default parameters and constants which define the state of the satellite, etc. For maintaining all this critical data on the satellite we ported a minimal embedded filesystem on to the SDCARD. For the SDCARD connected to OBC-1 we used licensed version of EFFS-THIN (FAT16) filesystem provided by HCC - embedded [8].

THIN is a highly optimized, reduced footprint version of highly successful FAT system. This DOS compatible file system is designed for configurations with limited resources. It works well with the 8051, MSP430 and H8S series MCUs and is suitable for use with most 8-bit and 16-bit CPUs. On an MSP430 the minimum build of THIN including an MMC card driver uses less than 800bytes of RAM.

Time required to access SDCARD using filesystem or through direct MMC APIs is almost the same with minimal overhead, hence we used FAT16 filesystem for data handling which provides structured way for storing data. Also this feature can be used during the *testing phase* of the satellite, logged data and other checkpoint data written onto the SDCARD during testing can be viewed and analyzed directly by

connecting the SDCARD to a laptop or PC with MMC card reader.

Integrated Development Environment for MSP430

The Integrated Development Environment (IDE) chosen for the MSP430 is CrossWorks from Rowley Associates Limited [16]. It has a powerful debugger, with multiple breakpoints. CrossWorks for MSP430 sets the standard for Texas Instruments MSP430 development tools. It has an ANSI C compiler, macro assembler, linker/locator, libraries, core simulator, flash downloader, JTAG debugger. IDE is very user friendly. EFFS-THIN library available from pumpkin, Inc. is compatible with Rowley Crossworks.

2.2 On Board Computer - 2

The purpose of using OBC-2 is to have a high end processing device for complex computational tasks and to provide real time operations. The OBC-2(ARM) board is designed indigenously by JUGNU team to meet the requirements of satellite. ARM 7 based Atmels microcontroller AT91SAM7X was chosen for this purpose as it has been tested to run successfully in space.

2.2.1 Hardware Specifications

Table 2.2: *Hardware specification of OBC - 2*

Mother Board	Indigenously developed
Microcontroller	ATMEL's AT91SAM7X512 (32 bit, RISC) [17]

Table 2.2 continues on next page

Table 2.2 continued from previous page

Internal RAM	128 KBytes of internal SRAM
Internal Program Flash	512 KBytes of High Speed internal flash
Processor Frequency	Operating clock of 48MHz through PLL
External flash memory	Industrial grade SDCARD of 1 GB size
Internal Modules	Watchdog timer - 12 bit 1 - 3 channel 16 bit timer/counter 1 - 20 bit counter with 12 bit interval 2 - UART modules 2 - SPI modules 1 - TWI modules 1 - ADC(10-Bit)
External RTC	M41T81S [15]

2.2.2 Software Specifications

We have indigenously developed OBC - 2, hence we tried to make software part of OBC - 2 as indigenous as possible, we have used open source RTOS and Filesystem and have ported them on our platform.

μC/OS-2 - RTOS (open source)

μC/OS-II is currently maintained by Micrium Inc. [4] and can be licensed on a per product or product line basis. Use of the operating system is free for educational non-commercial use. μC/OS-II is a fully preemptive real time kernel, most commercial

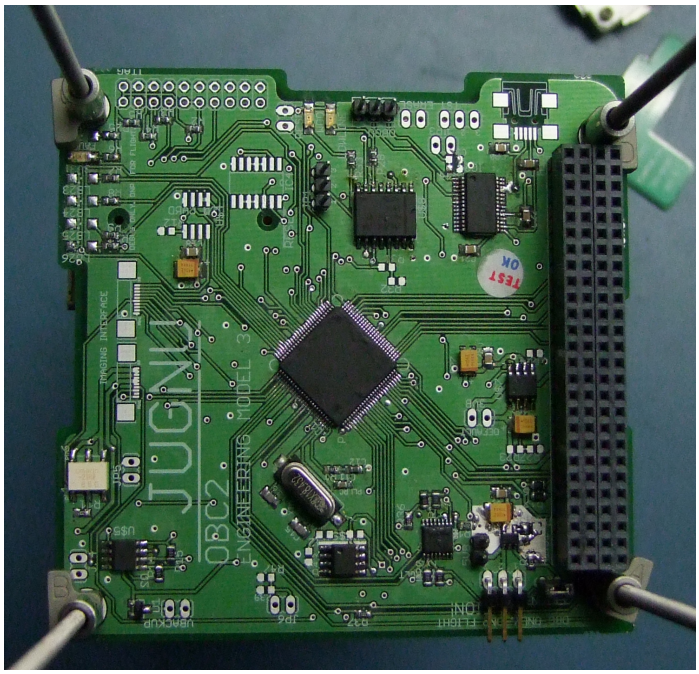

Figure 2.3: *OBC-2 ARM Board*

kernels are preemptive and μC/OS-II is comparable in performance with many of them. μC/OS-II is written in highly portable ANSI C, with target microprocessor specific code written in assembly language. Porting μC/OS-II on to SAM7(ARM) core was thus very easy for us. It is sufficiently robust to meet rigorous safety critical requirements. It has ROM footprint of 5Kb - 24Kb (scalable). We chose μC/OS-II over other open source RTOS because it comes with complete source code and documentation and is currently implemented in a wide array of high level of safety critical devices including Avionics, medical devices, etc [5].

Embedded Filesystems Library (open source)

We ported the Embedded Filesystems Library [6] to our platform, EFSL is Library for filesystems intended to be used in embedded projects.The library currently supports FAT12/16/32 reading & writing on SD-cards, and is easily expandable for use with other devices on any platform. This library can be used with as little as 1.5

kilobyte RAM, however if you have more at your disposal, an infinite amount can be used as cache memory. The more memory you commit, the better the performance will be. In order to minimize IO to the hardware 2 caching mechanism are in place on different layers. The first layer is the IO Manager which does "dumb" caching of sectors. This will of course work better if you have RAM to spare. The second level is higher up in the library, where some tricks are in place to minimize redundant reading of the FAT table (which is the primary bottleneck of the FAT filesystem, especially with hardware which has seek times). With this caching policies in place the speed up obtained is better than direct memory access to the SDCARD.

Integrated Development Environment for ARM

IAR Embedded workbench [18] is being used for programming and debugging codes on OBC - 2 (ARM). Some of the key features of IAR Embedded Workbench are : strong debugger with user friendly IDE, RTOS-aware debugging with built-in or 3rd-party plug-ins, plug-ins for μCOS is available which made debugging of RTOS based application simple, ready-made peripheral register definition files and example codes provided are helpful in building the application, compatible with J-Link and J-Trace (hardware debug probes)[19].

Chapter 3

Software Design of On Board Computers

Designing the software of satellite on board computers is done by considering the functionality of the entire satellite. Performance of the satellite depends upon performance of on board computers; in small/micro satellites there are separate modules for data handling, telemetry and telecommand handling, payload operation sequence, control systems etc. But, in the case of nanosatellites all these operations are performed by a single on board computer. Hence, on board computers for nanosatellites have to perform many tasks simultaneously, operation of on board computers decides the mode in which satellite functions. Also in small conventional satellites redundant devices are used to provide high reliability and long lifetime. Devices used are space graded and very costly, program and data memory are generally radiation hardened which is essential for survival of such satellite in space for longer duration of time even in harsh space environment. But for nanosatellites such devices cannot be used, main reason for building a nanosatellite is to provide reliable

performance using commercial-off-the-shelf (COTS) components. Same reliability and lifetime cannot be guaranteed for nanosatellites as that for small satellites using commercial-off-the-shelf components, generally the lifetime of nanosatellites is around one year. Reliability of commercial-off-the-shelf components is increased by designing software which can detect temporary failures and take corrective measures. On board computers for nanosatellite should be capable of performing different task simultaneously and the software should be such that it can provide reliable and controlled satellite operation even with commercial-off-the-shelf components.

Software design also consist of creating device drivers for all the devices interfaced with on board computers. Device drivers are interrupt driven and timeout based. RTOS is used on both the on board computers for scheduling various task on the satellite. Task are given priority and scheduled such that none of them cross their deadline. One of the important factor which is considered in embedded system software is software fault tolerance, We have taken measures to provide software fault tolerance on the satellite using redundancies. Following sections provide an overview of the on board computers software.

3.1 Health Monitoring on Satellite

Health monitoring plays an important role in the survival of the satellite in space. If the satellite stops working in normal mode then the health status telemetry data received at ground station can be used for decoding the problem and take the corresponding action by transmitting telecommands. Health status data is also transmitted through beacon. Health monitoring is also essential for the safety of devices in the satellite, also proper utilization of the available power is essential so that power

is reserved for performing critical operations on the satellite (power distribution). Also maintaining the status of different devices on the satellite (working or dead) is essential, turning "on" failed devices[1] for a long time may drain large current through the board which can also damage the other interfaced devices.

The different health monitoring components are:

1. Temperature monitoring
2. Power monitoring
3. Device health and performance monitoring

3.1.1 Temperature Monitoring

Temperature sensors are placed on all the critical boards of the satellite. OBC samples this sensors and monitors the temperature of different boards on the satellite. If the temperature of a particular board falls below the critical value then the temperature of that part of the satellite is increased by pointing it towards sun using ADCS. Similarly if the temperature of the board increases above threshold then power supply to that board is turned "off". Before turning "on" any of the devices or boards the temperature value of that board is monitored and the corresponding action is taken thus providing protection against thermal damage. OBC also maintains a Health table of the satellite which stores the current temperature values obtained from all the temperature sensors. This health table is updated after every 2 secs.

[1]Devices can fail because of latch up, burn out, etc.

3.1.2 Power Monitoring

Power monitoring involves proper power distribution on the satellite guaranteeing survival and normal operation of the satellite under different power conditions. For performing this operation, Power Consumption table is maintained by the satellite for different devices as well as for different modes of operation of the satellite. Thus whenever a device is turned "on" by OBC the corresponding entry in the Power consumption table is compared with the current power available if sufficient power is available then the device is turned "on". Also when OBC switches from one mode to another, the overall power consumption during that mode is compared with the available power. If available power is less, then the mode switch is not done and the satellite operates in the current mode until the available power is more than the required power. Before performing any payload operation power status is checked and if sufficient power is not available then that payload operation is not performed. Also whenever a device is turned "on", current drawn by the device is monitored initially so that if the device is faulty then it will draw large amount of current which will be detected and device will be turned "off" immediately. In case of lower power mode, all the non-critical devices will be turned "off" and OBC will wait for the power status to rise. Current battery status is transmitted continuously through beacon.

3.1.3 Device Health and Performance Monitoring

Device health table is maintained on OBC which indicates the status of all the devices interfaced with it. Device can be in "working" or "reset" or "dead" state. Device is in "working" state if it is functioning normally, if the device is turned

"off" because of temporary failures like Single event latchup(SEL), reconfiguration needed because of bit flip or faulty behaviour then the device status is initialized to RESET. Whenever a device in RESET mode is turned "on", then proper checks are performed on the device and the device is reconfigured before its status is initialized to "working". If the device failed OBC checks like current drawn, register readback, device interface initialize, etc. then the device status is initialized to "dead" and is no longer used for satellite operations. In case if redundancy is maintained on the satellite for that device then that redundant device is used. For example if the SDCARD interfaced with OBC fails then the internal flash memory will be used for logging of critical data and payload operations will be performed depending on the available internal memory, also if the receiver fails then protocol used for downlink will switch to broadcasting mode without waiting for ACKs from ground station. If solar panel failure on one face is detected then the satellite switches to the solar panel on third face by rotating the satellite by 90° using ADCS. Various such counter actions are implemented on the OBC to maintain the satellite in normal mode even if certain devices fail during lifetime of the satellite.

3.2 Event Sequencing on Satellite

On board computers being the central controller of the satellite, have to schedule all the events which includes payload operations, communication window, telemetry data formation, etc. All this operations are scheduled on the satellite using J2-Orbital propagation algorithm[20]. OBC calculates the time at which different operation are to be performed by passing the target coordinates to orbital propagation function. Orbital propagation function finds the exact time at which satellite

will come exactly above the target location. In the case of Imaging, target location is the coordinates whose picture is to be taken by the satellite. For communication window, target location is the coordinates of Indian Institute Technology, Kanpur ground station. Similarly, GPS and IMU operations are also scheduled using the orbital propagation algorithm and the coordinates of the satellites when the payload operation is to be performed. Other than these payload and telemetry operations which are dependent upon the position of satellite, there are events which needs to be scheduled periodically irrespective of the position of the satellite like Memory Check on satellite, running the ScheduleEvent (ScheduleEvent is a function used to add events into the sequence queue) operation, satellite resets to correct soft-errors in RAM these events are also scheduled using event sequencer.

Event queue is maintained on both the OBCs, events are added into the queue with timestamp indicating when these events will occur in the future and should be serviced by OBC. Events are maintained in the form of structure which has fields in it like start time, end time, periodic or non-periodic, period and name of the task which needs to be created to service the event. ScheduleEvent operation is performed periodically to add events onto the sequencer queue, it runs the orbital propagation code and finds the timestamp for all the events which needs to be scheduled and adds them into the queue. Each OBC have a task "Sequencer" running on them which keeps on looping over this queue and whenever the event timestamp matches with the current system time, task is created by the sequencer to service that event.

3.2.1 Periodic Events

When a periodic event is serviced it is added back into the queue with a timestamp equal to current time + period of the event. No ScheduleEvent operation is performed for these events as they are periodic, the periodic field in the event structure is set for such events and the period field is initialized with the periodic time value. Periodic events does not have any prerequisite operations associated with them. Example of such events are Memory Check (for soft error correction on flash), Periodic satellite resets (for soft error corrections in RAM), ScheduleEvent operation to find the events to be schedule in current period (in our case the satellite is propagated by 12 hrs and ScheduleEvent is called every 10 hrs).

3.2.2 Scheduled Events

Scheduled events are ones that are dependent upon the position of the satellite and instructions received from ground station. These events are added in to queue by using the orbital propagation method. Once these events are serviced they are removed from the queue. Also each event has some prerequisites to be performed before the event can be serviced, for example, if communication window event is to be added in to the queue then before that events like Make packet, ground station tracking, Momentum dumping are added into the queue. Thus while scheduling a particular event, its prerequisite events are also scheduled by ScheduleEvent operation. These prerequisite events are scheduled depending on their execution time.

3.3 Satellite System Time

For the proper functioning of satellite system which includes satellite and ground station, it is essential that system time maintained is synchronized. Same system time with minimal delay is essential because telecommands sent and the telemetry data received are timestamped based on this system time. System time on the satellite is maintained by using a 32 bit counter, this counter is incremented every 0.02 secs. Hence the time is maintained on the satellite at an accuracy of 20 ms. OBC-1 and OBC-2 periodically synchronize their system clock. Also each OBC has a Real Time Clock (RTC) module (M41T81S [15]) connected to it, these RTC modules have battery backup hence even under low power or power reset situations they will maintain the satellite system clock. System clock maintained by both the OBCs is also synchronized and corrected using these RTCs. Also for proper event scheduling and analysis of timestamped data received at ground station, ground station also maintains a system clock which is synchronized with the satellite system clock. Each time the satellite comes in communication window a special time-synchronization packet is exchanged between the satellite and ground station.

3.4 Tasks List

OBC-1 uses Salvo PRO which has a cooperative priority based scheduler and OBC-2 uses μCOS which has a fully preemptive real time kernel. RTOS is used for providing real time behaviour on the satellite and scheduling different tasks on the satellite using the RTOS scheduler. Depending on the operations to be performed on OBC, these operations are split in to tasks with priorities and scheduled. Figure 3.1 and

Figure 3.2 describes the tasks running on OBC-1 and OBC-2 respectively. Tasks shown in red are critical periodic tasks and are always in the ready queue of the scheduler. Tasks shown in blue are created and destroyed by the 'Sequencer' task depending on the event to be serviced. Tasks shown in green are replaced by its child task depending on the mode of operation of the satellite for example when satellite is in detumbling mode 'Detumbling Task' runs in place of ADCS Modes.

3.4.1 On Board Computers - 1

Salvo Pro [3, 7] RTOS is used on OBC-1. All the different tasks running on OBC-1 are as described below:

Health Monitoring Task: This task performs the operation described in Section 3.1. Its a periodic task and has highest priority. It updates the current health status of the satellite and depending on the status corrective actions are taken if required. Periodic time of this task is $\approx$ 1 minute.

Beacon Update Task: This task periodically updates the health status values to be transmitted by beacon. It has lowest priority and the periodic time of the task is $\approx$ 3 minutes. Beacon is transmitted by using a timer interrupt which keeps transmitting the updated beacon signal in the background.

Sequencer Task: This task performs the operation as described in Section 3.2. Its a periodic task with medium priority and periodic time of 2 minutes. It creates task to service the scheduled events.

Diagnostic Task: This task performs the diagnostic operation on satellite. It monitors the progress of all the tasks and if it finds that a particular task has hung,

it restarts that task. Also it does periodic log of subsystem level global data into SDCARD which can be used for system restore if there is a planned reset. Period of this task is $\approx$ 2 minutes.

OBC-2 Synchronize Task: It is a periodic task which communicates with OBC-2 and exchanges all the critical data required for synchronization of the two OBCs. Following parameters are synchronized current magnetometer reading, mode of operation, ADCS mode, system time, health and device status, etc. The period of this task is $\approx$ 5 secs.

Communication window Task: This task is created by the 'Sequencer Task' to perform the necessary operation when the satellite is in communication window. It has high priority but the priority is less than that of Health Monitoring Task. This task runs in the background and services the commands received from ground station while interrupt driven (foreground) technique is used to buffer the data received from receiver. If a packet is successfully received (CRC is valid) only then it is retrieved from the buffered and serviced by the communication window task. ACKs received from ground station are immediately forwarded to OBC-2. This task synchronizes the phase of communication window with OBC-2, both OBC-1 and OBC-2 have Communication window task running on them. Both these task run in the same mode and successfully execute the communication protocol.

ADCS Modes: Depending on the mode of operation of ADCS corresponding task is created and put into ready queue either by OBC-2 synchronization task or the Sequencer task. Previous task is replaced by the new task using RTOS

APIs. Only one of the ADCS Tasks run at any given instant of time. Each task perform separate set of operations and different devices are involved in those operations for example for Maneuvering mode magnetometer and reaction wheels are involved while in momentum dumping mode reaction wheels, magnetometer and torquer coils are involved.

Memory Check Task: This task performs memory check on the internal program flash of the OBC. If CRC check fails, which indicates that there is a bit flip in program memory then TMR is implemented on that block of memory in flash and the program memory is restored to correct state. This is a periodic task and the period of task is $\approx$ 2 hrs. This task also performs TMR on the filesystem and on Image of code stored on SDCARD.

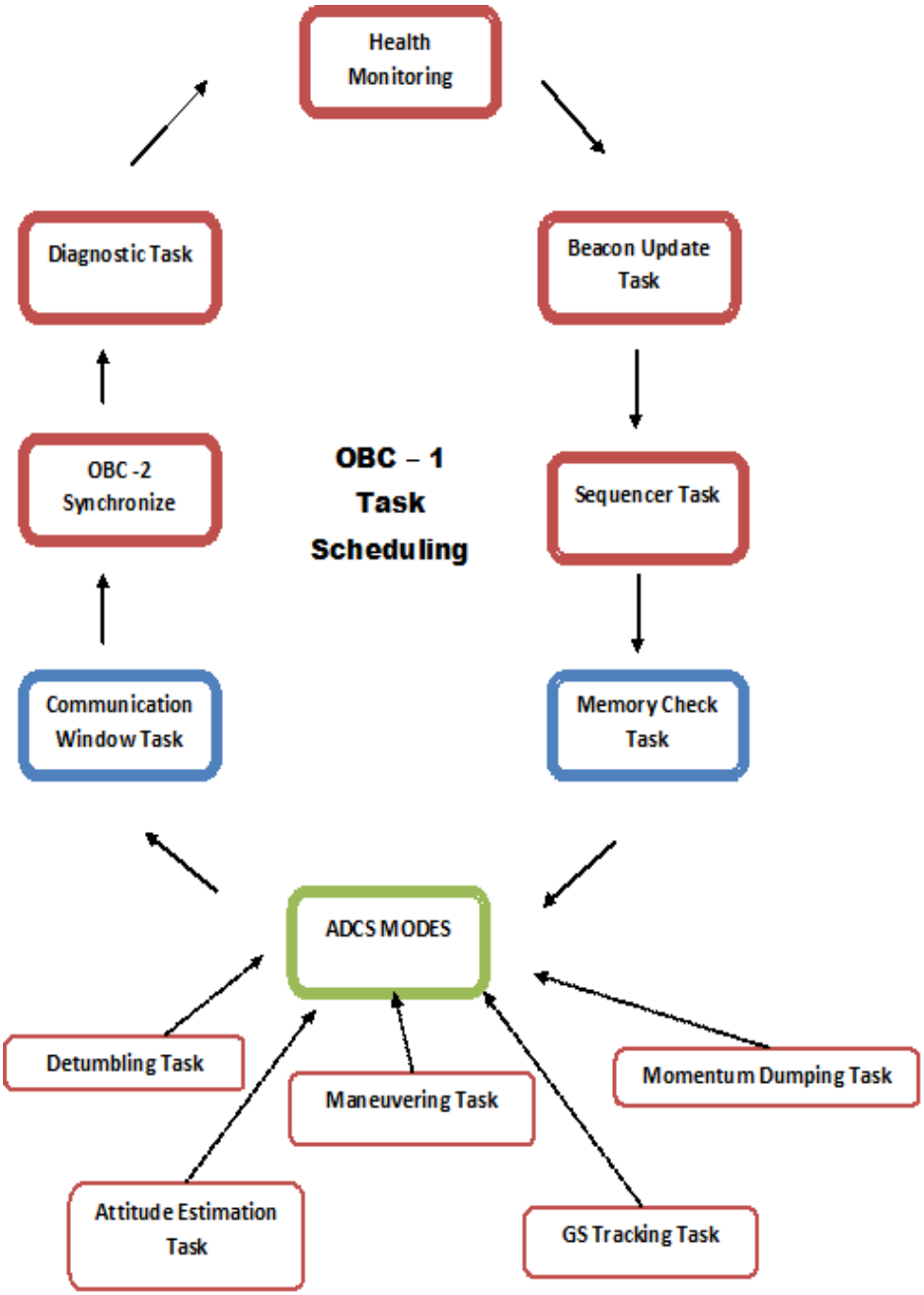

Figure 3.1: *Tasks scheduled on OBC - 1*

3.4.2 On Board Computers - 2

μC/OS-2 [4] is used as RTOS on OBC-2. All the different tasks running on OBC-2 are as described below:

Payload Operation Task: This task is created by the sequencer task, any one of the payload operation task is scheduled at any instant of time. This task performs all the payload control related operations right from turning the payload device "on" to receiving the log data and storing it onto SDCARD and turning the payload device "off". It has high priority.

Make Packet Task: This task is scheduled by 'Sequencer Task', it creates packets from the RAW data on SDCARD and stores them in to a separate file on SDCARD. It also reads the health log data stored on OBC-1 converts them into packets and stores them in critical data transmission file on SDCARD.

Communication window Task: This task is created by the 'Sequencer Task' to perform the necessary operation when the satellite is in communication window. It has high priority but the priority is less than that of Health Monitoring Task. This task retrieves packets from the buffer and appends header to it and sends it serially to the transmitter. Downlink communication protocol is implemented in this task, ACK received from OBC-1 are piggybacked and transmitted to ground station. This task runs synchronously with communication window task of OBC-1.

ADCS Modes: Depending on the mode of operation of ADCS corresponding task is created and put into ready queue by the Sequencer task or diagnostic task. Previous task is replaced by the new task using RTOS APIs. Only one of the

ADCS Tasks run at any given instant of time. Each task perform separate set of operations and different devices are involved in those operations for example for Attitude estimation task only reaction wheels and magnetometer are involved while for momentum dumping reaction wheels, magnetometer and torquer coils are involved. Normally attitude estimation task runs in the background with medium priority and periodic time of the task is $\approx$ 4 secs. Operations performed by ADCS tasks on OBC-1 and OBC-2 are different as devices controlled by these tasks are different. OBC-1 ADCS task mainly controls the magnetometer and torquer coils while OBC-2 ADCS task does all the computational job and controls the reaction wheels driver.

Health Monitoring Task, Memory Check Task, Sequencer Task and Diagnostic Task are similar to those on OBC-1.

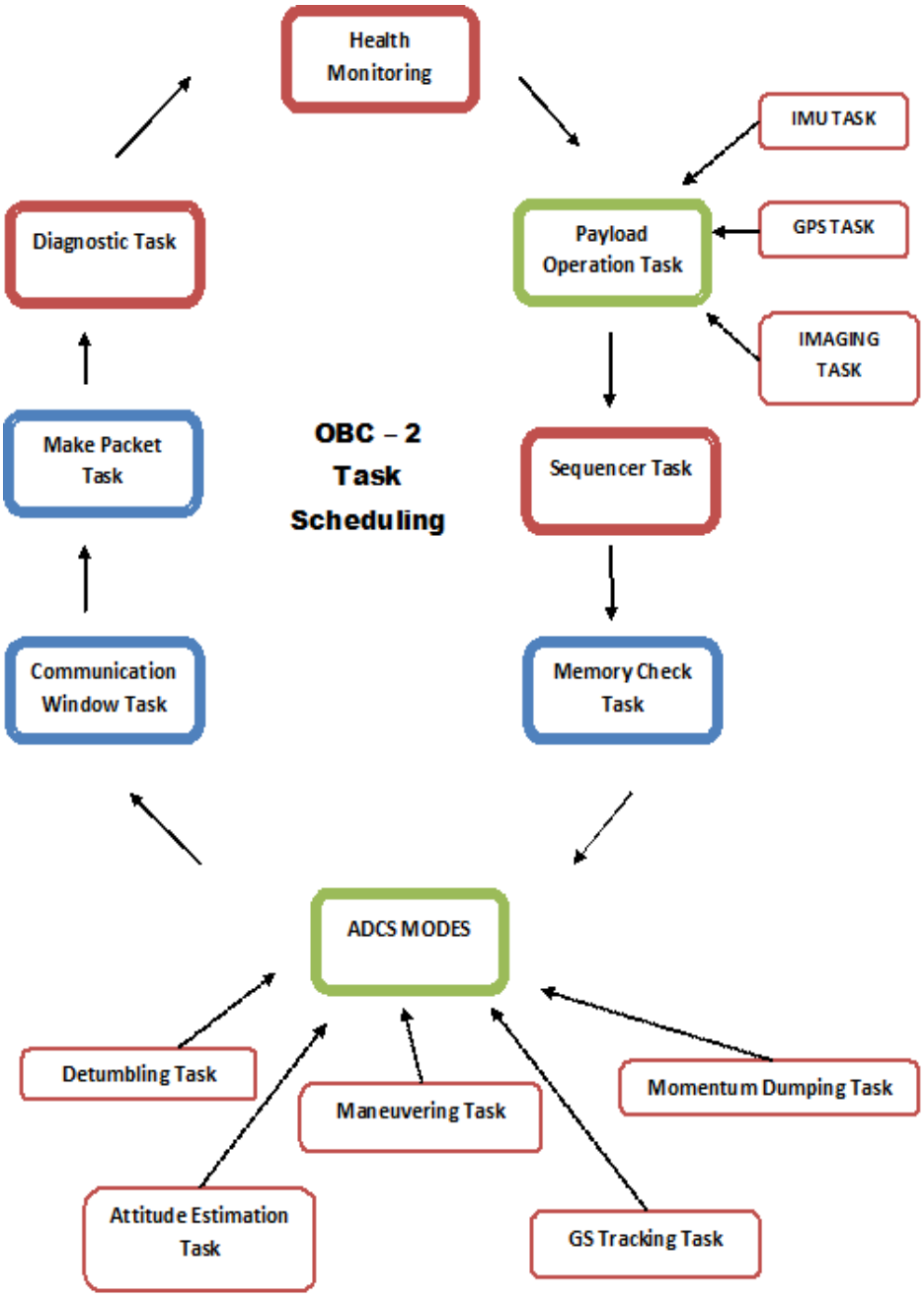

Figure 3.2: *Tasks scheduled on OBC - 2*

Chapter 4

Communication System for JUGNU

For the success of any satellite mission, one of the important mission requirement is the success of communication system of the satellite which consists of RF system and the communication protocol implementation.

The *communication RF system* starts with the link budget analysis which includes analysis of losses suffered by a signal and the noise that gets added to it during propagation as well as due to different devices in the system chain. A decent margin is aimed over the targeted BER for a particular frequency choice, keeping the constraints on transmitted power, receiver sensitivity, antenna gain realization, etc. in mind. Figure 4.1 describes the basics of the communication subsystem with the specifications in Table 4.1. Objective of the communication subsystem of the satellite is to provide the best link margin possible and also under the given variable link performance, provide maximum throughput along with data-reliability.

4.1 Communication Link Specifications

Table 4.1 below describes specifications of the uplink, telemetry and beacon for Nanosatellite JUGNU. The link budget has been calculated for 10° elevation when the satellite is at it's maximum slant range from the ground station.

Table 4.1: *Specifications for uplink, telemetry and beacon.*

Parameters	**Uplink**	**Telemetry**	**Beacon**
Frequency	145.92MHz	437.505 MHz	437.275 MHz
Transmit power	20W	1W	50mW
Modulation	FSK	FSK	ASK
Data Rate (bps)	1200	2400/4800/9600 [1]	12
Targeted BER	0.0001	0.0001	0.0001
GS antenna gains	12dB	18dB	18dB
Satellite antenna gain	-5dB	-5dB	-5dB

Basic block diagram of entire Communication system of the *JUGNU* Nanosatellite is given in Figure 4.1:

4.2 Introduction to Communication System

JUGNU communication system is composed of three major units Telecommand Uplink, Telemetry downlink, Beacon downlink.

[1] Variable adaptive baud-rate feature is used on the satellite to provide reliable data transfer

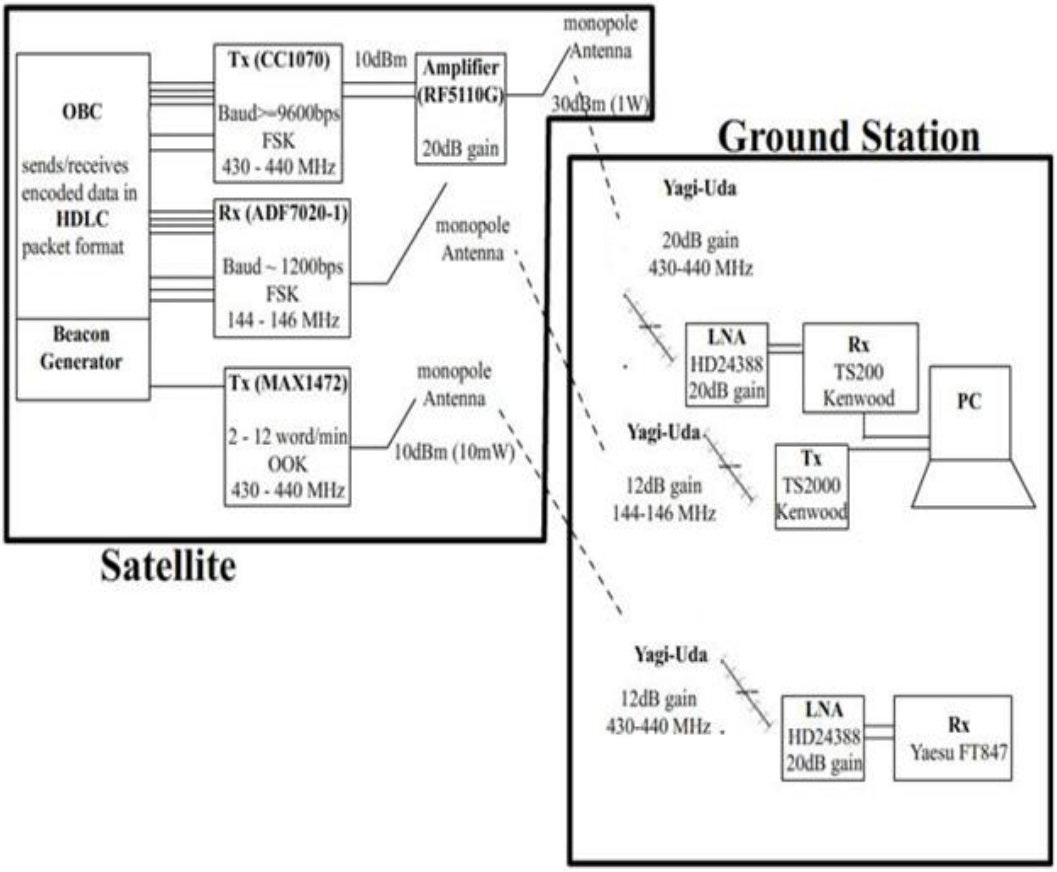

Figure 4.1: *Block diagram of communication subsystem.*

4.2.1 Telecommand Uplink

Uplink refers to the telecommands being transmitted by ground station to satellite. The uplink includes commands that will guide the satellite to perform certain operations as well as transmit acknowledgements to satellite in response to the data received at ground station. In any communication system for satellites, the uplink must be designed as efficiently as possible because at times it may be necessary to completely shutdown/reset the satellite by sending a telecommand to it or control the operation or performance of various devices on the satellite. So, data reliability and correct telecommand processing is important.

The power that is transmitted from the ground station is generally in control of the ground station controller but it is intended to make the uplink less susceptible to path loss; hence, selecting low frequencies and low baud rate than the downlink is common choice.

4.2.2 Telemetry Downlink

It refers to the payload data and the data corresponding to the health status of different sub-systems of the satellite being sent by the satellite to the ground station. Downlink is very important so that the ground station knows the exact status of the satellite through its detailed critical health data being received. Also the mission of the satellite highly depends on the performance and success of the payloads which can be verified only if the payload telemetry data is successfully received at ground station and analyzed. Hence, the downlink protocol should be such that maximum data can be received at ground station with maximum channel throughput also guaranteeing the reliability of the data received.

It is intended to design the downlink so that minimum noise gets added and the packets are obtained with minimum distortions. At higher frequencies the galactic noise is quiet low; hence, the spacecraft added noise is less. However, at the ground station one can boost up the signal using low noise amplifiers at the receiver front end that themselves add less noise. Hence, it is a common practice to keep downlink frequency higher than the uplink. Also one of the *mission objective* is data downlink rate of 9600 bps should be possible during communication.

4.2.3 Beacon Downlink

This is designed as the most reliable system in the communication subsystem. It usually consists of satellite's call-sign which is the unique identification of the satellite. If uplink and downlink fail for some reason then the only way to know that the satellite is still in the orbit is the beacon signal. Usually the beacon signal is accompanied by a general health status of the satellite. The beacon system design may include redundancies if necessary and it is intended to keep it as a separate unit not fully controlled by the on board computer. Beacon is always "on" after separation, and continuously transmits the call sign as well as the encoded satellite health status. To transmit *amateur radio beacon* signal all over the world is one of the *mission objectives* of the satellite.

4.2.4 Protocol Design

While designing the *communication protocol* for JUGNU Nanosatellite, some of the important points that were considered are :

1. JUGNU is a low earth orbit (LEO) satellite, hence have the property of pe-

riodic visibility over ground station. Because of this periodic nature, link management and ground station tracking play an important role in the performance of the communication protocol.

2. Limited power generated on the satellite. Computational power limitation on the satellite.

3. Providing reliable data communication with best possible channel utilization.

4.3 Communication System Block Diagram

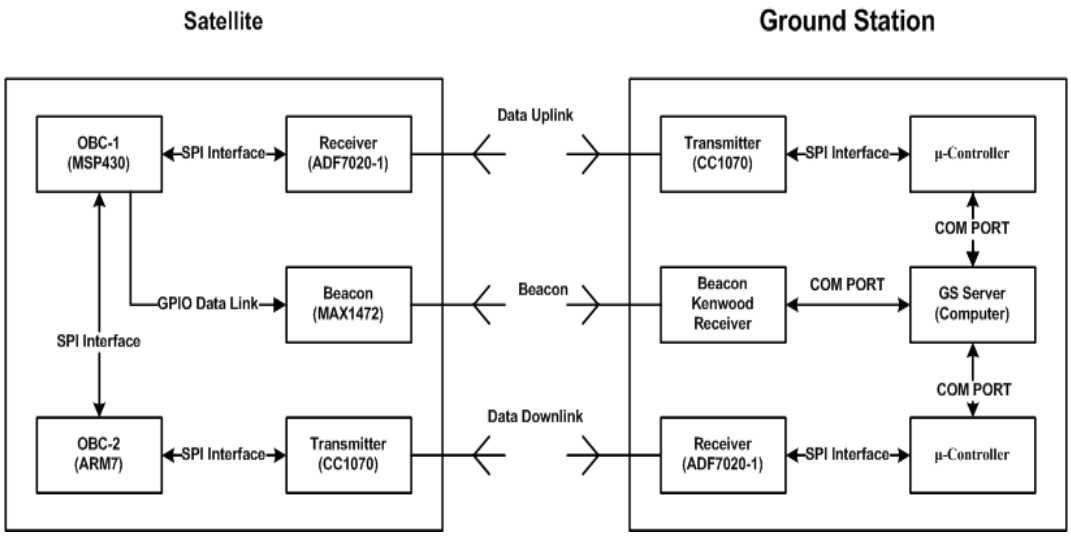

Figure 4.2: *Block Diagram of the satellite communication system*

The function block diagram of the communication system is as shown in Figure 4.2. The satellite has two *on board computers*, OBC-1 (TI MSP430 [9]) and OBC-2 (AT91SAM7X [17]). From the specifications given in Table 2.2, OBC-2 is computationally more powerful than OBC-1. Ground station consist of one server PC having sufficiently high computational power and memory.

4.3.1 Ground Station

Transmitter: Ground station uses the same transmitter module as on satellite (CC1070 [21]). Transmitter is interfaced with the server PC using a ATMEGA8 [22] microcontroller. ATMEGA8 [22] is used because there is no need for a faster and complex micro-controller, since it is only used as a parallel to serial converter.

Receiver: Ground station uses the same receiver module as on satellite (ADF7020 [23]). Interfacing of the receiver with server PC is same as that of transmitter.

Beacon: Beacon signals are decoded at the ground station is done using Kenwood transceiver whose serial output is directly connected to the server PC.

Entire communication protocol is being implemented on the server machine which has very high computational power so that the memory and timing and computation constraints are nearly eliminated. Two separate process will be running on the server PC for transmitter and receiver operation. Shared memory concept is used for Inter Process Communication(IPC) between the two process. There is a beacon process which will continuously decode the beacon signal received and display the decoded signal on Ground station (GS). The GS server has a GUI which displays the current status of the satellite along with payload data received from the satellite.

4.3.2 Satellite Communication System

Transmitter: Transmitter module (CC1070 [21]) is connected to OBC-2 on the satellite. SD-CARD is connected to OBC-2 which can be used for storing the data packets to be transmitted to ground station. Since OBC - 2 has

comparatively higher computational power (48Mhz, 32-bit RISC, 512KB flash) one can implement more complex protocol which can improve the throughput efficiency of the satellite downlink.

Receiver: Receiver module (ADF7020 [23]) is connected to OBC-1 on the satellite. SD-CARD is connected to OBC-1 which can be used to store the tele-commands received from GS. These commands are processed after the satellite is out of communication window. Since OBC-1 (8Mhz, 16-bit RISC, 128KB flash) is comparatively slower, uplink protocol should be designed such that minimal computation is done at the receiver end.

Beacon: Beacon module (MAX1472 [24]) is connected to OBC-1 on the satellite, since OBC-1 is the reliable and space proven of the two on board computers. Health data will be transmitted along with call-sign through the beacon signal.

Receiver and beacon are connected to OBC-1 because even if the transmitter on OBC-2 fails, satellite health data can be received from beacon and satellite can be controlled through tele-commands.

4.4 Communication Window

JUGNU is a LEO satellite with orbital height of ≈ 700 km. LEO satellites have this property of periodic visibility, hence are visible above the ground station only for a limited *period* of time. The *time period* during which the satellite is visible above the ground station is called *Communication Window* of the satellite. For calculating the communication window of the satellite we have considered the satellite to be visible when it is in the cone above Indian Institute Technology, Kanpur with conical angle

of 75°.

Communication with the satellite can be established even before it is inside the 75° cone, but for performance, safety margin reasons and calculating the time during which satellite can successfully communicate with ground station with best link margin, we have considered the 75° cone only.

We carried out simulations for calculating the average duration of the communication window. Communication window results are required for the protocol implementation so that the time available for communication with the ground station is known and the size of data packet and the information that can be communicated in one day can be calculated and tested. The simulation code developed calculates the communication window of the satellite orbit for a 75° cone above Kanpur (Latitude: 26.46667; Longitude: 80.35). The satellite will be placed in a *sun-synchronous orbit* around the Earth. The calculations have been done for the below mentioned orbital parameters Table 4.2 and we do not expect much changes in the results even if the parameters are varied slightly and/or the launch date/time is changed.

Table 4.2: *Orbital parameters*

Parameter	**Value**
a (semi-major axis)	700 km
e (eccentricity)	0.0013
inc	98.186172 deg
Date of launch	15-AUG-2010
Time of launch	1030 hrs
Time period	5926.39 seconds ($\approx$ 5927)

4.4.1 Simulation Results

	Number of orbits	Number of communication windows	Total communication window time (seconds)	Maximum communication window time (seconds)	Average communication window time (seconds)
For 1 day	15	3	1060	465	353.33
For 1 week	103	19	7015	475	369.21

Figure 4.3: *Communication window simulation results*

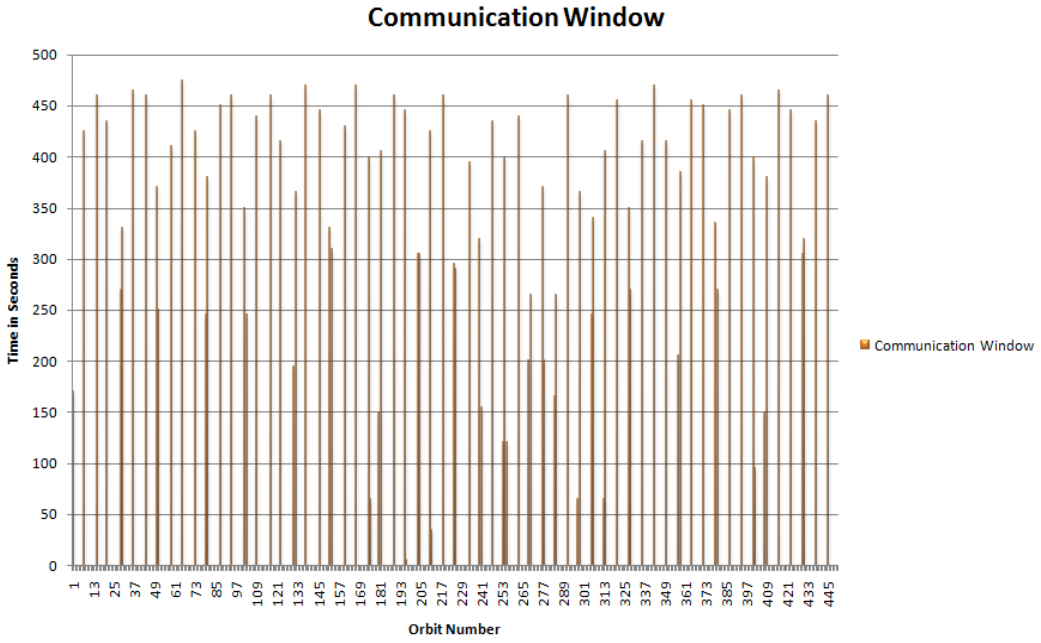

Figure 4.4: *Communication window simulation results*

Figure 4.4 shows the plot of length of communication window (time) vs. orbit number considering the initial orbital parameters shown in Table 4.2. From the simulation results shown in Figure 4.3 and Figure 4.4 obtained by propagating the satellite position using J2-orbital propagation algorithm we can deduce the following conclusions:

1. Satellite orbits around the earth approximately 15 times per day.

2. Satellite will be visible above the ground station (considering the 75° visibility cone) for 3 orbits per day on an average.

3. Some calculations:

   (a) Maximum communication window size is 475 secs

   (b) Minimum communication window size is 5 secs

   (c) Average communication window size is 337 secs

   (d) Standard Deviation of the communication window size is 124 secs.

4. From the graph shown in figure 4.4 we can see that out of 15 orbits per day the satellite is visible over the ground station approximately in 3 orbits and most of this orbits have communication window size more than 200 secs which is sufficient for transferring the health data along with payload data under normal communication link performance.

5. Also there are communication windows having size less than 60 secs (5 - 60 secs). In such cases the OBC will detect these windows and since payload data cannot be transmitted during this window we will transmit only the important health data.

   Depending on the size of communication window following actions will be taken by on board computers:

   (a) If communication window $\leq 60 secs$ then transmit only the health data from satellite to GS and transmit only the critical telecommands from GS to satellite.

(b) If communication window $\leq 300secs$ then transmit health data and important payload data. Priority of the payload data is given in the Table 4.3. Payload data will be sent depending on the window size and priority of the payload data.

Table 4.3: *Payload data priority*

Payload Data	**Priority**
IMAGE	1
GPS POSITION	2
IMU LOG	3
GPS SATELLITE ACQ	4
IMAGING PARAMETERS	5
ADCS LOG	6
COMMUNICATION LOG	7

(c) If communication window $\geq 300secs$ then the satellite can transmit all the health and payload data. Payload data is always transmitted in the order given in Table 4.3.

4.4.2 Telemetry Data Transfer

From Table 4.3 it can be seen that the average length of the communication window is $\approx 350secs$. As we will see in Section 4.5.6, data will be transmitted by the satellite in different baud rates. Initially the critical satellite health data will be transmitted at a lower baud rates of 2400bps and then the payload data will be transmitted at a

higher baud rate of 9600bps. Also variable packet length mentioned in Section 4.5.5 feature is being used on the satellite keeping in mind the changing nature of the communication link. Hence by default, critical data will be transmitted with packet length of 64 bytes and the payload data will be transmitted with packet length of 200 bytes. Also in Section 4.5.3, we will see that the overhead bytes because of control information and synchronization bits per packet is ≈ 24 bytes.

We have split the satellite to ground station data transmission in two parts, critical data transmission and payload data transmission. On an average communication window size (with the 75 degree cone angle) is about 350 secs but from the testing of our ground station antennas we have been able to track the satellites when they are at 5 degree angle with the horizon. Hence we can assume that by the time satellite comes in the 75 degree communication window, link would have been established and data transfer protocol can be started.

We split the 350 secs window in to 100 secs and 250 secs phases, assuming that the critical data will be transmitted during the first 100 secs and the payload data will be transmitted during the next 250 secs. The table below gives the rough idea about total data that can be transmitted during an average communication window with channel efficiency of 60% and overhead bytes per packet being 24 bytes.

Table 4.4: *Possible data transfer in one communication window*

Data	**Time Duration**	**Baud Rate**	**Channel Utilization**	**Total Data**
Critical Data	50 secs	2400 bps	60 %	5.635 KB
Payload Data	300 secs	4800 bps	60 %	95.04 KB

Table 4.4 continues on next page

Table 4.4 continued from previous page

Data	**Time Duration**	**Baud Rate**	**Channel Utilization**	**Total Data**
Payload Data	300 secs	9600 bps	60 %	190.08 KB

From Table 4.4, we can transfer $\approx$ 5.6 KB of critical data and $\approx$ 190 KB of payload data in an average communication window. Hence we can transfer all the critical and payload of the satellite mentioned in Table 4.5 and Table 4.6 under normal communication link behaviour and in an average communication window.

Table 4.5: *Health Status Data*

Health Status Data	**Size**
Reset time of the satellite if it was reset in the given time duration or else a NIL packet	$\approx$ 100 B
In the case of mode switch, reason for mode switch can be transmitted along with the time stamp. Example emergency mode, low power mode, etc.	$\approx$ 100 B
Power Log	$\approx 500 - 700$ B
Temperature Log	$\approx 500 - 700$ B
Satellite status like which devices are working fine, status of devices	$\approx$ 50 B
Switching ON/OFF time of all the devices (Success / Failure)	$\approx 500 - 700$B

Table 4.5 continues on next page

Table 4.5 continued from previous page

Health Status Data	**Size**
Time synchronization packets	≈ 50 B
Health Status of Solar panels	$\approx 100 - 300$ B
Performance of ADCS Algorithm	$\approx 500 - 1000$ B
Total	≈ 3.5 KB

Table 4.6: *Payload and non critical data*

Payload and non critical data	**Size**
IMU log data	≈ 10 KB
NIR Images	≈ 130 KB
GPS log data	≈ 10 KB
Communication signal strength related data	$\approx 500 - 1000$ B
All the temperature sensors reading to get temperature variation in different parts of the satellite	≈ 4000 B
ADCS log data	≈ 4 KB
Total	≈ 160 KB

4.5 Communication Protocol

Communication protocol is a set of rules and procedures for error detection, data representation, flow control, etc to send information efficiently and reliably over

a communication channel. Protocol is designed to ensure communication over a varying communication channel. JUGNU will be put in to low earth orbit which have this property of periodic visibility over ground station and also within a pass the channel performance varies depending on its azimuth and elevation angles.

Communication protocol for JUGNU is being designed keeping in mind this channel features and the specifications of our communication system. Maximum channel utilization and reliability is aimed, also considering the computational and memory constraints on the satellite. For experimental purposes we have also introduced features like variable/adaptive baud rate and variable packet length features into our protocol. The following sections describe JUGNU communication protocol in detail.

4.5.1 Communication Protocol Stack

Communication protocol stack is used to split the communication protocol into different layers which makes understanding the protocol easier and simplifies implementation of the protocol. JUGNU communication protocol is made up of three layers basically physical layer, data link layer and network-application layer. These layers are explained in more details in the following subsections.

Physical Layer

Physical layer is concerned with the following:

Hardware Interfacing: Figure 4.2 shows the detailed block diagram of the communication system of the satellite with all the interfacing details. On the satellite and the ground station its the responsibility of the physical layer to

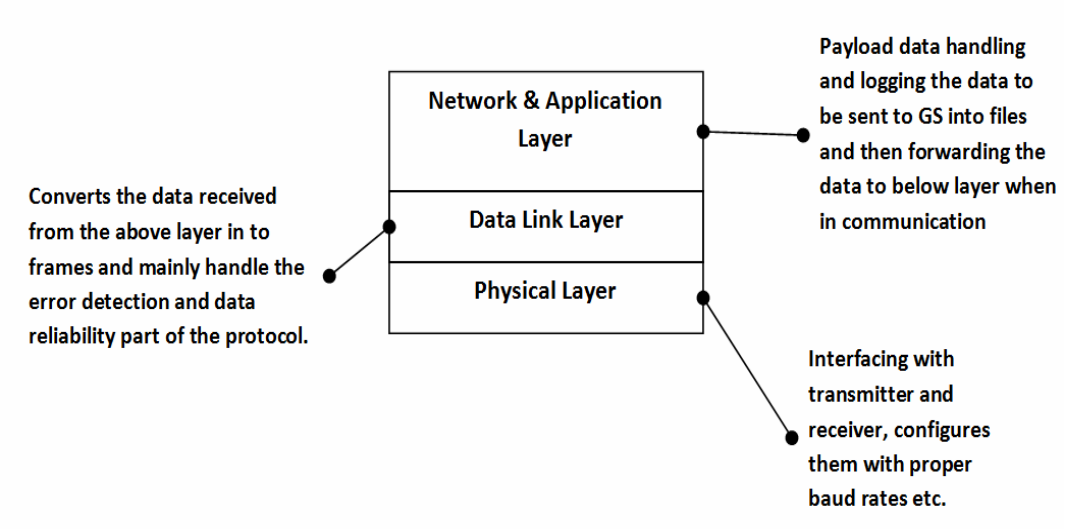

Figure 4.5: *Communication protocol stack*

monitor the health and performance of these devices. For example if the device is wrongly configured then detecting it and reconfiguring it. Further, in space, devices might latch up and require sudden reset and reconfiguration.

Baud rate: Physical layer is concerned with configuring the internal registers of the receiver and the transmitter such that they operate at the same baud rate. Also on the satellite we have implemented adaptive baud rate feature. Hence, it is the responsibility of the physical layer to change the baud rate when the corresponding command is received from the above layer.

Synchronization of bits: The physical layer provides bit synchronization between the transmitter and the receiver such that their clocks are synced and they use same bit duration and timing. In our case this is attained by sending 48 bits of preamble before each packet as synchronization bits.

Data Link Layer

Data link layer is concerned with framing, flow and error control and Automatic Repeat Request(ARQ) type of protocols to provide smooth and reliable data communication:

Framing: At the transmitter end, data link layer gets data to be sent from the above layer appends headers, CRC and control bits to it. In our case the data to be sent is previously logged in SD-CARD (satellite) and in files (GS). Data link layer retrieves this data, converts them in to packets and stores them in local memory so that the physical layer can directly access this packets and transmit them during the communication window.

Flow and Error Control: Acknowledgement based technique is being used to provide flow control and error control for the data transmission in both uplink and downlink. We have used Go-Back-N(GBN) ARQ for the uplink protocol(Section 4.5.8) and modified checkpoint and negative acknowledgement based protocol for downlink (Section 4.5.9).

Network and Application Layer

Network and Application layer is concerned with the following:

Tele-Command Handling: Network Layer deals with the tele-command handling of the communication system, some of them are mentioned below:

- On the satellite, processing the tele-commands received from the ground station and taking the necessary action.

- Depending on the health data received from the satellite on ground station, analyzing them and sending the control tele-command for immediate action on satellite.
- For certain payload operations, uploading the schedule from ground station using tele-commands.

Payload Data Handling: Maintaining the payload data on satellite and sending it in correct format so that they can be interpreted at the ground station.

Communication Window Connection Control: In LEO satellites communication *link management* plays a very important role. Network layer monitors the link and controls the operation of the satellite when in communication window. Since we are using adaptive packet length and adaptive baud rate features in our protocol it is the responsibility of the network layer to take the control actions depending on the link performance.

4.5.2 Error Control

Forward error correcting schemes can be used to correct the errors in packets thus avoiding the retransmissions; but, these schemes are useful only if propagation delay is long and computation power is not a concern as such schemes come with lots of redundancy overhead and encoding decoding computation overhead. However in our case because of small negligible RTT (Round Trip Time) and low computation power, highly reliable data link control protocol with ARQ which uses only error detection is used.

Error control on the satellite is implemented by using Automatic Repeat Request method. Packets with error are just discarded at the receiver end in the uplink

protocol while a NACK is sent corresponding to that packet in the case of downlink protocol. Error detection is done using 32 - bit CRC (Cyclic Redundancy Check) which is applied on the data-length as well as the data-region part of the packet (packet format is described in Section 4.5.3). Table Method for calculating CRC is being used, application report [25] from Texas Instruments is used as reference for implementation.

4.5.3 Packet Formats

Two types of frames are used in the protocol implementation, C-frame (Control frame) and I-Frame (Information frame).

Packet Format for Control Command Frames

C-frames are the frames used for link management and critical satellite control; these frames are also used as control frames in DLC protocol. Control frames are basically used during operations like RLIP, Change baud rate process, Link closure process, etc.Since these frames are control frames, there are no sequence number and ACK fields.

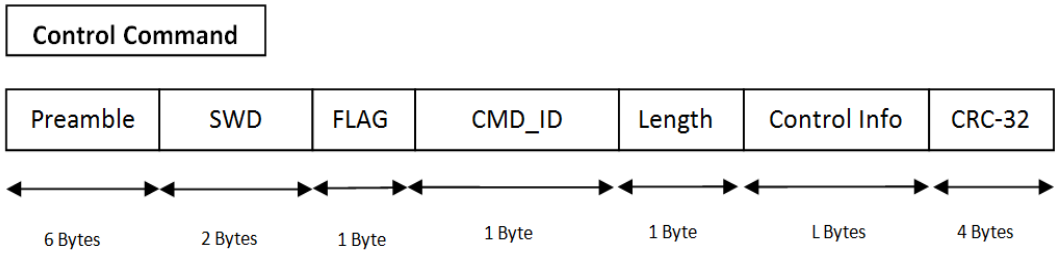

Figure 4.6: *Control command packet format*

Preamble: Its a 48 bit sequence transmitted before each packet to synchronize the receiver clock with the transmitter clock. 6 times "0xAA" is being transmitted.

SWD: Sync Word Detect is special sequence of bits transmitted before each packet indicating the start of a new packet. SWD is a special bit pattern which should not repeat in any other part of the packet; hence, *Bit Stuffing*[2] is being used. On reception of a SWD receiver generates a pulse on one of its pin which is used by the micro-controller to detect arrival of a new packet. We have used 0xACF0 as our SWD.

Flag: Flag is used to distinguish between C-frames and I-frames. Flag is transmitted just as a delay before actual packet transmission starts. This is being added as a padding so that even if the micro-controller fails to respond to the SWD immediately (8 bit time delay *flag*) it would still get ready to receive the actual data packet to follow. "0x0F" is transmitted as flag in the case of C-frames.

CMD_ID: Each control command has a ID corresponding to it which is used to process the command received.

Length: Length of the control-info field in bytes.

Control-Info: Control commands may carry control information with them, for example CP packets have error list in the control info field.

Packet Format for Uplink and Downlink

Difference between packet formats of the uplink and downlink protocol is the *ACK-Num* field. In the case of uplink protocol we are using *piggy backing* [26] technique for sending the acknowledgements to ground station. Hence the acknowledgements

[2]**Bit Stuffing**: Bit stuffing is the insertion of extra bits in the packet so that the receiver does not mistake the pattern 0xACF0 as a SWD when it is occurring in fields other than SWD field.

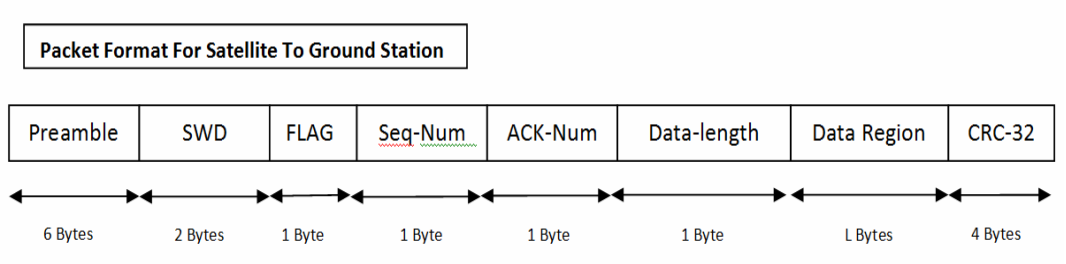

Figure 4.7: *Satellite to GS packet format*

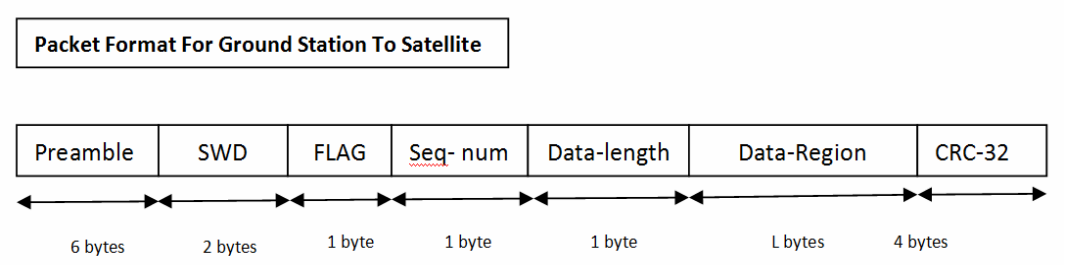

Figure 4.8: *Ground station to satellite packet format*

are piggy backed in the packets transmitted to ground station from satellite. In the case of downlink protocol piggy backing is not used. Instead, separate *Check Point packets* are transmitted which provide periodic Negative Acknowledgements.

Preamble: Its a 48 bit sequence transmitted before each packet to synchronize the receiver clock with the transmitter clock. 6 times "0xAA" is being transmitted.

SWD: Sync Word Detect is special sequence of bits transmitted before each packet indicating the start of a new packet. SWD is a special bit pattern which should not repeat in any other part of the packet hence *bit stuffing* is being used. On reception of a SWD receiver generates a pulse on one of its pin which is used by the micro-controller to detect arrival of a new packet. We have used "0xACF0" as our SWD.

Flag: Flag is used to distinguish between C-frames and I-frames. Flag is transmitted just as a delay before actual packet transmission starts. This is being

added as a padding so that even if the micro-controller fails to respond to the SWD immediately (8 bit time delay *flag*) it would still get ready to receive the actual data packet to follow. "0x00" is transmitted as flag in the case of I-frames.

Seq-Num: Sequence number of the packet transmitted. Used for in sequence assembly of packets and data reliability.

ACK-Num: Sequence number of the last successfully received packet from Ground Station to satellite is being sent in the form of cumulative positive acknowledgement.

Data-Length: Length of the data region to follow (number of bytes in the data region).

Data-Region: Actual data that needs to be transmitted from satellite to ground station.

CRC - 32: 32 bit CRC is used for error detection. CRC is calculated on the Data-Length + Data-Region part of the packet.

From the packet formats given in Figure 4.7 and in Figure 4.8 we can see that one packet transmission involves transmitting additional bytes along with the actual data that needs to be transmitted. These additional bytes act as an overhead when transmitting data in the form of packets. Overhead bytes play an important role while determining the packet length.

Table 4.7: *Packet overhead bytes*

Overhead Fields	**Bytes**
Preamble	6
SWD	2
Flag	1
Seq-Num	1
ACK-Num	1
Data Length	1
CRC -32	4
Control Info in data region	4
Total	**20**

4.5.4 Beacon

Beacon Signal

Beacon signal will be sent by satellite in the form of Morse code. This beacon signal will be received by all the amateur radio communities all over the world. Health status data of the satellite in quantized form is also by the satellite through beacon signal.

Morse Code Details

Morse code is a type of encoding technique used for transmitting information in the form of rhythm [27]. The Morse code uses a standard form of long and short

elements that represent alphabets, numbers and some special characters. They are commonly referred to as “dots” and “dashes” or “dits” and “dahs”. Time elapsed between the signals conveyed plays an important role in determining the meaning of the code.

Spacing and Length of Signals:

1. A dot (.) is the basic unit being transmitted. Signal is being transmitted during this interval time.
2. A dash (-) is equal to three consecutive dots (.).
3. The space between the signals forming the same letter is equal to one dot and no signal is transmitted during this interval.
4. The space between two letters is equal to three dots and no signal is transmitted during this interval.

Frequency Details

Beacon signal will be transmitted at a frequency of 12 units[3]/sec.

Beacon Packet Format

Call_Sign	Power Board 1	Power Board 2	Temp. Structure	Temp. Internal	Device_Status	Mode	B-dot value
X char	1 char	1 char	1 char	1 char	3 char	1 char	2 char

Figure 4.9: *Beacon packet format*

Call_Sign: Call sign ”*unknown*”[4] will be transmitted by satellite in Morse code.

[3]One unit is considered to be equal to one dot(.)

[4]To be given by IARU (International Amateur Radio Union)

Size of call sign is 'X' characters. Each satellite has unique call sign, its a combination of letters used to identify a broadcasting station.

Power-Board 1: Power status of the battery pack - 1 will be transmitted. This value will be quantized into 26 different states i.e. character 'A' - 'Z'.

Power-Board 2: Power status of the battery pack - 2 will be transmitted. This value will be quantized into 26 different states i.e. character 'A' - 'Z'.

Table 4.8: *Power status quantized values*

Power Range(Watt-Hr)	**Symbol Used**
0 - 0.65	A
0.65 - 1.3	B
1.3 - 1.95	C
1.95 - 2.6	D
2.6 - 3.25	E
3.25 - 3.9	F
3.9 - 4.55	G
4.55 - 5.2	H
5.2 - 5.85	I
5.85 - 6.5	J
6.5 - 7.15	K
7.15 - 7.8	L
7.8 - 8.45	M
8.45 - 9.1	N

Table 4.8 continues on next page

Table 4.8 continued from previous page

Power Range(Watt-Hr)	Symbol Used
9.1 - 9.75	O
9.75 - 10.4	P
10.4 - 11.05	Q
11.05 - 11.7	R
11.7 - 12.35	S
12.35 - 13	T
13 - 13.65	U
13.65 - 14.3	V
14.3 - 14.95	W
14.95 - 15.6	X
15.6 - 16.25	Y
16.25 - 16.9	Z

Temperature_Structure: Average value of the structure temperature will be transmitted. There are two temperature sensors on the structure at opposite faces such that atleast one of them will be facing the sun at any instant of time. Average value of these two temperature sensors will be quantized into 26 states represented by characters 'A' - 'Z'.

Temperature_Internal: Average value of the temperature of all the PCBs will be transmitted which will give information regarding the internal temperature of the satellite. There are temperature sensors on each of the critical boards of

the satellite. This average value will be quantized into 26 states represented by characters ‘A’ - ‘Z’.

Table 4.9: *Temperature quantized values*

Temperature Range(°C)	**Symbol Used**
-20 - -17	A
-17 - -14	B
-14 - -11	C
-11 - -8	D
-8 - -5	E
-5 - -2	F
-2 - 1	G
1 - 4	H
4 - 7	I
7 - 10	J
10 - 13	K
13 - 16	L
16 - 19	M
19 - 22	N
22 - 25	O
25 - 28	P
28 - 31	Q
31 - 34	R
34 - 37	S

Table 4.9 continues on next page

Table 4.9 continued from previous page

Temperature Range(°C)	Symbol Used
37 - 40	T
40 - 43	U
43 - 46	V
46 - 49	W
49 - 52	X
52 - 55	Y
55 - 58	Z

Device_Status: Status of all the critical devices on the satellite is being transmitted. Status of all the devices will be transmitted using 12 bit hexadecimal value. Three characters will be transmitted to represent the 12 bits and each device will be represented by a bit position. '1' received at that bit position means that the device is working fine and is "on" while a '0' being received means that the device is "off" or not responding properly. So a failure of device can be detected by checking the mode of the satellite and if in that mode that particular device should have been "on" and is not, indicates that the device has failed to respond. Status of the solar panels is represented using 2 bits; hence, it can be divided in to 4 different states or levels of degradation.

Table 4.10: *Devices list*

Device	Bit

Table 4.10 continues on next page

Table 4.10 continued from previous page

Device	Bit
Reaction Wheels	Bit 0
Torque Coils	Bit 1
Solar Panels	Bit 2
Solar Panels	Bit 3
Imaging	Bit 4
Magnetometer	Bit 5
GPS	Bit 6
IMU	Bit 7
Receiver	Bit 8
Transmitter	Bit 9
Temperature Sensors	Bit 10
OBC -2	Bit 11

Mode: Satellite will operate in different modes depending on the status of the different subsystems of the satellite.

Table 4.11: *Satellite modes*

Satellite Modes	Symbol Used
Emergency Mode	A
Communication Tracking Mode	B
Power Save (Shadow Region) Mode	C

Table 4.11 continues on next page

Table 4.11 continued from previous page

Satellite Modes	Symbol Used
Imaging Mode	D
Sun Pointing Idling Mode	E
Maneuvering Mode	F
Initialization Mode	G
Detumbling Mode	H
Momentum Dumping Mode	I
Satellite Cooling Mode	J
Reserved	K - Z

B_Dot Value: Magnitude of the B_Dot vector will be transmitted so that the angular velocity (stability) of the satellite can be determined at ground station. B_Dot values will be quantized and sent using 2 characters. Let rev1[5] and rev2[6] be the two characters received, they can take values from 'A' - 'Z'. Then the quantized B_Dot value is obtained by using the below expression.

$$Quantized\_Bdot = (rev1 - `A') \times 26 + (rev2 - `A') \tag{4.1}$$

Maximum value of magnitude of the bdot vector received is 'ZZ' and minimum value is 'AA'. 'ZZ' maps to value "X" gauss/s[7] and 'AA' maps to value "Y" gauss/s[8]. Exact bdot value is obtained by using the following expression.

$$Approx\_Bdot = Quantized\_Bdot \times (X - Y) \div 676 \tag{4.2}$$

[5]rev1 is the first character of B_Bot value received

[6]rev2 is the second character of B_Bot value received

[7]X = 1.5 gauss/s (maximum value for the satellite(tumbling))

[8]Y = 0 gauss/s (minimum value for the satellite(stable)

4.5.5 Variable Packet Length in Downlink Protocol

In LEO satellites, communication channel is subject to a lots of variations with time because of which the data link layer frame length plays an important role in the throughput efficiency of the channel. Determination of optimal frame length and the concept of variable frame length was introduced by Eung-In KIM, Jung-Ryun LEE and Dong-Ho CHO [28], proving that if the frame length of the data link layer is chosen adaptively in response to changes in the dynamically varying channel, maximum throughput could be achieved under both noisy and non-noisy error conditions.

We carried out simulations to determine the optimal length of packets for our satellite considering the varying nature of communication link at different elevation angles above earth. Hence satellite uses different packet lengths during different phases of the communication window.

Determination of Packet Length

The packet size has a lot of effect when we consider the Bit Error Rate and the Transmission Time required to download all the information.

In a clean channel; a long frame has some advantages in terms of quick transmission because of less Overhead per chunk of information transmitted coupled with less number of packets that needs to be transmitted. But in a channel with less SNR and high BER, a long packet is more likely to be corrupted, reducing the throughput drastically and, hence, a smaller packet size is required and advised.

So a very small packet will increase the overall number of overhead bits to be transferred (not good for clean or nearly clean channels) and a large packet will

require retransmission of packets since the probability of the packet being corrupted is very high; hence, a calculation for optimum Packet Size in different conditions is a must.

Simulation Channel's Model

The following assumptions have been made during the simulation of channel to estimate the Optimum Packet Size:

1. The calculations have been made for a baud rate of 10000bps.
2. Noise is considered to be AWGN (Additive white Gaussian Noise) noise.
3. Expected Worst SNR = 18dB.

BER Simulation Results

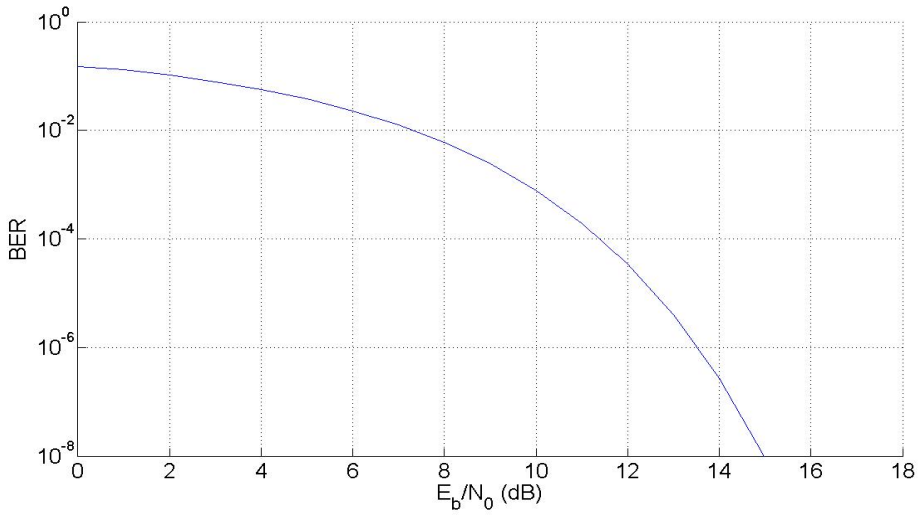

Figure 4.10: *BER vs SNR*

A simulation is done for various SNR. Each time, an analog BFSK signal is transmitted and decoded using correlated receiver. Figure 4.10 shows the plot of BER verses SNR for the given channel model. Results prove that improving the

SNR of the link reduces the BER, better the SNR less number of erroneous packets received. One of the ways of improving the SNR for a given transmitted power is by reducing the baud rate of the signal transmitted (Section 4.5.6).

Error Free Packets simulation

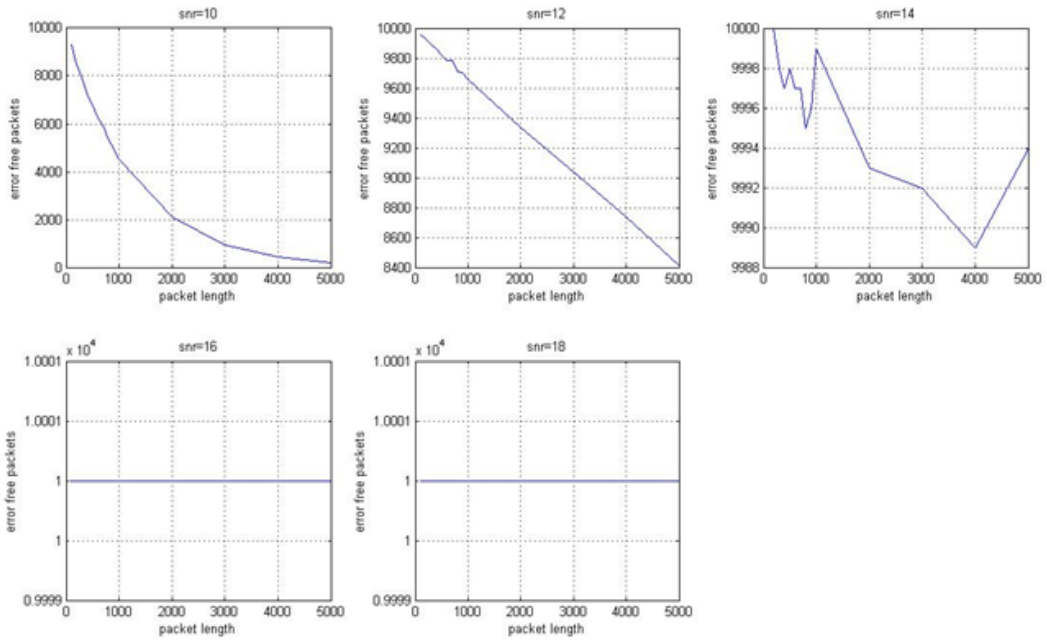

Figure 4.11: *Error free packets Vs packet length*

A Simulation is done for various SNR to observe number of error free packets versus packet length. Figure 4.11 shows the result of Error free packets versus Packet length for different SNR values. We can observe that at lower SNR value if the packet length is increased, number of error free packets received slowly drops to zero. Similarly at good SNR values packet length does not have any effect on the number of error free packets received.

Packet Length for BER = 1E-4

When the satellite is inside the communication window and in good visibility region, the expected BER will be around 1E-4 or better. For transmission of 20kB of data; the calculated time for different frame sizes and 1E-4 BER values is as follows:

Table 4.12: *20 KB data transmission time for BER = 1E-4*

Frame Length	**Data Length**	**No. Of Packets**	**Total Packets Transmitted**	**Duration**
64B	32B	6250	6586	337.203s
96B	64B	3125	3384	259.891s
160B	128B	1563	1792	229.376s
192B	160B	1250	1475	226.560s

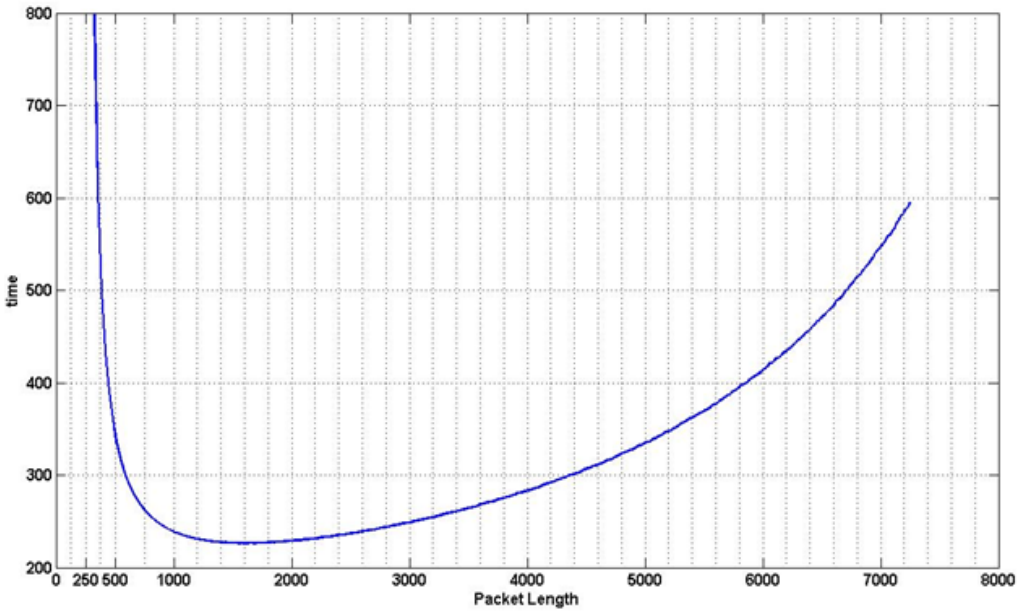

Figure 4.12: *Time vs packet length for BER=1e-4*

Figure 4.12 shows the time required to successfully transmit 20KB data for variable packet length at BER of 1E-4. We can observe that the transmission time increases as the packet length increases, indicating the increase in number of retransmissions.

Minimum Transmission Time = 226s

Packet Length = 1622 bits

Packet Size if BER = 1E-3

The BER might be a bit higher for sometime when the satellite is at lower elevation angles i.e., it is entering or exiting the communication window. Hence a comparatively smaller packet size will be required for such cases. The simulation of such a case presented the following graph.

For transmission of 20kB of data; the calculated time for different frame sizes is as follows:

Table 4.13: *20 KB data transmission time for BER = 1E-3*

Frame Length	**Data Length**	**No. Of Packets**	**Total Packets Transmitted**	**Duration**
64B=512	32B	6250	12800	655.4
96B=768	64B	3125	13453	1033.2

Figure 4.13 shows the time required to successfully transmit 20KB data for variable packet length at BER of 1E-3. We can observe that the transmission time increases as the packet length increases, indicating the increase in number of retransmissions.

Minimum Transmission Time = 655.0710 secs

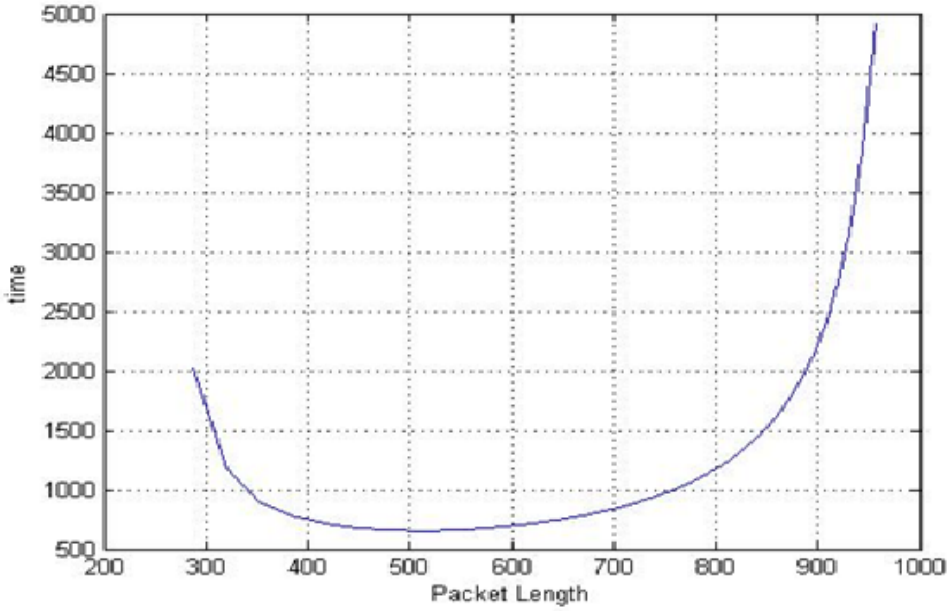

Figure 4.13: *Time vs packet length for BER=1E-3*

Packet Length = 505 bits

Packet Size if BER = 5e-4

For transmission of 20kB of data; the calculated time for different frame sizes is as follows:

Table 4.14: *20 KB data transmission time for BER = 5E-4*

Frame Length	**Data Length**	**No. Of Packets**	**Total Packets Transmitted**	**Duration**
64 Byte	32 Byte	6250	8396	429.9s
96 Byte	64 Byte	3125	5068	389.2s
160 Byte	128 Byte	1563	4332	554.4s
192 Byte	160 Byte	1250	5370	825.3s

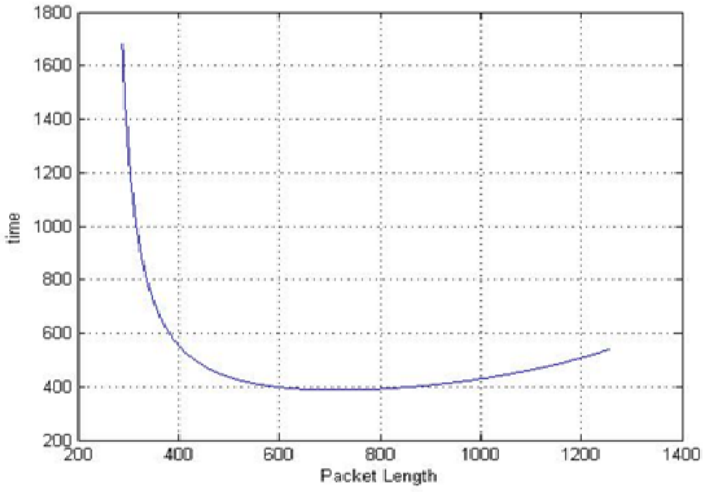

Figure 4.14: *Time vs packet length for BER=5e-4*

Figure 4.14 shows the time required to successfully transmit 20KB data for variable packet length at BER of 5E-4. We can observe that the transmission time increases as the packet length increases indicating the increase in number of retransmissions but packet length for minimum transmission time is greater than that for BER of 1E-3.

Minimum Transmission Time = 387.4519 secs

Packet Length = 720 bits

Conclusion

From the above analysis we can see that the performance of the channel decreases at lower BER if the packet length is increased. Hence the optimal packet length at a particular BER is one which takes minimum transmission time for sending 20KB data with minimum retransmission overhead.

For the case of 1E-4 BER, a packet length of 1622 bits is appropriate and for 1E-3 BER, length of 505 bits is appropriate.

Variable Packet Length

In Section 4.5.5 we carried out simulations to determine the optimal packet length for different BER values.

We can determine the performance of channel on the basis of number of erroneous packets received on GS or number of retransmission done by satellite. During the closing phase of the communication window, GS sends control frame to satellite which has information about number of erroneous packets received during the current communication window. Using this information performance of the link is calculated and if it less than LOW_BER_THRESHOLD then packet length is reduced by one level. If the performance is greater than HIGH_BER_THRESHOLD then the packet length is increased by one level. In short, the receiver feeds back BER and burst errors in the communication window to the transmitter and the transmitter will respond to this feed back by sending data in optimal packet length in the next pass.

We have not used adaptive packet length feature on our satellite in which depending on the current link performance packet length of the satellite is changed instantly during the transmission of data. Implementing such a feature on OBC would require it to have high computational power because when the satellite is in communication window there are various other critical task running on the OBC - 2. Hence to reduce the workload during communication window the data to be sent to ground station during the course of a particular communication window is already divided and converted in to packets and stored on SD-CARD. The conversion of data in to packets is done before each communication window and the packet length used depends on the performance of the link in the previous communication window.

Hence we can exploit the fact that the link behaviour in the two consecutive passes will be nearly same.

By default we have assumed that the value of BER in the initial phase (elevation angle of 0° - 20°) of the communication window will be 1e-3 and hence the packet size during this duration will be 512 bits i.e. the critical data will have packet size of 512 bits. We have also assumed that the satellite will be able to get a BER better than 1e-4 when its within the 70° cone. Hence, the payload data will be transferred using packet length of 1600 bits.

Table 4.15: *Packet length depending on channel performance*

Performance in Previous Communication Window	**Critical Data**	**Payload Data**
$\leq LOW\_BER\_THRESHOLD$	**512** bits	1024 bits
$\geq LOW\_BER\_THRESHOLD$ and $\leq HIGH\_BER\_THRESHOLD$	768 bits	**1600** bits
$\geq HIGH\_BER\_THRESHOLD$	1024 bits	2048 bits

Since the critical data will be transmitted during the initial phase of the communication window when the BER is assumed to be higher, packet length of the critical telemetry data is smaller. Payload data is transmitted with longer packet length to reduce the number of packets transmitted and increase the throughput.

4.5.6 Variable Baud Rate for Satellite Downlink

As we have noticed in Figure 4.10, as SNR of the link increases BER decreases. And also BER has a very high impact on the performance of the communication link if the packet length is fixed. One of the important factors that effect the SNR is the baud rate of the data transferred.

With every data rate is associated a certain bandwidth for that signal. The channel must be wide enough to reliably pass through that information, if it is too wide then we unnecessary waste the available bandwidth of the channel or if it is too narrow then the information symbols go through Inter Symbol Interference (ISI). Also due to a certain bandwidth of the channel there is inclusion of Nyquist/thermal noise proportional to the bandwidth of the channel which degrades the available SNR at the receiver and consequently the BER becomes worse. However if the channel is such that it utilizes just optimum bandwidth required to receive the signal corresponding a certain data rate then by reducing the data rate we can reduce the noise added and hence improve upon the BER.

Hence, depending on performance of the downlink channel proper baud rates can be selected. For example during the initial and ending phase of the communication window when the satellite is at lower elevation angle we will transmit signal at low baud rates to improve the SNR and also transmit the critical data at low baud rate so that it is successfully received at ground station. Once the critical data is transmitted, we will change the baud rate using the below mentioned protocol and the satellite will try to transmit the payload data at higher baud rates and by this time the satellite would have definitely come in good visibility region of the ground station so the SNR will be good enough to support higher baud rates.

Baud Rate Change During Communication Window

- Initially when the satellite is at an elevation angle between 0° - 10°, because of bad link margin and low SNR, the baud rate is maintained at 2400 bps.
- Once the critical data is transmitted, *change baud rate* process is started to increase the transmission baud rate to 9600 bps.
- Just before going in to the link closure phase, baud rate is again reduced down to 2400 bps.
- Whenever there is a LINK FAILURE, C-frames are always transmitted at 2400 bps. By default a LINK FAILURE causes the baud rate to be switched to 2400 bps.
- If the satellite fails to function at 9600 bps then payload data can be sent at 4800 bps.
- If there is a LINK FAILURE and the satellite is transmitting payload data then after link reinitialization baud rate is again switched to higher value of 9600 bps.

Performance Improvement

Changing the baud rate improves the SNR of the channel hence plays an important role when the link is bad and also throughput can be increased by changing the baud rate depending on the amount of data to be transmitted as well as channel performance.

Satellite Implementation

The commands sent and received during this process are in **C-frame** format.

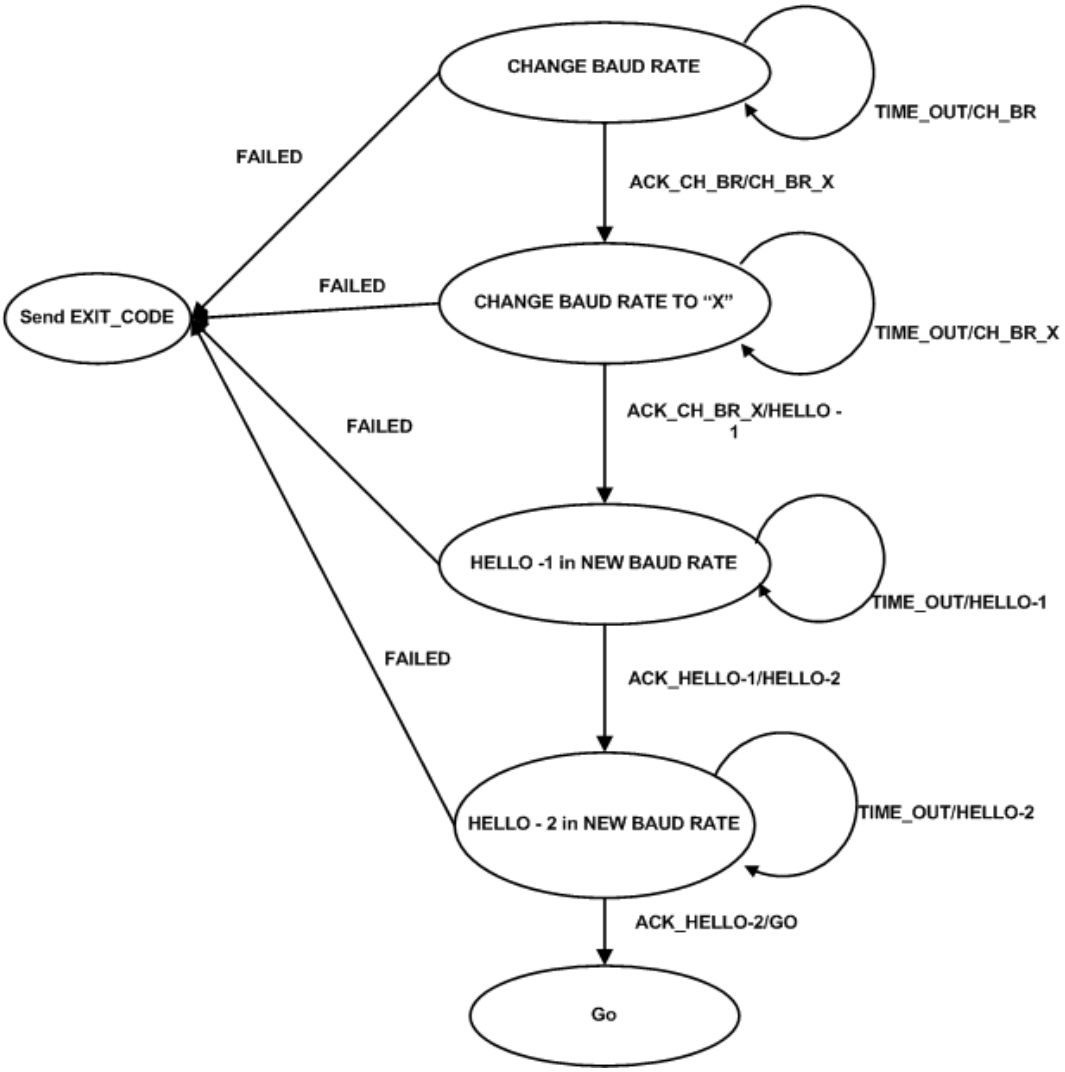

Figure 4.15: *Change baud rate protocol on satellite*

1. Change baud rate process is started by sending CH_BR command to GS by satellite continuously with time interval TIME_OUT[9] until it receives a ACK_CH_BR. After N[10] tries the process is marked as FAILED and exits by sending EXIT_CODE to GS. After transmitting EXIT_CODE and getting acknowledgement satellite starts transmitting data at 2400 bps (Default).

[9]TIME_OUT value is dependent on the response time of the channel

[10]N is a constant between 10 - 50

2. Satellite then sends CH_BR_X[11] to GS, indicating the baud rate to which it has to switch. CH_BR_X is transmitted continuously with some time interval TIME_OUT until it receives a ACK_CH_BR_X. After N tries the process is marked FAILED and exits by sending EXIT_CODE to GS. After transmitting EXIT_CODE and getting acknowledgement satellite starts transmitting data at 2400 bps (Default).

3. Satellite then configures the transmitter such that it can send data at new baud rate X. HELLO-1 command is transmitted to GS in new baud rate with some time interval TIME_OUT until it receives a ACK_HELLO-1. After N tries the process is marked as FAILED and exits by sending EXIT_CODE to GS. After transmitting EXIT_CODE and getting acknowledgement satellite starts transmitting data at 2400 bps (Default).

4. HELLO-2 command is transmitted to GS in new baud rate with some time interval TIME_OUT until it receives a ACK_HELLO-2. After N tries the process is marked as FAILED and exits by sending EXIT_CODE to GS. After transmitting EXIT_CODE and getting acknowledgement satellite starts transmitting data at 2400 bps (Default).

5. After receiving ACK_HELLO-2 satellite is in GO state and resumes the data link control protocol in new baud rate.

6. Once the satellite transmits EXIT_CODE, just to recover the satellite communication from failure mode RLIP is performed.

[11] X = 2400/4800/9600 bps

Ground Station Implementation

The commands sent and received during this process are in C-frame format.

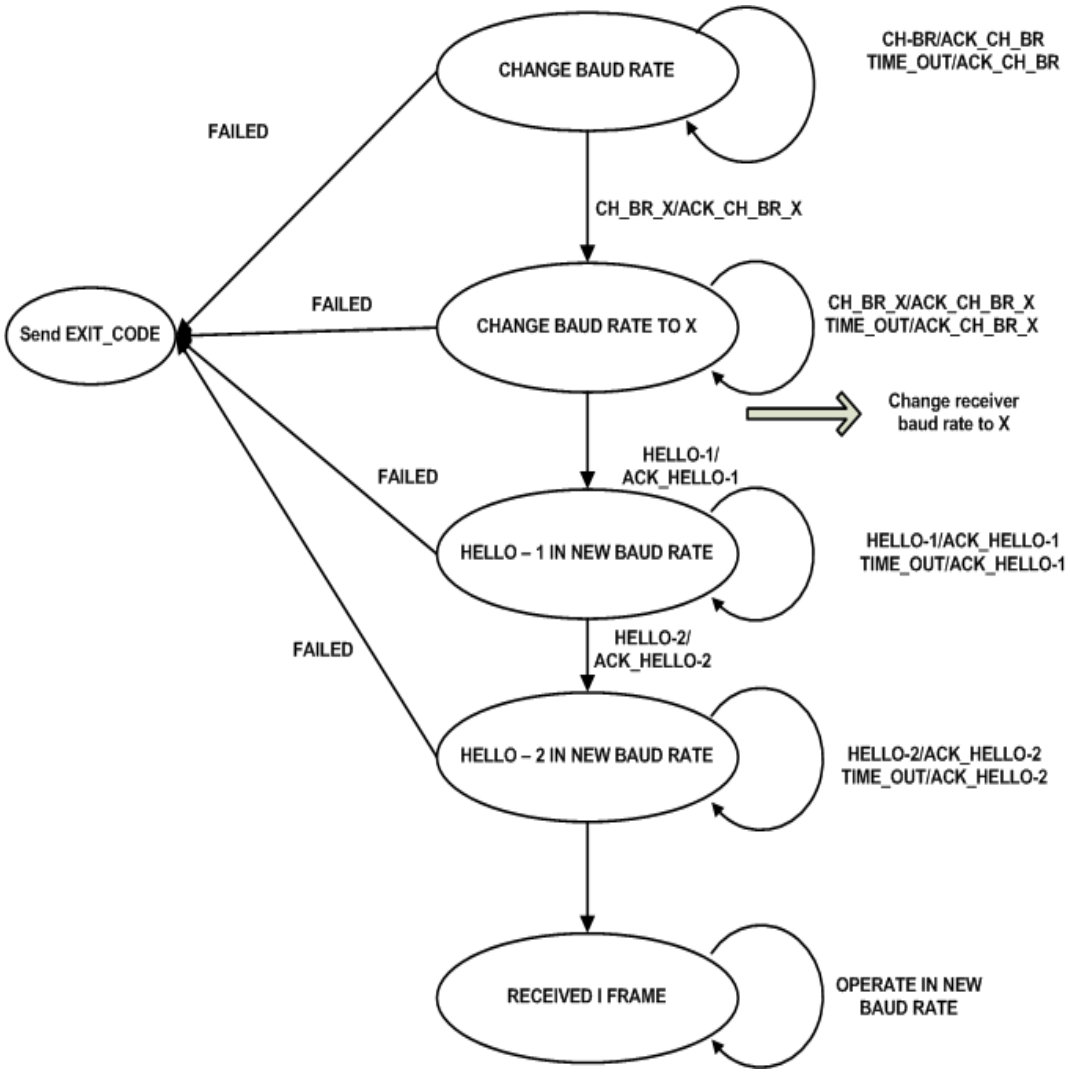

Figure 4.16: *Change baud rate protocol on ground station*

1. On receiving the CH_BR command GS starts sending ACK_CH_BR command continuously with some time interval TIME_OUT until CH_BR_X command is received. After N tries the GS detects FAILURE and configures the receiver to start receiving at 2400 bps.

2. On receiving the CH_BR_X command GS starts sending ACK_CH_BR_X com-

mand continuously with some time interval TIME_OUT until HELLO-1 command is received. GS configures the receiver to receive data at X baud rate. After N tries the GS detects FAILURE and configures the receiver to start receiving at 2400 bps (Default).

3. On receiving the HELLO-1 command GS starts sending ACK_HELLO-1 command continuously with some time interval TIME_OUT until HELLO-2 command is received. After N tries the GS detects FAILURE and configures the receiver to start receiving at 2400 bps (Default).

4. On receiving the HELLO-2 command GS starts sending ACK_HELLO-2 command continuously with some time interval TIME_OUT until I-frame is received. After N tries the GS detects FAILURE and configures the receiver to start receiving at 2400 bps (Default).

5. On receiving the EXIT_CODE GS sends ACK_EXIT_CODE and configures the receiver to receive data at 2400 bps (Default).

4.5.7 Phases in communication window

Communication Window of the satellite is divided in to different phases depending on the link conditions, the data to be sent and maintaining the reliability of the communication.

Reliable link initialization Protocol Phase: The link between ground station and satellite is being initialized during this phase. This protocol guarantees that the satellite is in the visibility region and the signal strength is good enough for data to be transferred. After the handshaking is done sequence

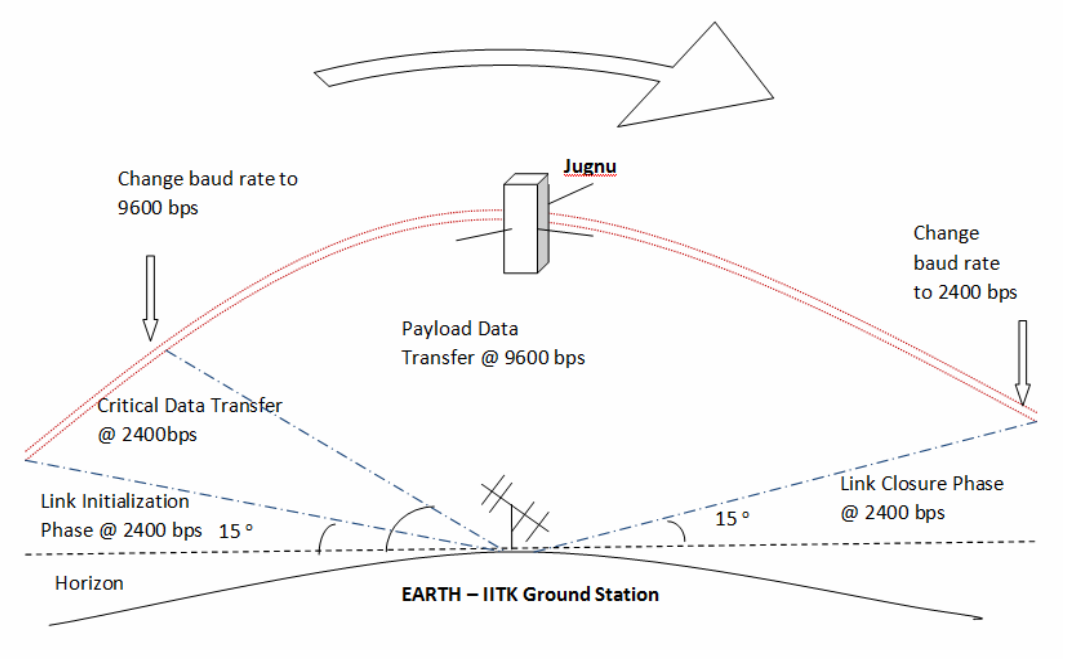

Figure 4.17: *Phases during the communication window of the satellite*

numbers and other default status variables of the protocol are initialized so that critical and payload data can be transfered reliably. Because of poor link margin initially the baud rate is kept low to improve SNR. Only control commands are exchanged during this phase and hence the packet size is small.

Packet Length : Small 32 - 36 Bytes

Baud rate : 2400 bps.

Critical Data Transmission Phase: Critical data transmission phase starts just after the link initialization is successful. In this phase of the communication window satellite will transmit the critical health data. Table 4.5 shows the critical data to be sent to ground station. During this phase of the communication window satellite is near the horizon and hence the link is comparatively weak and the BER will be near 1e-3. Hence during this phase for better data reliability and lower retransmissions we transmit the data at lower baud rate

and smaller packet size.

Packet Length : Small 64 Bytes

Baud rate : 2400 bps.

Payload Data Transmission Phase: Payload data transmission phase starts after the critical data have been transmitted successfully. By the time satellite have transmitted the critical data it will in the good visibility of the communication window and link margin, SNR will be better than 12 dB. Hence we increase the baud rate of downlink to 9600 or 4800 depending on performance of the link. After change baud rate process the satellite starts transmitting data from where it had stopped during the last payload data transmission phase.

In this phase of the communication window satellite will transmit the payload data. Table 4.6 shows the payload data and other noncritical health data to be transmitted to ground station. Downlink baud rate is increased during payload data transmission so that maximum data can be transmitted to GS in a single communication window. Also the packet size is increased to increase the throughput of the channel.

Packet Length : Medium 200 Bytes

Baud rate : 4800/9600 bps.

Satellite will autonomously calculate the start and end time of communication window using orbital propagation code. When the satellite goes out of the 75 degree visibility cone, it will start its closure phase and stop transmitting the payload data. The satellite and ground station will synchronize

their sequence numbers of packets sent and received successfully. Satellite will close the communication link by sending DISCONNECT command to ground station. DISCONNECT command can be sent by ground station to satellite or from satellite to ground station to inform the end of reliable 75 degree visibility cone. Once disconnected the satellite will again transmit critical data at lower baud rate and smaller packet size because of the same reasons as in link initialization phase.

Link Closure Phase: Packet Length : Small 64 Bytes

Baud rate : 2400 bps.

Reliable link initialization procedure

LEO satellite have property of periodic visibility; they keep moving in and out of the communication window which implies that the link between GS and the satellite is continuously established and broken. Hence we implemented link management protocol LAMSLM *(Low Altitude Multiple Satellite Link Management)*[29]. The LAMSLM [29] protocol is mainly designed for link management between multiple LEO satellites communicating with each other.

We modified the protocol idea and merged it with unbalanced type of *Reliable Link Initialization Procedure* (RLIP)[30]. Unbalanced because the link initialization is always started and controlled by satellite. LAMSLM uses a unique link closure mechanism at the conclusion of each active period that permits faster link reinitialization at a later point. The link closure mechanism is explained in Section 4.5.7, the link closure phase uses the same mechanism as in LAMSLM.

Since orbital propagation codes are running both on the satellite and the GS

with minimal error, prediction of start and close of the communication is possible and hence satellite can always initiate link initialization and link closure in synchronization with the GS.

Data reliability can be guaranteed by DLC protocols only if reliable link initialization is successfully completed before DLC protocol starts. The purpose of the Link initialization stage is to synchronize the two DLC processes, when the SUCCESS command is transmitted by the satellite, handshaking is performed between GS and satellite and the DLC parameters are exchanged in the Control-Info field.

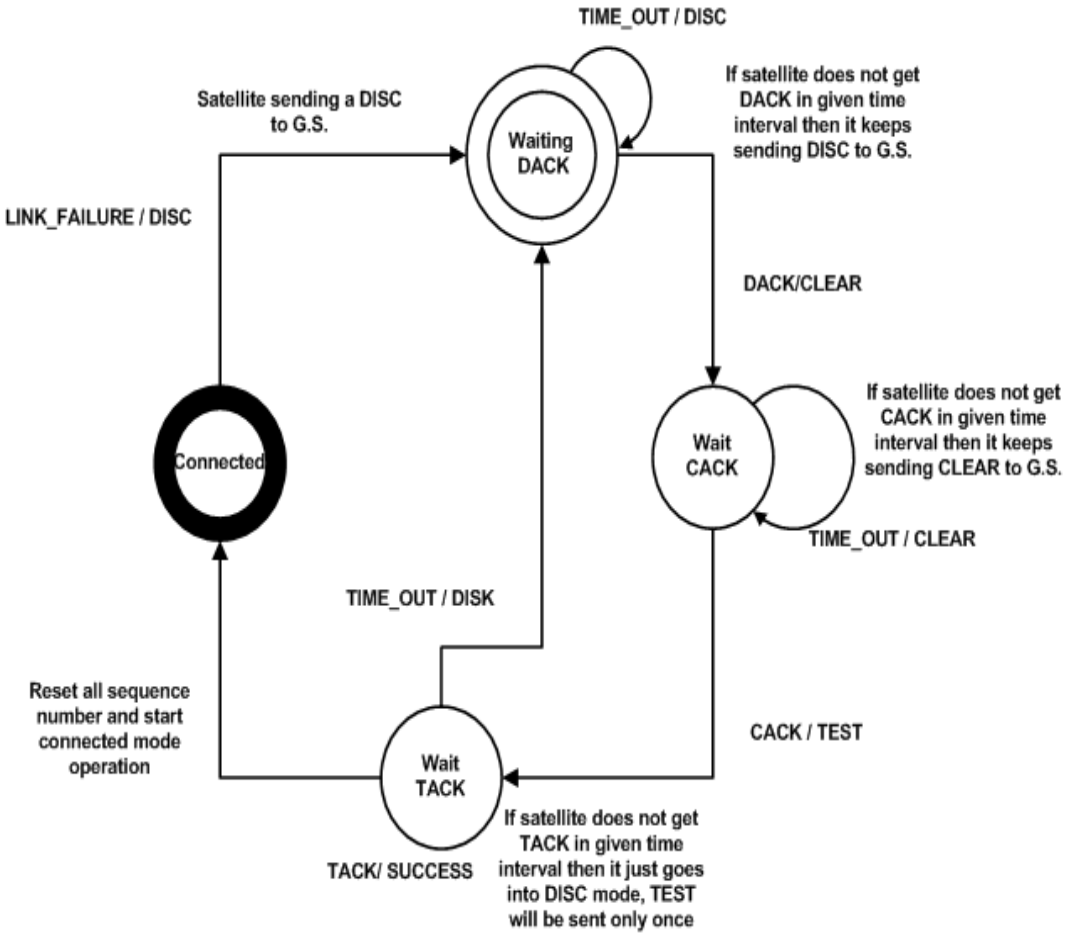

Figure 4.18: *Reliable link initialization protocol.*

Working of RLIP

All the control commands are transmitted in C-frame format; RLIP is not a part of the Data Link Control protocol. Hence, the control commands don't need sequence numbers. Working of the RLIP is as explained below:

- Once satellite and GS have detected the start of communication window (zero degree elevation) they will start transmitting PING command continuously. On receiving PING command, ACK_PING will be transmitted as a handshaking signal indicating that the satellite is in the visible region but with no guarantee about the link quality. Once the handshaking is finished it indicates that the target has been locked and RLIP can start.

- The satellite starts by transmitting DISC control frames at regular intervals using TIME_OUT until DACK control frame is received from GS.

- The satellite then sends CLEAR control frame to ground station at regular intervals using TIME_OUT until CACK control frame is received from GS.

- The satellite then transmits TEST control frame only once and waits for a TACK control frame in response. Once the timer is out the satellites goes back in to WAIT-DACK state and starts transmitting DISC control frame.

- If all the ACK control frames are received from GS within the TIME_OUT, then satellite transmit SUCCESS command and switches to connected mode and starts transmitting I-frames.

- TIME_OUT values don't have any effect on the performance of the RLIP.

Following Downlink Data Link Control parameters are exchanged between ground station and satellite during the RLIP phase:

1. newWindowStart: Pointer to the start of the downlink sender/receiver window.

2. newWindowEnd: Pointer to the end of the downlink sender/receiver window.

3. ErrorList: List of the erroneous packets in the current window, these packets are retransmitted.

No Uplink DLC parameters are transmitted for synchronization because once the link is broken all the incomplete uplink data is retransmitted.

Link Closure Phase

One of the advantages of link closure is that after the satellite is out of communication window no further packets are sent by satellite, hence lesser the loss of data more is the transmission power saved.

Also during the closing phase of the communication window, satellite will be at low elevation angles hence we need to transmit data at lower baud rate, Section 4.5.6 and also the packet size should be smaller, Section 4.5.5. Since before the communication window is started we have already formed the packets with fixed payload packet size and fixed critical data packet size(as mentioned in Table 4.15) therefore in the link closure phase we change the baud rate then perform the closure phase operation and then transmit the less critical satellite status data having smaller packet size.

After this phase the satellite stops transmitting the payload data and hence the payload data sender/receiver window parameters are synchronized by exchang-

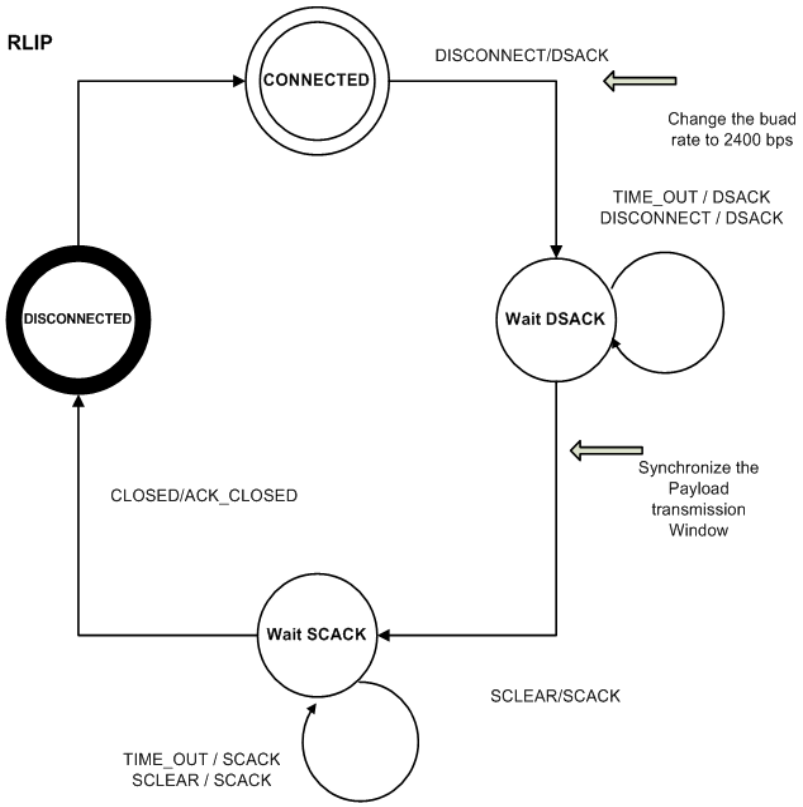

Figure 4.19: *Link closure phase*

ing the DLC parameters so that the payload data transmission phase of the next communication window starts with the same sender/receiver window parameters. If the link closure is done successfully then no window synchronization process is performed during RLIP.

Working of Link Closure Phase

1. On detection of the end of 75° communication window either the satellite or GS sends DISCONNECT command continuously until a DSACK commands is received.

2. Once the above event is done, baud rate of the downlink is changed to 2400 bps and then SCLEAR command is sent by the satellite continuously with TIME_OUT until the SCACK command is received from GS. Sender/ receiver

DLC window parameters are also synchronized during this step of the process.

3. Satellite then sends CLOSED command to GS continuously with TIME_OUT until ACK_CLOSED command is received.

4.5.8 Uplink Data link control protocol

Ground station will send the following data to satellite during the data transmission phase:

Control Telecommands: This control commands are sent from ground station to the satellite, controlling the operation of the satellite in orbit. These commands have highest priority and will be serviced by satellite immediately. These commands are transmitted using C-frames format 4.5.3. They may or may not carry control information in the control-info field with them. Example of such commands is DEVICE_OFF, used for turning "off" a particular device on satellite immediately or RESET to reset the satellite controller.

Configuration Data: Configuration data for configuring different subsystems of the satellite as well as the basic operation of the satellite may be controlled or improved. These data is sent in the form of I-frames format. Example of such configuration data is configuring the mode of operation of GPS, sending the co-ordinates of the target whose image is to be taken, system time synchronization.

CheckPoint packets: These acknowledgements are transmitted from GS to satellite as a part of the downlink protocol. Checkpoint packets are transmitted using C-frame format.

The I-frames are sent to the satellite using Go-Back-N ARQ sliding window protocol. From Figure 4.2, receiver is connected to OBC - 1 hence the uplink DLC protocol should as simple as possible with minimum memory buffer and computational power requirement.

Packet Length

Figure 4.8 shows the packet format for uplink, general configuration for the uplink protocol is as shown below:

Baud rate : 1200 bps

Packet Length of Telecommands: 32-36 bytes

Packet Length if Configuration Data: 64 bytes

Go-Back-N ARQ

Go-Back-N ARQ [26] simplifies the process at the receiver end. The receiver only have to keep track of the current packet to be received and there is no need for buffer at the receiver end. This protocol is ineffective in scenario where the link is noisy and BER is very low but in our case we have better link margin in the uplink and transmission power is also high hence number of retransmissions due to erroneous packets will be low.

The configuration data is sent from GS to satellite using Go-Back-N ARQ which is a special case of general sliding window protocol. We are using a sender window size of 5 and receiver window size of 1. Maximum sequence number is 255, acknowledgements are received from the satellite to GS using piggybacking technique. Since the downlink works at higher baud rate a window size of 5 will be enough for 100% link utilization of uplink. Since the sequence number can go up 255 we can increase the window size at ground station upto 255. The link utilization for Go-Back-N ARQ is:

$$a = \frac{Propagation\_time}{Transmission\_time} \tag{4.3}$$

$$u = \frac{(1-P)}{(1+2aP)} \qquad when\ n > 2a+1 \tag{4.4}$$

$$u = \frac{((1-P)\times N)}{((1+2a)\times(1-P+NP))} \qquad when\ n < 2a+1 \tag{4.5}$$

For satellite, P = 1E-4 (BER), N = 5 and a $\leq$ 1.8

4.5.9 Downlink Data link control protocol

Since LEO satellites have periodic visibility with continuously varying link performance when the satellite is in communication window, a lot of time is wasted in link management and packet retransmission. From the communication link specification and the transmitter power available we can consider the downlink to be more noisy than uplink and hence the downlink protocol should be such that it can handle noisy channel giving maximum throughput with minimum number of retransmissions. Go-Back-N ARQ is not used for downlink protocol because it is inefficient in noisy channel. Even a single erroneous packet will cause retransmission of multiple frames which reduces the performance of Go-Back-N ARQ. Selective Repeat ARQ protocols are suitable for noisy channels and sliding window protocols increase the link utilization.

Ward and Choi [31, 32] proposed a new link layer protocol, LAMS-DLC, suited to the low earth orbit environment, offering a reliable datagram service. The protocol is a negative acknowledgement(NAK) timer based checkpoint ARQ protocol, receiver sends periodic responses called *check point*(CP) commands having selective cumulative information regarding previous erroneous I-frames and is used to ensure that erroneous frames are properly retransmitted.

Also, transmitter on the satellite is connected to OBC -2 which has higher computational power (32-bit RISC, 48Mhz) and on-chip memory (512KB) which can be used as sender buffer hence we can go for more complex protocols.

Considering all the above factors, we combined the features of both sliding window selective repeat ARQ [26] and checkpoint ARQ [31, 32] to form DLC protocol which can be used for satellite downlink.

Satellite Downlink Protocol (SDP)

Satellite downlink protocol uses sliding window concept but instead of using periodic pos-ACK or NAK we use checkpoint based negative acknowledgements. This protocol increases the throughput efficiency with minimum complexity at the sender end and also has a separate mode for link failure detection.

The downlink protocol operation is split into two time intervals checkpoint interval and cumulative time interval as shown in Figure 4.20. After every checkpoint interval a CP command is transmitted by the ground station and then satellite starts sending the packets from the start of the window, resending the erroneous packets. Cumulative time interval is bigger than the burst error time and consists of many checkpoint intervals such that all the packets in the current sender window are transmitted successfully to ground station within a cumulative time.

We are using two types of frames in the protocol, I-frames are used for sending information data from satellite to GS and C-frames are used for CP packets, enforceCP packet, Request enforceCP packet. CP packets and enforceCP packets have information about the last successfully received packet as well as the list of sequence numbers of the packets which are discarded at GS due to error.

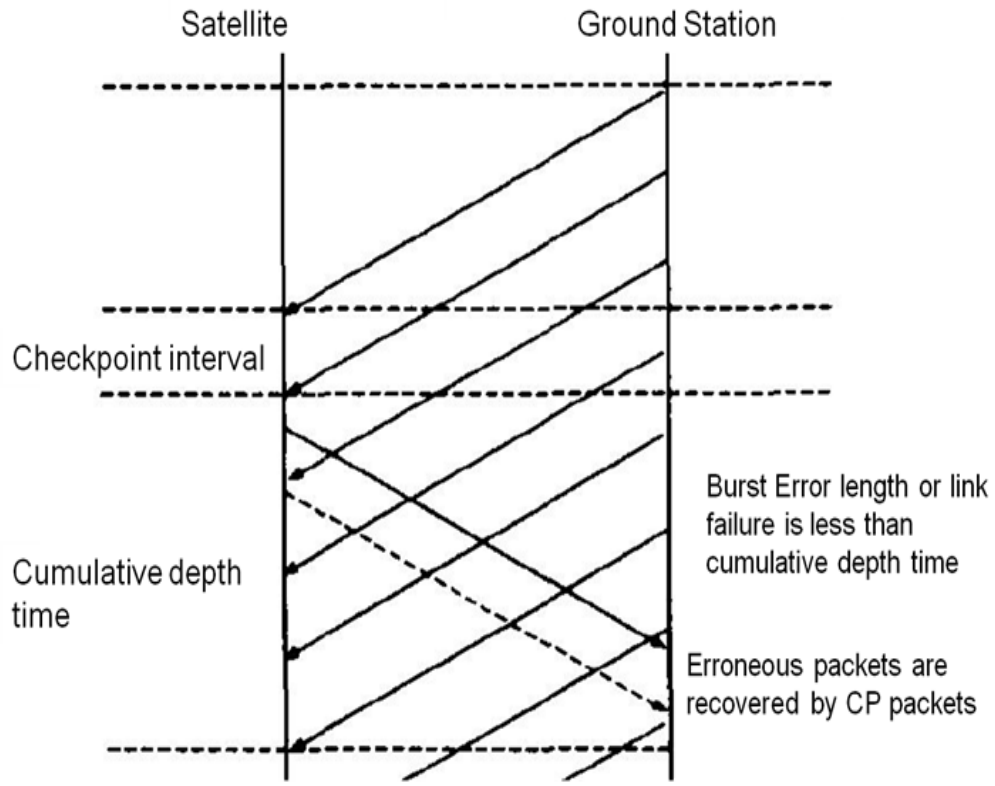

Figure 4.20: *Time Intervals in downlink protocol*

Sliding window protocol

For pipelining of packets in downlink data transmission, sliding window is maintained at the sender and receiver site. Sender and receiver have a window size of 128, sender maintains a buffer of 256 packets in flash. 8 bit sequence number is used in the protocol. Sender and receiver windows slide only after every cumulative time interval by amount which is dependent on the last successfully received packet sequence number. After every cumulative time interval the window slides till the position (lastSuccess[12] - 64) mod 256, hence it is enforced that the window slides till (lastSuccess - 64) mod 256 using clearbacklog mode. CP packets act as NAK and all the erroneous packets indicated in the CP packet are retransmitted each time a CP packet is received but the window slides forward only after the clearbacklog

[12]lastSucess is the sequence number of the last successfully received packet.

mode which occurs every cumulative time interval.

CP Frame Format

Control-Info field of CP packets carry information about the erroneous packets received at GS. 256 bits are used to transmit this NAK information, each bit corresponds to the packet with that sequence number in the sliding window. If there is a 1 at that bit position 'n' then the packet with sequence number 'n' was successfully received at the GS and a 0 indicates that the packet 'n' needs to be retransmitted. Along with this sequence number of the last successfully received packet at the GS is transmitted in the CP packet.

Format of enforceCP packet is same as that of CP packet with the only difference being the command Id.

Time Intervals

Two time intervals are used in the protocol to enforce data-reliability, checkpoint time interval[13] and cumulative depth time interval[14].

After every check point time interval, CP packet is transmitted from GS to satellite thus the error list is synchronized and the satellite then retransmits all the erroneous packets received till this check point time. After every cumulative time interval, sender and receiver operate in clear backlog mode and the window update procedure slides the window forward.

[13]Check point time interval $\approx$ transmission time of 10 packets

[14]Cumulative time interval $\approx$ transmission time of 120 packets

Functions Used

LoadPacket_SDCARD: This function loads the packet stored in SDCARD on to the buffer maintained in the internal flash of micro-controller. This function is called to initialize the sender window and after each slide window operation.

SendPacket(x): This function transmits the packet at position 'x' in the buffer. A buffer size of 256 packets is maintained at the sender size and the sender window slides over this buffer space.

initializeWindow: This function is called to initialize the sender/receiver window depending on the parameters synchronized during the last link closure procedure or RLIP. After this the sender and receiver windows are identical, ready to start the data transfer.

windowUpdate: This function is called to update the sender and receiver window after the backlog is cleared. The sender/receiver window slides forward from newWindowStart to the position of the first erroneous packet in the list and new packets are loaded from SDCARD onto the flash buffer.

clearBackLog: This function is called when cumulative timer expires. It makes sure that all the packets from newWindowStart till (lastSuccess - 64) mod 256 are successfully received at GS. During this mode of operation GS increases the sending frequency of CP packets.

enforeCP: *enforceCP* function sends Request-enforeCP packet continuously with a TIME_OUT interval until an enforeCP packet is received from GS. After receiving the enforceCP packet errorlist is updated.

SendCP/SendeCP: *sendCP* function send the CP packet and *sendeCP* function sends *enforeCP* packet to the satellite.

onPacketRev: This packet creates an event whenever a packet is received at GS and depending on whether the packet is successfully received or not the *errorlist* is updated.

Variables/Parameters Used

newWindowStart: Start position of the current window in the buffer.

newWindowEnd: End position of the current window in the buffer.

errorlist: Error list maintained using 256 bits where each bit represents the corresponding packet in the buffer. '1' at a bit position means that a valid packet is received with that sequence number.

oldWindowStart: Start position of the window before windowUpdate (slide window) function was executed. Used during the window initialize operation.

oldWindowEnd: End position of the window before windowUpdate (slide window) function was executed. Used during the window initialize operation.

lastSuccess: Sequence number of the last successfully received packet on Ground station.

Sender Process (Satellite)

After RLIP has successfully finished and the link is properly initialized, the following sender process as explained in flowchart in Figure 4.21 is executed:

STEP 1: Initialize the sender window, load packets from SDCARD onto the buffer in flash. Initialize all the Variables/Parameters of the protocol.

STEP 2: Start the cumulative depth interval timer and call windowUpdate procedure to update the current sender window. If there are any empty slots in the window, packets are loaded from SDCARD onto the flash.

STEP 3: Start sending packets from the newWindowStart till newWindowEnd position using errorlist, the packet corresponding to which there is a '0' in the errorlist is retransmitted.

EVENT 1: If CP packet is received then the errorlist and lastSuccess values are updated. Control returns back to the STEP 3.

EVENT 2: If cumulative depth timer expires then:

1. If no CP packet was received during the entire cumulative time interval then protocol goes into enforeCP mode and waits for a enforceCP packet. Once the enforceCP packet is received errorlist is updated and the protocol goes into clearbacklog mode. Once the backlog is cleared control goes back to STEP 2.
2. If CP packets were received during the cumulative time interval then the protocol goes into cleabacklog mode and once the backlog is cleared control goes back to STEP 2.
3. If in the enforeCP mode satellite fails to receive any enforeCP packet over a TIME_OUT interval then the DLC protocol declares the link to be broken and goes in to LINK FAILURE MODE.

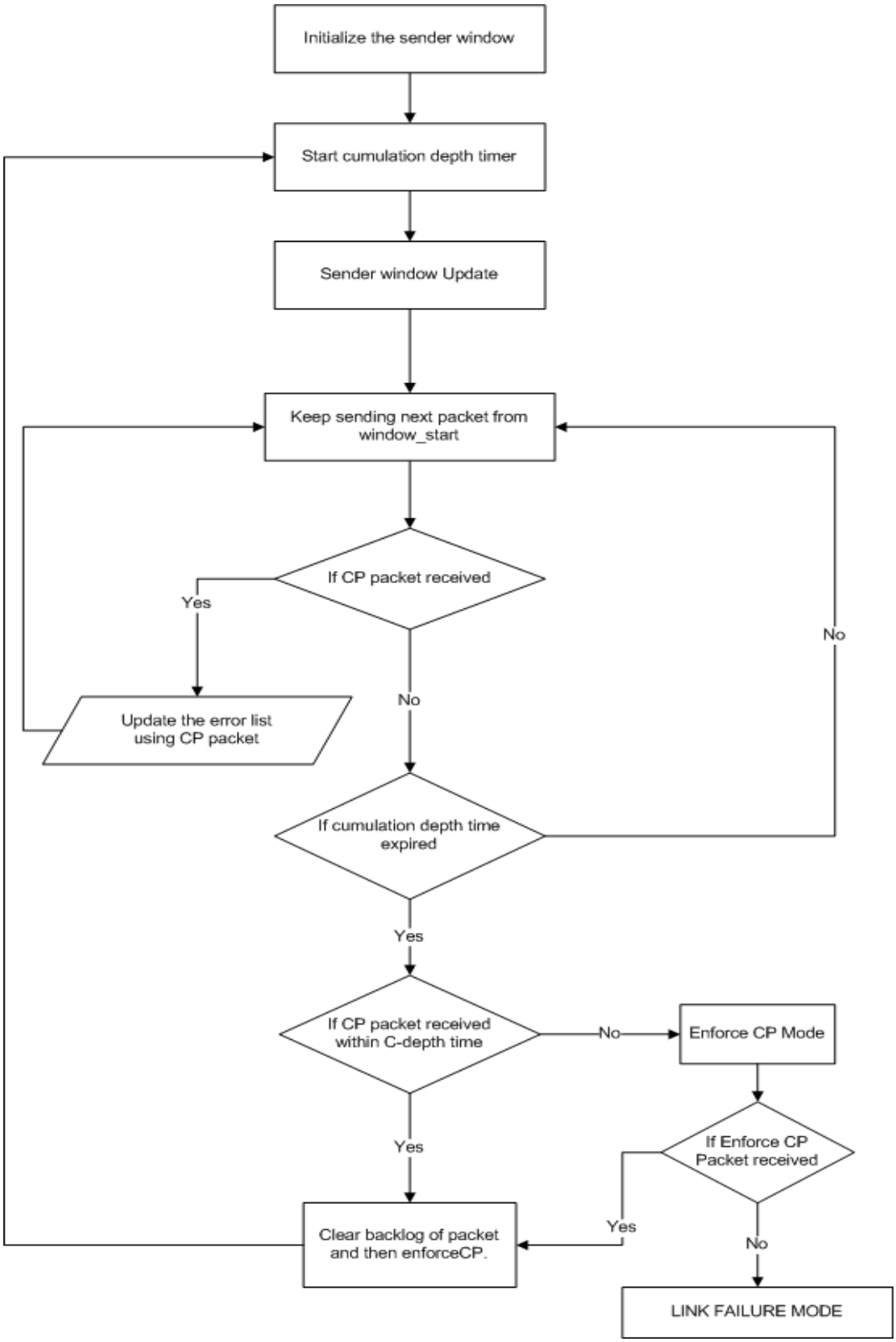

Figure 4.21: *Satellite sender process flow chart*

Receiver Process (Ground Station)

After RLIP has successfully finished and the link is properly initialized, the following receiver process as explained in flowchart in Figure 4.22 is executed:

STEP 1: Initialize the receiver window, load packets from 'GS file' onto the buffer if required. Initialize all the parameters mentioned in Variables/Parameters list.

STEP 2: Start the cumulative depth interval timer and call windowUpdate procedure to update the current receiver window. Packets from oldWindowStart till newWindowStart are logged on to a file on GS.

STEP 3: Keep receiving packets and process or log the packet depending on priority of the command Id. If the packet received is erroneous then the corresponding bit is updated in errorlist.

EVENT 1: If CP timer expires then latest errorlist and lastSuccess values are transmitted by ground station. Control returns back to the STEP 3.

EVENT 2: If cumulative depth timer expires then, If there is backlog to be cleared then protocol goes into clearbacklog mode and GS increases the sending frequency of CP packets. Once the backlog is cleared enforceCP mode is executed and the control returns back to STEP 2.

EVENT 3: If Request-enforceCP packet is received then GS keeps transmitting enforeCP packet until an I-frame is received.

EVENT 4: If no frames are received from satellite for a TIME_OUT[15] interval of

[15]TIME_OUT in this case is sufficiently large so that GS does not falsely declare link failure

time. Then the DLC protocol declares that the link is broken and goes into LINK FAILURE MODE.

4.5.10 Link Failure Mode

Link failure can occur because of many reasons like hardware or system errors, periodic long burst errors. If there is a link failure then before the DLC protocol can be restarted, link should be reinitialized.

Uplink failure can be determined if no CP packet is received over a certain TIME_OUT. The satellite then goes into forceCP mode and waits for the enforceCP packet. If satellite fails to receive any CP packet then DLC protocol declares that the link is broken and the satellite switches to lower baud rate of 2400 bps and starts the RLIP. If the RLIP also fails over N [16] number of tries, the satellite detects emergency, does a system check over the devices and starts transmitting the data in broadcast mode. If in the failure mode or emergency mode a packet is received from GS then satellite again tries the RLIP and reinitializes the link. After satellite is gone in to broadcast mode during the next RLIP all the parameters of sender/receiver window are initialized to default state (RESET condition).[17]

Downlink failure can be determined if no I-frames are received over a certain TIME_OUT on GS. Ground station then goes into forceCP mode and starts sending the enforceCP packet with higher frequency and waits for an I-frame. After a certain TIME_OUT then GS declares the downlink to be broken and configures the receiver to receive packets at 2400 bps. GS waits for the satellite to detect link failure and

[16]N is sufficiently large for the GS to detect link failure and switch the receiver to 2400bps

[17]RESET condition : Receiver has not received packets in its window and the error list is also all zeros indicating no erroneous packet received

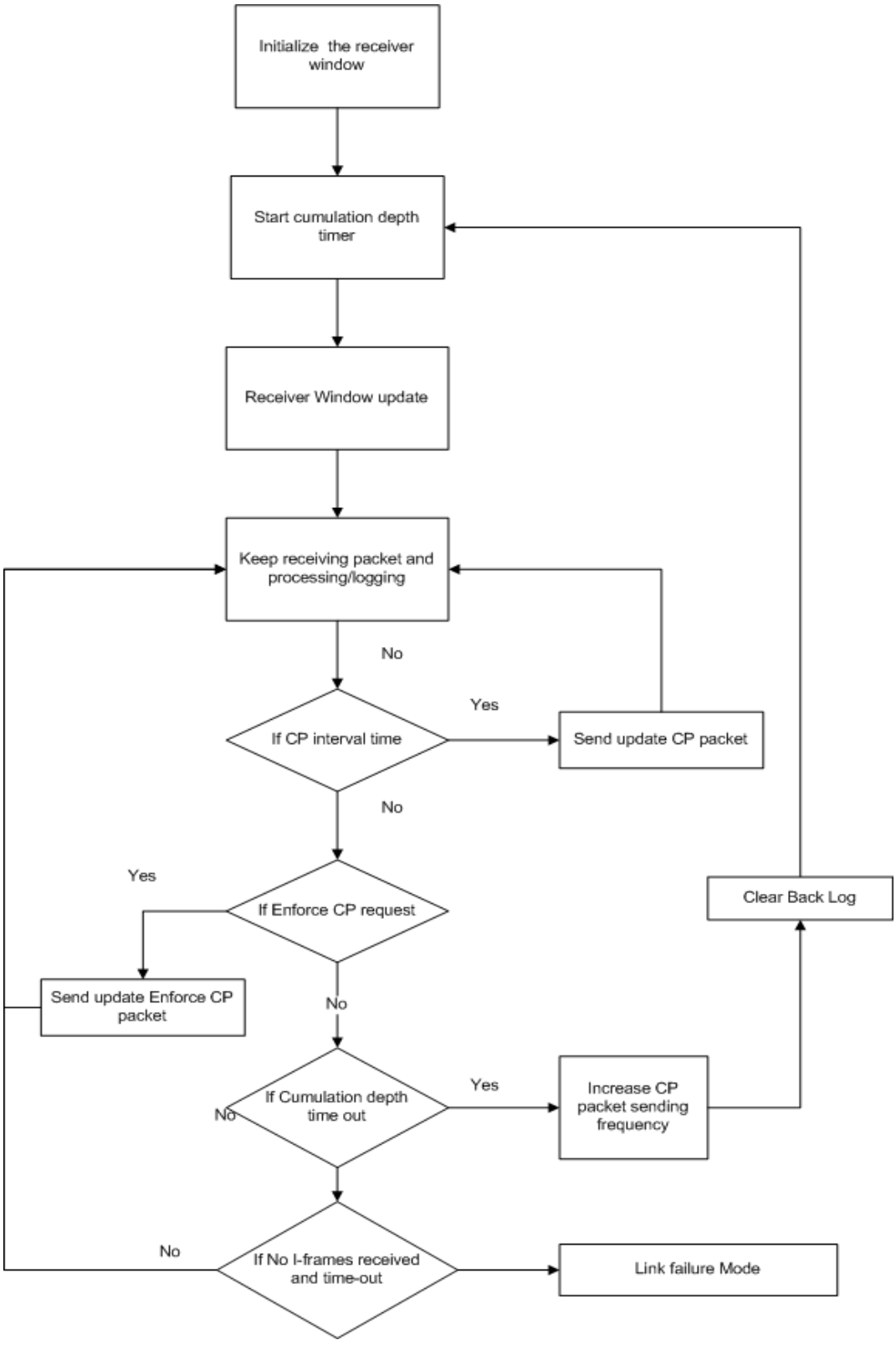

Figure 4.22: *Ground station receiver process flow chart*

start the RLIP. If in failure state no RLIP is initiated within a certain TIME_OUT, GS goes into emergency state. If in emergency mode GS receives a I-frame, it means that the satellite is broadcasting frames and hence GS keeps receiving and processing the frames as well as sending PING control packet to satellite.

If recovery from link failure was not successful in the previous communication window, then depending on whether satellite went into broadcasting mode or not the sender/receiver window are initialized to RESET state or normal state during the RLIP of the new communication window.

Chapter 5

Software Fault Tolerance

Software fault tolerance is essential because of the inability of the developers to produce error-free code. Its hard to guarantee bug free software design and its implementation. Also in space environment errors and faults may get added into the system, software design techniques should guarantee system operation even in the presence of such limited upsets at the device level.

It important that OBC is tolerant against single point failures, hence efforts are taken to make the OBC as robust as possible. Various software fault tolerance techniques [33] are being used for these reasons. There are two types of fault software fault tolerance techniques: single version and multi-version software techniques. Single version, improves the tolerance of single piece of code by adding techniques which can detect and handle errors caused due to design or device fault. Multi-version, improves the tolerance by using multiple versions of the same piece of software, this multiple versions are used to correct error in any one of the version. Section 5.1, Section 5.2 and Section 5.3 describe single-version based techniques used by OBC to detect and handle faults. Software triple modular redundancy is also implemented

on program and data memory of the satellite to provide fault tolerance against soft-errors caused due to radiations in space.

5.1 System Reset Based Mechanism

In case of errors which are non-recoverable, like normal task restart techniques fail to resume the normal operation then under such situations the crude RESET operation is performed. Such non-recoverable errors can be resolved by using various timer modules. The following timer reset modules are used :

5.1.1 External Watchdog Timer

Real Time Clock (RTC) M41T81S [15] is interfaced with both the OBCs on the satellite, it is used for maintaining the system time of the satellite. This module also has internal watchdog timer which generates an interrupt on one of the IO pins of OBC whenever the watchdog timer overflows. This interrupt can be used to generate software (SW) reset which will cause the satellite to resume normal operation starting with the initialization mode. Whenever the OBC is reset, TMR is implemented on the program memory which solves the soft error bugs while all the devices interface with OBC are reset and reconfigured which solves bugs because of device failure.

5.1.2 Internal Watchdog Timer

Both the OBCs have internal watchdog timer as well which is used for same recovery operations on the satellite. Diagnostic operations are performed by the satellite so that the cause of software fault can be detected before the processor is reset. Also

before resetting the satellite, backup of all the important state variables is taken. We have implemented self-test using diagnostic task based technique as explained by `Niall Murphy` in his article [34].

5.1.3 OBC-1 & OBC-2 Periodic Check

Also the reset pin of OBC-1 is connected to OBC-2 and reset pin of the OBC-2 is connected to OBC-1. Hence in situations where ISR based SW reset method fails to reset the processor then in such cases the other OBC can cause an external hardware reset. Both the OBCs periodically "ping" each other, if the ping is not receive within a certain TIME_OUT over multiple checks then the hanged OBC is reset. Thus if some software or hardware error causes OBC to hang and stop responding then it is resolved by reseting the OBC. There are three redundancies on the satellite for performing this operation as mentioned above. By default all the devices interfaced with OBC are in off-state and are turned "on" only after checking in the initialization mode. OBC goes into Initialization mode each time it is reset, hence resetting the OBC will switch "off" the faulty device interfaced to it and won't be turned "on" again.

5.2 Checkpoint and Recovery Based Mechanism

Along with a crude reset method of software fault tolerance, we also use the technique of checkpoint based system log and recovery [33]. We have created a diagnostic task in the OS of both OBCs which periodically logs the critical data like:

- Device health status which indicates working and failed devices.

- ADCS status variables
- Global variables which maintain the data handling on the satellite.
- Global variables which define the state of the satellite.

This critical data is periodically logged into a file on the SDCARD by the diagnostic task, if the processor is reset then these logged data is used to recover the satellite to its previous state of operation. This helps in maintaining the state of the satellite and the operations performed by payloads and other subsystems are still retained even if some hardware or software error causes the processor to reset.

5.3 Task Restart Based Mechanism

Diagnostic task also keeps track of the progress of all the tasks which are in ready queue. If over a certain number of scans diagnostic task finds that a particular task has hung then that particular task is restarted. This plays an important role in cases when the task has hung because of stack corruption or device failure etc. Stack memory corresponding to each task is located in RAM and corruption of this memory may cause the task to hang, this can be resolved by restarting the task.

5.4 Effect of Radiations in Space

Radiation effects in space on circuits are caused due to high-energy particles. Radiation environment is of concern for nanosatellites because of the use of COTS devices, COTS devices have low tolerance as compared to the radiation hardened devices used in small-satellites. Radiation effect can be divided into two categories:

Total ionizing doze: Total ionizing dose effects are caused due to absorbed charge or displacement damage caused by charged particles energy deposition within device structure. The total-dose tolerance of "commercial-off-the-shelf" devices is around 2-10 krad(Si), typical dose rates due to trapped electrons and protons is around 1 krad for LEO satellites [35]. Hence commercial-off-the-shelf components with typical total-dose values in the range of 2-10 krad can be used without any protection layers over them. Since lifetime of nanosatellites is around 3-12 months shielding is not required and device can survive total-dose effects.

Single event effects (SEE): Single event effects are the upsets that occur in digital ICs with small feature size, because of high rate of ionization produced by a ion passing through a sensitive node in the device [35]. SEE can cause device latchup called single event latchup (SEL) and the device may draw large amount of current within a short duration of time, damaging the device permanently if not powered off. SEE can also directly upset bits in memory or register of a device, such bit-flips are called soft-errors or single event upsets (SEU).

SEL problem can be handled by using current limiting circuits, if the device draw high current then the power supply to that device is turned "off" and the device is protected against permanent damage. Bit-flips (SEU) in RAM memory can be rectified by resetting the processor hence the error in the dynamic data in code segments in volatile memory is resolved. SEU in non-volatile memory like program flash and SDCARD can be removed by using triple modular redundancy, explained in more details in the following sections. Errors caused due to bit-flips in

the internal registers of the devices are removed by resetting and then reconfiguring those devices.

5.4.1 Triple Modular Redundancy (TMR)

A TMR system consists of three redundant copies of information. A voter module compares the three copies and then selects the value of majority. The TMR system is therefore designed to tolerate the failure of any single copy by only producing output on which at least two modules agree. This majority voted data is also written on the third copy in which the bit flip might have occurred. Triple Modular Redundancy, or TMR, is based on the assumption that the probability of an SEU-generated fault is finite, small and localized [36]. We have implemented software triple modular redundancy on SDCARD and program memory by creating three copies of data on the same memory and using software to compare values in the three copies and correct them using majority voting algorithm. The TMR model will be implemented at bit level on the memory by using following expression:

If A, B and C are three bytes on which majority voting algorithm needs to be implemented at bit level then the below expression gives the voted value which will be overwritten over all 3 copies.

$$Correct\_Value = ((A \oplus B) \cdot (A \oplus C)) \oplus C \tag{5.1}$$

5.5 TMR on Program Memory

Single event upsets can cause errors in the following memory segments:

1. Code and data segments resident in volatile memory (internal RAM).

2. Data stored in non-volatile memory (internal flash).

3. Program memory stored in the non-volatile memory (internal flash).

The result of this errors could be wrong output, wrong behaviour or even exceptions in cases where instruction gets converted to an unknown op-code for the processor. To avoid this scenario we implemented triple modular redundancy on the internal program memory. Figure 5.1 describes the memory structure after implementing TMR on it. Different sections of the memory are as described below:

RAM structure: Initial section of the internal RAM is reserved for "TMR implementation function". Whenever TMR needs to be implemented on the flash memory code is loaded from flash into this section and executed. For example whenever the Memory check Task explained in Section 3.4.1 and Section 3.4.2 detects bit-flip in program memory it stop the current execution, loads TMR code into this section and read/write operation is performed on the flash memory. Other parts of the RAM are divided in to stack (RAM section), initialized data (IDATA) and Uninitialized data (UDATA).

Interrupt vector: For the reliable operation of the interrupts, it is essential that there is no bit-flip in the Interrupt Vector Table (IVT). Hence three copies of IVT is also maintained in the internal flash memory, TMR is applied on it whenever a reset occurs. Also before resetting the microcontroller TMR is applied on the IVT inside the exception handler. So that after reset processor starts executing from the correct reset handler.

Code Segment: Three copies of code segment is also maintained in the flash memory, TMR is applied on this segment each time the processor resets.

CRC of program memory: To detect if there are any bit flips in the flash memory, CRC calculated on the entire flash memory is stored after each code segment. TMR is applied on this CRC bytes as well. Memory Check task explained in Section 3.4.1 and Section 3.4.2 uses this CRC to detect bit flips in the program memory.

Bootstrap Loader: This section of code resides at the start of code segment, whenever the microcontroller resets, reset handler calls the bootstrap loader which implements TMR on the flash memory and then returns the control to the reset handler which in the end calls the "main" function.

As explained in Section 5.1, reset can occur because of the three different modules, in each of the interrupt handlers we implement TMR on the bootstrap loader and interrupt vector so that after reset processor executes code from the correct reset handler. The implementation of bootstrap loader is such that on each reset the processor starts executing from next bootstrap loader, for example if initially bootstrap loader-1 is called then on next reset bootstrap loader-2 is called and after bootstrap loader-3 again bootstrap loader-1 is called and this sequence is followed. Thus processor will fail to recover only if all the three bootstrap loader gets corrupted. In processor idle time we keep syncing the redundant copies on flash with the image of code on the SD-CARD which will again reduce the probability of having bit flip in the redundant copies. Bit flips in RAM section of the memory can only be corrected by reseting the microcontroller, hence we do a planned reset of the microcontroller after every 6 hrs. This planned reset will also resolve problem which can occur because of bit-flips in the control registers of the microcontroller, no TMR can be applied on this control registers.

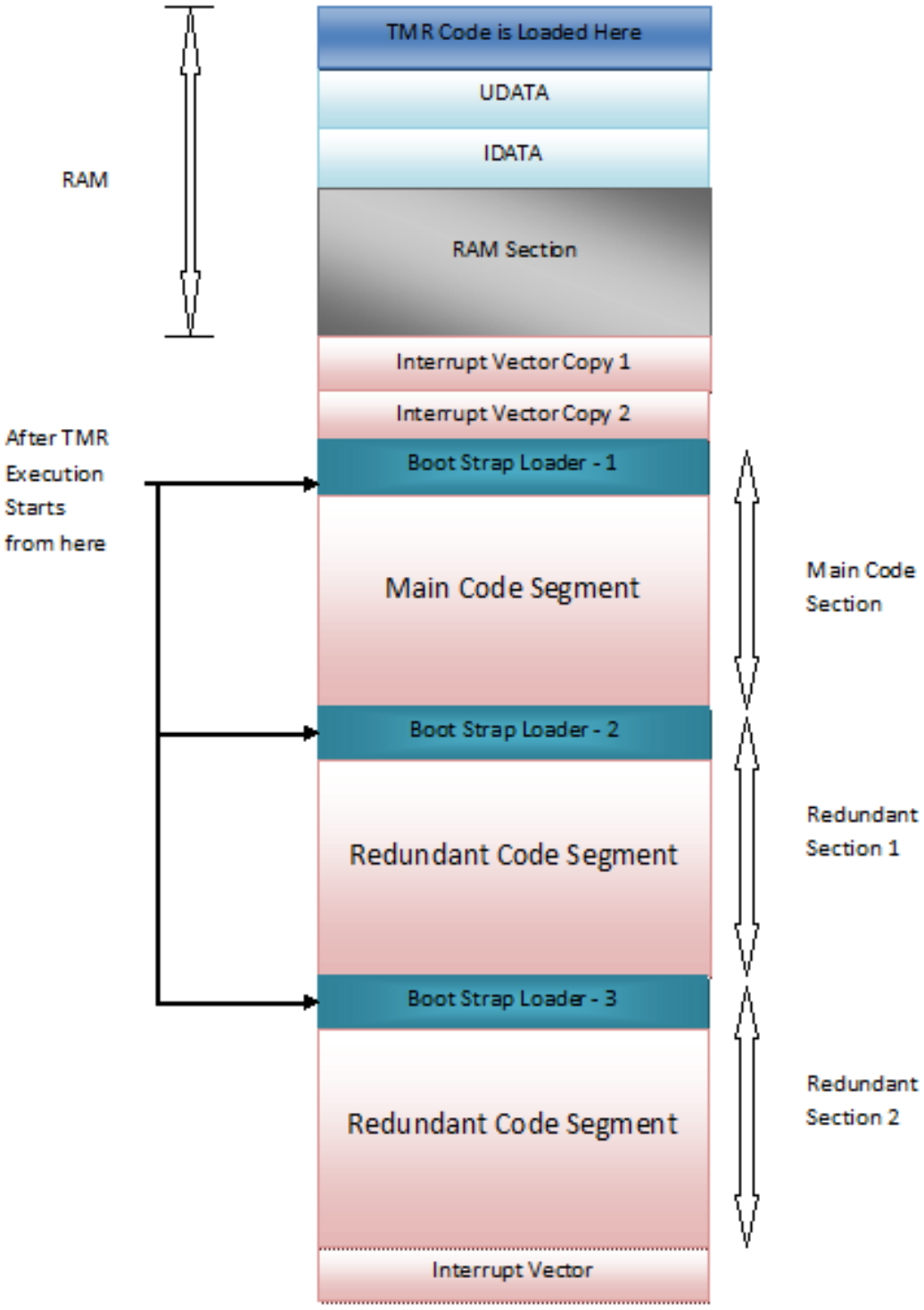

Figure 5.1: *TMR implementation on internal program memory.*

5.6 TMR on SDCARD

We are using SDCARD on satellite for storing system log, payload data, image of the program memory and other default state initialization data. This data is critical for the satellite, hence we implemented a filesystem which will apply TMR on this data each time it is read from the SDCARD. We need a file system which provides all the features of a normal file system along with being reliable to accidental bit flips. It will have features to detect an accidental bit flip, and will be able to self correct any such anomaly. As mentioned in Section 2.1.2 and Section 2.2.2 we are using FAT16 filesystem on SDCARD. To make this filesystem robust against soft error we have implemented a scheme which applies TMR both on the data in the SDCARD and filesystem on the SDCARD.

Figure 5.2 describes the structure of SDCARD after software TMR is implemented on it. Different sections in the SDCARD are as described below:

Filesystem: Two copies of filesystem structure which includes boot record, FAT structure and root directory is maintained at the end of the SDCARD. After every fixed interval of time TMR is applied on this filesystem structure to remove any bit-flips and guarantee normal operation of the filesystem. Filesystem can get corrupted because of SEU which is detected by filesystem APIs, for correcting this error TMR is applied on the filesystem structure.

Image of OBC Code: Three copies of OBC code are maintained on the SDCARD, each time before the code image from SDCARD is copied onto the flash memory, TMR is applied on the SDCARD copies. Image of OBC code is maintained on SDCARD so that it can be loaded on to the flash memory

periodically removing the bit-flips in redundant flash memory.

Data region: Data is stored on the SDCARD in the form of files. Three copies of the same file are maintained on the satellite, each time a read/write operation is performed on a data, these operations are performed on all the three copies. For example before reading data from a file TMR is applied on it and while writing data, same data is written on all the three files.

TMR implementation on the filesystem structure and Image of OBC code is performed not by using filesystem APIs but by using direct SDCARD memory read/write APIs.

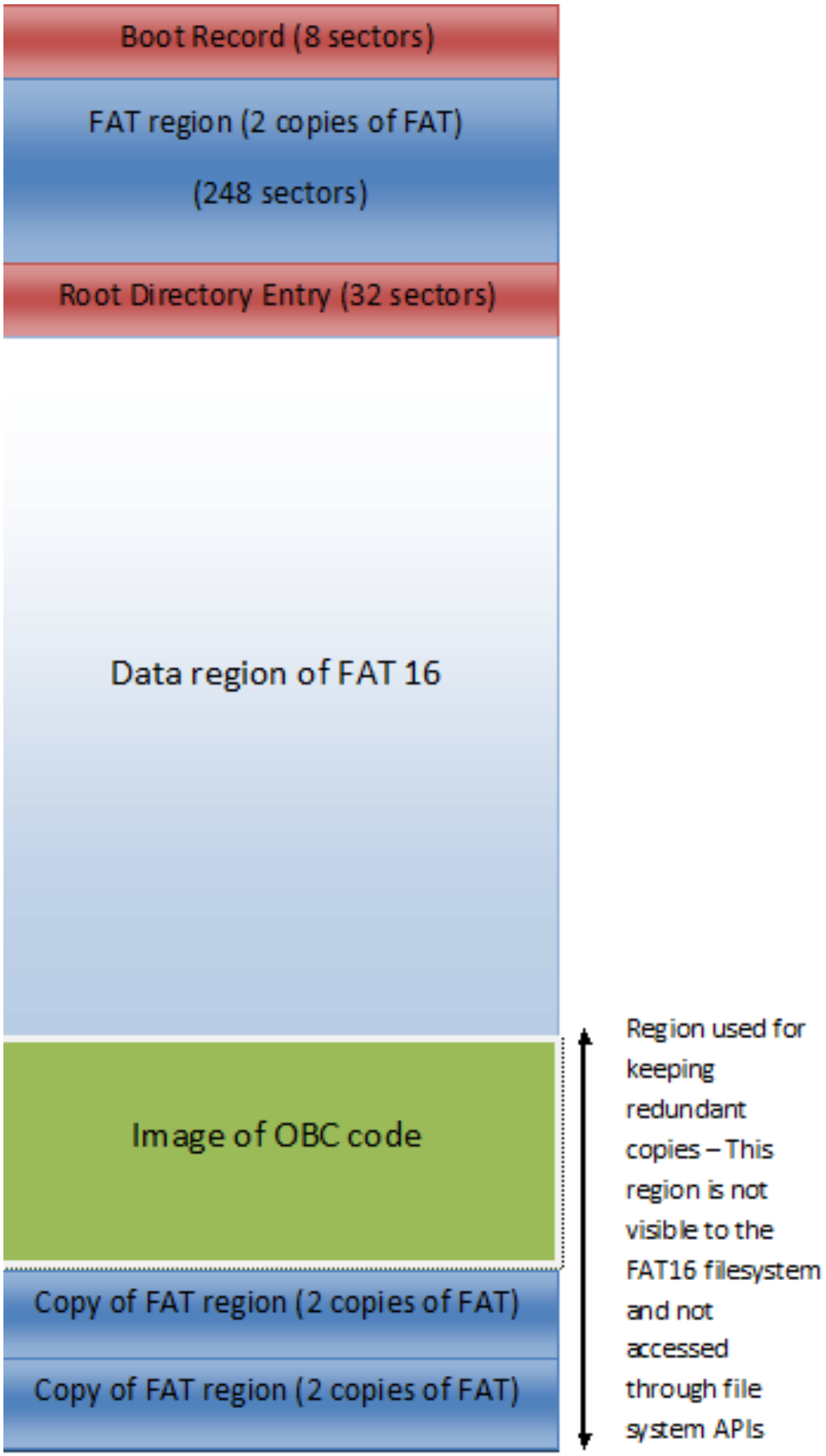

Figure 5.2: *TMR implementation on SDCARD.*

Chapter 6

Conclusions

For system software integration testing, we interfaced all the devices used on the satellite with on board computers and tested the device drivers. The software design of the system was tested by generating different unit test cases and checking on board computer's response. Communication protocol was tested on hardware and performance of the protocol was verified under different phases. All the control algorithms of the protocol like adaptive packet length and adaptive baud rate were tested separately. Also software fault tolerance was tested by inducing random bit-flips in the memory. On board computers could self-recover from such SEUs caused in both program memory and filesystem.

In this thesis, we have presented the design requirements and the development process for the on-board computer software for the nanosatellite JUGNU. The details of the communication protocol suitable for low earth orbit satellite, developed for reliable command and data transfer over the available communication channels have been discussed. The various methods developed to confer fault tolerance to the system have also been discussed. The analysis of these methods suggests that their

use will enhance the reliability of the system as a whole, and will allow the use of commercial grade electronics in the harsh space environment. The use of these methods also relaxes the constraints on communication channel fidelity, causing an overall reduction in the hardware complexity and subsequently system size. Use of Real time OS and embedded filesystems reduces the implementation complexity of the satellite computers by allowing code reusability and makes the development faster. The software was tested successfully against random bit-flips in the program memory and the data memory.

The imminent launch of JUGNU and subsequent analysis of our design's performance over the one year duration of the mission will prove its viability and also indicate areas of future work to further enhance the system reliability of the next generation nanosatellites.

Bibliography

[1] "California polytechnic state university." "http://www.calpoly.edu/".

[2] "Stanford universitys space systems development lab." "http://ssdl.stanford.edu/ssdl/index.php".

[3] "Pumpkin, inc.." "http://www.pumpkininc.com/".

[4] "Micrium, inc." "http://micrium.com/page/home".

[5] "Ucosii datasheet." "http://micrium.com/newmicrium/uploads/file/datasheets/ucosii_datasheet.pdf".

[6] "Efsl embedded filesystem library." "http://efsl.be/".

[7] "Salvo pro 4 user manual." "http://www.pumpkininc.com/content/doc/manual/SalvoUserManual.pdf".

[8] "Effs - thin v 1.87 implementation guide - hcc embedded." "http://www.hcc-embedded.com/en/products/file_systems/thin/msp430".

[9] T. I. Inc., "Datasheet - msp430f261x." "http://focus.ti.com/lit/ds/slas541f/slas541f.pdf", December 2008.

[10] "Msp430x2xxx family user guide (rev e)." "http://focus.ti.com/lit/ug/slau144e/slau144e.pdf", March 2008.

[11] "Cubesat kit." "http://www.cubesatkit.com/content/overview.html".

[12] "Datasheet cubesat kit ppm a3 - rev b." http://www.cubesatkit.com/docs/datasheet/DS_CSK_PPM_A3_710-00516-B.pdf.

[13] "Cubesat kit in space." http://www.cubesatkit.com/content/space.html.

[14] "Datasheet cubesat kit fm430 rev c." "http://www.cubesatkit.com/docs/datasheet/DS_CSK_FM430_710-00252-C.pdf".

[15] "Datasheet m41t81s - stmicroelectronics." "http://www.st.com/stonline/products/literature/ds/10773.pdf".

[16] "Crossworks for msp430 user guide." "www.rowley.co.uk/msp430/msp430_manual.pdf".

[17] "Datasheet atmel at91sam7x512." "http://www.atmel.com/dyn/resources/prod_documents/doc6120.pdf".

[18] "Iar embedded workbench for arm." "http://www.arm.com/community/partners/display_product/rw/ProductId/2107/".

[19] "Segger jlink." "http://www.segger.com/cms/jlink.html".

[20] "J2 orbital perturbation." "http://www.braeunig.us/space/orbmech.htm".

[21] "Datasheet cc1070." "http://focus.ti.com/lit/ds/swrs043a/swrs043a.pdf".

[22] "Datasheet atmel avr atmega8." "http://www.atmel.com/dyn/resources/prod_documents/doc2486.pdf".

[23] "Datasheet - analog devices adf7020-1." "http://www.analog.com/static/imported-files/data_sheets/ADF7020-1.pdf".

[24] "Datasheet - maxim max1472." "http://datasheets.maxim-ic.com/en/ds/MAX1472.pdf".

[25] T. I. Inc., "Crc implementation with msp430." "http://focus.ti.com/lit/an/slaa221/slaa221.pdf", November 2004.

[26] B. A. Forouzan, *Data Communications and Networking*. New York, NY, USA: McGraw-Hill, Inc., 2003.

[27] wikipedia, "Morse code." http://en.wikipedia.org/wiki/Morse_code.

[28] E.-I. KIM, J.-R. LEE, and D.-H. CHO, "Performance evaluation of data link protocol with adaptive frame length in satellite networks(satellite and space communications)," *IEICE transactions on communications*, vol. 87, no. 6, pp. 1730–1736, 2004-06-01.

[29] C. Ward, S. Mitra, and T. M. Phillips, "Throughput efficiency of an enhanced link management procedure," *Computer Communications*, vol. 17, no. 9, pp. 626–636, 1994.

[30] A. E. Baratz and A. Segall, "Reliable link intialization procedures," in *ICC (1)*, pp. 24–28, 1984.

[31] C. Ward and C. H. Choi, "The lams-dlc arq protocol," *SIGCOMM Comput. Commun. Rev.*, vol. 21, no. 4, pp. 249–257, 1991.

[32] C. Ward, C. H. Choi, and T. F. Hain, "A data link control protocol for leo satellite networks providing a reliable datagram service," *IEEE/ACM Trans. Netw.*, vol. 3, no. 1, pp. 91–103, 1995.

[33] T. Wilfredo, "Software fault tolerance: A tutorial," tech. rep., 2000.

[34] N. Murphy, "Watchdog timers." "http://www.embedded.com/2000/0011/0011feat4.htm".

[35] B.R.Bhat, "Single event effects," tech. rep., ISRO SATELLITE CENTRE, 1999.

[36] "Triple modular redundancy." "http://www.embedded.com.au/pages/TMR.html".

[37] J. M. Spinelli, "Reliable communication on data links 1," 1988.

[38] "Jugnu - nanosatellite." "http://www.iitk.ac.in/me/jugnu/".

[39] "Adcs subsystem of jugnu," tech. rep., Indian Institute of Technology, Kanpur.

[40] "Communication subsystem of jugnu," tech. rep., Indian Institute of Technology, Kanpur, 2010.

[41] "Space radiation effects on electronic components in low-earth orbit," tech. rep., NASA, 1996.