Development and Implementation of the attitude control algorithms for the micro-satellite Flying Laptop

A thesis accepted by the Faculty of Aerospace Engineering and Geodesy of the Universität Stuttgart in partial fulfillment of the requirements for the degree of Doctor of Engineering Sciences (Dr.-Ing.)

by

Muhammad Yasir

born in Sargodha, Pakistan

main referee:	Prof.Dr.rer.nat. Hans-Peter Röser
co-referee:	Prof.Dr.-Ing. Walter Fichter
co-referee:	Prof.Dr.-Ing. Felix Huber

Date of defence: March, 15, 2010

Institute for Space Systems

Universität Stuttgart

2010

Abstract

Growing interest in small satellites is propelled due to their cost effectiveness, less developmental time, evolveability, and performance. The fast growth is also connected with more challenges which are associated with small satellites. The Flying Laptop (FLP) is one such satellite which is under development at Institute of Space Systems (IRS) at University of Stuttgart. Beside from the scientific observation the FLP is demonstrator of several technology experiments including FPGA-based reconfigurable OBC with high computational capability. The ability to reconfigure enables the brain of the system to evolve and adapt to the changing requirements. Fast speed and true parallel computation are accompanying advantages of using FPGAs. The research presented in this thesis attempts to lay out essential foundations for developing, implementing and testing FPGA specific attitude control system.

As a first step, the overall structure of ACS algorithms, its command hierarchy and its interaction with other sub-systems and system FDIR was developed in accordance with the requirements and constraints. The need of different operating modes suitable for different mission scenarios was identified at this stage. For scientific observation inertial pointing, nadir pointing, and target pointing modes are developed to achieve the pointing accuracy of 150 arc-sec. Apart from the payload image acquisition modes, several supporting modes of ACS operation are provided to facilitate its functioning in different set of conditions. A clear concept of safe mode with solar panels pointed toward the sun is established which will be used during any contingency situation. Detumble mode is provided to ease out the orbit insertion phase and Idle mode will be used for charging the on-board batteries by pointing the solar panels toward the sun. The sensor measurements will be obtained by using magnetometers, sun sensors, GPS, star tracker, and FOGs while the actuation will be provided by magnetic torquers and reactions wheels.

In a developmental phase the implementation of the ACS algorithms is carried out in Matlab. The structure of ACS algorithms is developed in accordance with its implementation in FPGA as hardware. Embedded Matlab functions are used at this stage. Matlab is used at this stage to carry out the performance verification. In a next phase these algorithms are ported into FPGA using Handel-C developmental language which directly generates the hardware configuration of FPGA from a code similar to ANSI-C. A lot of effort was required at this stage to reduce the size of resulting hardware and to fit the ACS algorithms within the limited resources of FPGA. FPGA testing boards are

used at this point to verify the results and performance. Another issue related with the implementation was to generate hardware libraries for fixed point computation of several mathematical functions.

The testing of these algorithms is first carried out using Matlab/Simulink based simulation environment. For this the already present simulator was improved with the addition of several key blocks like the adding Earth's albedo effect , eclipse flag generator and simulation of power supply system. The communication interface between FPGA and simulation environment is realized by using RS-232 serial port. In a second step these algorithms are also tested with Model-based Development and Verification Environment (MDVE) originally developed by EADS Astrium. The simulation environment based on MDVE was developed at Institute of Space Systems, University of Stuttgart. The communication between MDVE and FPGA is realized by using RS-422 to obtain the maximum baud rate.

Generally, this work proves the possibility of using ACS algorithms as embedded hardware logic to meet the challenging requirements of accuracy. This work also presents different issues of developing, implementing and testing FPGA specific ACS algorithms. Different ways of optimizing the resulting hardware design are also discussed in this thesis which has not only proved their effectiveness in the ACS algorithms but these are also helpful in the development of other subsystems.

Zusammenfassung

Das wachsende Interesse an Kleinsatelliten wird hauptsächlich durch ihre Kosteneffizienz, geringe Entwicklungszeit, Anpassungsfähigkeit (Evolvierbarkeit) und Leistungsfähigkeit getrieben. Dieses schnelle Wachstum ist aber auch mit immer größeren Herausforderungen an die Kleinsatelliten verbunden. Der Flying Laptop (FLP) ist ein solcher Satellit und wird zurzeit am Institut für Raumfahrtsysteme (IRS) der Universität Stuttgart entwickelt. Neben der wissenschaftlichen Beobachtung ist der FLP ein Demonstrator für verschiedene Technologie Experimente wie zum Beispiel der FPGA-basierte rekonfigurierbare OBC mit hoher Rechenkapazität. Die Rekonfigurierbarkeit erlaubt es dem Gehirn dieses Systems sich zu entwickeln und an sich verändernde Herausforderungen anzupassen. Die hohe Rechengeschwindigkeit und echte parallele Rechnungen sind weitere Vorteile die der Einsatz von FPGAs mit sich bringt. Die in dieser Arbeit vorgestellten Untersuchungen sollen den Grundstein für die Entwicklung, Implementierung und das Testen eines FPGA basierten Lagekontrollsystems legen.

In einem ersten Schritt wurde die allgemeine Struktur der ACS Algorithmen, die Kommandohierarchie und die Interaktionen mit anderen Subsystemen und des System-FDIR in Übereinstimmung mit den Requirements und Constraints entwickelt. Bereits zu diesem Zeitpunkt wurde der Bedarf verschiedener operationeller Modi passend zu verschiedenen Missionsszenarien identifiziert. Für die wissenschaftliche Beobachtung wurden der Inertial Pointing, der Nadir Pointing und der Target Pointing Mode mit einer Ausrichtgenauigkeit von 150 Bogensekunden entwickelt. Neben den Nutzlast Modes für Bildaufnahmen gibt es weitere unterstützende ACS Modi, die das Funktionieren und Überleben des Systems unter verschiedenen Voraussetzungen sichern. Ein existiert Safe Mode Konzept, das die Solarzellen des Satelliten bei jeder Art von Notfall Richtung Sonne ausrichtet. Der Detumble Mode wird für die Phase nach dem Start und der Idle Mode zum Laden der Batterien verwendet, indem die Solarzellen aktiv auf die Sonne ausgerichtet werden. Die Sensormessungen werden durch Magnetometer, Sonnensensoren, GPS, Sternenkameras und Fiberobtische Kreisel vorgenommen, wohingegen die Stellbewegungen von Magnettorquern und Reaktionsrädern sichergestellt wird.

In der Entwicklungsphase erfolgt die Implementierung der ACS Algorithmen in Matlab. Die Struktur der ACS Algorithmen wird in Übereinstimmung mit ihrer Realisierung als FPGA Hardware entwickelt. In dieser Phase werden Embedded Matlab Funktionen verwendet. Matlab wird genutzt, um die Leistungsverifikation des Systems durchzuführen.

In der nächsten Phase werden die Algorithmen auf einen FPGA mittels der Entwicklungssprache Handel-C portiert, die direkt die Hardwarekonfiguration für den FPGA aus einem ANSI-C ähnlichen Code generiert. Zu diesem Zeitpunkt wurden große Anstrengungen nötig, um die Größe der resultierenden Schaltungen zu reduzieren und diese auf den begrenzten Ressourcen des FPGA unterzubringen. Ein weiterer Aspekt der mit der Implementierung zusammenhing war die Generierung von Hardwarebibliotheken für Fixed-Point Berechnungen verschiedener mathematischer Funktionen.

Die ersten Test dieser Algorithmen wurden mit einer Matlab/Simulink basierten Simulationsumgebung durchgeführt. Dafür wurde der bereits bestehende Simulator um mehrere Schlüsselfunktionen wie z.B. der Effekt der Erdalbedo, der Eclipse-Flag Generator und die Simulation des Energieversorgungsystems erweitert. Die Kummunikationsschnittstelle zwischen dem FPGA und der Simulationsumgebung wurde mit einer seriellen RS-232 Schnittstelle verwirklicht. In einem zweiten Schritt werden diese Algorithmen auch mit der von EADS Astrium entwickelten Model-based Development and Verification Environment (MDVE) getestet. Die auf der MDVE basierende Simulationsumgebung wurde am Institut für Raumfahrtsysteme der Universität Stuttgart entwickelt. Die Kommunikation zwischen der MDVE und dem FPGA wird mit einer RS-422 Schnittstelle realisiert, um eine maximale Baudrate zu erhalten.

Im Großen und Ganzen zeigt diese Arbeit die Möglichkeiten der Nutzung von ACS Algorithmen als eingebettete Hardwarelogik zum erreichen anspruchsvoller Herausforderungen an die Lagegenauigkeit auf. Ebenso präsentiert diese Arbeit verschiedene Möglichkeiten der Entwicklung, Implementierung und des Testens von FPGA spezifischen ACS Algorithmen. Verschiedene Wege, wie das resultierende Hardwaredesign optimiert werden kann werden diskutiert. Diese Verfahren haben sich nicht nur bei den ACS Algorithmen bewährt, sondern erweisen sich auch bei der Entwicklung anderer Subsysteme als nützlich.

Acknowledgments

First and foremost, I am grateful to The Almighty, without HIS help and mercy it was not possible for me to complete this work. I pray that HE will make this work useful for many.

I would like to profoundly thank my advisor Professor Dr. Hans-Peter Roeser. His unparalleled compassion, kindness and generosity allowed me to cruise through the demands of this research. One thing among many which I learned from him is how to stay positive in every problem you face. Thank you so much.

I would also like to say thank to Georg Grillmayer who initiated this research and laid the foundations. His invaluable support not only during his stay at IRS but also even after that must be appreciated. I learned a great deal more than I could have imagined because of his direction. Thank you so much.

Many thanks go to small satellite team members. Toshi, Albert, Sebastian, Michael (Le), Felix, Oliver, Michael (Fr), Claas, Michael (La) , Uli, Fabian, Alex. Thank you so much for your support and help at every step.

Many Thanks to SUPARCO and Higher Education Commission (HEC) of Pakistan for financially supporting this research. Without their help it was not possible for me to take this initiative. A heartfelt thank to Dr. Ata-ur-Rehman, former Chairman HEC and Maj. General (Retd) Raza Hussain, Chairman SUPARCO. It was your vision which helped a number of students like me to broaden their horizons. Thank you very much.

Also I want to say thanks to DAAD (German Academic Exchange Service) whose cooperation with HEC accommodated a number of students in Germany. Especially I want to say thanks to Ms. Monika Osman whose help was always there when required. Her kindness and generosity must be appreciated. Thank you very much.

I also want to say thanks to German people and Germany. I have learned lot of things from your rich culture. This will help me throughout my life. I also want to thank my neighbours Petra and her little daughter Hana who were so friendly throughout our stay in Untertürkheim, Stuttgart. Thanks for making this a wonderful stay with lot of beautiful memories.

I also want to say thanks to my parents. I owe my success to them and am extremely grateful for the inspiration and encouragement I received from them.

Last but not least, my family deserves a special thanks and gratitude. The incredible sacrifice of time offered by my wife Shahnaz is invaluable. Without her support it was not

possible for me to complete this research. My son Uzair also deserves a lot of praise whose smiles and hugs were a source of energy for me. My daughter Raghad who enlightened my life also deserves a special praise. Thank you so much for making my life and my stay joyful and beautiful.

Contents

List of Figures

LIST OF FIGURES

Nomenclature

Symbols

A	Sensor Alignment Matrix
b	Earth Magnetic Field
I	Moment of Inertia
m	Magnetic Dipole Moment
N	Earth's Nutation Matrix
ω	Angular rate
P	Earth's Precession Matrix
q	Quaternion
r	Position
S	Sun Vector
T	Torque
t	time
v	Velocity

Superscripts

^{PI}	Pseudo Inverse
^{T}	Transpose

Subscripts

$_{B}$	Body Coordinate System
$_{F}$	Fixed Coordinate System
$_{I}$	Inertial Coordinate System
$_{max}$	Maximum Value

meas	Measured Value
sensor	Sensor Coordinate System

Acronyms

ACS	Attitude Control System
ADC	Analog to Digital Converter
AMR	Anisotropic-Magnetoresistive
BPL	Back Plane
BRDF	Bi-Directional Reflectance Distribution Function
CDV	Command Decoder and Voter
C-FORS	Commercial Fiber Optic Rate Sensor
CHU	Camera Head Unit
CIB	Connector Interface Board
CLB	Configurable Logic Block
CPN	Central Processing Node
DCM	Direction Cosine Matrix
DK	Developer's Kit
EM	Experimental Model
FDIR	Fault Detection, Isolation and Rectification
FLP	Flying Laptop
FM	Flight Model
FOG	Fiber Optic Gyro
FPGA	Field Programmable Gate Array
GaAs	Gallium Arsenide
GENIUS	GPS Enhanced Navigation Instrument for Universität Stuttgart
GPS	Global Positioning System
I^2C	Inter-Integrated Circuit
IBIS	Integrated Bus for Intelligent Sensors
I/O	Input/Output
IRS	Institute für Raumfahrtsysteme
JD2000	Julian Date 2000
JD	Julian Date
LEO	Low Earth Orbit
LNA	Low Noise Amplifier
LTDN	Local Time of Descending Node
LUT	Lokk Up Table

μASC	Micro Advanced Stellar Compass
μDPU	Micro Data Processing Unit
MDVE	Model-Based Development and Verification Environment
MGM	Magnetometers
MGT	Magnetic Torquers
MICS	Multi-spectral Imaging Camera System
NEA	Near Earth Asteroids
OBC	On-Board Computer
PAL	Platform Abstraction Layer
PAMCAM	Panoramic Camera
PCDU	Power Control and Distribution Unit
PSL	Platform Support Library
PSS	Power Supply System
RW	Reaction Wheels
SHIP	Satellite Hardware Interface Protocol
STR	Star Tracker
SuS	Sun Sensors
TC	Tele-Command
TM	Telemetry

CHAPTER 1

Introduction

Advancement in the payload technologies has made small satellites capable of performing very sophisticated tasks in addition to their suitability as an attractive test bed for technology evaluation. Another motivation behind the growth of small satellites is relatively low overall costs and less development time. Due to the large size and heavy costs, the satellite projects were mainly limited to the public sector who were more interested in weather, communication or military satellites but now with the reduced over all costs and less development time, small satellites are attracting the interest of private sector. Even few institutions under the universities are having some of their satellites in the orbit. These small satellites are addressing problems which were either technically impossible or too expensive to attempt before.

Setting ambitious objectives from such a small satellites elevated the expectations of accuracy and agility from the attitude control system. The requirement of using light weight and cost effective hardware have put further bounds on the designer of the attitude control system. The challenge further increases with the demands from the subsystems such as maximizing the on-board power, facilitating communication, carrying out scientific mission objectives and along with that meeting the reliability constraints by keeping satellite alive with reduced functionality even under abnormal conditions.

1.1 Scope of this work

The work presented in this report is focused on the issues of designing, developing and testing an attitude control system for a micro-satellite *Flying Laptop* (FLP). The FLP will be the first satellite within a Stuttgart Small Satellite Program initiated by the Institute of Space Systems, University of Stuttgart. Despite of being the first satellite of this program, many ambitious objectives are set for the FLP. One of such objective is to use reconfigurable FPGA technologies for the on-board processing.

On-board computer based solely on reconfigurable FPGA is not only new technology experiment but it is also challenging for the development and implementation of attitude control algorithms which generally require a great deal of computations. The prevailing method to perform these computations is to use a software based design along with the processor. With the advent of high speed processors, it is now possible to use complex and computationally exhaustive algorithms within the limited time cycle to obtain more accurate attitude knowledge and attitude control with the same hardware.

Use of FPGA based OBC implies a hardware design instead of a software design and limited hardware resources restricts the choice of a attitude control algorithms to a great extent. Thus a lot of effort is required to evaluate and modify algorithms to conform to the available hardware resources. The main objective of this work is to develop and implement attitude control algorithms that could run as a dedicated hardware circuit instead of software. To provide easy reconfiguration these hardware circuits should be defined by a software.

Another task associated with hardware-based design is the testing and verification. Most of the hardware designs are specific to a particular hardware unlike software-based design which remains the same for most of the cases and could be tested with similar kind of hardware before porting the design in the real flight hardware. To develop a platform independent hardware design is a solution to this problem which could be easily tested and verified with verification hardware instead of a flight hardware.

In addition, high agile and accurate pointing control along with reliable safe mode control are other requirements on the attitude control system. A challenging task is to ensure the pointing accuracy in arc-second range. There are several other requirements associated with the design of attitude control system which are listed in Chapter 2

1.2 Reconfigurable computing in space

There are several technological, economical and political aspects that set space systems apart from the other type of systems. Space systems often incorporate highly sophisticated materials, components and other technologies customized or specifically produced for the space environment. Another distinct property which set space systems apart from all other technical systems is their location. They are deployed and operated at largest distances from the point of control which makes it almost impossible to repair or to service this system. This contributes to the requirements of increased reliability and robustness.

The core of any space system is its processing or computing unit. The idea to reconfigure this computing unit in space is a hardware designer's dream since the notion of fixing a hardware bug or updating a hardware-implemented algorithm has never been available. Space missions are critical by nature and, therefore, having the option of reconfiguring a system once launched provides numerous advantages and a great deal of flexibility.

FPGA stands for Field Programmable Gate Array, which is programmable semiconductor logic device. As both the speed and the gate count of FPGAs have increased, computing engines as well as peripherals can be designed directly into these devices, the notion of creating reconfigurable platform for on-board processing is of great interest for the space systems. The advent of these reconfigurable platforms assured reduced size, complexity, reduced design cycle times and opportunities for in-space upgrades and repairs.

The capacity and performance of FPGAs suitable for space flight have been increasing steadily for more than a decade. Higher computational power along with better timing performance and data throughput guarantee are the additional advantages associated with FPGAs. These advantages have made FPGAs begin to rival sophisticated tasks traditionally intended for processors and made them best candidate for small Earth observation satellites where parallel processing of payload images is required along with the accurate attitude control.

1.3 Stuttgart small satellite program

Stuttgart Small Satellite's Program was initiated in 2003 to develop and build several small satellites at the Institute of Space systems, Universitaet Stuttgart.[3]. The program's long term goals are the launch and operation of small satellites in orbit for evaluation of new technologies, earth observation and lunar exploration. The completion of four satellites is

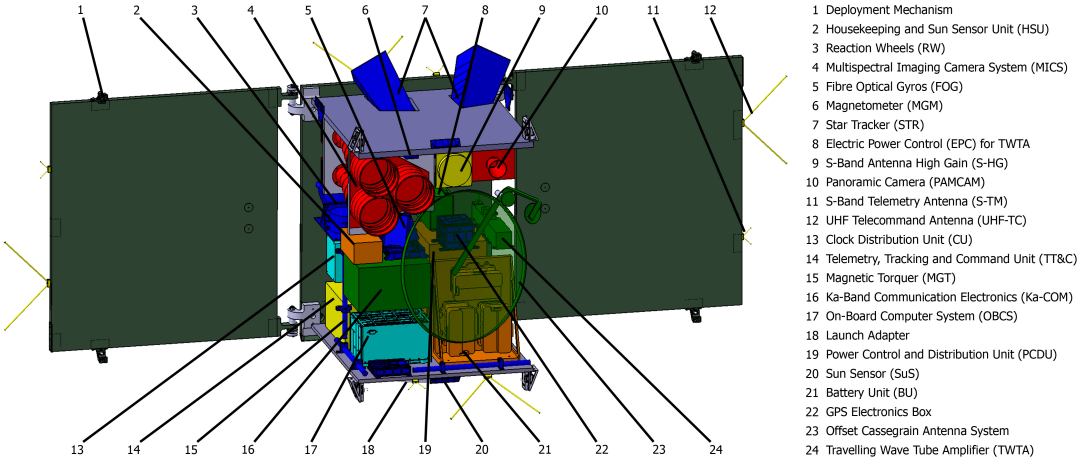


Figure 1.1: FLP System Configuration

planned to achieve the above mentioned goals. All the satellites will be designed, built and tested by the IRS. After completion of several testing models, the following four satellites are the core of this program

Flying Laptop Earth observation and technology demonstration.

Perseus Propulsion testbed and technology demonstration.

BW1 Lunar mission

Cermit Re-entry mission.

Building Institutes's ground station, a satellite clean room, integration room with an optic area and thermal vacuum chamber are also included in this program. In June 2007 the local Government agreed to fund the construction of a new establishment, the Space-center Baden Württemberg, for expansion of the current activities.

1.4 The Flying Laptop

The micro-satellite Flying Laptop is currently under development at the Institute of Space Systems, University of Stuttgart.[4; 5; 6] As it is shown in the Figure:, it is cubical shaped of dimensions 60 cm x 70 cm x 80 cm and its mass is around 120 kg. The launch is planned as a piggyback payload with a PSLV rocket from India in a sun-synchronous, low earth orbit between 500 km to 900 km. The Flying Laptop is a three axis stabilized satellite having a two years mission design lifetime .

1.4.1 Mission objectives

	Mission Objectives
Technology Demonstration	FPGA based on-board computer (OBC)
	Ka-band bi-directional communication (up to 500 Mbit/s)
	Electrical Solar Panel Deployment Mechanism.
	GENIUS experiment (attitude determination by using GPS)
	Rent-a-Sat Mode
	Space-use demonstration of several components like FOGs, solar cells etc.
Scientific Mission Objectives	Multi-spectral earth observation (GSD: 25m)
	Thermal infrared earth observation (GSD: 100m)
	Bidirectional Reflectance Distribution Function (BRDF) measurements
	Precipitation measurement (Ka/Ku-band)
	Atmospheric attenuation measurements
	Atmospheric trace gas detection
	Near earth Asteroid (NEA) detection using Star tracker

Table 1.1: Mission objectives of the FLP

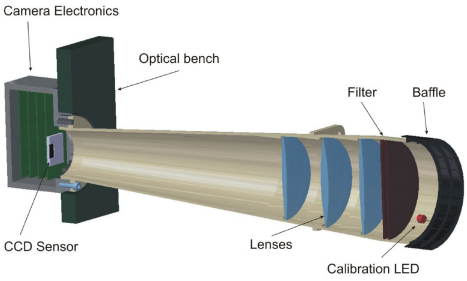
The mission objectives of the Flying Laptop project could be classified into technology demonstration and scientific earth observations and are listed in the Table 1.1.

1.4.2 Payload of the FLP

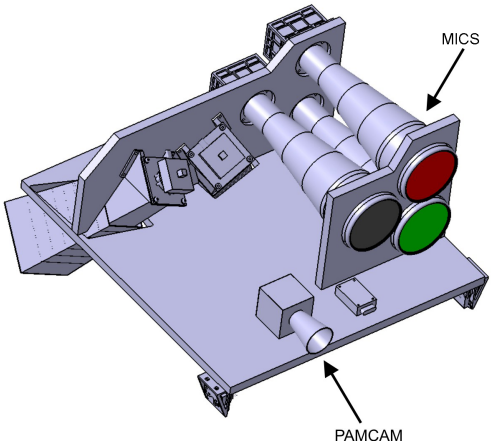
The payload of the FLP consists of Multi-spectral Imaging Camera System (MICS), Panoramic Camera (PAMCAM) and a communication system in the Ka-Band frequency range. The Thermal Infrared Camera System (TICS), as well as a signal generator in the Ku-band were originally intended to be part of the FLP mission but currently these instruments are not under any further investigation.

The MICS (Multi-spectral imaging camera system) instrument will be the main imaging payload on the Flying Laptop. It is designed for two purposes. The main purpose is the measurement of the BRDF, which is explained in detail in [7]. In a second application MICS will be used for earth observation in the nadir pointing mode, as well as in the target pointing mode. Since the first application is more ambitious MICS is designed for the BRDF usage, fulfilling the second application in the best possible manner. The schematic section drawing of MICS is shown in the Figure 1.2a.

The narrow field-of-views of the scientific camera system MICS and the rather small swath width of a few kilometers enforce the request for an additional camera system capable to provide an overview of the observed landscape. This so called Panoramic



(a) Schematic sectional drawing of a MICS camera



(b) Location of MICS and PAMCAM on the optical bench

Figure 1.2: Optical Payload of FLP

Camera (PAMCAM) should be able to take color images in order to increase the public outreach of the Small Satellite Program. During MICS operations the PAMCAM will be used in parallel, but with a reduced capture frequency. The location of PAMCAM and MICS on the optical bench of the FLP is shown in the Figure 1.2b.

Ka-Band communication is a new technology which will be demonstrated by the FLP. Due to the higher operating frequency this system is capable of delivering double data rate compare to the X-band. The objective of the mission is to demonstrate a downlink with minimum 500 Mbit/s.Further details about this experiment are presented in [7].

1.5 Previous work

The FLP is developed by a team mainly consisting of PhD students responsible for a particular subsystem. The present work is the second doctoral research conducted on the subject of attitude control system development. In the first research conducted by Mr. Georg Grillmayer [1], following tasks have been already done before the start of this work.

- ◇ The selection and procurement of ACS hardware including sensors and actuators except for the sun sensor.
- ◇ Actuator sizing.

[1] Research stay at IRS from 2004 to August 2007

- ◇ Hardware development of magnetic torquer power unit, GPS unit, fiber optic gyro unit.
- ◇ Development of device drivers library for sensors/actuators.
- ◇ Set-up of a hardware assembly to develop and test the FPGA interface library for communication with sensors/actuators.
- ◇ Analysis of potential orbit including the evaluation of star tracker blinding.
- ◇ Development of preliminary algorithms in Simulink environment.
- ◇ Development of Simulation environment based on Matlab/Simulink. The eclipse flag generator, effect of the Earth's albedo and simulation of Power Supply System (PSS) were added later in this simulation environment.

1.6 Thesis outline

This thesis is organized in such a way that first three chapters gives the introduction of the overall project and attitude control system, mission objectives, requirements along with necessary details of the ACS hardware.

The next three chapters are related with the details of ACS algorithms and their implementation. Beginning is marked with the overall structure of the ACS algorithms, followed by a detailed description of all the relevant algorithms together with the details of implementation of these algorithms.

The last three chapters are related with simulation and verification of the ACS algorithms. They give a brief introduction of the simulation environments used to verify the results, additionally the interfaces for conducting this verification are explained followed by a detailed discussion regarding the results, the summary of which is encapsulated in the conclusions.

Chapter 1

The first chapter is Introduction which provides basic know-how to the FLP project and it also highlights the scope of this work along with the previous work which is conducted in this regard.

Chapter 2

This chapter provides the system level design of the ACS by highlighting the requirements and constraints associated with the ACS design. This chapter also provides details of the ACS modes and ACS budgets followed by a brief description about the orbit analysis and effect of the radiation environment.

Chapter 3

This chapter lay down all the relevant details about the ACS hardware used for this design. For each sensor and actuator, its placements in the satellite, its specifications and the details about its electrical interface are provided in this chapter.

Chapter 4

This chapter provides details about ACS algorithm architecture. This chapter starts with the introduction of the overall control algorithm of the satellite having ACS as one subsystem in it followed by the details of each component of the ACS subsystem.

Chapter 5

This chapter is related with the ACS algorithms. To provide a systemized view this chapter is divided into sections similar to those in which these algorithms are realized in the hardware. This chapter provides all the mathematical relations used to describe the algorithms

Chapter 6

This chapter is connected with the algorithm implementation. It starts with the description of the algorithms implementation in the Matlab/Simunlink environment, after which details concerning the implementation of the algorithms as hardware on an FPGA chip are provided followed by the optimization details which are necessary to fit these algorithms with in the limited resources of the FPGA hardware.

Chapter 7

This chapter is about the simulation environment developed in Matlab/Simulink environment. Only a brief detail about each building block of this simulator is specified in this chapter to keep it concise.

Chapter 8

This chapter is about testing and verification of the algorithms. The testing was conducted both with the Matlab and MDVE simulation environment. The testing interface along with both simulators is described and a brief introduction of MDVE simulation and verification environment is also presented here.

Chapter 9

This chapter provides a variety of simulation results and discussion about the results and also outcome of this discussion is gathered to draw conclusions.

CHAPTER 2

ACS-Requirements, Constraints and Design

The process of devising the attitude control system to meet the stated objective is closely linked to constraints and performance requirements from the mission statements as well as from the other subsystems. This chapter outlines the overall design of the ACS, the objectives associated with this and also all the constraints which play important role in the overall design. To present a systemized view, the chapter starts with the objectives and constraints from the ACS followed by the overall architecture of the ACS hardware, its modes and the mode transitions. As stated in the first chapter, the FLP is planned to be launched as a piggy-back payload, therefore the parameters of the final orbit are not fixed which is another design constraint. So the second part of this chapter provides analysis of set of different potential orbits for FLP. Several properties of these orbits are calculated for better visualization of the environment which is helpful in simulations and studying different parameters.

2.1 Requirements and constraints on the ACS

The requirements on the ACS of the FLP are dictated by several different factors. The primary requirements are placed by the mission goals and the payload subsystem. These demands are of prime importance which mainly dictates the overall structure of the ACS and draw the ACS performance boundaries.

The secondary requirements are not related directly with the mission goals yet they are equally important for the proper functioning of ACS and the satellite itself. These requirements are mainly placed by the other subsystem like power and structure subsystem. Some of these are also related with the environment of the satellite in which it is functioning. The primary and the secondary requirements are listed in the following sections of this chapter.

2.1.1 Primary requirements on the ACS

The primary requirements could be listed as:

- ◇ Pointing the payload cameras toward nadir direction for the earth observation. The pointing accuracy required during this maneuver is comparatively low and is about 0.5°.
- ◇ Pointing the satellite toward inertially fixed target for calibration and for carrying out the experiment of Near Earth Asteroids detection (NEA).
- ◇ Pointing the payload toward a fixed target on the surface of the earth to carry out several experiments like BRDF measurements, and Ka-band bi-directional communication.
- ◇ Pointing knowledge of $7''$[1] (2σ) and pointing stability of $2.45''$ with in the duration of 10 ms, across the camera bore sight is required.
- ◇ An absolute pointing accuracy of $150''$ (2σ) is required around all axes with reference to the ideal attitude profile.
- ◇ To provide sun avoidance strategy to prevent the payload cameras facing toward the sun at lower body angular rates.
- ◇ Implementation of ACS algorithms as a hardware on a FPGA based OBC.
- ◇ Use of FOGs and solar cells as sensors to conduct technology demonstration and also to use GPS for GENIUS-GPS experiment.

[1] $1'' = 1°/3600$

2.1.2 Secondary requirements on the ACS

The secondary requirements could be listed as:

- ◇ Rate damping after the initial launcher separation or during operation in case of accidental increase of body rates.
- ◇ To provide a safe mode to ensure the survival of satellite by using only very reliable components during the contingency situation.
- ◇ To orient the solar panels toward the sun during stand-by periods or upon the request from the power subsystem.
- ◇ As it was decided to use star tracker to obtain the required accuracy so the requirements arises from the operation of a star tracker to avoid its blinding by the earth or by the sun.
- ◇ Maximum slew rate should not increase beyond 1.2°/s to obtain the optimum performance of the star tracker.
- ◇ Also it was decided to use reaction wheels for the accurate and agile maneuver during target pointing so to desaturate the accumulated angular momentum by the reaction wheels using external torques is another requirement coming from the use of reaction wheels.
- ◇ As the number of ground stations from which FLP could be controlled are limited so a firm concept of Fault detection isolation and recovery (FDIR) for the ACS components is required for its autonomous functioning.

2.1.3 Constraints of designing the ACS

The ACS design constraints are:

- ◇ Design life time of 2 years.
- ◇ A sun-synchronous orbit of height ranging from 500 km to 900 km.
- ◇ Satellite mass of about 120 kg.
- ◇ Limited number of ground stations with the primary ground station at IRS.
- ◇ Limited test facilities.

- ◇ Financial constraints as this project is initiated by an institute of a university which do not generate funds directly and has to rely heavily on the university and its partners.
- ◇ To be launched from PSLV.
- ◇ Environmental constraints of vibration. (mainly derived from the launcher)
- ◇ Environmental constraints of temperature, functioning in very low atmospheric pressure, and in a radiation environment depending upon its orbit height.

2.2 System level architecture of ACS

The requirements on ACS clearly demand the three axis stabilized system with the high pointing accuracy capabilities. Due to the limited hardware resources of the FPGA filtering techniques could not be utilized to improve the accuracy, therefore high pointing accuracy and pointing knowledge requirements could only be met with the fine pointing sensors. This situation compel toward the use of fine pointing devices like star trackers and rate sensors. The rate sensors are also the part of technology demonstration which is the main mission goal of FLP.

Table 2.1: ACS Components

Type	ACS Components	Quantity
Sensors	3-axis Magnetometers	2
	Hot redundant coarse sun sensors	8
	GPS Receiver	3
	Star Tracker camera head units	2
	Rate Gyros	4
Actuators	Magnetic Torquers	3
	Reaction Wheels	4

The use of earth sensor is quite common in the earth pointing satellites as they provide directly the orientation of spacecraft relative to the Earth about two orthogonal axes. But to the near earth satellite (in LEO), the Earth is the second brightest celestial object and covers up to 40% of the sky.[8] Due to this it becomes difficult for an Earth sensor to maintain the accuracy of the attitude. This was the prime reason for not selecting the Earth sensor as an attitude sensor in the FLP. Moreover the FLP is equipped with the

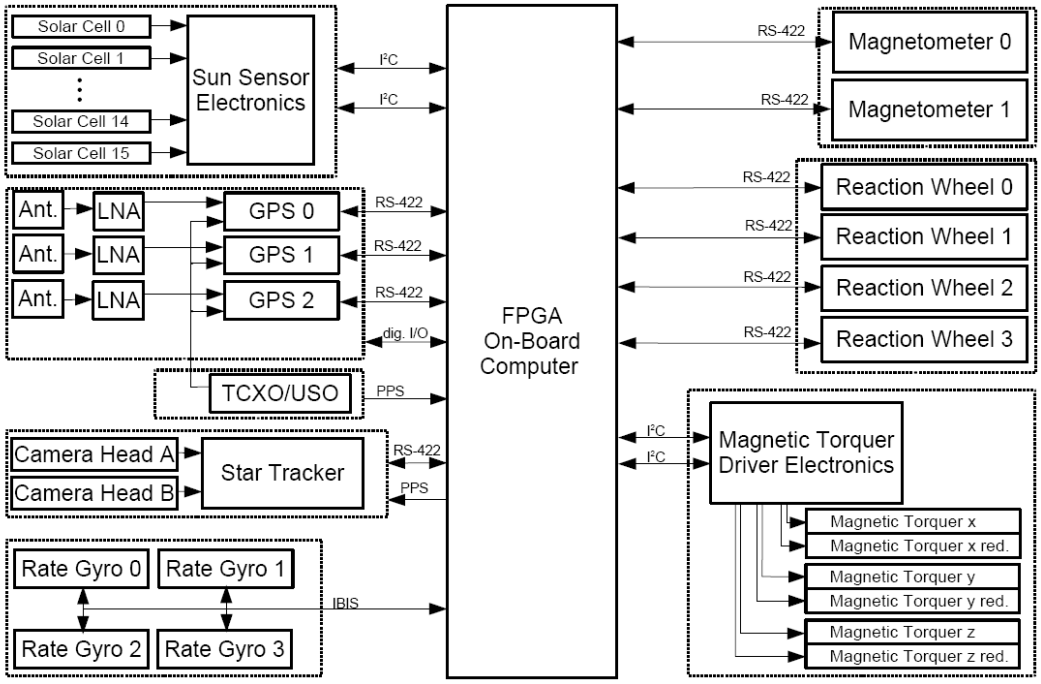


Figure 2.1: ACS Hardware Connections

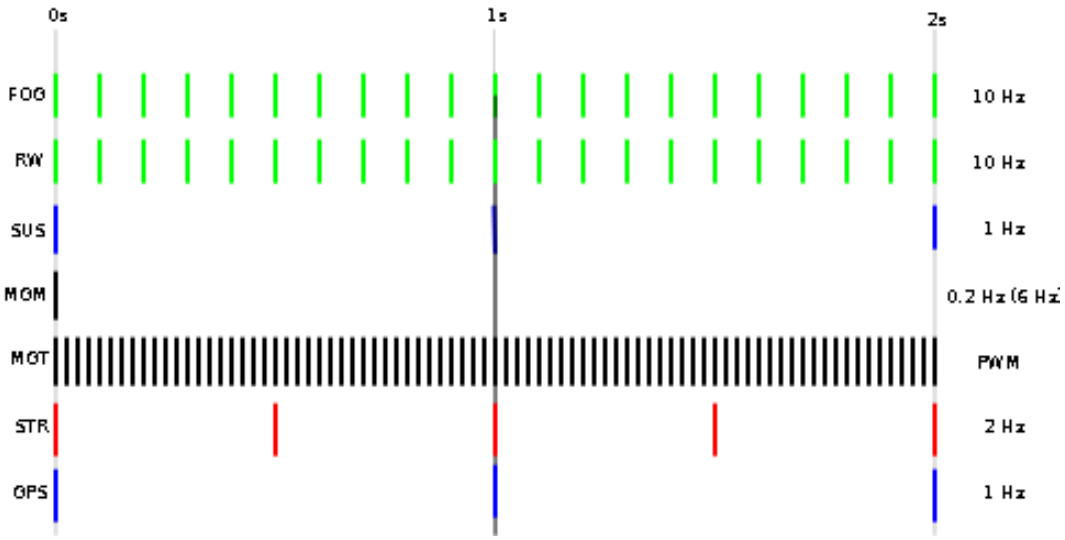


Figure 2.2: ACS timing Cycle

GPS as a part of technology demonstration experiment (GENIUS) so GPS could deliver the orientation of a spacecraft with respect to the Earth and more accurately than the Earth sensor. The cost and weight of an Earth sensor is another constraint due to which it is not considered as a choice for the attitude determination.

Five different kind of sensors and two different types of actuators are selected as shown in the Table 2.1 to meet the performance criteria laid down in above section. Many requirements influence ACS hardware design, but the primary drivers are requirement for pointing knowledge, its ability to work in safe mode with reliable hardware, control stability and slew-rate requirements during various satellite operations.

The selected ACS hardware is also a part of technology demonstration experiments which has to be carried out by the FLP. The magnetic torquers and reaction wheels are space qualified. Star tracker and magnetometer are the new developed technologies having no significant flight heritage. GPS and FOGs are products which are not intended for space use. The sun sensor is built in-house in cooperation with Astrium. The electronic boards and mechanical housings of ACS hardware are also developed in-house.

The ACS hardware structure ensures single redundancy to avoid single point failure. The power and communication interfaces are also realized in such a way that they ensure safety from single point failure. Figure 2.3 shows the power interfaces of the hardware and it is clearly redundant.

ACS hardware is connected to FPGA on-board computer in a star like configuration and the communication between OBC and each hardware is realized with different interface as shown in the Figure 2.1 . This architecture allows parallel and independent processing of each hardware interface. This architecture also ensures that the failure in one hardware interface will not effect the other hardware interfaces. The communication between each hardware is carried out with different frequency. The frequency of overall ACS loop is 10 Hz, Figure 2.2 shows the frequency with which each hardware communicates with the OBC. More details about the ACS hardware is presented in the next chapter.

2.3 ACS control modes

Depending upon the requirements the ACS control modes could also be categorized in two different categories, i.e. primary control modes and secondary control modes. Primary control modes are to fulfill the primary requirements laid down in section 2.1.1. Based on these requirements the primary control modes are classified into, `Inertial Pointing`

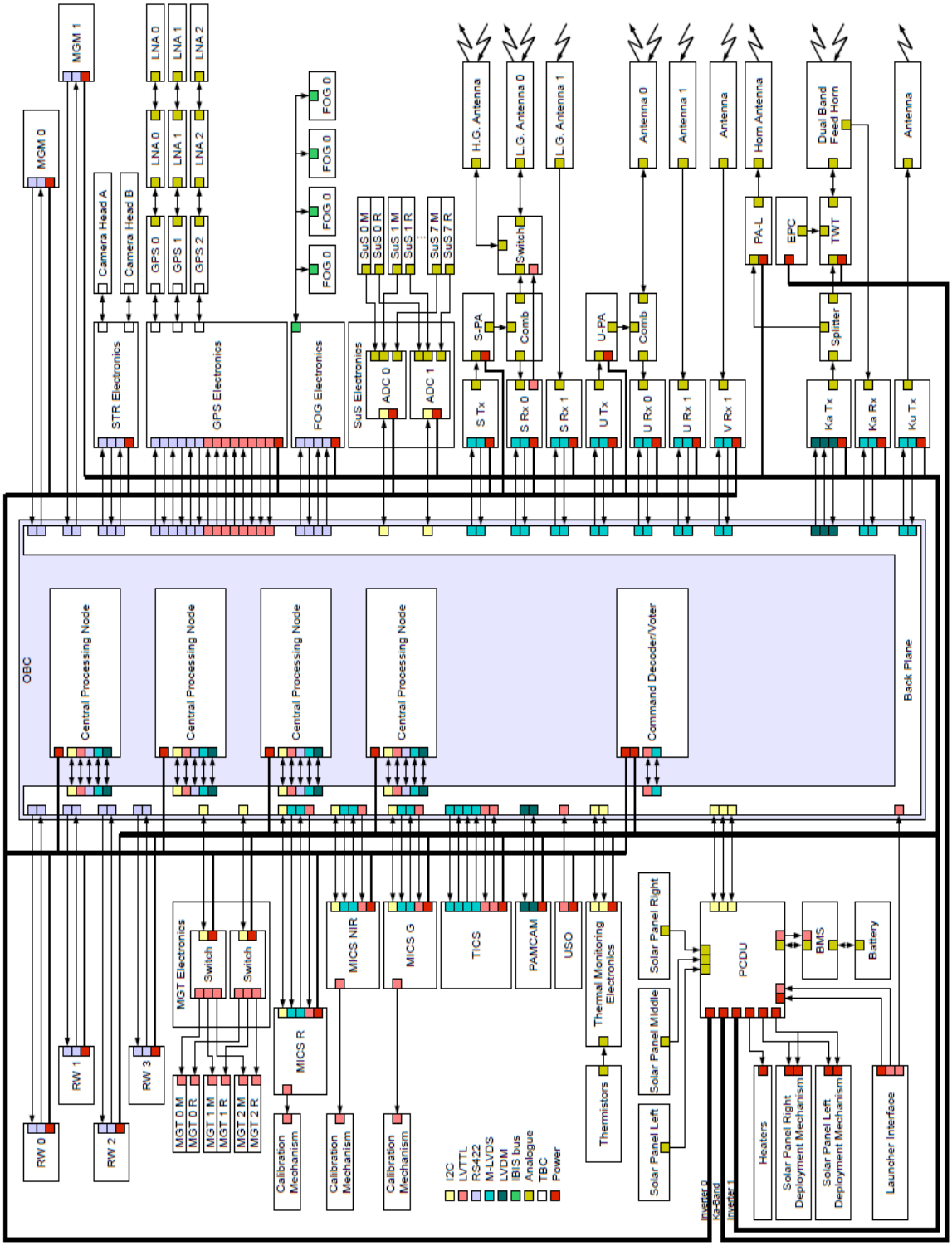


Figure 2.3: FLP Electrical Architecture

`Mode`, `Nadir Pointing Mode` and `Target Pointing Mode`. All the experiments planned to be conducted by FLP will be carried out using primary control modes, therefore these modes could also be called as science modes. These modes are also shown in the Figure 2.4.

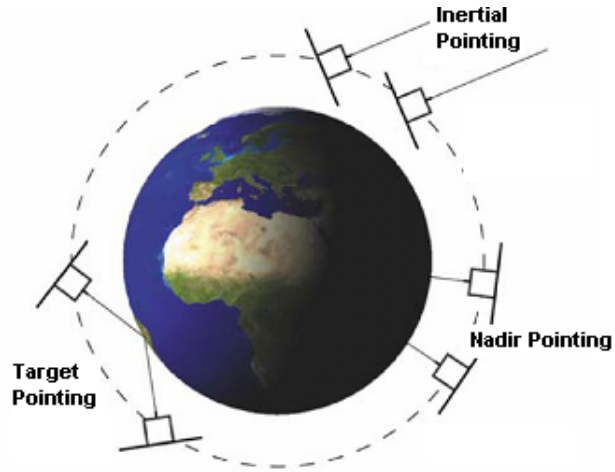


Figure 2.4: Pointing Modes of FLP

The secondary control modes are required to fulfill the requirements presented in section 2.1.2. Based on these requirements, secondary control modes are classified into `Detumble Mode`, `Safe Mode` and `Idle Mode`. These modes are non-science operation modes and they are required to cater the need like orbit insertion, house keeping and working with the minimum functionality.

There are also two supportive modes which support the functioning of reaction wheel, these are `Null-Space Mode` and `Desaturation Mode`.

2.3.1 Detumble mode

A small satellite is likely to tumble after the separation from the launcher. Detumble mode as the same suggests will be used to reduce the angular rates after the separation of the satellite from the launcher, also in the case of accidental increase in the body rates during normal operation, the satellite will automatically switch to the detumble mode. This mode uses pure magnetic control using magnetic torquers which interact with the geomagnetic field to generate the control torques required to reduce the body rates of a satellite. The satellite body rates will be estimated using magnetometer and the applied control law is the famous `B-dot` law. The control law is based on the measurement of the rate of change

of body-fixed magnetometer signals. Using only magnetometer and magnetic torquers, the `B-dot` controller despins the satellite relative to the Earth's magnetic field vector.

2.3.2 Safe mode

The safe mode for the FLP is defined as the situation when the satellite is able to continue its normal operation with reduced functionality on the occurrence of serious anomaly in the software or the hardware of the satellite, which cannot be automatically processed and corrected on board. This mode is used to ensure the satellite's vital functions (e.g. on board power) while in the meantime, the failure is being analyzed on ground. In safe mode, the satellite is controlled through autonomous software dedicated for such a condition. The sensors used for this mode should reflect high reliability and the sensor outputs need to be available all the time, however only a coarse pointing knowledge is required during this mode.

As FOGs used in FLP are not qualified for the space use and also due to their unreliability it was decided not use them in the safe mode. Star tracker is also not reliable to be used during this mode. Reaction wheels are also not considered to provide actuation during this mode due to their moving parts which reduces their reliability. Only magnetometers and sun sensors will be used for attitude determination during this mode and magnetic torquers will be used to provide actuation.

The major constraints in designing the safe mode are to keep the solar panels oriented toward the sun to keep the satellite alive and also to protect the exposure of payload cameras to the sun. Since attitude information is obtained by the sun sensors in this mode so this information is no longer available when satellite is in the umbra phase which is from 30-35% of the time period depending upon the final orbit. As a consequence when satellite is coming out of umbra phase, there is a high probability that payload cameras could get the exposure of the sun. To overcome these problems and to meet the constraints a simple but effective approach was selected. The satellites rate information is extracted from sun sensors and magnetometers and the satellite's negative principle inertial axis $-Z_j$ is aligned with the sun vector. Meanwhile for attitude hold, a spin is built around the negative principle inertial axis therefore without having any attitude information during the umbra phase, the payload cameras remain protect from the sun light while satellite is coming from the eclipse to the sun phase. This is illustrated in the Figure 2.5. Further details about safe mode could also be found in [9]

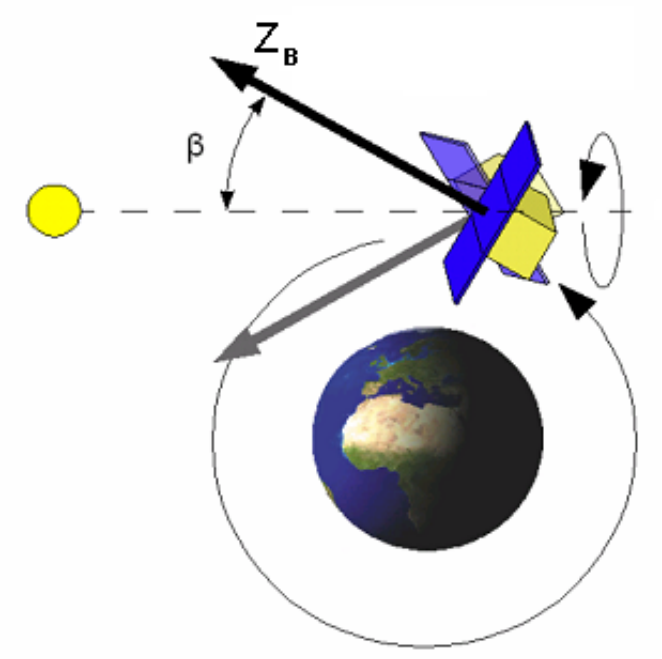


Figure 2.5: Safe Mode Illustration

2.3.3 Idle mode

This is also a non-science operational mode. This mode will also be used to cover the power supply demands furnished by the power subsystem by orienting the solar panels toward the sun, on the other hand this mode will be used when satellite is not performing any commanded action and waiting for the next instructions. Therefore this is called Idle mode.

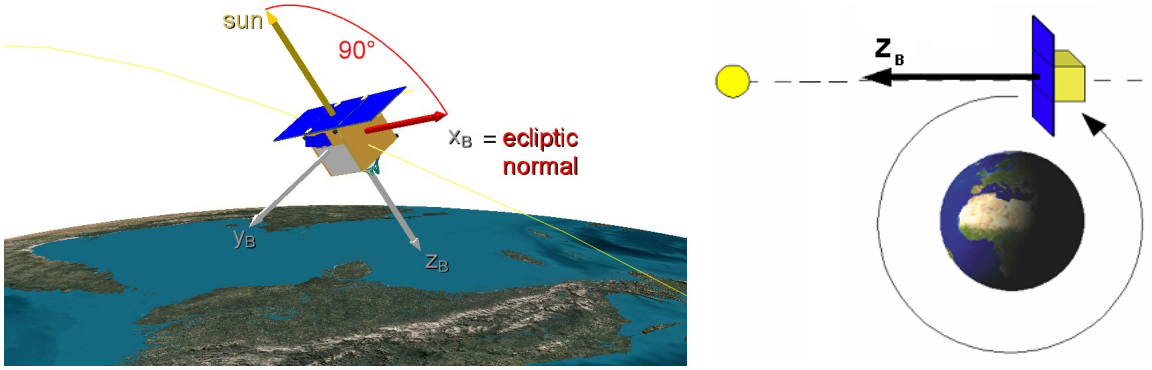


Figure 2.6: Concept of Idle mode

The accuracy requirements are not demanding in this mode but principally all the

attitude hardware is available in this mode. So it was decided to use magnetometers, sun sensors, FOGs and star tracker as attitude sensors in this mode and actuation will be obtained through reaction wheels. Magnetic torquers will facilitate the use of reaction wheels. Sun sensor will provide the information of the sun vector while STR will help in providing the attitude in the inertial reference. FOGs will be used for the rate damping.

The concept of the Idle mode is shown in the Figure 2.6. The solar panels are oriented toward the sun to obtain the maximum power which is helpful in charging the batteries, meanwhile to define the rotation around the sun vector and to align the satellite in the inertial frame, the body x-axis is pointed toward the ecliptic normal vector.

2.3.4 Inertial pointing mode

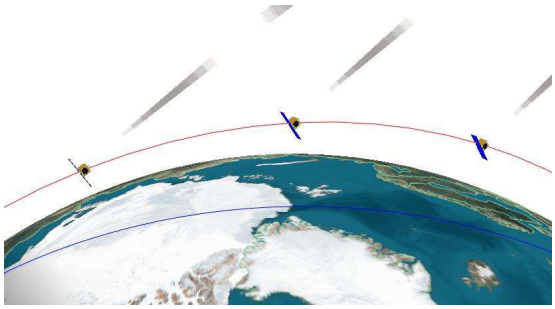


Figure 2.7: Inertial Pointing Mode

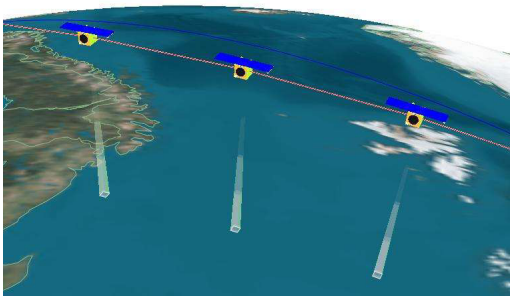


Figure 2.8: Nadir Pointing Mode

As depicted in the Figure 2.7 inertial pointing mode will be used to point the payload toward the fixed target relative to the inertial frame. The attitude remain fixed relative to the stars. The attitude information is acquired by STR and FOGs in this mode and the actuation is provided with the reaction wheels. The STR provides directly attitude information in the fixed inertial reference frame therefore no additional navigation is required for the attitude control. Information from the other sensors is not required in this mode, however MGTs, MGMs and SuS are also used during all the pointing modes to facilitate the working of reaction wheels and sun sensor information is obtained to calculate the position of the sun. This information helps in preventing the payload cameras looking directly into the sun.

2.3.5 Nadir pointing mode

This is also a science observation mode which will be used primarily for the earth observation. The satellite's body coordinate system is aligned with the nadir coordinate system and the payload cameras face toward the center of the earth during satellite's motion under this mode. The satellite conduct one rotation about its body y-axis during each orbit. The motion of the satellite is shown in the Figure 2.8.

As FLP is not equipped with any earth sensor, therefore the information of the nadir vector is not directly available during this mode. This information is extracted from the GPS. The conversion of ECI to ECF frame is conducted using the earth rotation angle which could be calculated using GPS time. The nadir reference frame is calculated based on velocity and position vector provided by the GPS. The center of the earth defined in this mode is the origin of WGS-84 frame and the attitude is calculated relative to the J2000 frame. This mode could also be extended to an off-nadir earth observation by simply changing the reference quaternion accordingly.

2.3.6 Target pointing mode

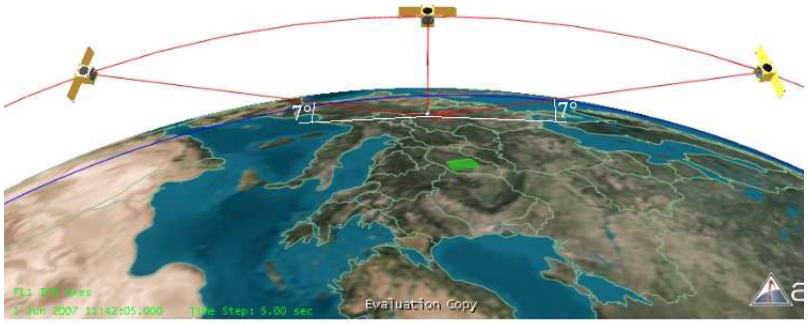


Figure 2.9: Target Pointing Mode

This is the most demanding mode of the FLP with respect to the accuracy considerations. This mode will also be utilized for conducting the scientific experimentation. BRDF measurements and Ka-band communication will be done using this mode. As it is shown in the Figure 2.9 the payload during this mode focuses a fixed spot on the surface of the earth and it remain pointed toward this spot throughout the flyover. The absolute pointing accuracy of 150″ and the pointing knowledge of 7″ is required in this mode. The

slew rate required for this motion during a flyover follow a non-linear bell shaped curve with a maximum rate of about 1°sec.

The attitude control algorithm for this mode is based on the same principle as in the nadir pointing mode. It will also use the same hardware for navigation and control.

2.4 Mode transitions

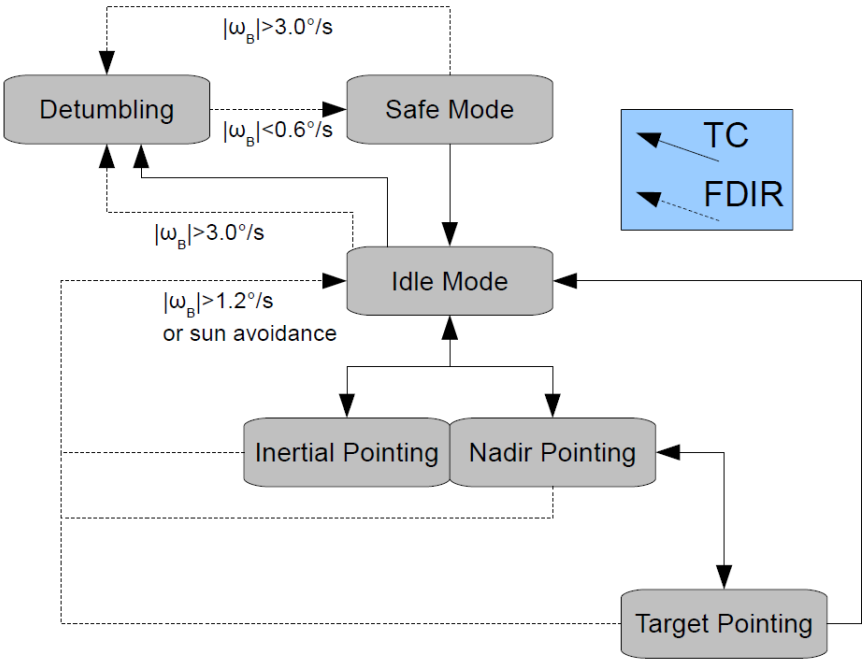


Figure 2.10: Mode Switch Logic

The mode transitions in ACS of the FLP could be triggered either by using telecommand (TC) or it is triggered automatically by the Fault Detection, Isolation and Recovery (FDIR). Mode switching by FDIR could be initialized upon occurrence of some logical error like rate limit violation, sun avoidance or off-target pointing and it is also initiated when there is some hardware failure. ACS mode transitions in FLP are shown in the Figure 2.10. The dotted lines in this figure shows the mode transitions which are initiated automatically by the FDIR and the mode switching which is shown with the solid lines is initiated by the TC.

Inter-mode switching is also limited in the ACS design. Mode switching from every mode to the other mode is not possible in order to ensure the normal functional and safety requirements. The allowed mode transitions in the ACS are tabulated in the Table 2.2.

This table shows all the possible mode switching options and the allowed mode switching is represented with the "1" and the mode transition which is not allowed is represented with the "0".

Table 2.2: Allowed Mode Switching

Current Mode	**Commanded Mode**					
	Detumble mode	Safe mode	Idle mode	Inertial mode	Nadir mode	Target mode
Detumble mode	1	1	0	0	0	0
Safe mode	1	1	1	0	0	0
Idle mode	1	0	1	1	1	0
Inertial pointing	0	0	1	1	0	0
Nadir pointing	0	0	1	0	1	1
Target pointing	0	0	1	0	1	1

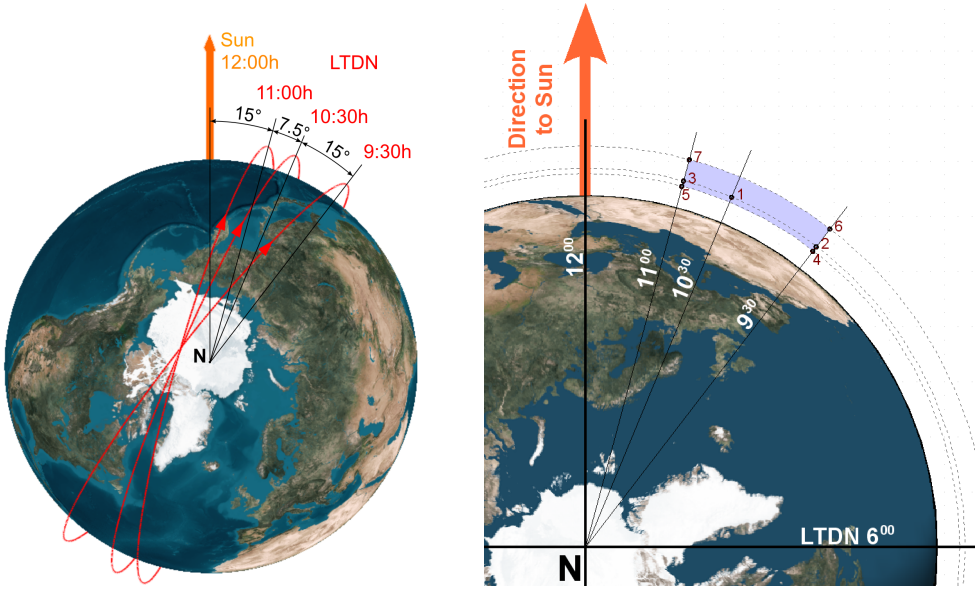
2.5 ACS budgets

Mass, power, and memory are the main constraints while designing any subsystem of a satellite. Since FLP is based on FPGA based OBC, therefore the attitude control algorithms will run as a dedicated hardware instead of a software. So there was no memory constraint in designing the ACS algorithms, however the FPGA based OBC is limited in terms of number of look up tables (LUTs) which is a major constraint in developing the ACS algorithms.

The values of mass and power of ACS components are listed in the Table 2.3. The total mass of the satellite is estimated to be 123 kg including 10% margin. The mass of the ACS components listed here also contains 10% of the margin. The power consumption of ACS components is also listed here. The maximum power consumption of a satellite is 313 W during target pointing mode while performing Ka-Band communications. The minimum power consumption of a satellite is 69 W during safe mode which is a survival mode. The values of voltage supply mentioned in this table refers to voltage supply of a individual sensor which is reliazed redundantly.

Table 2.3: ACS Budgets

ACS Components	Mass [kg]	Power [W]	Voltage [V]
Magnetometers	0.105	1	12 ± 0.1
Sun sensor system	0.182	2.5	5.1 ± 0.1
GPS system	0.926	9	5.1 ± 0.1 3.3 ± 0.1
STR system	2.356	5.5	20 ± 1
FOGs Unit	1.970	11	5.1 ± 0.1
MGTs and Electronics	1.008	1.5	5.1 ± 0.1
RW Unit	4.250	16 max 8 nominal	20 +0/-0.5 5.1 ± 0.1



(a) Effect of LTDN

(b) Effect of LTDN and orbit altitude

Figure 2.11: Orbit visualizations

2.6 Orbit analysis

The exact orbital parameters for the FLP could not be estimated as it is planned to be launched as a piggy-back (secondary) payload and the orbit is defined according to the needs of the primary customer. In spite of that it is possible to set the boundaries for the relevant parameters. It is therefore planned for FLP to be launched in a sun synchronous orbit (SSO) with an altitude between 500 to 900 km and a Local Time of Descending Node (LTDN) between 0930 hrs and 1100 hrs. By taking different values of LTDN and set of different altitudes seven different orbits are defined for FLP to carry out simulations for the verification of analysis of the impact of the different orbital parameters. The parameters on the basis of which this selection is made are shown in the Table 2.4

Table 2.4: Properties of the defined orbits

Orbit	Property
01	Nominal orbit
02	Nominal altitude and largest LTDN
03	Nominal altitude and smallest LTDN
04	Lowest altitude and largest LTDN
05	Lowest altitude and smallest LTDN
06	Highest altitude and largest LTDN
07	Highest altitude and smallest LTDN

Table 2.5: Orbit definitions

Orbit	a	h	e	i	LTAN/LTDN	T_{orbit}		Revolutions	
	[m]	[km]	[%]	[°]	[hh:mm]	[s]	[min]	[1/day]	[1/year]
01	6 978 137	600	0	97.79	22:30/10:30	5801.23	96.69	14.89	5440
02			0		21:30/09:30				
03			0		23:00/11:00				
04	6 878 137	500	0	97.40	21:30/09:30	5677.98	94.63	15.22	5558
05			0		23:00/11:00				
06	7 278 137	900	0	99.03	21:30/09:30	6179.33	102.99	13.98	5107
07			0		23:00/11:00				

2.6.1 Orbit definitions

The two important orbital factors which play an important role in the earth observing satellites are LTDN and orbit altitude. LTDN defines the lighting conditions while the orbit altitude is important for the optical resolution of the payload cameras and it also affects the slew rates during imaging maneuvers. The effect of these parameters on the orbit is shown in the Figure 2.11.

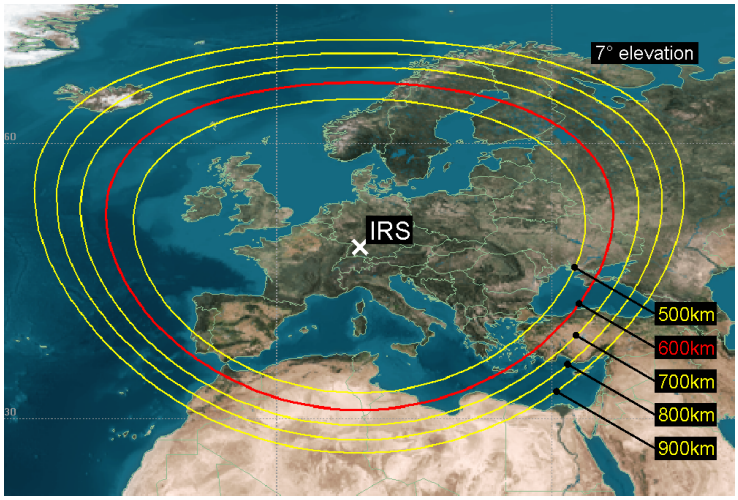
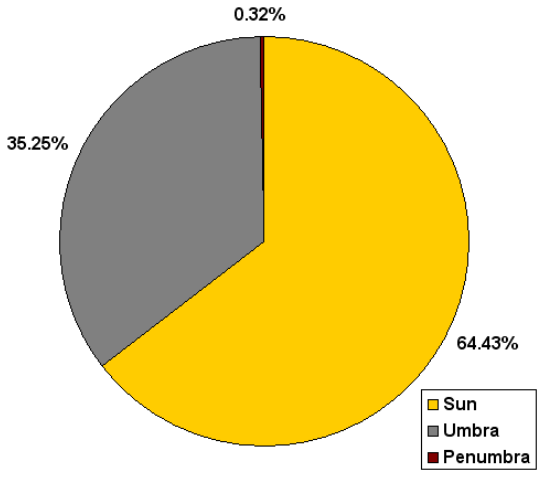
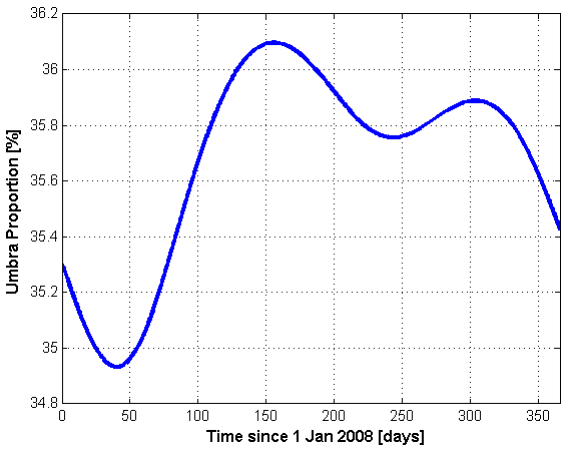


Figure 2.12: Contact area, IRS ground station



(a) mean proportion of sun, umra and penumbra (Orbit 01)



(b) Change in the sun phase over a year (Orbit 01)

Figure 2.13: Sun, umbra, and penumbra

The orbit Epoch selected for the simulation is 1st of Jan, 2012 at 00:00:00.000 UTCG

(MJD 55927.0), orbit definitions for the other parameters are listed in the Table 2.5. More details about the orbit definition is provided in [10]

2.6.2 Contact to IRS ground station

A minimal elevation angle of 7° is required for any contact to the IRS ground station. This value is based on experiences of present ground stations and will be used for the accessibility prediction of the FLP. Values of the expected contacts per day varies from 4 to 6 contacts per day depending upon the altitude of the orbit. Contact area of IRS ground station is shown in the Figure 2.12. Further details are provided in [10].

2.6.3 Sun, umbra and penumbra phases

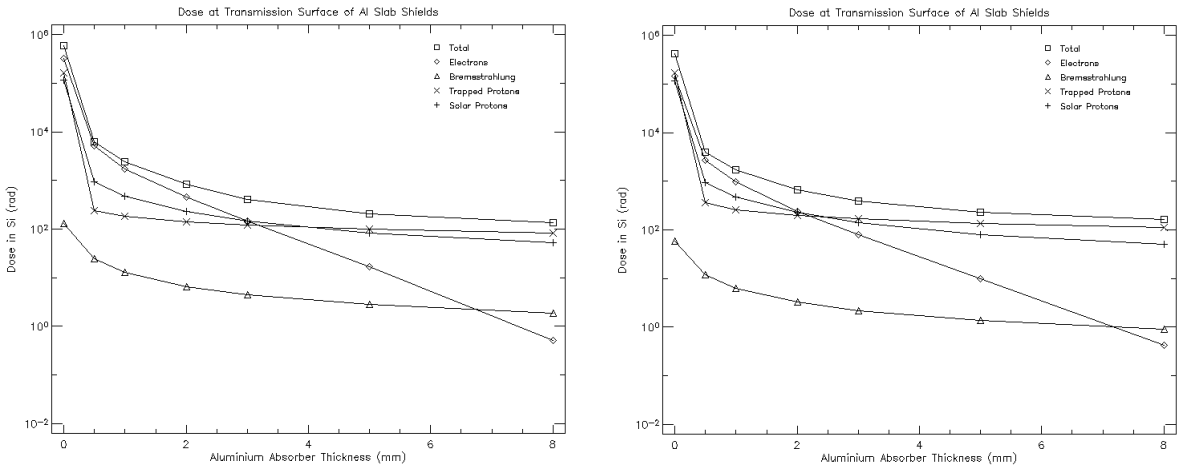
The satellite stays most of the time in the sun light during a complete orbit. Depending upon the orbit the sun proportion varies between 62.7% to 70.2%. The partial phase of eclipse or penumbra is relatively very small and is negligible for most of the simulations as its percentage is below 0.5%. The eclipse phase or Umbra varies from 29.44% to 36.97% of the orbital time period depending upon the orbit. The different phases of the orbit-01 are shown in the Figure 2.13a. The proportion of the phases where the satellite travels in the sunlight, the earth shadow or in the transition of one to another is also not constant during the year and there is a slight change in these parameters. Figure 2.13b shows the change during one year for the orbit-01.

2.7 Radiation environment

SPENVIS [11] is an on-line tool developed by Belgian Institute for Space Aeronomy. SPENVIS was used to calculate the expected total ionization dose (TID) for the different possible orbits of the FLP. The simulations were carried out for the mission design life time of two years.

To calculate the ionizing radiation dose, radiation parameters should be calculated for the expected solar activity. There are two solar activity models present in SPENVIS which are solar maximum and solar minimum and there are two magnetosphere models which are available in SPENVIS which are stormy magnetosphere and quiet magnetosphere. There are total four possible combinations but only two combinations, stormy magnetosphere/solar maximum and quiet magnetosphere/solar minimum are simulated to

predict the worst case scenario. The radiation sources considered are Trapped Proton and Electron Fluxes and Solar Proton Fluences.



(a) TID Solar Maximum (Orbit 01) (b) TID Solar Minimum (Orbit 01)

Figure 2.14: Total Ionization Dose (Orbit 01)

AP-8 is used as Proton model and AE-8 is used as Electron model for the simulation purpose. These models are considered to be the standard models for the calculation/-modeling of radiation effects as they are the only models that cover all possible radiation effects. Within these models there is an option for selecting solar maximum or solar minimum models. Both of the options have been utilized to obtain the results. Simulations are also performed for the CRRESPRO/CRRESELE Model present in SPENVIS, setting quiet magnetosphere for the solar proton models and average flux for the trapped electron model The default model JPL was used for the calculation of solar proton fluences. The radiation dose was calculated for the simple geometry and dose model SHIELDOSE2 was used to calculate the results. The shielding configuration was set to finite Al slab and the target material was set to be silicon. The results are shown in the Figure 2.14. The results are also summarized in the Table 2.6. Further details about this study could be found in [12].

Table 2.6: TID for different orbits

Orbit	Altitude [km]	Inclination [°]	Total Ionization Dose [rad] For 0.001 mm Al			Total Ionization Dose [rad] For 3 mm Al		
			Solar Min	Solar Max	CRRESELE Model	Solar Min	Solar Max	CRRESPRO/ CRRESELE Model
01-03	600	97.79	42390	60140	484700	389.3	415.8	1136
04-05	500	97.40	27980	38700	466800	277.1	303.4	932.4
06-07	900	99.03	155900	223200	526800	1063	1079	2375

CHAPTER 3

ACS Hardware

This chapter gives the detail description about the ACS hardware components, their specifications, their placement in the satellite as well as their communication and power interfaces. As described in the previous chapter the attitude knowledge is acquired using five different kind of sensors and the actuation is provided with two different kind of actuators. The detail about ACS components is presented in Table 2.1. The following sections will give a detailed description of each hardware and will also discuss the placement of this hardware in the body, how it communicates and how the power connections are realized.

3.1 Sensors

Five different kind of sensors will generate the required attitude information to meet the mission definitions and they will also allow ACS to perform its functions in all the mission phases.As the pointing accuracy requirements on the ACS of Flying laptop are quite stringent so star tracker is selected to give fine pointing knowledge. As an inertial pointing reference, the star tracker provides an absolute attitude knowledge less than $3''$ for rates below 1 °/sec.

Sun sensors are selected to orient the solar panels toward the sun to ensure power supply when required by the power supply system. The will additionally provide the coarse attitude information during the safe mode.

Figure 3.1: ZARM AMR Magnetometer

GPS is required to conduct GENIUS experiment and it will additionally assist in the determination of satellites position and velocity vectors on-board. GPS will also be utilized to give the nadir and the target vector on the surface of earth. This orbital information is also helpful in conducting different house keeping operations such as routine tracking for health monitoring, payload data processing and annotation, orbit maneuver planning etc. Use of GPS eliminates the requirement of earth sensor or horizon sensor for earth pointing and also the accuracy provided by the GPS for nadir pointing in a low earth orbit is much better than the earth sensor.

Use of FOG will not only contribute to the main objective of FLP technology demonstration but its measured body rates will also help in controlling the satellite. Magnetometers are used to measure the earth magnetic field in orbit and they will also support the functioning of Magnetic torquers. The detailed explanation of the ACS hardware is described in the following sections:

3.1.1 Magnetometers

The ZARM-Technik AMR Magnetometers are selected to conduct earth magnetic field determination and for the prediction of the torque applied by magnetic torquers during the satellite operation. The ZARM-Technik AMR Magnetometer is a micro-controller based magnetometer with a digital output and therefore very easy to use [13]

An integrated three-axis Anisotropic-Magnetoresistive (AMR) sensor HMC-1023 from Honeywell measures the external magnetic field in all the three directions. The AMR sensor

Table 3.1: Specifications of MGM

Parameter	**Digital AMR-MGM**
Number of Axes	Three, orthogonal
Field Measurement Range	$\pm 150 \mu T$
Sensitivity	8.5 nT
Accuracy	$<\pm 1\%$
Sampling Rate	60, 120 and 240 Hz
Output Rate	1.5, 3 and 6 valuess
Supply Voltage	5V to 16 V
Interface Communication	Serial: RS422
Baudrate	57600 Baud
Power Consumption	<0.3 W 12 V
Temperature Range	-40 °C to 85 °C
Dimensions	56 x 36 x 17 mm
Mass	<50 g

uses three independent permalloy bridges for magnetic measurement. The output is linear proportional to the applied external magnetic field. The sensor uses a special set/reset-technology to eliminate temperature offset drifts and to maximize the output sensitivity. The analog sensor output voltage is converted into digital value by a precision sigma delta A/D converter. The digital communication interface is RS422. A temperature sensor is also integrated for the temperature compensation.

3.1.1.1 Specifications

The technical data of the digital AMR-Magnetometer is summarized in Table 3.1

3.1.1.2 Placement in the satellite

The MGMs are mounted on the cover plate of the payload module. Magnetometers are not placed in the direct neighborhood of MGTs as there is a possibility of magnetometer saturation due to the magnitude of the magnetic field produced by MGTs. Further details regarding this topic could be found in [14] Orientation of MGMs in FLP body axis is shown in the Figure 3.2.

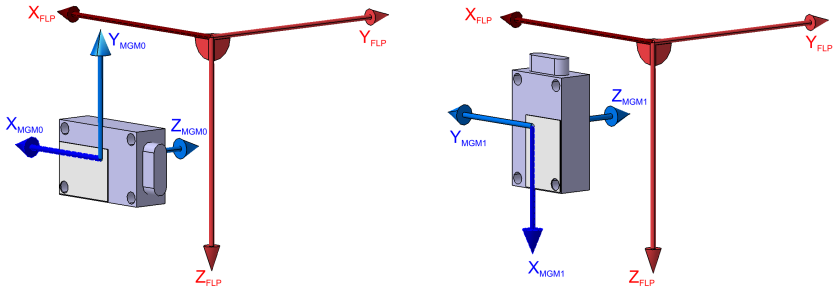
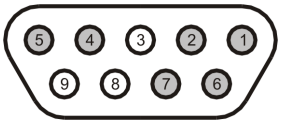


Figure 3.2: Orientation of MGMs in Body Axis

3.1.1.3 Electrical interface

The AMR-Magnetometer uses a micro miniature nonmagnetic female 9-pin Micro-D connector. The pin assignment of the Micro-D connector is shown in the Table 3.2. Pin 9 (A) and Pin 5 (B) are MGM input pins while Pin 4 (Y) and Pin 8 (Z) are MGM output pins.

Table 3.2: Pin Assignment of Mgm Connector



Pin	Signal
1	Power Return, Signal Ground
2	Power, VCC
3	not connected
4	(Y) TxD +
5	(B) RxD -
6	not connected
7	not connected
8	(Z) TxD -
9	(A) RxD +

Between Pin 2 (Power, VCC) and Pin 1 (Power Return, Signal Ground) a supply voltage higher than 4.9 V, but not exceeding 12 V, is required for MGM operation. The MGM is not protected against reverse polarity or higher voltages than 12 V.

The MGM flight models are operated at 12 V. During the tests the MGM was operated with a stabilized power supply at 5.13 V because there is no difference in the MGM's performance when the supply voltage is within the specified voltage range.

The measured current consumption in the Power-Up mode at a supply voltage of 5.13 V is approx. 16.22 mA, the measured current consumption at normal operation (Read Data mode) is at approx. 16.33 mA. In stand-by mode the current consumption is

approx. 4.41 mA. All values are within the range specified by ZARM-Technik (less than 17 mA during operation, less than 5 mA in stand-by mode).

The MGM interface diagram is shown in the Figure 3.3. After the MGM is connected to the power supply, the state of the MGM is On (Power-Up mode). It was observed that a few bytes were generated randomly during the Power-Up phase until the supply voltage is stable. These values have to be ignored and deleted from the buffer before the communication can start.

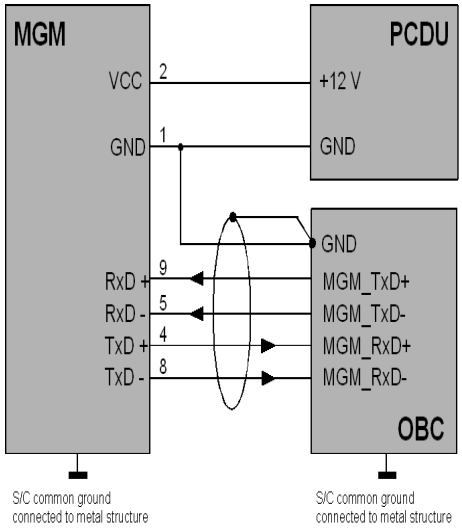


Figure 3.3: Interface diagram of MGM0 and MGM1

The serial communication is always started by one start bit which is followed by 8 data bits and one stop bit. There is no parity bit. The commands needed for magnetometer communication consists of only one byte. A hexadecimal value of `0x98` is used for Power-Down/Stand-By mode and `0x99` is needed for Power-Up. The magnetometer data and the temperature could be read by sending the Read-Data commands to the magnetometer. Three different sample rates using four different commands are selectable. `0x50` is used for slow, `0x51` is used for medium and `0x52` is used for fast sampling rate. For the FLP the fast sampling rate is selected to minimize the time required by magnetic torquers to be switched off.

After the measurement a response data is sent back by the MGM. The response data consists of 2 bytes for each sensor axis (X, Y, Z) and the temperature followed by one command byte for acknowledgment (Power-Up/Power-Down) . The response sequence is: channel X, channel Y, channel Z and the temperature. For better understanding

Table 3.3: MGM command description table

Command	Send	Response	Description
Power Up/Down	0x98	0x98	Off (Power-Down/Stand-By)
	0x99	0x99	On(Power-Up)
Read Data	0x50	8 bytes + 0x50	Sample rate slow
	0x51	8 bytes + 0x50	Sample rate medium
	0x52	8 bytes + 0x50	Sample rate fast

the command values are tabulated in Table 3.3. A detailed description of magnetometer interface is provided in [15]

3.1.2 Sun sensors unit

The sun sensors units are employed to provide the coarse sun direction which is required in orientation of the solar panels toward the sun during safe mode and idle mode. The sun sensors units are in house developed and each unit consists of two solar cells having dimensions of 20 cm x 20 cm. The solar cells are GaAs solar cells of type 3g-27% manufactured by Azur Space Solar Power GmbH. The bonding of the solar cells on solar cell plate made of carbon fiber is performed using the facilities of Astrium Ottobrunn using space qualified manufactureing process. Eight of such sun sensor units are used to obtain the complete 4π view.

3.1.2.1 Specifications

The solar cells are analog, single axis sensors which are very small size, low mass and low power and they provide coarse sun vector determination with 180° field of view. Two solar cells are used in one sun sensor unit to obtain hot redundancy. This is shown in the Figure 3.4

In ambient temperature of 25°C and solar irradiation of 1365 W/m^2, the solar cells generates an open loop voltage of about 2.5 V and a short circuit current of about 62 mA as mentioned by the manufacturer [16]. The characteristics are also verified after manufacturing two Experimental Models (EM). The specifications of the solar cells are stated in the Table 3.4.

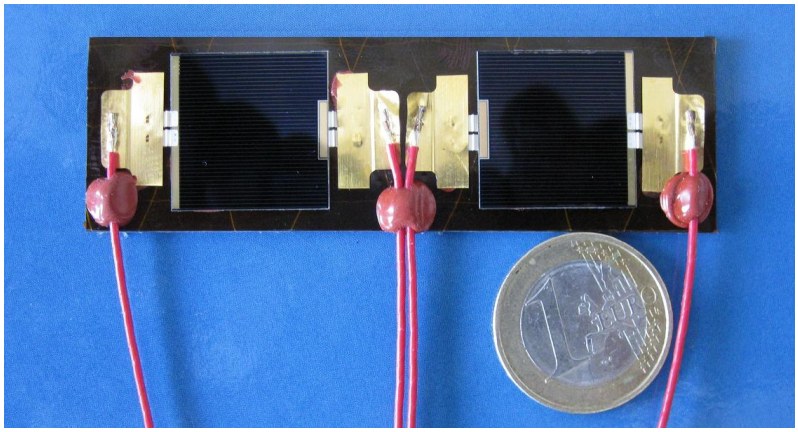


Figure 3.4: SuS Unit (EM)

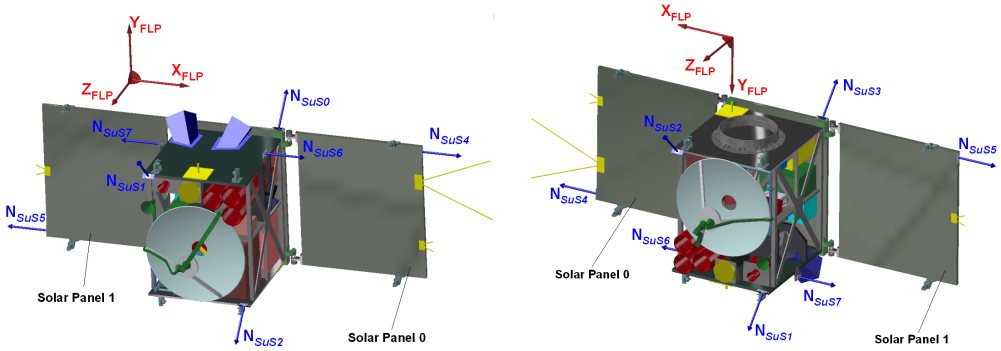


Figure 3.5: Location of SuS Units in FLP

Table 3.4: SuS Unit Specifications

Parameter	SuS Unit
Number of Solar Cells	2 per SuS Unit
Accuracy	5°
Field of View	2π
Mass	$\approx 120g$
Dimensions	80 mm x 25 mm x 1.5 mm
Average Efficiency	26.8%
Average Open Circuit Voltage	2.575 V
Average Short Circuit Current	62 mA

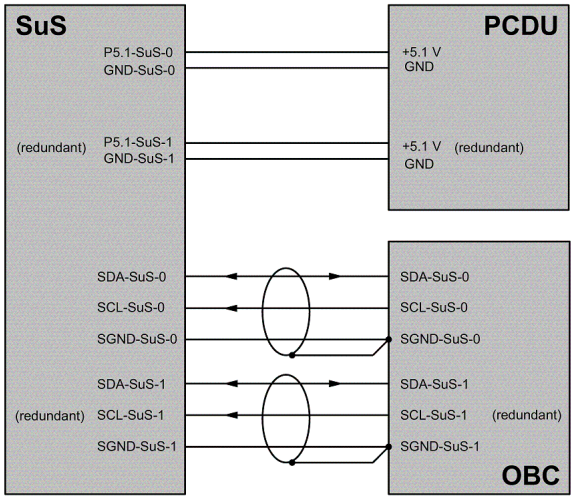


Figure 3.6: SuS Interface

3.1.2.2 Placement in the satellite

Four sun sensors are mounted at a 45° angle at the edges of the cubical structure. Two sensors are mounted on solar panel 0 and solar panel 1. Another two sensors point in the solar panel directions but are mounted high enough that they can look over the not yet deployed solar panels. The sensors mounted on both of the solar panels will be used after the deployment of the solar panels. The whole system is shown in Figure 3.5. The sensor pointing vectors span an orthogonal system with a minimum of 6 sun intensity measurements, each with a 2π field of view, which are selected depending on the solar panel deployment [14].

3.1.2.3 Electrical interface

The system of SuS electronics is designed in a redundant way with two identical separated circuits. Four ADCs are used to achieve the redundant design. I^2C bus is used for the communication between SuS electronics and OBC.

The connection of OBC/PCDU to the SuS power electronics is realized by using the MDM 15 pol plug connector. The connector contains both, the voltage and the communication signals. The interface diagram of the SuS Power Electronics to the OBC/PCDU is shown in the Figure 3.6.

A Sub-D (50 pol.) connector is used for connecting the sensors with on-board electronics. Figure 3.7 shows the the electrical design of the sun sensor PCB. As from the

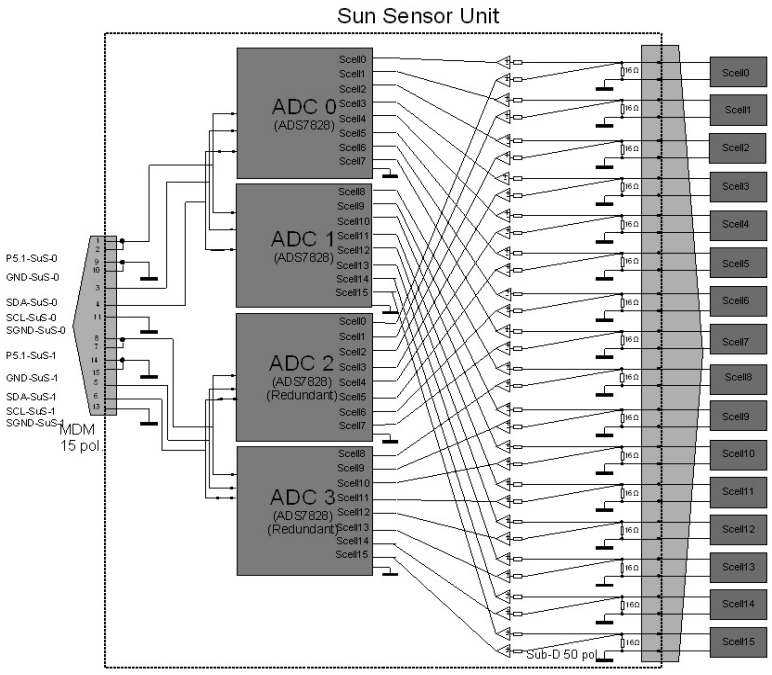


Figure 3.7: SuS Electronics Unit

Figure 3.7 it is shown that the PCB is designed in a redundant way with two identical separated circuits.

The value of the shunt resistor is 16 Ohms and ADS7828 is used for converting the analog signal into digital signal. The ADS7828 is a single-supply, low-power, 12-bit data acquisition device that features a serial I^2C interface and an 8-channel multiplexer. The Analog-to-Digital (A/D) converter features a sample-and-hold amplifier and internal, asynchronous clock. Each of these circuits can be controlled from the OBC by its own separate I^2C bus.

Figure 3.8 shows the flow control diagram. OBC is the master and ADS7828 is connected as slave to the interface. Redundancy is realized by two separate and identical circuits. Each of the circuit is controlled by a separate I^2C bus. ADC 0 and ADC 1 are connected with the first I^2C bus and ADC 2 and ADC 3 are connected with the second I^2C bus. The start condition is initiated by the Master (OBC) and after that slave address is transmitted by the master. The slave (ADC 0) then returns an acknowledge bit. Next, data bytes are transmitted by the slave. The master returns an acknowledge bit after receiving each byte except for the last byte. At the end of the last received byte , a not-acknowledge bit is returned. The master device generates all of the serial clock pulses and

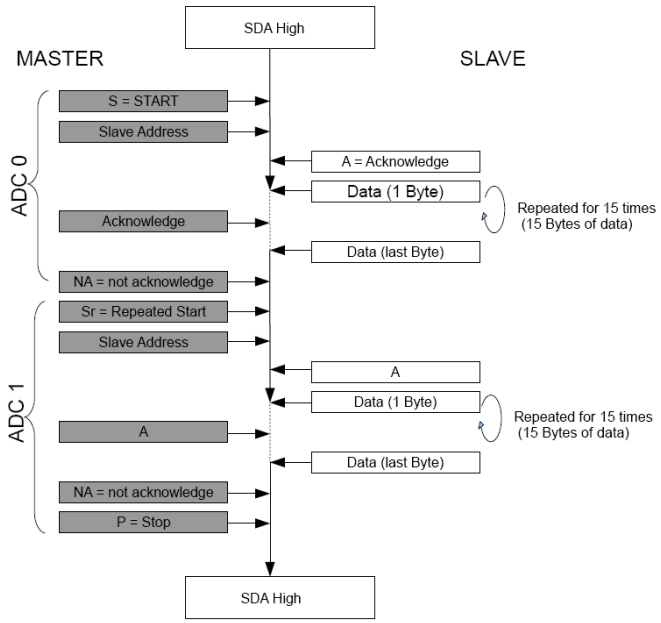


Figure 3.8: Communication Flow Diagram

the start stop condition. A transfer is ended with a Stop condition or a Repeated start condition. Since Repeated start is also a beginning of the next serial transfer therefore the bus is not released. Further details describing functioning of sun sensor in FLP are documented in [17].

3.1.3 GPS system

The GPS system in FLP consists of three separate GPS units. GPS would be used to derive the nadir coordinate system required by nadir pointing and target pointing control. In addition a technology demonstration experiments GENIUS (GPS Enhanced Navigation Instrument for the University of Stuttgart's micro-satellite) would also be conducted with the help of GPS system.

GENIUS is developed as an experiment for accurate determination of the spacecraft attitude. As a novelty, all receivers are driven by an single on-board oscillator. Raw code and carrier phase measurements will be recorded and dumped during ground station contacts for off-line analysis. Each Phoenix GPS receiver is connected to a separate GPS antenna and low noise amplifier. The antennas are placed on three corners of the satellite in an L-like arrangement. By synchronizing the reference oscillators of three receivers to a single oscillating source the accuracy of attitude determination can be increased leading

to real-time position, velocity and timing information with envisaged accuracies of 10 m, 0.1 m/s and 1 μs. This experiment is conducted in cooperation with the German Space Operations Center (DLR/GSOC). The Phoenix receiver is a commercial GPS receiver board with a new developed firmware for space and high dynamics applications. Following are the goals set for this experiment.

- ◇ The GENIUS GPS system shall offer real-time position, velocity and timing information with envisaged accuracies of 10 m, 0.1 m/s and 1 μs, for on-board usage.
- ◇ The three receivers shall assure a high level of redundancy for GPS on-board navigation.
- ◇ All receivers shall be driven by an ultra stable 10 MHz crystal oscillator on-board the Flying Laptop. This offers the possibility to study GPS navigation in situations, where less than four satellites can be tracked simultaneously.
- ◇ An orbit determination of down to 1 m shall be achieved.
- ◇ The spacecraft attitude shall be determined with an accuracy of 0.1° to 1° by off-line analysis of the GPS carrier phase measurements.
- ◇ On-the-fly software uploads shall be supported.

3.1.3.1 GPS system overview

The GPS system consists of

- ◇ Three GPS antennas
- ◇ Three Low Noise Amplifiers (LNA)
- ◇ The GPS receiver box containing three GPS receivers boards and an interface board integrated in a common aluminum housing

The GPS system is illustrated in Figure 3.9 and Figure 3.10

3.1.3.2 Specifications

GPS antenna

The three GPS antennas are mounted on the middle solar panel. The chosen antennas are of type S67-1575-20, L1 Series operating at 1575.42 Mhz. The chosen models are the smallest ones of the available series and the other models of the same company with

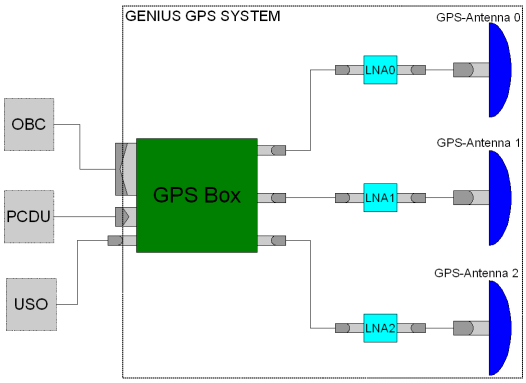
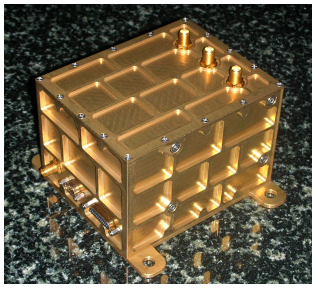
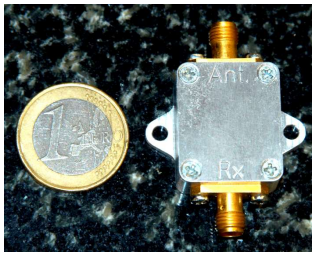


Figure 3.9: GPS System Overview



(a) GPS Box

(b) GPS Antennas



(c) GPS LNA

Figure 3.10: GPS System Components

the same manufacturing process have space flight heritage over large satellites. The parameters of antenna are listed in the Table 3.5

Low noise amplifier (LNA)

A miniature preamplifier for the Phoenix GPS receiver has been design by DLR based on M/A-COM's AM 50-0002 low noise amplifier chip. More details about LNA could be found in [18]. The parameters of DLR LNA are listed in the Table 3.6

GPS box

GPS box is in house developed and its details could be found in [19]. GPS box contains three Phoenix receivers and an interface board. The Phoenix GPS receiver is a miniature GPS receiver specifically designed for high dynamics applications such as sounding rockets and low Earth satellites. The receiver is based on SigTechs commercial-off-the-shelf

Table 3.5: GPS Antenna Parameters

Parameter	Value
Size	Diameter: 88.9mm, Width: 10.2mm
Weight	85 g
Connectors	TNC jack female
Frequency	1575.42
Impedance	50 Ω
Input voltage standing wave ratio	2:1
Output voltage standing wave ratio	1.5:1
Maximum Input Power	17 dB
$V_{DD}(nom/max)$	5/10 V
Temperature	-55°C to +85°C
Total Ionization Dose	>30 krad at 1-8 rad/sec

MG5001 receiver board and has been qualified for space use in a series of thermal-vacuum, vibration and total ionization dose tests. It employs a GP4020 baseband processor which combines a 12 channel GP2021 correlator and an ARM7TDMI microprocessor kernel. At a power consumption of less than one Watt and a board size of 50 x 70 mm the receiver is among the smallest of its kind and particularly well suited for satellites with limited on-board resources. The Phoenix receiver is extensively used in European sounding rocket missions and has been selected for the Proba-2, X-Sat, ARGO and Flying Laptop micro-satellites as well as the PRISMA formation flying mission.

DLRs proprietary receiver [18] firmware enables code and carrier measurements with a noise level of 0.4 m (pseudorange) and 0.5 mm (carrier phase) at a representative carrier-to-noise density ratio of 45 dBHz. The carrier phase measurements are ensured to exhibit integer double-difference ambiguities which allows use of the Phoenix receiver for carrier phase differential GPS applications such as formation flying and attitude determination. Despite a high raw data quality, the achievable single-point navigation accuracy is, however, limited to typically 10m (3D rms) due to broadcast ephemeris errors and unaccounted ionospheric path delays.

3.1.3.3 Placement in the satellite

The flat antennas are mounted on the middle solar array in an L-like arrangement as shown in the Figure 3.11. This direction is opposite to the payload cameras hence it will

Table 3.6: GPS LNA Parameters

Parameter	Value
Size	28.2 mm x 21 mm x 10.5 mm
Gain	25/27/29 dB
Noise	6061-T6 Alu/thermoset plastic
Finish	Skydrol resistant enamel
Connectors	Two SMA straight jack female
Frequency	1.575 GHz (1.4 to 1.8 Ghz)
Impedance	50 Ω
Temperature	-40°C to +85°C

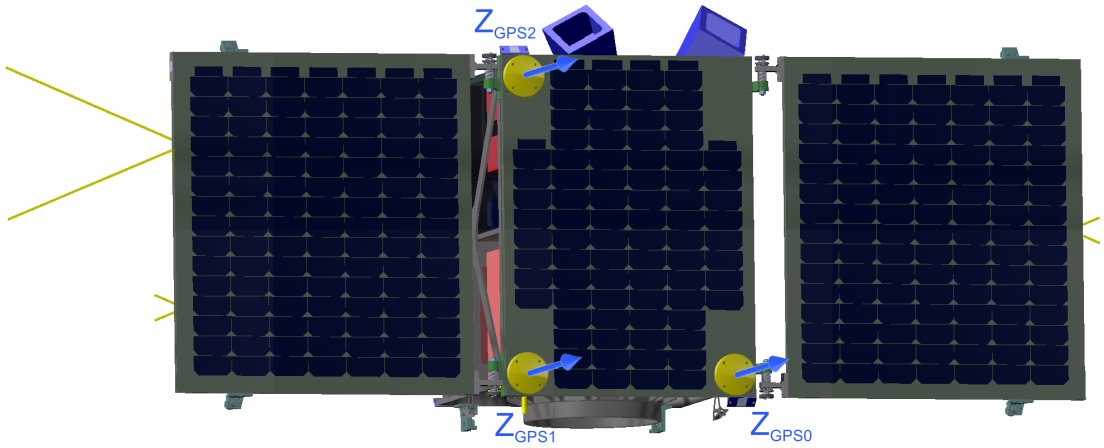


Figure 3.11: Arrangement of GPS Antennas in FLP

provide the optimum reception of signals from GPS satellites during earth observation. The distance between GPS0 and GPS1 antenna is 440 mm and the distance between GPS1 and GPS2 antenna is 610 mm. [14].

3.1.3.4 Electrical interface

The electrical interface is shown in the Figure 3.12. Connector J1 is 25 pin MicroD connector which is used for communication of data between GPS box and OBC. Connector J2 is a 9 pin MicroD connector used for power supply. Connector J3 is a SMA female connector used for Oscillator input to GPS box and J4, J5 and J6 are also SMA female connectors used for antenna signal input. Complete details of the electrical interface is presented in [19].

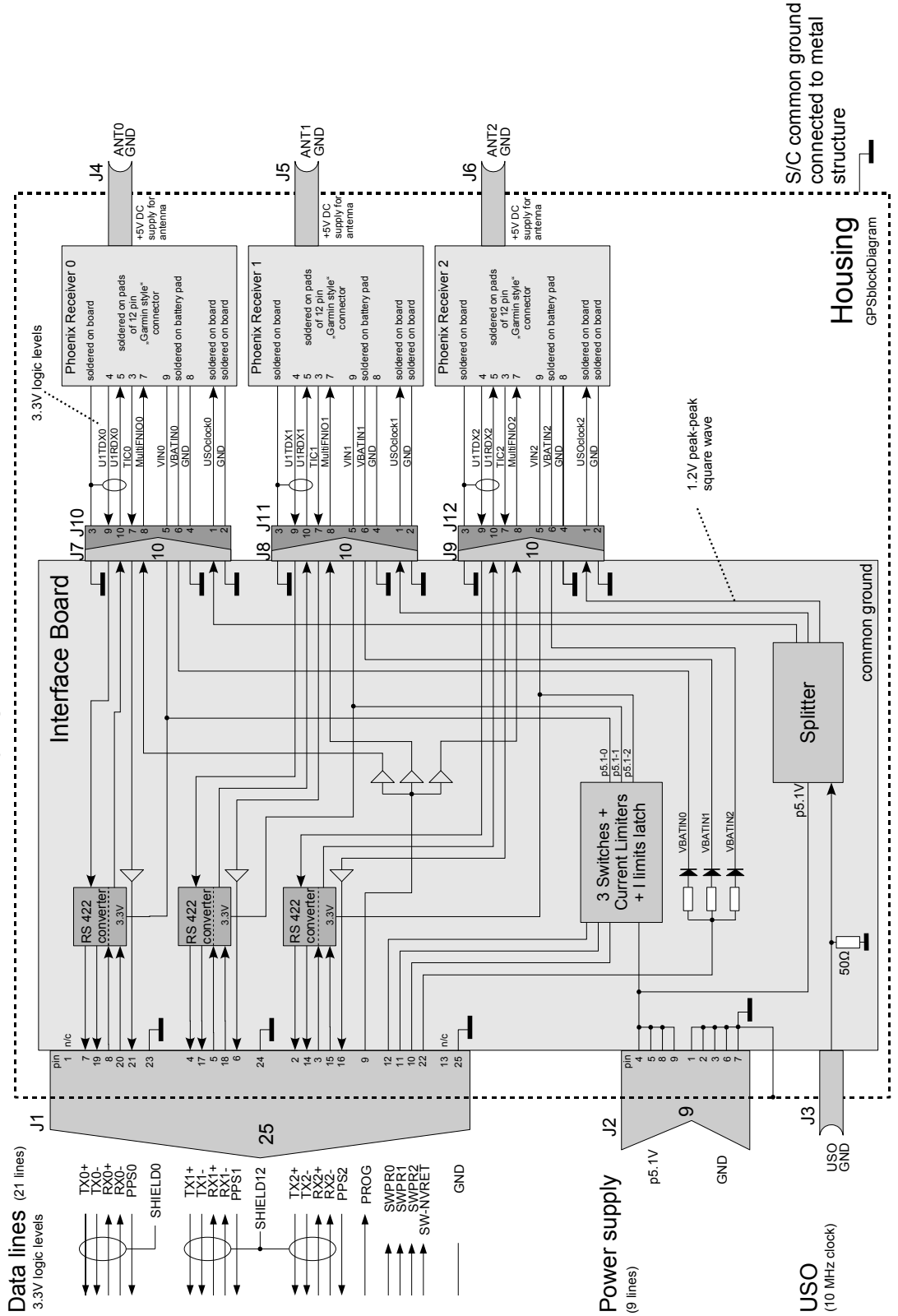


Figure 3.12: GPS Block Diagram

Table 3.7: Message sent from GPS receivers

Message	Content	Time Period	bytes
F40	Cartesian Navigation Data	1 s	104
F80	Filtered navigation data	1 s	104
F48	Configuration and status parameters	10	48
F62	Raw measurements and channel status	10	601

Several modifications were made to the Phoenix GPS receiver boards to make them suitable to achieve the FLP mission objectives. Figure 3.13 shows these changes which are also listed below.

1. The 12-pin plastic connector is unsoldered. The data/power wires are directly soldered to the pads of the removed connector and fixed by epoxy adhesive (Araldit AV 138 M / HV998).

2. The MCX antenna connector is replaced by a SMA connector (Rosenberger SMA 32 K246-400 E3). According to the ECSS standards, soldering directly to gold plating has to be avoided. Hence, the gold plating of the connector-pins was removed before soldering.

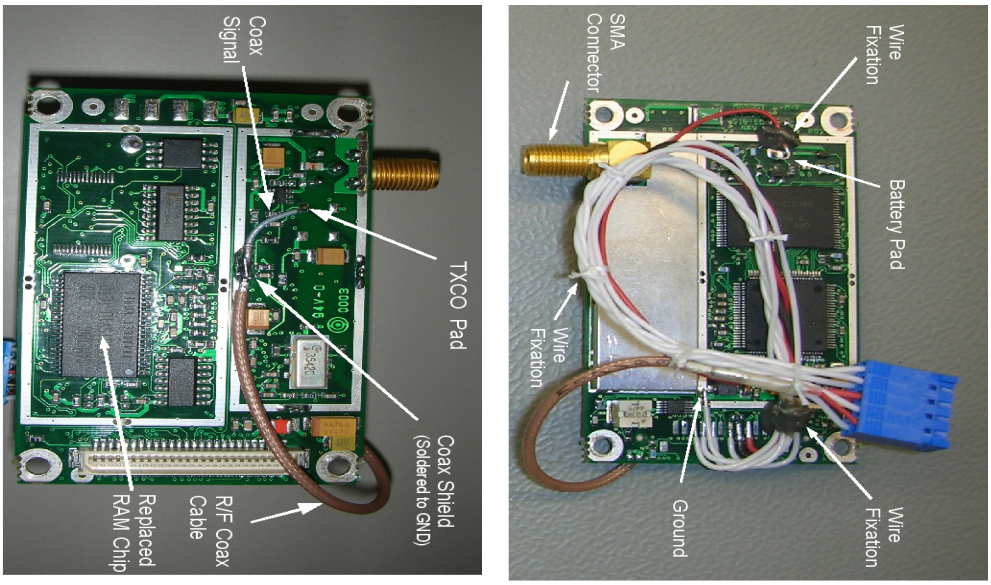
3. The lithium backup battery was removed. A keep alive +3.3V wire is directly soldered to the pad of the removed battery.

4. Due to the modified firmware the 256 kB RAM memory is replaced with a 512 kB RAM memory (Samsung K6R4016V1D-TC10).

5. The temperature controlled crystal oscillator (TCXO) is unsoldered. A RF coaxcable is used to feed the reference frequency into the Phoenix board.

There are two serial communication ports provided in the Phoenix GPS receiver. The second port is designed to receive differential GPS data, therefore this port is not utilized in FLP. Port A is the prime port for command input and message output.

The Phoenix receiver is pre-configured to regularly output 4 messages summarized in Table 3.7. The F40 message is transmitted every second and it is used for the ACS as it contains the position, velocity and time. The F48 and F62 message are read and stored every 10 sec for the attitude experiment and will be dumped during ground station passes. The F80 message is issued from the orbit propagator integrated in the firmware of



(a) GPS receiver bottom side (b) GPS receiver top side

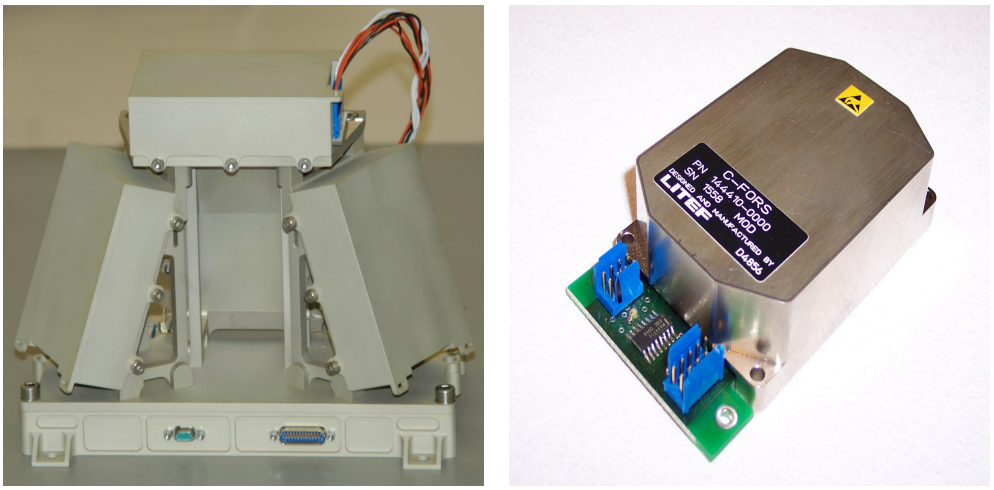
Figure 3.13: Changes made in GPS Phoenix board

Phoenix GPS receiver. This is called the extended navigation system (XNS). It consists of 24-state Kalman filter with a 50th order gravitational model. Due to the availability of such a high precision orbit propagator the internal ACS orbit propagator could be kept fairly simple in case of restricted GPS availability. The two-line element (TLE) orbit propagator contained in the firmware for signal acquisition aiding can be updated with the Load Orbit (LO) command. The TLEs need to be uploaded from the ground station on a regular basis and should not be older than a week.

Complete details about GPS unit are documented in [19] and [18]. The complete technical drawings, circuit diagrams and mounting instructions could be found in these references.

3.1.4 Fiber optic gyro unit

Fiber Optic Gyro unit is composed of four fiber optic gyros (FOG) used in a tetrahedron like configuration for the measurement of the body angular rates. The tetrahedron like configuration allows autonomous detection of a single failure. This configuration is shown in Figure 3.14a. The type of the sensor is C-FORS (Commercial Fiber Optic Rate Sensor) from Litef and it is shown in the Figure 3.14b. The sensor works on Sagnac effect, using a beam of coherent light that is split and propagated both clockwise and anticlockwise



(a) FOG Unit (b) FOG Sensor

Figure 3.14: FOG Unit and Sensor

through a coil of optic fiber. When the sensor experiences a movement around its axis, one half of the split beam travels further than the other half relative to an observer outside the coil. A photo-detector measures the phase-difference patterns between the two beams, which are directly related to the rotation. As the gyro uses the closed loop techniques, the output is linear and high angular rates can be measured

The FOG can be configured to one of three measurement modes

- ◇ Angle Increment: returns the angle the gyro has moved through since the last data output.
- ◇ Angle Accumulated: returns the total angle moved through since the last reset.
- ◇ Rate: In this mode, the gyro returns the current angular rate

In FLP, the measurements would be done in the rate mode of FOG. The output value is the mean angle increment that is turned through since the last data output, divided by the time elapsed since the last data output.

3.1.4.1 FOG sensor specifications

The specifications are summarized in Table 3.8. Further details could also be consulted in [20].

Table 3.8: FOG Sensor Specifications

Parameters	FOG C-FORS Sensor
Sensor Performance	
Rate bias	$\leq 2°/h$ (1 σ)
Random walk	$\leq 0.15°/\sqrt{h}$
Scale Factor	$\leq$ 1000 ppm (1 σ)
Axis misalignment	$\leq$ 10 mrad (Absolute) $\leq$ 1 mrad (Stability)
Maximum measurement range	$\pm$ 1000°/s
Initialization time	$\leq$ 120 ms
Mechanical Properties	
weight	$\leq$ 130 g
Dimensions	78 mm x 53 mm x 22 mm
Case	Hermetically sealed
Electrical Properties	
Power Consumption	2.8 W
Communication Interface	IBIS
Supply voltage	+5.0 V -5.0 V +3.3 V

3.1.4.2 Placement in the satellite

The axes of fiber optical gyros are arranged in tetrahedron configuration to avoid single sensor failure. The measurement axis of each FOG is aligned to the rotation axis of a corresponding RW. Due to magnetic sensitivity, the FOG unit is placed not closed to the MGTs. [14]. Each rate sensor measures the angular velocity around its z-axis as defined in Figure 3.15 [14].

3.1.4.3 Electrical interface

Each FOG sensor uses 3 voltages for power supply (i.e. +5 V, -5 V and +3.3 V) and the IBIS for synchronous serial communication. The power supply is generated separately for each FOG for redundancy reasons. The FOGs use one common bus for communication with the OBC. The possibility of disconnecting a FOG from the common bus is given,

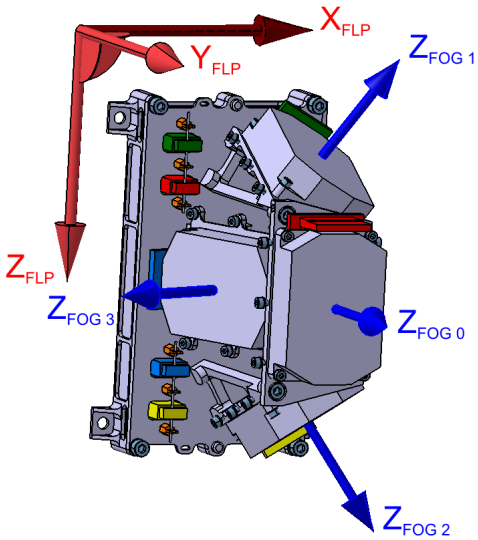


Figure 3.15: Arrangement of FOG unit

if the FOG is not working properly. Thus, the bus and communication with the other sensors is not affected.

Figure 3.16 and Figure 3.17 shows the detailed interface diagram of the FOG-unit. An interface to the PCDU is provided for separate power supply of each FOG sensor and an interface to the OBC is provided for symmetric communication between the FOGs and the OBC.

Each FOG uses an interface board which is equipped with a single connector for power supply and data transfer. The entire electronics is combined on the main electronic unit which consists of power supply unit, the communication bus IBIS, analog switches, and the signal converter. Each FOG is supplied with a voltage of +5.1 V by the PCDU. The +5 V line of each FOG is powered directly by the incoming +5.1 V which is within the tolerance range specified by the manufacturer. The voltages for the +3.3 V line and the -5 V line are generated at the FOG-unit using the voltage of +5.1 V provided by the PCDU.

A voltage of +5.1 V for the +5 V line is selected to minimize the risk of a fall of voltage between PCDU and the FOGs below the specified lower voltage limit. The value of the voltage on the +5 V line must not fall below +4.75 V according to specifications of a sensor provided by the manufacturer. This voltage takes into account the voltage ranges of the single components : 5 V ± 5% for the RS-422 signal converter, 3-16 V for the -5 V DC-DC converter and 2,7-14 V for the 3.3 V DC-DC converter.

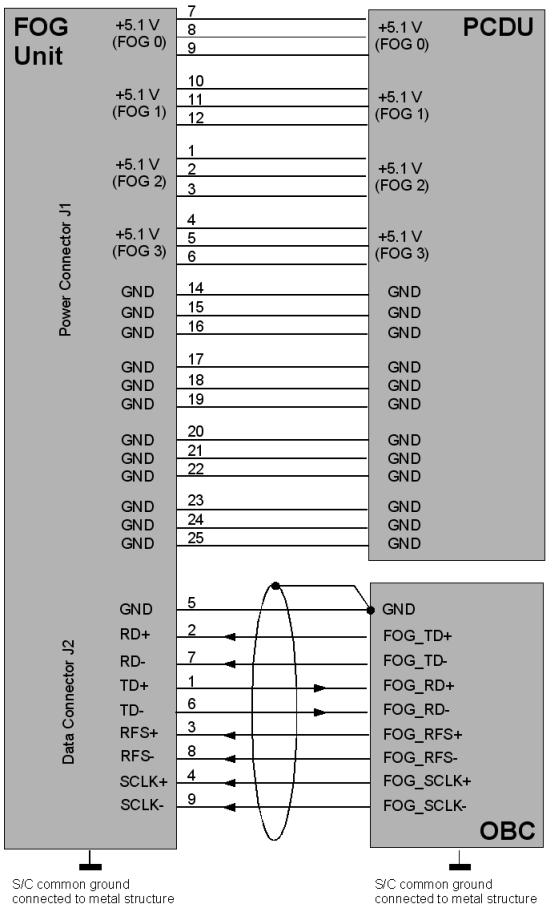


Figure 3.16: FOG Interface Diagram

A powered signal converter split into receiver and transmitter transforms the ground asymmetric signals from the IBIS into ground symmetric signals for data transmission between FOG-unit and OBC. According to Figure 3.17 the signal converter is directly powered by the voltage of +5.1 V provided by the PCDU which is transformed into +5 V to power the signal converter. The power line of each of the 4 FOGs is connected to the converter to ensure that the converter is always in operation when at least one FOG is switched on. To avoid interaction between a powered and a not powered incoming power supply line the lines are separated by diodes.Furthermore, if any FOG is powered down, it is disconnected from the IBIS by an analog switch. Thus, the bus is not affected by an defective and/or powered down FOG. When a FOG is powered down it is no longer visible to the IBIS due to the open switch.

An asynchronous UART and a synchronous four wire bus called IBIS (Integrated Bus for Intelligent Sensors) are available for communication with the C-FORS FOG sensor. On

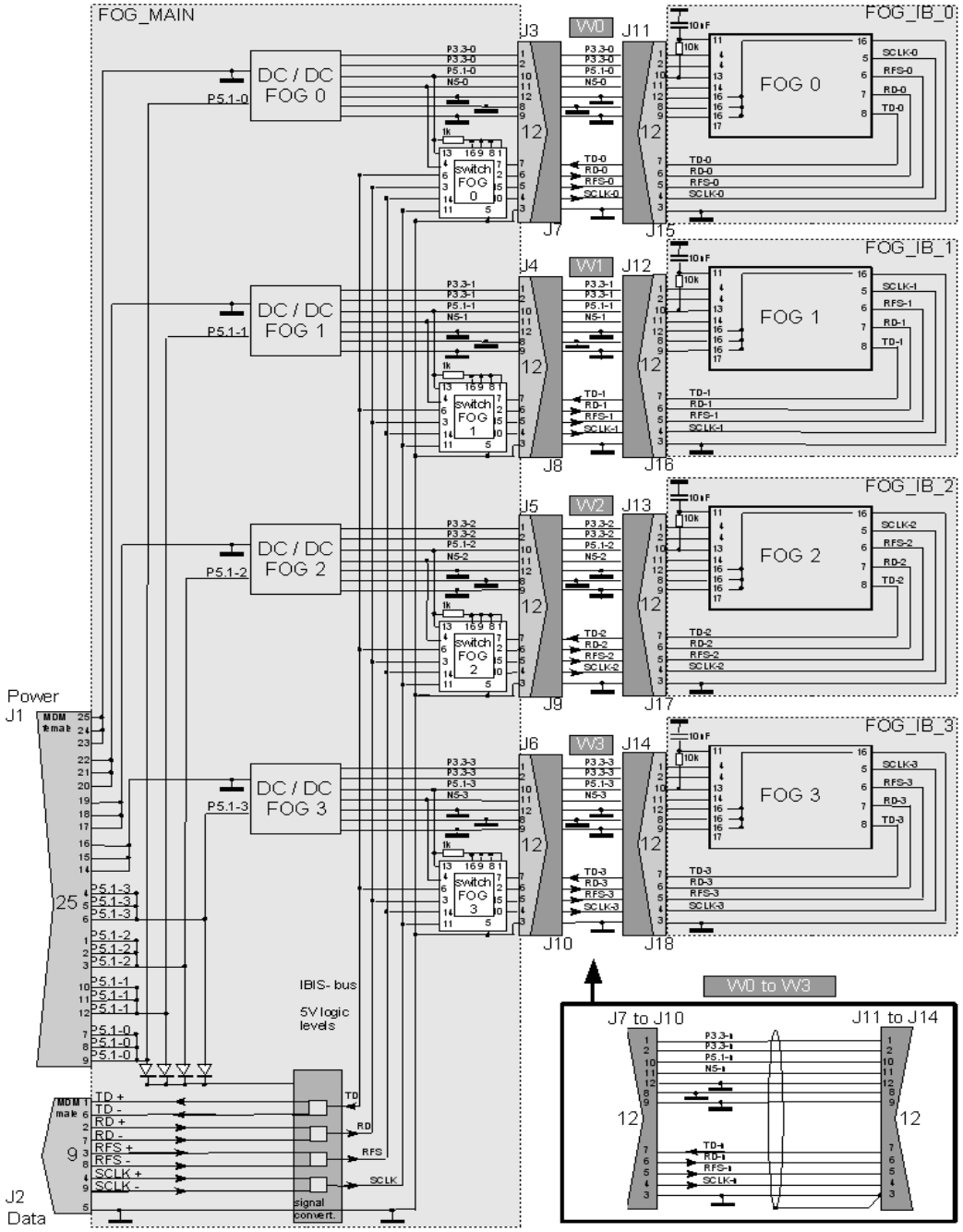


Figure 3.17: FOG Detailed Block Diagram

the Flying Laptop the IBIS interface is used because it is has the advantage to synchronize the 4 sensors and to control them via a single interface to the on-board computer, also the data request is issued to all four sensors at exactly the same time. Thus the data from each sensor is exactly synchronized with the OBC time. Moreover the synchronous interface allows a maximum data rate of 4 kHz while the asynchronous interface could support data rate of 1 kHz.

The rate sensors of the Flying Laptop are operated in slave mode i.e. the IBIS clock rate is produced by the master which is the OBC and is transmitted via the SCLK line. The maximum SCLK frequency for the FOG is 2048 kHz. The use of this clock rate is recommended by Litef [20]. The SCLK rate for the FOGs of the Flying Laptop is 2 MHz, because the clock rate of the OBC divided by the SCLK rate for the FOGs has to result in an integer value. The measurement range of a FOG sensor is set to $\pm 16.777216°$/sec and its resolution is 24 bits which leads to a least significant bit of 2.10^{-6} °/sec.

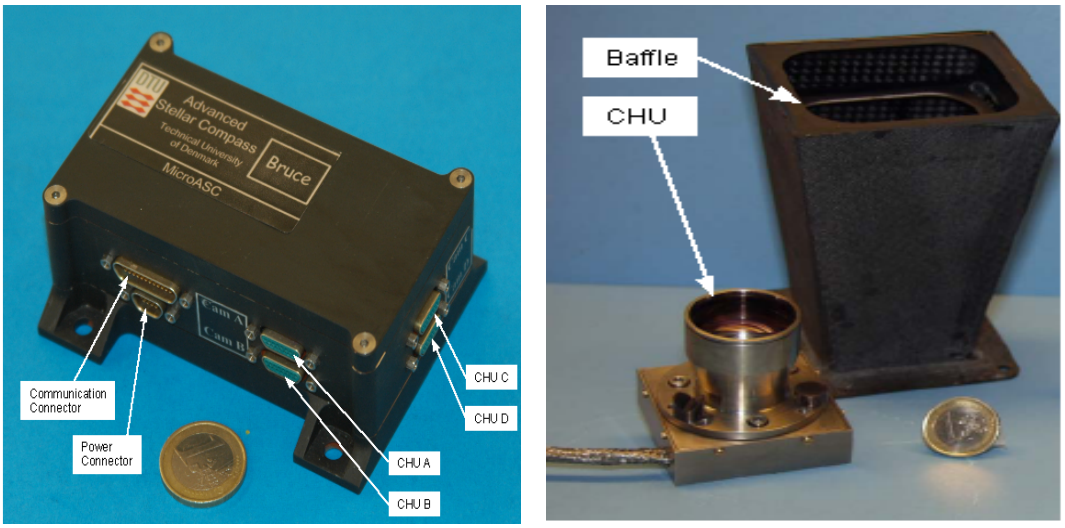
More information about the FOG unit, its communication with the OBC and its mechanical assembly could be found in the [21].

3.1.5 Star tracker system

The star tracker system is employed in FLP to obtain the fine attitude knowledge. The selected sensor is the micro-Advanced Stellar Compass (μASC) developed by the Technical University of Denmark (DTU). The μASC is the latest generation of the Advanced Stellar Compass (ASC) which was used in not less than 15 different projects carried out by NASA, ESA, CNES, DLR and other leading space agencies. The μASC is light in weight is more suitable for the small and micro-satellites.

The Star Tracker System of Flying Laptop consists of two camera head units (CHU), one micro-data Processing Unit (μDPU), and two baffles as shown in the Figure 3.18. Two CHUs are included in this design to avoid the Star Tracker blinding and to allow a single failure in this system. The μDPU could drive up to 4 CHUs and its electronics is fully redundant.

The μDPU uses 486 microprocessor to process the optical data acquired by CHU. It is equipped with 64 kB PROM, 8 MB flash and 8 MB RAM. During normal operation, the μDPU creates a digital image of the acquired analogue data received from the CHUs every 0.5 s and stores it in the internal RAM. In further computing, the lens distortion is compensated and the image is adjusted for adequately bright objects. The amount of bright objects that are found in an image can be influenced by varying a software



(a) μDPU (b) CHU and Baffle

Figure 3.18: Star Tracker System

parameter, but should never be below 16 stars, which is the minimum to guarantee reliable information. The developers recommend between 20 and 80 detected stars. The maximum figure of 199 stars could be employed.

3.1.5.1 Specifications

The performance envelop and the specification of μASC Star Tracker are listed in the table. [22].

3.1.5.2 Placement in the satellite

The CHUs are oriented on the satellite body so that simultaneous blinding or occultation by the Earth, the Moon and the Sun is avoided. The orientation is reached by aligning the cameras with the FLP's y-axis, spreading them by 56° and elevating them by -35° toward the z-axis. This results in an angle of 45° between the two bore-sights and an angle of 125° between the each bore-sight and the FLP's z-axis as shown in the Figure 3.19b . More details about the orientation and position could be found in 3.19b. The baffle system is mounted in order to minimize the effects from stray light and offers a rectangular Field of View (FOV) of 13.4° x 18.4° as shown in Figure 3.19c. Thus its diagonal covers 22.76°.

Table 3.9: Star Tracker Specifications

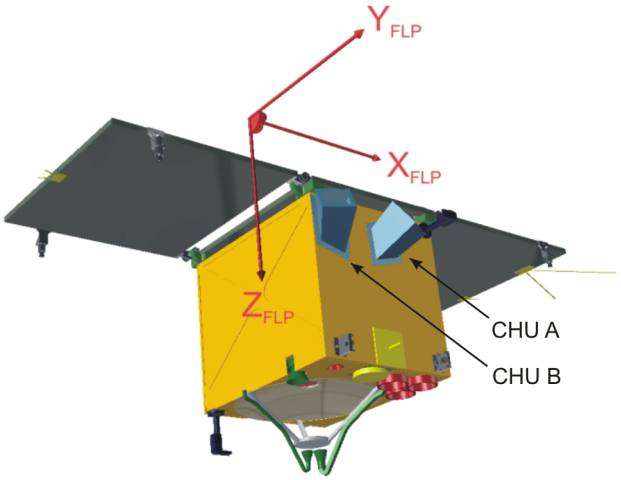
Parameter	μASC Component	Specification
Dimensions	Camera Head unit	50 x 50 x 54 mm including lens
	μDPU	100 x 100 x 47 mm
	Inner Baffle	76 x 76 x 130 mm
	Outer Baffle	212 x 228 x 152 mm
Mass	Camera Head unit	225 g including mounts
	μDPU	555 g including shielding
	Inner Baffle	65 g
	Outer Baffle	318 g
Precision [arcsec]	μASC	1 (1σ) pointing 8(1σ) roll
Rectangular Field of View	μASC	13.4° x 18.4°
CCD-Chip	CHU	7.95 mm x 6.45 mm 752x558 pixels
Temperature	μASC	-40°C to +70°C
Power	μDPU	3.7 W
	CHU	0.6 W
Communication	μDPU	RS422

3.1.5.3 Electrical interface

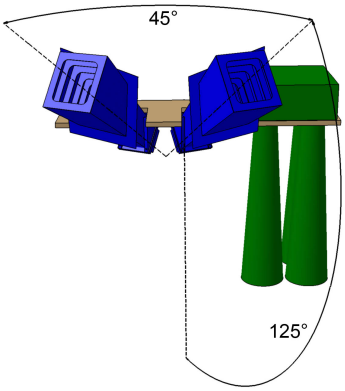
The power to the μDPU is supplied via Micro-D plug connector with 9 pins. The μDPU could accept unregulated voltages from 16.8 V to 75 V as stated by the manufacturer. The voltages for the CHU's are generated inside the μDPU. For FLP μDPU is supplied with 20 V. The power consumption of μDPU itself is 3.6 W. Each CHU connected to the μDPU increases the power consumption by 0.7 W. The electrical connections of μDPU with OBC and PCDU are shown in the interface diagram (Figure 3.20)

The μASC STR communicates with OBC in terms of packets using full duplex RS422 serial communication. A number of distinct packets have been defined. A list of TC and TM-packets accompanied by comprehensive explanations is presented in [22]. From a total of 18 available types of TC-packets FLP uses 10 different types and from 12 different available types of TM-packets FLP uses 7 different types of TM-packets. The details of these TC- and TM-packets used by FLP is presented in [23].

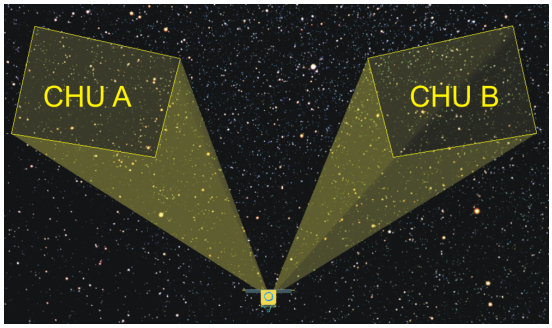
The μASC STR operates in seven different modes, which can be initiated by the



(a) STR in FLP Body



(b) CHU's Orientation



(c) CHU's FOV

Figure 3.19: STR Orientation in FLP

relevant TC-packet. The modes cover on ground modes for ACS testing and in flight operational modes. Three of these modes (i.e. Safe Mode, Stand-by Mode, and Attitude Mode) are used during flight and rest of the modes (i.e. Simulation Mode, Test Image Mode, Stimulator Mode, and Single Shot Mode) are used for ground testing. Safe mode is part of the power-up sequence. In the first step μDPU boots in the safe mode after switching on the power supply or on the issuance of `AscBootTC` command.

3.2 Actuators

Two different kind of actuators will provide actuation during the attitude control of FLP. To achieve agility and accuracy four reaction wheels are employed in a hot redundant

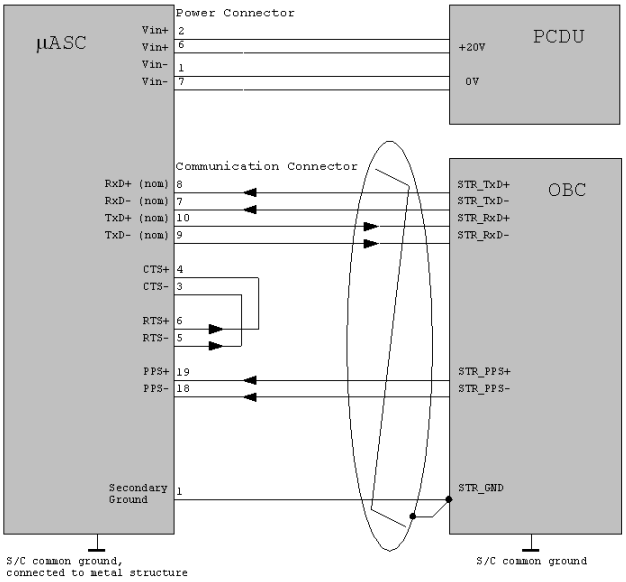


Figure 3.20: Star Tracker Interface Diagram

configuration. Each wheel could store angular momentum of 0.12 Nms resulting in a reaction torque of 5 mNm with in the rotational speed limits of ±2800 rpm. To dump the accumulated momentum by the reaction wheels, three magnetic torquers are also engaged as actuators in FLP. Magnetic torquers will additionally serve for actuation in detumble and safe mode of satellite where reaction wheels are not available. Their high reliability makes them ideal for the choice of detumble and safe mode. The linear dipole moment of each magnetic torquer is 6 Am^2. The double coil in these torquers makes them completely redundant.

3.2.1 Magnetic torquers

The magnetic torquer system consists of three magnetic torquers (MGT) and one MGT power unit which connects them to the OBC and PCDU as shown in the Figure 3.21. The selected magnetic torquers are of type MT6-2 produced by ZARM-Technik. Each MGT consists of two simultaneous winded coils around high permeability core material. Both coils are interfaced to their individual and independent interface connector. One of these coils is actively used and other is cold redundant. The magnetic torquers generates a magnetic dipole moment when a drive current is supplied to the coils which results in interaction with the earth's magnetic field. The result torque is a product of dipole moment of the torquer and the Earth'magnetic field.

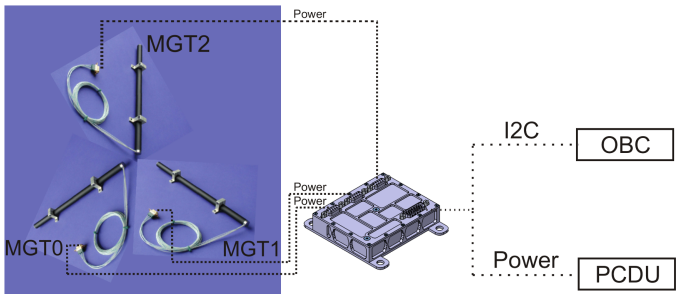


Figure 3.21: MGT Block Diagram

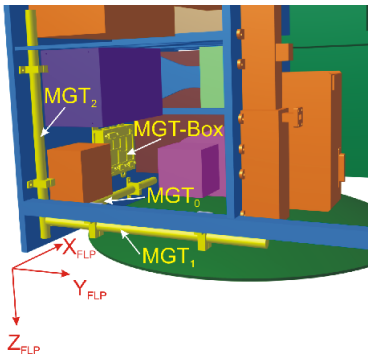


Figure 3.22: Placement of MGT in FLP

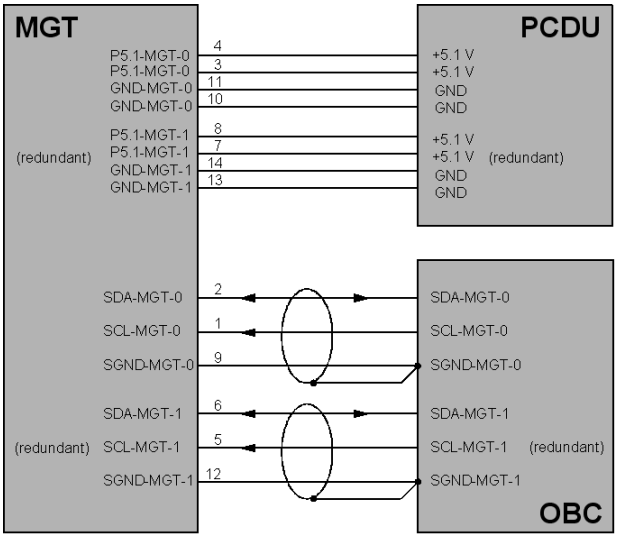


Figure 3.23: MGT Interface Diagram

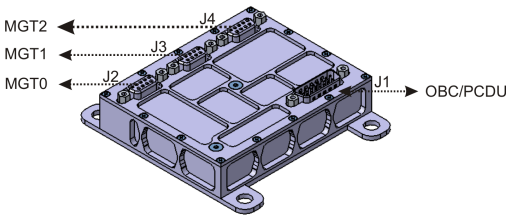


Figure 3.24: MGT Power Electronics Box

Table 3.10: Specifications of MGT

Parameter	Specifications
Linear dipole range	± 6 Am^2
Linear supply current	95 mA
Linear power cosumption	0.5 W
Linear supply voltage range (one coil)	± 5.0 V
Linearity error	$< 5\%$
No. of coils	2
Mass	< 0.25 kg
Dimensions	dia = 14.5 mm length = 325 mm

3.2.1.1 Specifications

The specification of MT6-2 MGT are presented in the Table 3.10 [24].

3.2.1.2 Placement in the satellite

The magnetic dipole vector for each torquer points to its positive x-axis. Because the generated magnetic field is rotationally symmetric the y- and z-axes do not need to be considered. Each MGT is assigned to one of the axes of FLP's body coordinate system [14]. MGT0 is aligned with the body X-axis of FLP, MGT1 is aligned with the Y-axis, and MGT2 is aligned with the body Z-axis of FLP as shown in the CAD model (Figure 3.22).

3.2.1.3 Electrical interface

MGT interface diagram is shown in the Figure 3.23 and MGT power electronics box is shown in the Figure 3.24. There are four connectors provided in the MGT power electronics box as shown. The connector J1 realizes the connection of MGT power electronics box to the OBC and PCDU while the connectors J2, J3 and J4 connects the MGT power electronics box to each MGT. The two coils of MGT can be set in three different states; i.e. off, positive current direction, and negative current direction.

The current from PCDU is passed through a H-bridge transistor DRV592VFP which based on ACS commands digitally actuates the MGTs with positive or negative current direction. This is also called bang bang actuation. As it can be seen in Figure 3.25 the PCB is designed in a redundant way with two identical separated circuits. Each circuit is

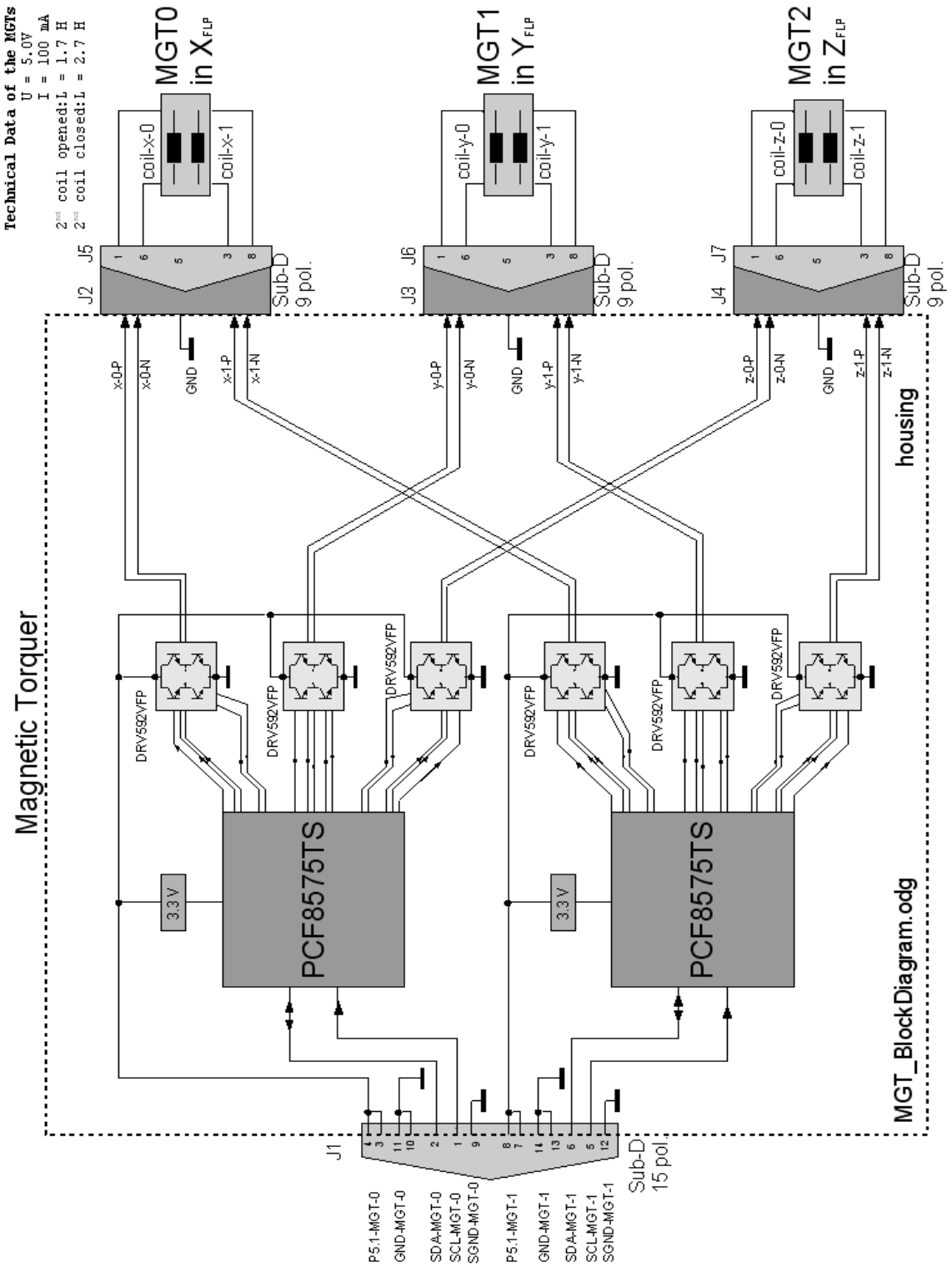


Figure 3.25: MGT Power Electronics

equipped with one PCF8575TS chip and three DRV592VFP chips. The PCF8575TS chip provides general purpose remote I/O for the OBC via the two-line bidirectional I^2C-bus. This chip allows to communicate in both directions, i.e. the PCF8575TS chip is an I^2C-bus slave transmitter/receiver. The receiver option is utilized to return the possible fault flags. The PCF8575TS chip is supplied with 3.3 V, derived from the 5.1 V supply voltage via a DC/DC converter. This allows direct communication with the OBC at the same voltage levels. The DRV592VFP chip is a H-bridge and drives the current to the MGT. The direction of the magnetic field (i.e. the current direction) depends on the output of the PCF8575TS chip.

As shown in the Figure 3.25 both the circuits communicates with OBC by its own separate I^2C-bus. The SCL of the I^2C-bus has to be driven with a specified clock cycle. The clock frequency for the MGT power electronics is defined to 100 kHz. Further details about the communication using I^2C-bus are provided in [25].

3.2.2 Reaction wheels

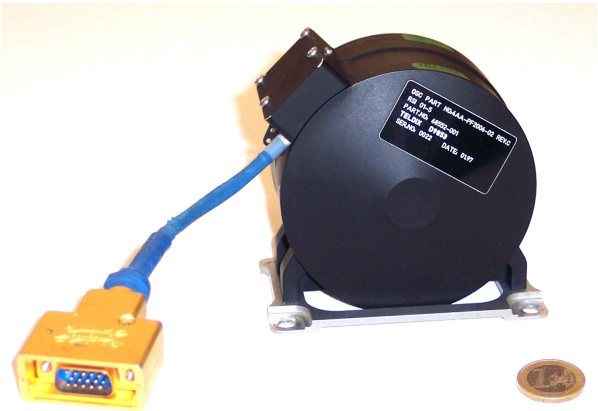


Figure 3.26: Reaction Wheel RSI 01-5/28

The reaction wheel system consists of four reaction wheels of type RSI 01-5/28 manufactured by Rockwell Collins. The reaction wheels are connected with OBC in a star like configuration via RS-422 bidirectional interface at 9600 buad to utilize the parallel execution capabilities of the FPGA. The selected RW is a ball bearing momentum/reaction wheel system where the digital wheel drive electronics is completely integrated into the wheel housing. It provides an angular momentum storage capacity of 0.12 Nms and a reaction torque of 5 mNm at its maximum speed of ±2800 rpm.

Table 3.11: Specifications of RW

Parameters	Specifications
Mass	0.70 kg
Dimensions	dia = 95 mm height = 102 mm
Angular Momentum	0.12 Nms
Operation Speed Range	±2800 rpm
Speed Limiter	< 3000 rpm
Torque (2800 rpm)	5 mNm
Power Consumption	< 2 W Steady state at nom. speed < 4 W max. torque at nom. speed
Supply Voltage	20 V - 0.5 V, 5 V ± 0.25 V
Electrical Interface	RS485, full duplex (9600,n,8,1)
Operating Temp.	-20°C to +60°C
Random Vibration	14.9 grms (max. qual. level)
Life Time	5 years (in-orbit) 2 years (storage)

3.2.2.1 Specifications

The specification of RSI 01-5/28 RW are tabulated in the Table 3.11 [26].

3.2.2.2 Placement in the satellite

Since the total reaction torque generated by the reaction wheels is independent of the point of action so there is no special requirement for the location of the reaction wheels. For reasons of redundancy, the reaction wheels are arranged in tetrahedron configuration. The rotation axis of one reaction wheel is aligned to the Y-axis of FLP's body coordinate system in order to optimize the configuration for the target pointing maneuver. The arrangement of RWs is shown in the Figure 3.27.

3.2.2.3 Electrical interface

As shown in the Figure 3.28 each RW is supplied with two voltages of +20 V and +5.1 V. 5.1 V voltage is used by the electronics in RW and +20 V are required by the electric motor. The connection between each RW, PCDU and the OBC is realized by a 15-pin

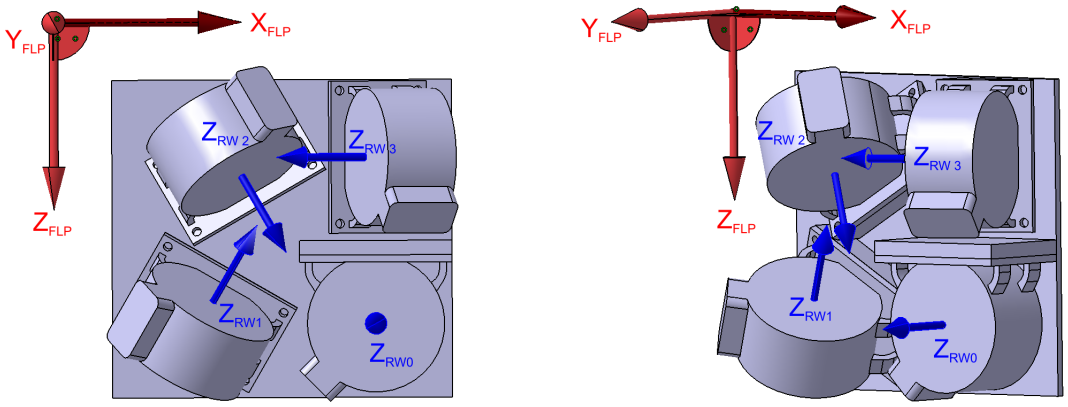


Figure 3.27: Arrangement of the reaction wheels

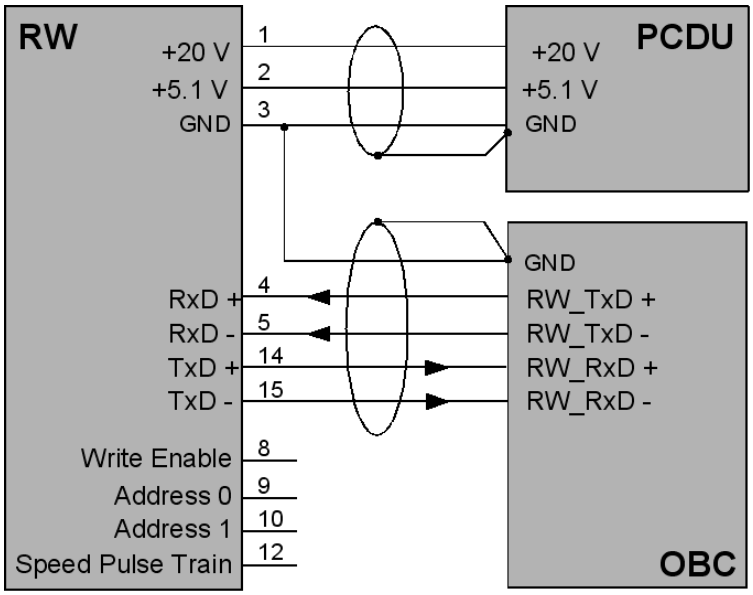


Figure 3.28: RW Interface Diagram

high density Sub-D connector. Further details and pin assignment is given in [27]. RW operates in 3 different modes which are: Standby mode, Speed-loop and Torque-loop. The ACS of FLP uses Torque-loop of the RW. Further details about each operational mode could be found in [26] and [27].

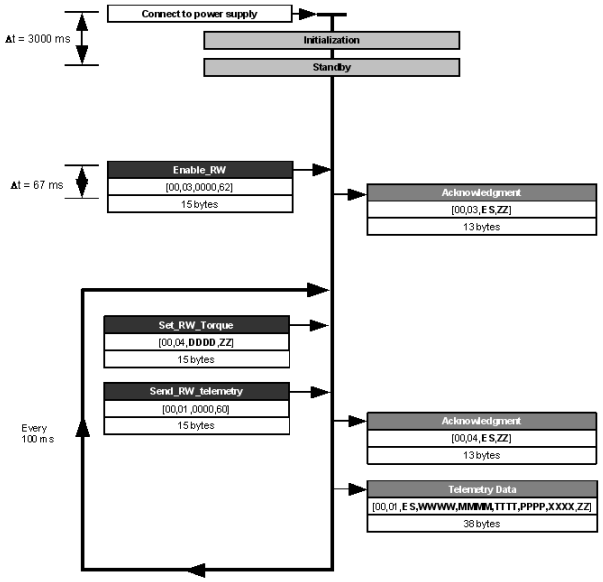


Figure 3.29: RW Communication Flow Chart

Figure 3.29 shows the communication flow of RWs with the OBC. The OBC commands are shown on the left side while RW response is shown on the right side of the flow chart. After the connection to the power supply the RW runs through the initialization-mode to the active standby mode. Communication with the RW is possible. The `enable_RW` command is sent until a correct confirmation is received which ensures that the RW is in the command-mode. Then the 10 Hz command cycle consisting of a `set_RW_torque` command immediately followed by a `send_RW_telemetry` command starts. The received responses have to be translated. Telemetry data corresponding to an invalid response messages has to be ignored and an error message has to be given. If a `wheel off` status is reported without a preceding `disable_RW` command the RW has to be enabled again by the `enable_RW` command.

CHAPTER 4

ACS On-Board Algorithms Architecture

The majority of satellite's processing requirements are accomplished with software executed on microprocessor-based computers. Software allows engineers to quickly configure and reconfigure a computer. Tools like operating system and programming languages have significantly improved the efficiency and productivity of software based design. Software based design is comparatively easy to develop, implement and change. However software based design has its limitations. Performance requirements such as timing constraints and data throughput are difficult to guarantee in software, especially when there are multiple software applications running on the same processor. Modern operating systems allow multiple programs to share the same processor but as the number of program increase, the amount of time given to each program is reduced. Much effort is needed to ensure software based design meets time critical requirements.

Another alternative to software based design is to use hardware based design which is adopted in the FLP. The primary motivation to develop hardware as opposed to software is performance guarantee. Unlike software, timing constraints and data throughput requirements can be guaranteed in hardware. Hardware allows engineers to carry out processing requirements independently. Specification that involve multiple tasks can be separated and implemented using independent hardware devices which provides true parallelism. The main limiting factor in hardware design is the physical amount of hardware resources available.

Another complication associated with the hardware based design is that it is always targeted for the specific hardware and this design could not be very easily ported to another similar hardware. This makes testing of these algorithms with engineering models a bit complicated as most of the time engineering models are necessarily not the same as the flight hardware. To overcome this problem, the overall OBC algorithm of the FLP is realized in layers. The lower layers are generally hardware specific and the upper layer is the application layer which is independent of the hardware. This makes it easy to test and qualify the on-board algorithms.

This chapter initially presents the architecture of FPGA-based OBC after which a brief introduction of satellite's overall control algorithms structure is presented which is followed by a detailed discussion about the ACS on-board algorithm architecture.

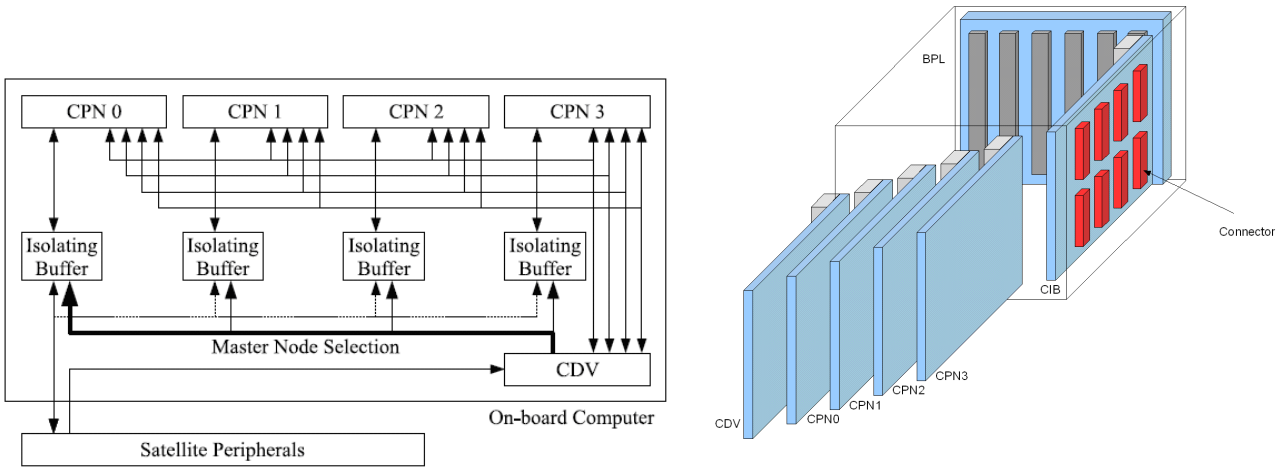
4.1 OBC hardware architecture

The FLP is the testbed for an On-Board Computer (OBC) with a reconfigurable, redundant and self-controlling high computational ability which is based on FPGA technologies. Field Programmable Gate Arrays (FPGAs) are programmable semiconductor devices that are based around a matrix of configurable logic blocks (CLBs) connected via programmable interconnects. As opposed to Application Specific Integrated Circuits (ASICs) where the device is custom built for the particular design, FPGAs can be programmed to the desired application or functionality requirements. Although one-time programmable (OTP) FPGAs are available, the dominant type are SRAM based which can be reprogrammed as the design evolves. There are mainly three different types of FPGAs.

SRAM-FPGA The first is SRAM-FPGA which could be reprogrammed. The first ever FPGA was also based on SRAM technology by Xilinx [28].

Anti-fuse FPGA The second type is one-time programmable (OTP) which are also called Anti-fuse FPGA which is made of anti-fuses and the desired logic circuit can be realized by burning them off.

Flash FPGA The third type is Flash-FPGAs which are also reconfigurable. Further details about these different types of FPGAs could be found in [28], [29], [30], and [31]



(a) Schematics of OBC Configuration

(b) OBC Structure

Figure 4.1: OBC Hardware Configuration

A combinational use of different types of FPGAs are applied for the design of the OBC. The main computing element of the OBC is the multi-module redundant SRAM-FPGA node monitored by a Voter based on SEU (Single Event Upset) effect tolerant FPGA. The selected SRAM-FPGA which will be implemented into the nodes is Virtex II-Pro, XC2VP50 of the Xilinx. This is partly because the chip was the state-of-the-art SRAM-FPGA at the time of design fix of the OBC and also partly because that the chip offers additional processor core, named PowerPC inside the chip, which can be used for normal floating point calculation with an operation system. The selected SEE (Single Event Effect) tolerant FPGA for the voter of the OBC is the state-of-the-art RTA3P Flash-FPGA of the Actel. Using these FPGAs, the redundancy degree of the node System is decided as four, for incorporating uncertainty of space radiation model and the case of failure of one of the four nodes. The merits of this approach are

- ◇ Extended lifetime of the OBC in the event of a permanent error in a node hardware.
- ◇ Short outage times (higher availability) in the event of SEU induced errors.
- ◇ Redundant computations allow for voting resulting in higher reliability.

In order to actually realize this concept, first of all, all of the redundant nodes are connected to the relevant satellite peripherals in parallel via isolating buffers so that each single CPN can execute full functions and control tasks of the OBC. Secondly, the nodes are physically identical. Thirdly, a failure in one node shall not be propagate through the

system. The schematics of the OBC of the Flying Laptop and the hardware configuration is illustrated in Figure 4.1.

The OBC consists of four central processing nodes (CPNs) and one command decoder and voter (CDV). They are connected with each other through a back plane (BPL) and with other peripheral electronics through a connector interface board (CIB). The CPNs are based on a reconfigurable FPGA with high computational ability and planned being reconfigured during on-orbit operation controlled by the CDV, which is also a FPGA-based computer. The CPN can also be loaded with a soft-core of a microprocessor. The data handling system of the Flying Laptop is integrated system on OBC and it holds the highest authority and performs most of the data handling on-board.

The CDV selects a master node from the four CPNs. Because this design is based on the assumption that each of the four CPNs performs the exact same tasks at the same time, all input/output signals, in total more than 200 lines, from peripheral electronics are connected to all CPNs in parallel. Due to the demands of peripheral electronics the interfaces of RS422, LVDM, M-LVDS, TTL and I^2C are implemented and connected with each node.

4.2 OBC control algorithm architecture

As discussed above to keep the application level of the algorithms independent of the platform, the overall control algorithm is realized in a layered architecture. Each layer represents a different abstraction level which hides the implementation details of a particular set of functionality. The most important feature of this layered structure is that the application level control algorithms remain the same in the flight and in the verification and qualification with the simulators such that the verified control algorithms could be directly ported into the real flight configuration. This layered structure is illustrated in the Figure 4.2. The layers from lower level to the higher level could be written as;

- ◇ PSL - Platform Support Library
- ◇ PAL-API - Platform Abstraction Layer-Application Programming Interface
- ◇ PAL-Core - Platform Abstraction Layer Core
- ◇ Application Layer

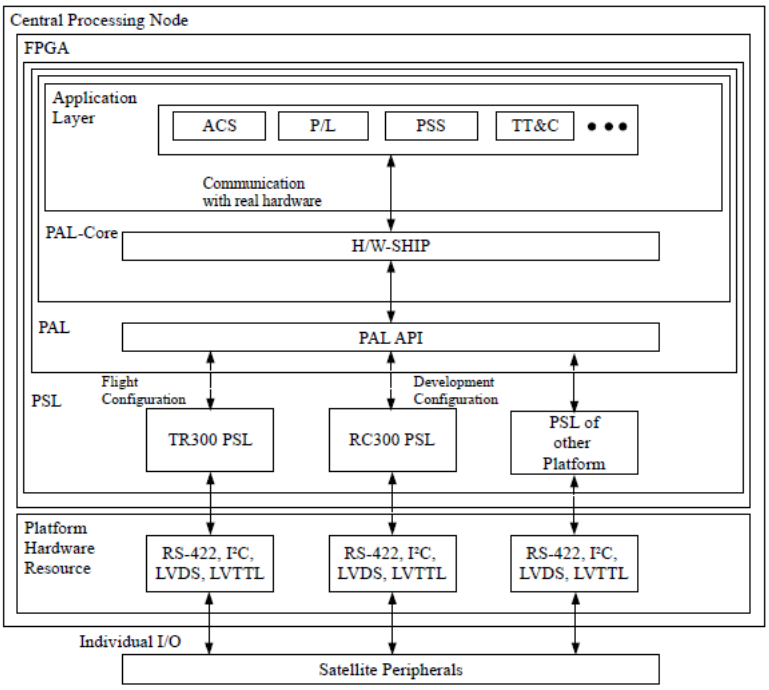


Figure 4.2: Layered structure of the control algorithm

PSL

The platform support library contains the settings for a specific FPGA board. This library provides the low-level interfaces to specific platform resources, such as I/O or memory. The number of a specific type interfaces like serial interface is defined and also I/O pins are assigned to the corresponding hardware sensors and actuators. The PSL is then accessed from the hardware applications on the FPGA using a simple and consistent Application Programming Interface (API).

PAL-API

PAL shields development engineers from low-level hardware interfaces in order to ease the integration of FPGAs with physical resources. PAL-API provides generic functions or macro calls that provide the hardware application with access to and communication with PSL.

PAL-Core

This is implementation specific function layer, in which routine tasks such as periodical sensor data collection for every satellite peripheral is implemented. This is also called as SHIP (Satellite Hardware Interface Protocol). In this layer, every component is represented by relevant variables and functions, which enables an application programmer to access the peripherals quite easily without knowing their exact technical specifications.

Application Layer

This is the highest level of implementation in which all the user specific algorithms are implemented such as algorithms of the attitude control, payload, power sub system and algorithms regarding telecommunication etc. This layer is completely platform independent and could be easily ported for verification or to the flight environment.

4.3 Control algorithm architecture

The application layer and the PAL-Core layer of the developed control algorithm is shown in the Figure 4.3. As illustrated in this figure each subsystem is implemented independently and there is no governing central process unlike conventional CPU-type OBCs. The important characteristic of this design is that the system-level algorithms could be decentralized, letting each subsystem be responsible to a part of system tasks, which can be implemented as parallel process inside an FPGA.

Each subsystem communicates with other subsystems through `Inter-agent I/O`. The central logic or the algorithms are implemented in the `Central Control Logic` which receives inputs from the hardware through PAL-Core and instructions from System FDIR (Failure Detection, Isolation and Recovery). Further details regarding `Central Control Logic` are given in the subsequent sections. System FDIR is designed to locate and isolate the potential risks of hardware and software anomalies using watch-dog functions by switching the satellite into appropriate control mode. This is the decision making function of the satellite which is also connected with the hardware for monitoring and in certain conditions it could switch on and switch off any hardware component. The request for a particular system mode is also furnished in the FDIR function and it selects the appropriate ACS mode related with this system mode and commands the ACS subsystem accordingly.

Each subsystem has its own `Fault Detection` (FD) function. The output of FD function is interconnected with the `Central Control Logic`. The functions of `TC` and `TM` receives and send the required information. The PAL-Core layer for each subsystem could be further decomposed into components. The example of ACS subsystem is shown in the Figure 4.3 in which ACS PAL-Core is further divided into several domains each representing a specific ACS hardware component. Even these domains are sub-divided into several functions as elaborated in the case of sun sensor (SuS).

4.4 ACS algorithm architecture

As explained above, ACS runs as an independent unit and in parallel with the other subsystems. It is commanded by TC or in the case of hardware or software anomaly the commands from the TC are overwritten by the FDIR. Inter-agent (I/O) is used for the communication with the other subsystem. Figure 4.4 shows the expatiated view of the application layer of ACS subsystem in which the central control logic is further expanded. The main blocks of this figure are explained in the subsequent sections.

4.4.1 Algorithms

This is the core of the ACS subsystem. All the attitude determination algorithms and control commands are calculated in this block. The inputs from the sensors are received through ACS PAL-Core which are processed to determine the attitude of a satellite and the desired control actions are performed to calculate the commands for the actuators which are then transmitted to ACS PAL-Core through the output. This block is discussed in detail and complete details regarding the algorithms are provided in the next chapter. (Chapter 5)

4.4.2 Constants

This is used to store all the constants required during attitude control computations. These constants include the satellite inertia matrix, linear gains for the controllers and the alignment matrices to convert the sensor information into the body fixed coordinate system. These constants are stored separately and are accessible all the time to the ACS algorithms. The controller gains and the alignments matrices are the estimated values which are not fixed. These estimates are improved after the reception of the flight data.

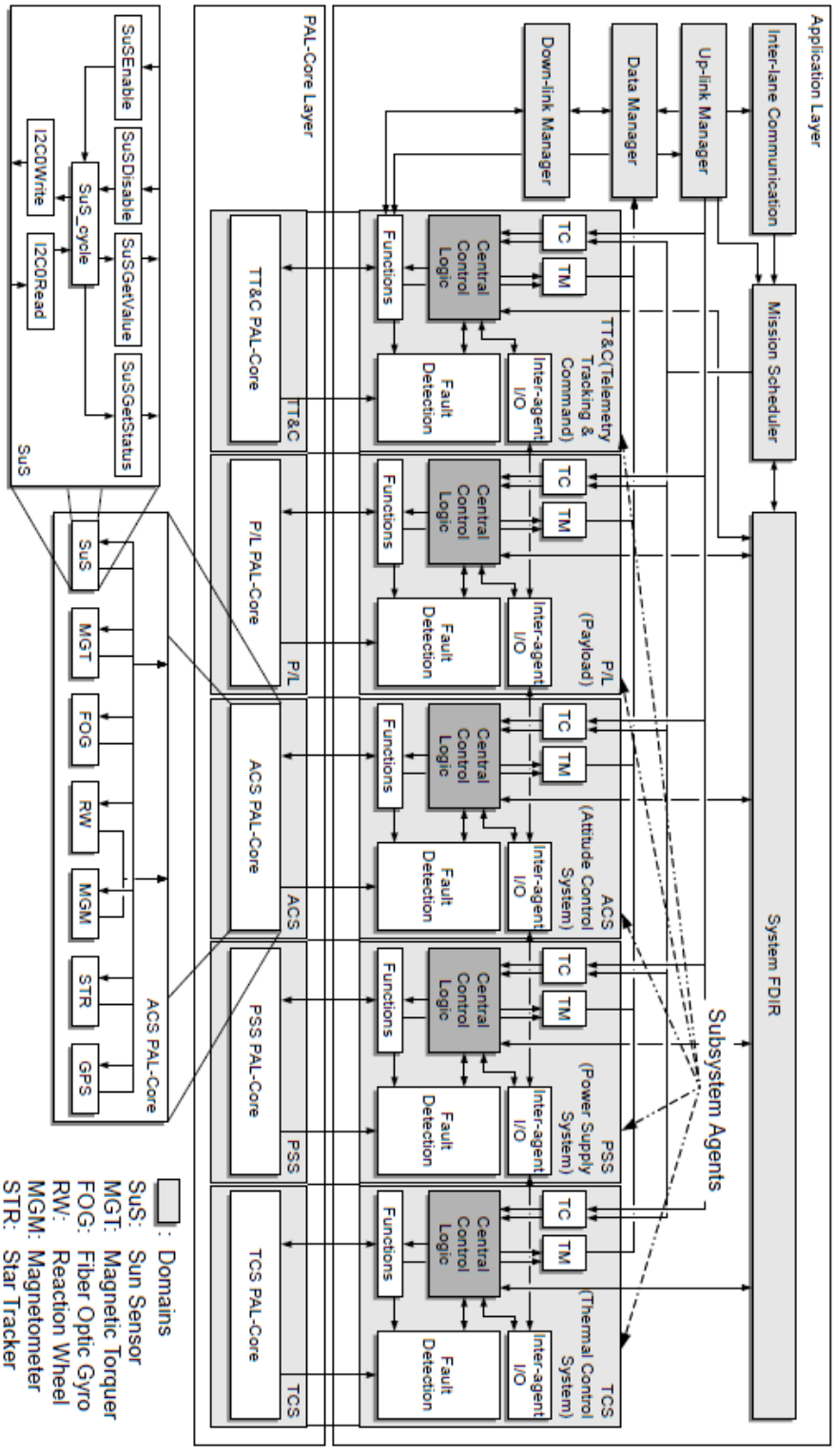


Figure 4.3: Control Algorithm Architecture

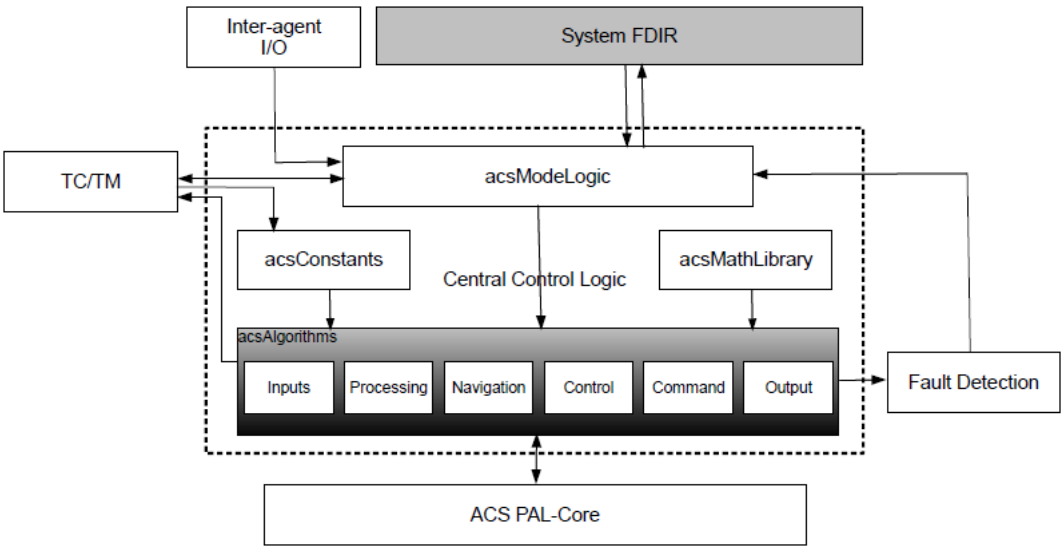


Figure 4.4: ACS Control Algorithm Architecture

Therefore this information is stored in a separate location to facilitate any change in these constants through telecommand (TC) during the flight without making any change in the main algorithm hardware.

4.4.3 Math library

The attitude control algorithms involve extensive use of matrix and quaternion algebra. This library facilitates the algorithm with the math functions. Some functions are called multiple times and from different locations during the computations. As these algorithms has to be implemented as a hardware on FPGA so limited hardware resources is always a major constraint. Therefore the mathematical functions which are called multiple times and from multiple locations are realized as a shared hardware which could be accessed from multiple locations during an ACS cycle. Although this reduces the parallelism but still it is realized in such a way that it produces no delay in the ACS cycle time. The details about the implementation are provided in the Chapter 6.

4.4.4 Fault detection

The fault in the ACS subsystem could be induced either by the ACS hardware or by the execution of faulty algorithms/conditions. These faults are then reported to the `acsModeLogic` which directly communicates with the system FDIR which collects the

relevant information from the other subsystems as well. The final decisions are made in the system FDIR which are then communicated to the `acsModeLogic`, that is responsible for implementing these decisions. The ACS itself could not induce a mode change itself and always follows a command either communicated through TC or dictated by System FDIR. ACS also could not turn-on/turn-off the hardware components as these decisions are dependent upon the input from various subsystems and only ACS could not take such decisions on the behalf of other subsystems so these are decided at a higher level of system FDIR.

The block of Fault Detection receives inputs from the hardware directly in the form of hardware flags, further it receives body rates, sun position in body coordinate system and the current mode of the ACS from the algorithms block. The relevant hardware flags based upon the current mode are checked and if there is some flag missing or faulty then it is reported as a fault. The fault is not reported immediately but this reporting mechanism is realized with a series of counters which when reach some threshold limit report this fault to the higher level.

Apart from the hardware there are several logical failures which could occur. These are

Rate limit violation Every mode has its own rate limits in which it performs its specified functions and if the body rate of the satellite violates this limits then this error is reported after passing through a counter to the higher level

Sun avoidance The satellite payload is not allowed to look directly into the sun, therefore the position of the sun is directly monitored in the body axis. If the angle of the Sun with the positive z-axis of the body coordinate system decreases below 60°, a warning of *Sun Proximity* is issued to the TM and finally an error is reported when this angle decreases below 45°.

Off target Target flag is calculated in the ACS algorithms (explained in the next chapter) and if this flag during Target pointing is low for certain amount of time period then this error is reported to the higher level.

4.4.5 acsModeLogic

Within the ACS subsystem this is the highest level of the algorithm which commands the current ACS mode of the satellite and it also receives direct instructions from the TC and it also sends the TM messages. Additionally this function receives the output of Fault

Detection and deliver this information to the system FDIR. It also received instructions from the system FDIR which in certain conditions overwrite the TC.

The information received through telecommand (TC) is of two types; High-priority TC and Low-priority TC. Low-priority TC could be overwritten by the FDIR under certain conditions where satellite is facing some hardware or software anomaly. The High-priority TC commands could not be overwritten and are executed directly without any alteration. TC information contains the desired ACS mode, reference quaternion q_{ref} and the target position in ECF frame r_{tgtF}. It also updates $matNP$ periodically which is a product of Earth nutation and precession matrix and explained in Appendix A.

The only mode change which is induced by the ACS subsystem itself is a change from detumble mode to the safe mode which is based on the body rates. Satellite is shifted into the safe mode automatically if the body rates of the satellite are lower than 0.6 °/s and the satellite is in detumble mode. That is why safe mode of the satellite could not be requested from outside directly. So in any contingency the satellite is asked to go to the detumble mode and then it is automatically transferred to the safe mode within the ACS subsystem if the body rates of the satellite are lower than the specific limit. The decision of shifting the satellite from detumble to the safe mode is also taken in acsModeLogic.

This function detects the mode change, in case if the current and desired mode are not the same, the up date flag `ptUpdtFlg` is set to high and the new requested mode is stored in the variable `modeGet`. As long as the `ptUpdtFlg` is high the logic tries to update the ACS mode and as soon as the ACS mode is updated the `ptUpdtFlg` is set to low once again. The satellite remains in the current mode until the `ptUpdtFlg` is set to high once again.

This function also sets the `desatFlg` to high for the modes which uses reaction wheels (if the value of the mode is 2,3,4 or 5). This flag activates the wheel desaturation. This flag is set to 0 for the modes which do not use reaction wheels (i.e. mode = 0,1).

CHAPTER 5

ACS Algorithms

The accuracy of orientation control of a satellite eminently counts on the algorithms selected for the attitude determination and control. The objective of this chapter is to provide the details of the attitude determination and control algorithms. The accuracy of attitude determination of a satellite deteriorates due to the noise and inaccuracies of the measuring sensors. There are many available techniques and algorithms which could deal with this problem. Most of such algorithms are based on filtering techniques. The only drawback of these techniques is their computationally exhaustive algorithms. As FLP is based on FPGA-based OBC therefore every algorithms runs as a dedicated hardware instead of a software. Although this increases its speed remarkably and it also provides the true parallelism but at the same time this also impose lot of restrictions in the freedom of choice while selecting the ACS algorithms.

Most of the filtering techniques due to their computationally exhaustive nature could not be employed in this design. Use of trigonometric functions, matrix inversions and use of floating point require to build few libraries in the hardware which occupy significant amount of hardware gates in FPGA so this was also avoided while developing the ACS algorithms. This impose a huge amount of restrictions in developing the algorithms or selecting different available techniques.

This chapter summarizes all algorithms used in the ACS design of the FLP. There are many points on which alternative solutions were also used and simulated but to keep this chapter concise only few of them will be discussed and compared to the baseline. The

ACS algorithms are implemented in a structured way and to present a systemized view, this chapter is also branched into similar sections as they are implemented on-board.

5.1 Structure of the ACS algorithms

The structure and flow of the ACS algorithms is shown in the Figure 5.1. ACS algorithms could be classified into six categories based on the vertical sections of the Figure 5.1. These six categories are:

1. Inputs
2. Processing
3. Navigation
4. Control
5. Command
6. Outputs

5.2 Inputs

There are two types of inputs which are driving the ACS algorithms. Most of them are result of a direct communication with the ACS hardware (i.e. ACS sensors, OBC time) and rest of the inputs are logical inputs which are mainly dictated by the FDIR. FPGA communicates directly with the hardware using Satellite Hardware Interface Protocol (SHIP) which is low level of the algorithm implementation and is discussed in more detail in Section 4.2. The user commands coming from the TC (Telecommand) are implemented through FDIR which decides the ACS mode depending upon the subsystem/user request or this decision is based upon the conditions which are vital for the satellite survival. Inputs also collect the speed information from the reaction wheels required for the Null-space and Desaturation control modes.

5.3 Processing

Processing is mainly done to convert the sensor data into useful data. The main functions of processing blocks are:

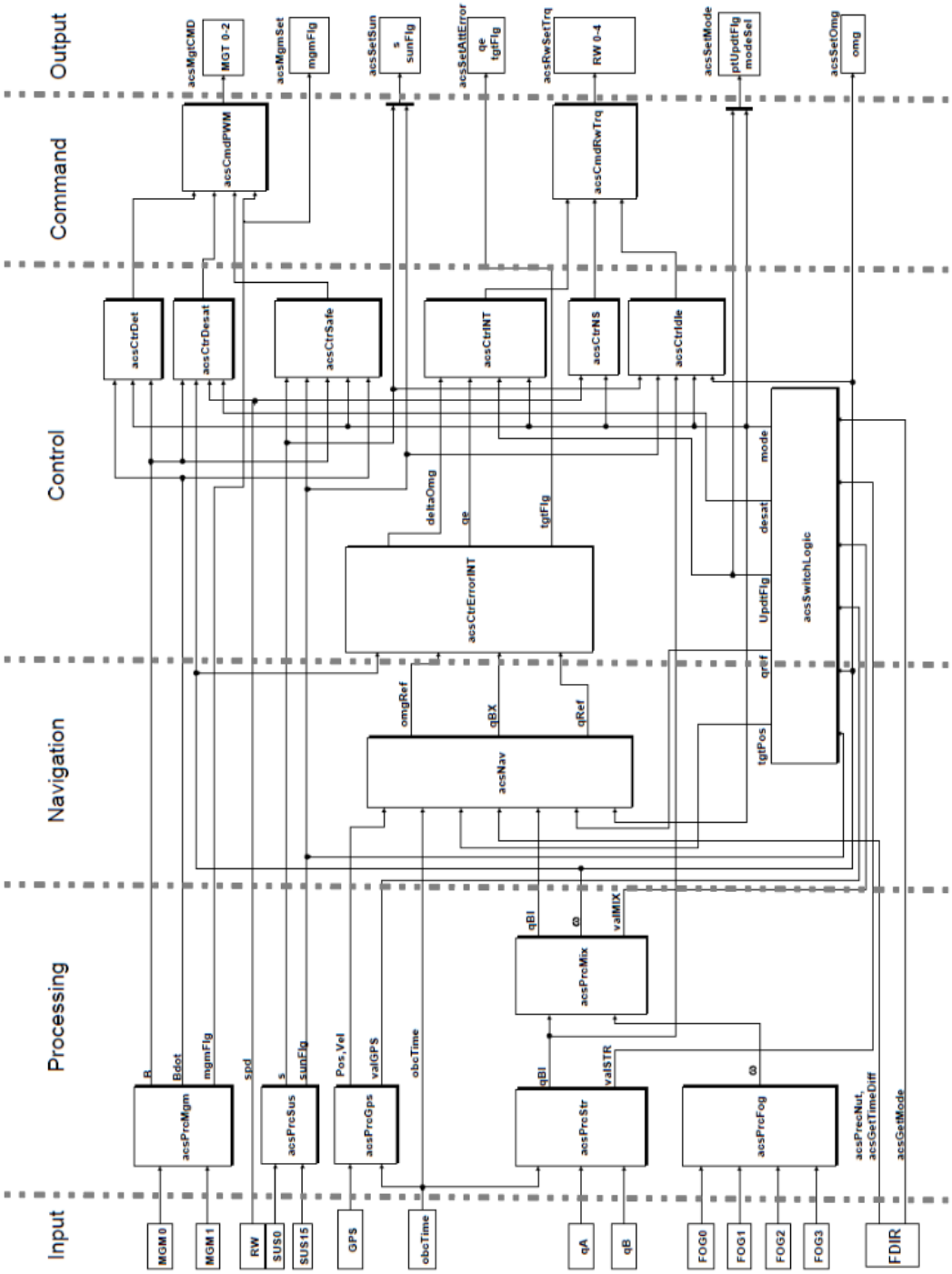


Figure 5.1: Structure of the ACS Algorithms

- ◇ Selection of useful sensors
- ◇ Conversion of the sensor information from the sensor axis to the FLP body axis system
- ◇ Propagation of the sensor data to the ACS time.
- ◇ Improvement of pointing accuracy by mixing FOG and STR data.
- ◇ Extraction of rate information from quaternions
- ◇ Extraction of attitude by integration of the body rates

This category is further subdivided into six functions as it is shown in the Figure 5.1. For each sensor there is a corresponding function shown as a block in this figure. The nomenclature of these functions is realized in such a way that the first three characters always represent the sub-system which in this case is `acs` and the three characters followed by the category they belong to, like in the case of processing it is `Prc`, and the function name comes after that. The functions under this category could be listed as follows:

1. `acsPrcMgm`
2. `acsPrcSuS`
3. `acsPrcGps`
4. `acsPrcFog`
5. `acsPrcStr`
6. `acsPrcMix`

The name of the each function clearly indicates the sensor whose outputs are being processed in this block. `acsPrcMix` is a function used to improve the pointing accuracy by mixing the STR data with the FOG sensor. The details about each of these functions are provided in the subsequent sections.

5.3.1 `acsPrcMgm`

The inputs to this function are the magnetometer outputs comprising of the Earth's magnetic field measurement and the valid flag coming from the two three axis magnetometers. The alignment matrices from the magnetometer sensor axis to the FLP body axis are also provided as an input to the function from the `acsConstants`. The output of this function is estimate of the Earth magnetic field and its derivative in the FLP body axis. The

magnetometer readings are not reliable when MGTs are functioning. So in each 5 s of operation, the first 4.5 s are allocated to MGTs actuation and the remaining 0.5 s are utilized for magnetometer measurements. Therefore this function also returns `mgmFlg` which is set to 0 while MGTs are functioning and it is set to 1 when magnetometers are used to measure the Earth's magnetic field.

Depending upon the valid flag from the MGM, this function could have three conditions which are:

Both the sensors are valid In this condition, both magnetometer outputs are combined to decrease the sensor noise. The matrices of transformation or the alignment matrices are also combined to form a pseudo-inverse matrix.

$$A_{MGM}^{PI} = \begin{bmatrix} A_{MGM0}^{T} \\ A_{MGM1}^{T} \end{bmatrix} \tag{5.1}$$

A_{MGM0} and A_{MGM1} are the matrices of transformation and A_{MGM}^{PI} is the pseudo-inverse matrix. Now using this matrix, the outputs of the MGMs could be converted in to the FLP body axis.

$$b_B = A_{MGM}^{PI} \begin{bmatrix} b_{MGM0} \\ b_{MGM1} \end{bmatrix} \tag{5.2}$$

The derivative of the magnetic field could be computed by using the present and the previous value of the earth's magnetic field sensed by the MGMs.

$$\dot{b}_B = \frac{b_B(t_i) - b_B(t_{i-1})}{t_{MGM}} \tag{5.3}$$

An internal counter is used for the calculation of `mgmFlg`. As ACS cycle time is 0.1 s and this function `acsPrcMgm` is called every time. `mgmFlg` is set to 1 when the counter is in between 45 and 50 and it is set to 0 otherwise. The counter is reset to 0 when it reaches 50.

One of the sensors is valid if one of the sensor is valid then the value of magnetic field is only calculated using that sensor.

$$b_B = A_{MGM0} b_{MGM0} \tag{5.4}$$

$$b_B = A_{MGM1} b_{MGM1} \tag{5.5}$$

The value of the derivative and the `mgmFlg` is calculated in a same manner.

None of the sensors is valid This is the third condition and if there is no valid flag from both of the MGMs then the output is set to the previous estimated value and the `mgmFlg` is set to 0.

5.3.2 acsPrcSuS

This function is used to process the sun sensor information. The input to this function are the measured currents of the sun sensors and the matrix of transformation of SuS currents from sensor axis to the FLP body axis. The output of this function is the sun vector in the body axis (s_B) and the `sunFlg` which is 0 during eclipse and is unity during the illuminated portion of the orbit.

Analog sun sensors are used in the FLP. The functioning of analog solar cell is explained in [32]. The current measured (I_{meas}) by the solar cell is directly related with the sunlight incident angle and could be approximated with the sine function as:

$$I_{meas} = I_{max}.\sin\theta \tag{5.6}$$

Using the same principle the normalized sun vector (s_B) in the body coordinate system could be obtained.

$$s_B = A_{sus}\left(\frac{I_{meas}}{I_{max}}\right) \tag{5.7}$$

Where A_{sus} is the transformation matrix used to transform the sun vector from sensor axis to the body axis and I_{meas} is a (3x1) vector measured by the sun sensors. The actual measurement vector is a (8x1) vector but among these only three brightest sun sensors (orthogonal) are selected based on their current value. An alternative method of using 6 solar cells instead of 3 brightest was also in consideration but later after simulations with the albedo effect included in the simulation environment, method of using only 3 brightest sun sensor was selected.

The sun flag `sunFlg` is also generated in this function to determine whether satellite is in illuminated part of the orbit (`sunFlg` = 1) or it is in the eclipse (`sunFlg` = 0).

$$sunFlg = \begin{cases} 1, & \text{if } \|s_B\| > 0.5 \\ 0, & \text{otherwise} \end{cases} \tag{5.8}$$

s_B is normalized before the output to ensure the unit sun vector and to remove noise. If satellite is in eclipse or current data from the sun sensor is not valid then `sunFlg` is set to 0 and the output sun vector s_B is set to $[00-1]^T$ to avoid division by zero in further computations.

5.3.3 acsPrcGps

The inputs to this function are the data from all the three GPS receivers and the OBC time. The data from GPS includes position, velocity and the time information. This information from GPS is updated every second and the time at which this measurement is taken t_{GPS} is included in the received data but this information is not synchronized with the OBC time t_{OBC} so this function propagates this information at 10 Hz and also it shifts this information to the ACS time t_{ACS}. t_{ACS} has a constant offset t_{off} relative to the t_{OBC} to compensate the delay of reaction wheels.

The first part of this function relates to the selection of one among the three GPS receivers. There were two methods of selection of GPS receiver under consideration. The first is based on the GPS PDOP (Positional Dilution of Precision) value which is an indicator of GPS accuracy. Its value is higher for a weak geometry providing lower accuracy and for strong geometry and better accuracy PDOP value is smaller. This value is also supplied by the GPS receiver in the navigation message. So first it was decided to select the GPS receiver based on lower PDOP value but as all the three GPS antennas are oriented in the same direction, therefore it is more likely that they will have the same satellites in their view and their PDOP value would be almost the same.

Later to avoid frequent switching among the receivers which could also produce additional noise it was decided to use one GPS receiver as master and the information obtained from the master will be selected for the further use. Initially GPS0 is treated as master but if GPS0 outputs invalid data then GPS1 is treated as master. If GPS0 and GPS1 both are giving invalid information then GPS2 is treated as master and in case none of the data is valid then position and velocity are propagated based on the last measured values.

The method used for the propagation of GPS data is very simple. To shift the GPS information to the t_{ACS} first total delay Δt between the GPS measurement and the ACS time is calculated, which is

$$\Delta t = t_{OBC} - t_{GPS} + t_{off} \tag{5.9}$$

The acceleration at the time of measurement is approximated with the latest available information at t_i and the previous information at t_{i-1}. Index i denotes the current measurement.

$$\dot{V}_i = \frac{V_i - V_{i-1}}{t_i - t_{i-1}} \tag{5.10}$$

Using estimated acceleration and measured velocity the information of the GPS is shifted to the ACS time t_{ACS}.

$$V_j = V_i - \dot{V}_i.\Delta t \tag{5.11}$$

$$R_j = R_i + V_j.\Delta t \tag{5.12}$$

where index j denotes the propagated value at the ACS time. The above equations are used to shift and synchronize the GPS information with t_{ACS}. Since GPS information is only available once in a second so this shifting is done once in a second. In the subsequent ACS cycle where no GPS information is available following equations are used to update this information. Δt_{cyc} is the time required to complete one ACS cycle which is 100 ms.

$$V_{j+1} = V_j - \dot{V}_i.\Delta t_{cyc} \tag{5.13}$$

$$R_{j+1} = R_j + V_j.\Delta t_{cyc} \tag{5.14}$$

5.3.4 `acsPrcFog`

The inputs to this function are the rates sensed by four FOGs which are mounted in a tetrahedral configuration. The inputs also include the valid flags from the rate sensors. This function estimates the satellite's angular speed in the body coordinate system.

The rate information could be estimated if at least three out of four sensors are having valid data. Based on this there could be five different combinations of these sensors. For each combination there is a pseudoinverse alignment matrix A^{PI}_{FOG} pre-calculated and stored in `acsConstant`. The pseudoinverse matrix also helps in reducing some noise. The information from all of the four FOGs is collected in a (4x1) matrix ω_{FOG} and to obtain the body rates ω_B which is (3x1) vector, it is multiplied with the (3x4) A^{PI}_{FOG}.

$$\omega_B = A^{PI}_{FOG}\omega_{FOG} \tag{5.15}$$

In case all of the four sensors are working, A^{PI}_{FOG} combines the output of all the sensors to obtain better estimate of the ω_B but if the data of any one of the FOGs is invalid then the appropriate alignment matrix is selected and used. If more than one FOGs are having invalid data then the previous value of the body rate is sent at the output and the `validFog` flag is set to zero.

5.3.5 acsPrcStr

The function receives the attitude measurement and the valid flags from both the Camera Head Units (CHU). The information is in the form of a attitude quaternion q_{SI} which provides attitude information from the sensor axis to the inertial frame (J2000). The output of this function is the attitude quaternion q_{STR} providing the attitude information from the body coordinate system to the inertial frame.

There could be three possibilities to output the required information based on the valid input flags of the CHUs. If both of the CHUs are delivering valid data then both the inputs quaternions from the CHUs are mixed to improve the accuracy of the output. The other two possibilities are if one of the CHU is not delivering valid information.

if only one CHU is providing a valid data then the CHU alignments matrices (A_{CHUA} and A_{CHUB}) which are precalculated are converted into transformation quaternion (q_{BS}) and is multiplied with the measured quaternion q_{SI} to obtain the attitude quaternion q_{BI}.

$$q_{STR} = q_{BI} = q_{BS}q_{SI} \tag{5.16}$$

Since the measurement of the star tracker across the boresight is more accurate than the measurement around the boresight (roll) therefore the information from the star tracker could be improved in terms of accuracy by mixing both the input quaternions, if both the CHUs are providing valid data. To mix the input quaternions q_A and q_B, first a transformation matrix T_{BJ} for converting the information from both the CHUs to body system is calculated using the alignment matrices A_{CHUA} and A_{CHUB} .

Each CHU's boresight direction could be obtained as

$$v1 = [A_{CHUA,13}\ A_{CHUA,23}\ A_{CHUA,33}] \tag{5.17}$$

$$v2 = [A_{CHUB,13}\ A_{CHUB,23}\ A_{CHUB,33}] \tag{5.18}$$

Now using these vectors the joint transformation matrix could be calculated as

$$v_x = \frac{v1 + v2}{\|v1 + v2\|} \tag{5.19}$$

$$v_y = \frac{v1 - v2}{\|v1 - v2\|} \tag{5.20}$$

$$v_z = v_x \times v_y \tag{5.21}$$

The transformation matrix T_{BJ} could be now calculated

$$T_{BJ} = [v_x\ v_y\ v_z] \tag{5.22}$$

This matrix could be pre-calculated and is stored in `acsConstants` similar to the other alignment matrices. Now we have a transformation matrix from both of the CHU's boresight direction to the body coordinate system and we need to calculate the CHU boresight vectors from sensor axis to the inertial coordinate system and by multiplying these two we could calculate the quaternion from the body coordinate system to the inertial frame. So first we need to calculate the DCM vectors which express CHU boresight vectors from sensor to the inertial frame. These DCM vectors could be calculated from the input quaternions as

$$p_{A,1} = 2(q_{A,1}q_{A,3} + q_{A,2}q_{A,4}) \tag{5.23}$$

$$p_{A,2} = 2(q_{A,2}q_{A,3} - q_{A,1}q_{A,4}) \tag{5.24}$$

$$p_{A,3} = 2(q_{A,3}^2 + q_{A,4}^2) - 1 \tag{5.25}$$

Similarly DCM vector p_B could be calculated from q_B and these two vectors result in a joint system given by the following equations.

$$p_x = \frac{p_A + p_B}{\|p_A + p_B\|} \tag{5.26}$$

$$p_y = \frac{p_A - p_B}{\|p_A - p_B\|} \tag{5.27}$$

$$p_z = p_x \times p_y \tag{5.28}$$

The transformation from inertial to joint sensor system T_{JI} could be obtained by

$$T_{JI} = [p_x\ p_y\ p_z] \tag{5.29}$$

Finally the desired attitude transformation of body to inertial frame could be done

$$T_{BI} = T_{BJ}T_{JI} \tag{5.30}$$

The transformation matrix T_{BI} could be converted into q_{BI} quaternion which is the functions output q_{STR} q_{BI}. If data from both of the CHUs is not valid then the output is set to $q_{BI} = [0\ 0\ 0\ 1]^T$ and the `validStr` flag is set to zero.

5.3.6 `acsPrcMix`

The inputs for this functions are the outputs of `acsPrcStr` and `acsPrcFog`. This function mixes the information from the star tracker and rate gyros. STR outputs the attitude quaternion at 2 Hz, this information is not synchronized with the t_{OBC} but this contains

a time stamp t_{STR} at which this measurement is made. This function also propagate the attitude quaternion using body rates measured by the FOGs. In case if any one of `acsPrcStr` or `acsPrcFog` does not output the valid data then this function additionally extract the rate information from the STR measured data or it estimates the attitude quaternion from the FOG measured data and outputs this for the navigation.

There could be four different conditions in this function based on the validity of the input data. These conditions are:

- ◇ Propagation if attitude quaternion and rate are valid.
- ◇ Propagation if only rate is valid.
- ◇ Propagation if only attitude quaternion is valid.
- ◇ Propagation if both the inputs are invalid.

5.3.6.1 Propagation if attitude quaternion and rate are valid

The information received from STR is updated every 500 ms and it contains a time stamp t_{STR} with each measurement. The difference between OBC time t_{OBC} and t_{STR} is not fixed and it could be around 500 ms. So the delay between t_{OBC} and t_{STR} could be very simply calculated as:

$$\Delta t = t_{OBC} - t_{STR} \tag{5.31}$$

To synchronize the quaternion with the OBC time, the amount of ACS steps (k) between the last measurement time and the current OBC time are required as ACS cycle is updated every 100 ms. This could be determined by using the delay calculated in the equation 5.31.

$$k = floor(10\Delta t) + 1 \tag{5.32}$$

Different notation and different times used in this function are illustrated by the use of an example in Figure 5.2. In this example the value of k is equal to 4 and it is shown that the value of body rates is measured in every OBC time cycle. t_{off} is an offset between real OBC time t_{OBC} and the artificial ACS time t_{ACS} to compensate the delay of the reaction wheels. t_{ini} is the time difference between STR measurement and the following OBC cycle in the past and this could be calculated as

$$t_{ini} = \Delta t - \frac{k-1}{10} \tag{5.33}$$

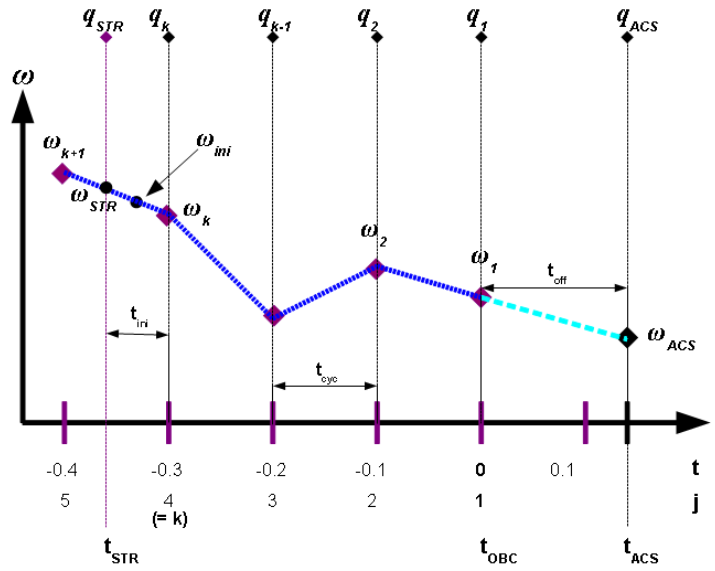


Figure 5.2: Timing example for propagation of attitude to the t_{ACS}

As the new rate information is only available at OBC time so it has to be shifted to ACS time. This is done by linear extrapolation using the latest (ω_1) and the previous rate (ω_2) as shown in the Figure 5.2.

$$\omega_{ACS} = \omega_1 + (\omega_1 - \omega_2).\frac{t_{off}}{t_{cyc}} \tag{5.34}$$

As angular rates are needed to propagate an accurate quaternion from a delayed STR measurement, the last 10 rates at OBC time are saved, independent on how they are calculated in the subsequent sections. Additionally, always the last quaternion and measurement time from the STR are saved over several cycles until new STR information arrives. Finally, the last propagated rate at ACS time is saved to be available in the next cycle. Since t_{ini} is already calculated so the rate at the time of STR measurement could be interpolated with the help of ω_k and ω_{k+1} shown in the Figure 5.2.

$$\omega_{STR} = \omega_k - (\omega_k - \omega_{k+1}).\frac{t_{ini}}{t_{cyc}} \tag{5.35}$$

ω_{ini} is the average rate of ω_{STR} and ω_k and it is used to shift the quaternion to time t_k.

$$\omega_{ini} = \frac{\omega_{STR} + \omega_k}{2} \tag{5.36}$$

Now we can calculate the quaternion at time t_k by using following equation

$$q_k = q_{STR} + \frac{1}{2} t_{ini} \Omega(\omega_{ini}) q_{STR} \tag{5.37}$$

With the same procedure the quaternion is propagated to the OBC time in steps of t_{cyc} again using the average rates between two ACS cycles

$$q_j = q_{j+1} + \frac{1}{2} t_{cyc} \Omega(\frac{\omega_j + \omega_{j+1}}{2}) q_{j+1} \qquad for \quad j = k, k-1, ...1 \tag{5.38}$$

Finally the desired quaternion at ACS time is:

$$q_{BI} = q_{ACS} = q_1 + \frac{1}{2} t_{off} \Omega(\frac{\omega_1 + \omega_{ACS}}{2}) q_1 \tag{5.39}$$

If there is no STR update then the quaternion is propagated in a very same manner using latest and the previous angular rates along with the last measured quaternion as described in the equation 5.40.

5.3.6.2 Propagation if only rate is valid

Again in this part the body rates which are available at OBC time need to be shifted to the ACS time as presented in the equation 5.34. As there is no STR measurement available so quaternion is propagated using latest and the previous angular rates along with the last measured quaternion.

$$q_{BI} = q_{ACS,i} = q_{ACS,i-1} + \frac{1}{2} t_{cyc} \Omega(\frac{\omega_{ACS,i} + \omega_{ACS,i-1}}{2}) q_{ACS,i-1} \tag{5.40}$$

5.3.6.3 Propagation if only attitude quaternion is valid

If there is no direct rate information available then rates are estimated using the attitude quaternion. Following equation could be used to estimate body rates from quaternion.

$$\omega = -2(q_{1:3} \times \dot{q_{1:3}} + \dot{q_4} q_{1:3} - q_4 \dot{q_{1:3}}) \tag{5.41}$$

where is the derivative of the quaternion and could be approximated by using the latest and the previous STR measurement.

$$\dot{q} = \frac{q_{STR,i} - q_{STR,i-1}}{t_{STR,i} - t_{STR,i-1}} \tag{5.42}$$

Now for quaternion propagation we could have two cases for the propagation, the first case is if there is a new STR measurement available. In this case the quaternion is shifted to the ACS time as

$$q_{BI} = q_{ACS} = q_{STR,i} + (t_{OBC} + t_{off} - t_{STR,i}) \dot{q} \tag{5.43}$$

For the next case in which no new STR measurement is available, new quaternion could be derived using the quaternion derivative as

$$q_{BI} = q_{ACS} = q_{BI,i} + t_{cyc} \dot{q} \tag{5.44}$$

5.3.6.4 Propagation if both the inputs are invalid

If no information is valid the last body rate measured by gyro or estimated using attitude quaternion is held constant over the subsequent cycles. The quaternion is propagated using this constant rate as

$$q_{BI,i} = q_{BI,i-1} + \frac{1}{2} t_{cyc} \Omega(\omega_{ACS,i-1}) q_{BI,i-1} \tag{5.45}$$

The flag `validMix` is set to zero in this case.

5.4 Navigation

Function name: `acsNav`

Navigation is used for the pointing modes to determine the reference quaternion, reference rates and current attitude quaternion in a respective coordinate system. Based on the desired pointing modes the navigation function is executed for three different cases.

- ◇ Case 1: Inertial pointing mode
- ◇ Case 2: Nadir or target pointing mode
- ◇ Case 3: No pointing mode

Case 1: Inertial pointing mode

For inertial pointing no additional calculation have to be made as the output of FOGs and STR is already in the inertial frame. The output of navigation attitude is q_{BX}, which is a quaternion from the required frame of reference X to the body frame of reference B. As in this case the required frame of reference is inertial frame so the output is simply

$$\omega_{ref} = [0 \;\; 0 \;\; 0]^T \tag{5.1}$$

$$q_{BX} = q_{BI} \tag{5.2}$$

$$\tag{5.3}$$

Case 2: Nadir or target pointing mode

Principally nadir and target pointing mode are the same and nadir pointing mode is a special case of target pointing mode in which the target position is the center of the Earth.

Therefore the algorithms for these two modes are the same and are represented by the same equations.

Pointing the satellite toward the surface of the Earth requires the attitude information of the satellite's body relative to the earth. This knowledge is provided by combining the GPS information with the STR attitude quaternion. GPS provides the position and the velocity knowledge in the ECF [1](Earth Center Fixed) coordinate system and STR provides the attitude information in the ECI (Earth Center Inertial) frame, therefore a transformation from ECI to ECF is required. So the first step in this case is to estimate this transformation matrix T_{FI}.

T_{FI} could be estimated by combining the Earth rotation angle information with the Earth nutation and precession matrix as

$$T_{FI} = \Theta matNP \tag{5.4}$$

Where matNP is a product of Earth nutation and precession matrix. The two matrices of the Earth nutation and precession are not computed on-board but they could be uploaded regularly via TC as change in their values is very slow over time. Another reason for not calculating them on-board is the commonly use of trigonometric functions in the calculation of these matrices which require significant amount of hardware in FPGA if implemented on-board. A detailed description of these matrices and how their values are derived for the simulation is presented in [33].

The Earth rotation matrix Θ is derived using the angle between vernal equinox and the Greenwich meridian which is related to Greenwich Mean Sidereal Time (θ_{GMST}) [34]

$$\Theta = \begin{bmatrix} cos(\theta_{GMST}) & sin(\theta_{GMST}) & 0 \\ -sin(\theta_{GMST}) & cos(\theta_{GMST}) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{5.5}$$

θ_{GMST} could be calculated as

$$\theta_{GMST} = \theta_{GMST,2000} + \omega_e t$$

$\theta_{GMST,2000}$ is a value of θ_{GMST} at 00:00:00 on 1st of January, 2000 and $\omega_e = 0.0000729211585530$ is the Earth's angular velocity [33]. Further details about $\theta_{GMST,2000}$ and $matNP$ are provided in the Appendix.

[1] A detailed description about different coordinate systems is provided in the Appendix

As the change in the $matNP$ over time is very slow so based on this assumption we could calculate the derivative of the transformation matrix presented in equation 5.4.

$$\dot{T_{FI}} \approx \frac{d\Theta}{dt} matNP = \omega_e \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \Theta matNP \tag{5.7}$$

As T_{FI} and $\dot{T_{FI}}$ are orthogonal matrices therefore

$$T_{IF} = T_{FI}^T \quad and \quad \dot{T_{IF}} = \dot{T_{FI}^T} \tag{5.8}$$

Now using these matrices the GPS position vector r_F and the target position r_{tgtF} represented in ECF frame could be expressed in ECI frame as r_I and r_{tgtI}

$$r_I = T_{IF} r_F \tag{5.9}$$

$$r_{tgtI} = T_{IF} r_{tgtI} \tag{5.10}$$

Velocity vector in ECI could be calculated as

$$v_I = T_{IF} v_F + \dot{T_{IF}} r_F \tag{5.11}$$

In the next step the transformation matrix T_{IX} for the respective system (nadir or target) is calculated. For the nadir coordinate system the place holder X could be replaced with N as $T_{IX} = T_{IN}$ and for target coordinate system this could be replaced with T as $T_{IX} = T_{IT}$, the derivation and the calculation remains the same.

Normalized difference of satellite and target position gives a position vector pointing toward the center of the target which by definition is the z-axis of the coordinate system.

$$z = \frac{r_{tgtI} - r_I}{\|r_{tgtI} - r_I\|} \tag{5.12}$$

The normal to the satellite's orbit plane could be written as a cross product of position and velocity vector.

$$n_{orb} = \frac{r_I \times v_I}{\|r_I \times v_I\|} \tag{5.13}$$

As no yaw should occur relative to the target, the y-axis is chosen as follows

$$y = \frac{z \times \left(n_{orb} \times \frac{r_I}{\|r_I\|} \right)}{\left\| z \times \left(n_{orb} \times \frac{r_I}{\|r_I\|} \right) \right\|} \tag{5.14}$$

The x-axis completes to a right-hand system

$$x = y \times z \tag{5.15}$$

The transformation matrix T_{IX} now could be derived as

$$T_{IX} = [x \quad y \quad z] \tag{5.16}$$

The quaternion q_{IX} could be derived from the above matrix. The detail of deriving quaternion from the DCM (Directional Cosine Matrix) is given in the Appendix B. In order to prevent sudden changes in q_{IX} which would result in errors in subsequent rate computations, a sign memory is introduced and realized with a (4x1) multiplication factor m. If the j^{th} element of q_{IX} was larger or zero in the previous cycle, the j^{th} component of m is set to 1, and to 0 otherwise. In the actual cycle the computed quaternion q_{IX} is multiplied by the component of m, which is equivalent to the number of the maximum element computed during the conversion of the quaternion. The new multiplication factor is calculated and saved for the next cycle.

In case if off-nadir or off-target pointing is required then the reference quaternion does not points to the origin $q_{ref} \neq [0 \quad 0 \quad 0 \quad 1]^T$ and it is multiplied with the q_{IX} to obtain the new desired value.

$$q_{IX} = q_{IX} \cdot q_{ref} \tag{5.17}$$

This quaternion is saved and it is used in the subsequent ACS cycle for the computation of the reference rate. The desired quaternion q_{BX} describing the attitude of the satellite with respect to the target or nadir coordinate system is finally obtained after multiplication of the star tracker quaternion.

$$q_{BX} = q_{BI} \cdot q_{IX} \tag{5.18}$$

The reference rate ω_{ref} is also calculated from this quaternion. As ω_{ref} is equal to the rate of the target coordinate system relative to the inertial frame, which is ω_{XI}, therefore to obtain this the quaternion q_{IX} needs to be inverted as

$$q_{XI} = q_{XI}^{-1} = [-q_1 \quad -q_2 \quad -q_3 \quad q_4]^T \tag{5.19}$$

The method of deriving rate from quaternion is described in the Appendix. In case of a mode change, the reference rate is set to $[0 \;\; 0 \;\; 0]^T$ to avoid large error which might could appear due to the change of a reference system.

Case 3: No pointing mode

The navigation algorithms are not executed if satellite is not operating in any of the pointing mode. In this case the outputs are set to

$$\omega_{ref} = [0 \;\; 0 \;\; 0]^T \tag{5.20}$$

$$q_{BX} = [0 \;\; 0 \;\; 0 \;\; 1]^T \tag{5.21}$$

$$q_{ref} = [0 \;\; 0 \;\; 0 \;\; 1]^T \tag{5.22}$$

5.5 Control

All the control algorithms are implemented in this section of the ACS algorithms. As described in the Section 2.3 different control modes are defined to fulfill the requirements in different situations. This section of ACS algorithm is also classified in the same manner. For each mode there is a different function which runs only if the respective mode is commanded either by FDIR or by user via TC. The name of each function starts with the prefix of `acsCtr` followed by its name. These functions are listed as

1. `acsCtrDet`
2. `acsCtrSafe`
3. `acsCtrIdle`
4. `acsCtrINT`
5. `acsCtrNS`
6. `acsCtrDesat`

5.5.1 `acsCtrDet`

This function is executed if satellite is in the detumble mode. As already explained in Section 2.3 this mode is used to reduce the angular rates which are build either due to the separation of satellite from the launcher or due to some failure during the flight. Satellite during this mode is controlled using famous B-dot law [35]. B-dot is an extremely simple control law which is mostly used for despinning satellites. It relies on MGTs as control actuators. The control law is based on the measurement of the rate of change of body-fixed magnetometer signals. Using only a magnetometer measurements and MGT, the B-dot controller despins the satellite relative to the Earth's magnetic field vector.

B-dot law is based on the fact, that the satellite rotates with the higher angular rate so the change in the magnetic field vector is mainly due to the change of its attitude and the change of magnetic field due to the satellite's motion in a orbit is negligible compared to this. Hence this assumption is only valid when satellite's attitude is changing very fast. The rate of change of magnetic field in this condition is a good approximation of the body rates and this could be used to reduce the angular rates of the satellite.

In general, B-dot control laws command, a magnetic moment whose sign is opposite to that of the rate of change of the magnetic field. There are two strategies to implement the B-dot law. The first is B-dot proportional control, in which a constant positive gain is multiplied with the normalized vector of rate of change of magnetic field. The B-dot proportional control law could be written as

$$m_{Det} = -k_b \frac{\dot{b_B}}{\|b_B\|} \tag{5.1}$$

k_b is the proportional gain which is positive and $k_b = 1024$ is used for the simulations.

As an alternative to the proportional B-dot controller, consider a bang-bang system where, instead of using proportional control to calculate the dipole command, the maximum torquer strength is always used. The bang-bang B-dot control law could be written as

$$m_{Det} = -m_{max}.sign(\dot{b_B}) \begin{cases} -m_{max}, & \text{if } \dot{b_B} > 0 \\ +m_{max}, & \text{otherwise} \end{cases} \tag{5.2}$$

Where $m_{max} = 6\ Am^2$ is the maximum torquer dipole. The simulation results obtained by both of the algorithms were almost the same as both of them are robust against all the perturbations and extremely simple to implement. The bang-bang B-dot law was later selected as it requires even smaller space in the FPGA hardware.

5.5.2 `acsCtrSafe`

This function is executed only if the satellite is commanded in the safe mode. The satellite is commanded to the safe mode subsequent to the initial detumble phase or in later mission upon the occurrence of serious malfunctioning of a software or hardware which cannot be corrected on-board automatically. In this mode the satellite is kept alive by reduced functionality and performance. The system waits in this mode and eventuality is diagnosed on ground.

During the safe mode the satellite spins keeping the solar panels oriented toward the sun. Satellite uses only two sensors (i.e. Magnetometers and Sun Sensors) and one actuator (i.e. Magnetic torquer) in this mode. The selection of the sensors and actuators is based upon the reliability instead of performance, therefore only coarse attitude information is obtained in this mode. The difficulties which were challenging during the design of the control law for this mode are listed below

- ◇ The rate information is not directly available from the sensors used in the safe mode.
- ◇ The sun sensor information is also not available during the eclipse phase.
- ◇ Another design constraint is to avoid the payload cameras facing toward the sun while satellite is moving out of the eclipse.

This was the most challenging control mode to design. There were few techniques [36; 37; 38; 39] which were selected and simulated but finally a very simplified but reliable approach is selected to overcome these complex problems. The satellite rate information is calculated indirectly from the sun sensor and magnetometers, the satellite's negative principal inertial axis $-z_j$ is aligned with the sun vector $s_B = [0 \;\; 0 \;\; 1]^T$. Meanwhile to protect the payload cameras from sun light, a spin is built around this axis; therefore, without having any input from the sun sensors during eclipse, the satellite holds its attitude. This ensures the protection of payload cameras while the satellite is moving from the eclipse to the sun phase.

The control law is divided into two parts to achieve these goals. The first part (T_{sun}) deals with the sun acquisition by pointing the solar panels toward the sun in the illuminated phase and the second part (T_{spin}) generates and controls the spin rate which is necessary to hold the attitude in the eclipse phase. Deviation from the desired attitude relative to the sun vector can be calculated as a cross product of satellite's negative principal inertial axis $-z_j$ with the sun vector in the body coordinate system s_B as $(-z_j \times s_B)$. The rate perpendicular to the sun vector could be calculated as

$$\omega_{\perp} = (\dot{s_B} \times s_B) \tag{5.3}$$

Now control law for the first part (T_{sun}) could be calculated as

$$T_{sun} = k_{sun}(-z_j \times s_B) - k_{\perp}(\dot{s_B} \times s_B) \tag{5.4}$$

k_{sun} and $k_{\perp}$ are the scaler proportional gains. The control law for (T_{spin}) could be written as

$$T_{spin} = k_{\parallel}(\omega_{\parallel,ref} - \omega_{\parallel}) \tag{5.5}$$

$k_{\parallel}$ is a scaler proportional gain. $\omega_{\parallel}$ is the rate of the satellite's negative principal inertial axis $-z_j$ around (parallel) to the sun vector. $\omega_{\parallel,ref}$ was selected as 2 °/s. When the satellite is aligned to the sun, the rate parallel to the sun vector could not be determined, therefore magnetic field information is additionally used to approximate this value.

$$\omega_{\parallel} = \frac{\dot{b_B}(b_B \times s_B)}{\|b_B \times s_B\|} \tag{5.6}$$

T_{spin} is set to $[0 \;\; 0 \;\; 0]^T$ if $(\|b_B \times s_B\|) = 0$. Finally magnetic dipole command for the safe mode could be calculated as

$$m_{SM} = \frac{b_B \times (T_{sun} + T_{spin})}{\|b_B\|^2} \tag{5.7}$$

m_{SM} is set to $[0 \;\; 0 \;\; 0]^T$ if satellite is in eclipse or not in the safe mode.

5.5.3 `acsCtrIdle`

This function is executed if satellite is commanded into the idle mode. This is a sun pointing mode in which solar panels are oriented toward the sun and to define the attitude in inertial space, the body x-axis is aligned with the ecliptic normal vector n_I. The ecliptic normal could be assumed constant during the mission time. Ecliptic normal is orthogonal to the plane of Earth's orbital revolution around the sun, and the Earth's orbital plane is fixed in its orientation in space. The value of n_I could be approximated as [33]

$$n_I = [0 \quad -0.39778 \;\; 0.91748] \tag{5.8}$$

The transformation from the inertial frame to the body frame could be carried out using star tracker quaternion q_{BI} which is converted to the transformation matrix T_{BI} and then multiplied with the n_I.

$$n_B = T_{BI} n_I \tag{5.9}$$

For sun alignment coarse sun sensors only provide coarse sun vector information which is sufficient for this mode as there is no strict requirement for the accuracy. The control law for the idle mode could be divided into three parts as

$$T_{IM} = T_{sun} + T_n + T_\omega \tag{5.10}$$

T_{sun} is the control law which provides the sun pointing for the solar arrays. This could be defined by multiplying a constant proportional gain k_{sun} with the cross product of desired

body vector ($s_{ref} = [0 \;\; 0 \;\; -1]$]) and the sun vector in body coordinate system s_B. T_{sun} is set to zero in case satellite is in eclipse.

$$T_{sun} = k_{sun}(s_{ref} \times s_B) \tag{5.11}$$

T_n provides alignment of the body x-axis $n_{ref} = [1 \;\; 0 \;\; 0]$] toward the ecliptic normal. This is also set to zero in the case of star tracker blinding. This could be written as

$$T_n = k_n(n_{ref} \times n_B) \tag{5.12}$$

T_ω is used for the rate damping. It is simply negative feedback of the rate measured by the rate gyros.

$$T_\omega = -k_\omega \omega_B \tag{5.13}$$

The output T_{IM} is set to zero in case satellite is not in the idle mode.

5.5.4 `acsCtrINT`

This function is executed if the satellite is commanded into any of the pointing modes (i.e. inertial pointing mode, nadir pointing mode or target pointing mode). Another function `acsCtrErrorINT` is also executed prior to `acsCtrINT`. This function is used to calculate the error quaternion q_e and the rate error $\Delta\omega$. q_e is calculated by multiplying the reference quaternion q_{ref} with the actual attitude q_{BX}. In case if the fourth element of q_e is negative then the whole error quaternion is multiplied by -1, this is to ensure that the slew maneuver always select the shortest possible path as q_e is controlled toward [0 0 0 1]. Another output of `acsCtrErrorINT` is the $tgtFlg$ which is set to 1 if the required accuracy of 150″ is achieved and is set to 0 otherwise.

The outputs of the function `acsCtrErrorINT` are the inputs to the `acsCtrINT`. An error quaternion feedback controller is selected for the pointing modes. This controller is described in detail in [40]. The control law could be written as

$$T_{INT} = -Ksat(Pq_e) - C\Delta\omega \tag{5.14}$$

$sat()$ is a saturation function which limits the values of Pq_e within the range of $-1 \leq (Pq_e) \leq 1$. K, P and C are the control gains which are recalculated whenever the commanded mode is changed. This change is indicated by the $updtFlg$ which is set to 1.

According to [40] these gains could be defined as

$$K = diag(k1, k2, k3) \tag{5.15}$$

$$P = diag(p1, p2, p3)I \tag{5.16}$$

$$C = c_{INT}I \tag{5.17}$$

where I is the satellite's inertia matrix and k_i, p_i and c are scaler gains which could be determined using selected damping ratio ζ and natural frequency ω_n for each mode. The values used for the simulation are shown in the Table 5.1.

Table 5.1: Gains selection table

Mode	ζ	ω
Inertial	1	0.4
Nadir	1	0.4
Target	1.5	0.75

From the values presented in the Table 5.1 we could calculate

$$k_{INT} = 2\omega_n^2 \tag{5.18}$$

$$c_{INT} = 2\zeta\omega_n \tag{5.19}$$

The gain k_i depends upon error quaternion vector elements $q_{e,i}(0)$ at the time of mode change. if one or more values of $q_{e,i}(0)$ are zero or close to zero then they are replaced by the value of $q_{min} = 0.1$ to ensure that the gains are large enough. k_i could be calculated as

$$k_i = c_{INT}\frac{q_{e,i}(0)}{\|q_{e,i}(0)\|}\omega_{max} \tag{5.20}$$

ω_{max} is the maximum allowed rate limit of 1 °/s during the slew maneuver. Now K and P could be calculated as

$$K = diag(k1, k2, k3) \tag{5.21}$$

$$P = K^{-1}k_{INT}I \tag{5.22}$$

As K is a diagonal vector so K^{-1} need not to be calculated, it is only the reciprocal of the diagonal elements. This is important for the implementation of the algorithms into the hardware because a significant amount of hardware in FPGA is required to calculate the inverse of a matrix.

The output T_{INT} is now calculated according to the equation 5.14 using all the gains and this output is set to $[0\ \ 0\ \ 0]^T$ if satellite is not operating in any of the pointing modes.

5.5.5 acsCtrNS

This function is executed for the nullspace control. This is not an independent mode and it is to support the functioning of reaction wheels therefore it is only used when reaction wheels are activated. If the reaction wheels are not activated, its output T_{null} is set to zero.

To calculate the null space control, first the wheel momentum is computed from the wheel speeds ω_{RW} and the wheel inertial I_{RW}

$$h_{RW,i} = I_{RW}\omega_{RW,i} \qquad fori = 0, 1, ..3 \tag{5.23}$$

The applied control is explained in [41].

$$T_{null} = k_{null}u_{null}u_{null}^{T}h_{RW} \tag{5.24}$$

The constant null space vector u_{null} is computed by a singular value decomposition (SVD) [1] of the wheel alignment matrix A_{RW}^{PI}. The constant null space vector u_{null} used for FLP is $u_{null} = [-0.5 \;\; 0.5 \;\; 0.5 \;\; 0.5]$.

5.5.6 acsCtrDesat

This function is executed for the desaturation control. Like nullspace this is also not an independent control mode and it is designed to facilitate the functioning of reaction wheels. As from the name it is clear that this mode is used unloading the accumulated angular momentum or for the desaturation of the wheels. This is done by using magnetic torquers and basically the angular momentum of the satellite and the angular momentum of the wheels is controlled to zero using the torquers. The total angular momentum consisting of wheels and the satellite motion is computed is body coordinate system as

$$h_B = I\omega_B + A_{RW}^{PI}I_{RW}\omega_{RW} \tag{5.25}$$

The reference angular momentum h_{ref} is set to zero and it is controlled using a control law

$$m_{Des} = -k_{offload}\frac{b_B \times (h_B - h_{ref})}{\|b_B\|} \tag{5.26}$$

$k_{offload} = 1000$ is a scaler gain. The magnetic dipole moment for wheels desaturation is forwarded to the magnetic torquer command block.

[1] This could be done by using *svd* command in Matlab

5.6 Command

In this section of the ACS algorithms the commands for the actuators are calculated. The inputs from the different functions of control blocks are summed together and the commands for the actuators are calculated from it. As two types of actuators are used in the ACS of FLP therefore in this section there are two functions, each for each type of actuator. The name of each function has a prefix of `acsCmd` followed by its name. These functions are

1. `acsCmdPWM`
2. `acsCmdRWTrq`

5.6.1 acsCmdPWM

This function is used to command magnetic torquers. First of all, all the magnetic dipole commands from the control blocks are added together as

$$m_{mgt} = m_{Det} + m_{SM} + m_{Des} \tag{5.1}$$

The command m_{cmd} is calculated by multiplying the m_{mgt} with the transpose of actuator alignment matrix A_{MGT}.

$$m_{cmd} = A_{mgt}^{T} m_{mgt} \tag{5.2}$$

The output command is a row vector that consists of three two bits elements, each specifying with its first bit the on/off state of the torquer and with its second bit the sign of the torque.

$$cmd = [cmd_1 \;\; cmd_2 \;\; cmd_3] \tag{5.3}$$

$$cmd_i = [set_i \;\; sign_i] \tag{5.4}$$

As the command should be pulse width modulated with a period of 5 s, a counter is initialized running from 1 to 50. It is increasing by one with every ACS cycle and reset to one when reaching a value larger than 50. To determine the fraction of the whole PWM (Pulse Width Modulation) cycle in which the torquers have to be active, the ratio between magnitude of each component of m_{cmd} and the maximum torquer dipole is computed

(i.e. a 3 Am^2 command should activate the torquers for 50% of the cycle). To make it comparable to the counter value, this ratio is multiplied by 50.

$$\alpha_i = 50.\frac{|m_{cmd,i}|}{m_{max}} \qquad for \;\; i = 0, 1, 2 \tag{5.5}$$

As long as the counter value is smaller than or equal to the value of α_i , the appropriate torquer is activated (first bit of $cmd_i = set_i$ is set to 1) taking into account the sign of $m_{cmd,i}$ (second bit of $cmd_i = sign_i$ is set to 1 if $m_{cmd,i} > 0$ and to 0 otherwise).

5.6.2 acsCmdRwTrq

This function adds the torque requests from the control blocks and computes the final torque to be sent to the reaction wheels. First of all, all the control torques are added together

$$T_{RW} = T_{INT} + T_{IM} + T_{null} \tag{5.6}$$

This torque is finally transformed from body to wheel coordinate system by multiplication with the pseudoinverse A_{RW}^{PI} of the wheel alignment matrix.

$$T_w = -A_{RW}^{PI} T_{RW} \tag{5.7}$$

The negative sign is required as the resulting torque on the satellite is the reaction to the commanded wheel torque. The input torque T_w is not scaled and it may exceed the torque limit of $T_{max} = \pm 5mNm$ that can be realized by a single reaction wheel. In this case, the input torque is scaled along its largest element to keep the desired torque direction in the body system. If $max(T_w)$ is less than T_{max} then the output $T_{cmd} = T_w$ otherwise it is scaled by using following equation.

$$T_{cmd} = T_w.\frac{T_{max}}{max(T_w)} \tag{5.8}$$

5.7 Outputs

As inputs the outputs are also of two types. The first type is the information which is forwarded to the hardware directly. These are the outputs of the command blocks which are sent to the magnetic torquers and the reaction wheels. The other type of outputs are TM (telemetry) messages and the information required to FDIR such as current ACS mode, target flag, body rates, sun vector for the calculation of sun proximity.

CHAPTER 6

Algorithms Implementation

This chapter gives an overview of the implementation of the ACS algorithms on FPGA. In this chapter only the highest abstraction level details are provided which are platform independent and could be implemented and tested on any platform with its platform specific supporting libraries. This chapter in the beginning also provides some details of the developmental path of the ACS algorithms which finally led to the implementation and testing of these algorithms. The algorithms during the developmental stage were first implemented as a software in Matlab/Simulink environment where extensive testing and optimization of these algorithms were carried out. After verification in the Matlab/Simulink environment these algorithms were ported into Handel-C, which is a Hardware Description Language (HDL) for implementing these algorithms as a hardware. So in this chapter both of the steps are elaborated, first starting with the implementation in the Matlab/Simulink followed by the Implementation as a hardware.

6.1 Developmental path

The whole developmental path consists of various stages which are shown in the Figure 6.1. The ACS algorithms are developed in the Matlab/Simulink environment based on the system level requirements. Extensive testing of the algorithms in this stage is conducted to verify and validate the results. In the next phase these algorithms are ported manually into the Handel-C language. DK design suite provides the environment and compiler for

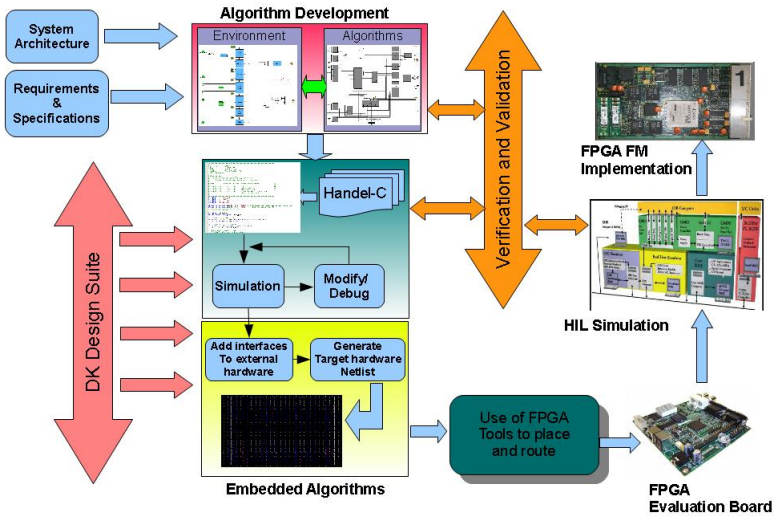


Figure 6.1: ACS Algorithms Developmental Path

the Handel-C language. DK design suite also provides simulation environment which was used for conducting the performance verification and the whole design passed through many Modify/debug cycles before it became ready to be tested with the hardware. DK generates directly the logical netlists of the target hardware. These netlists could be after then used to program the target FPGA using manufacturer's place and route tool. FPGA evaluation boards like RC10 and RC240 manufactured by Agility are used for the HIL (Hardware in the loop testing) to evaluate the performance and accuracy of these algorithms. As these algorithms are platform independent so the same algorithms after verification and validation will be implemented on the flight hardware with its hardware specific support libraries.

6.2 Implementation of the algorithms in Matlab environment

Matlab[42] is a high performance language for technical computing which has a vast collection of computational algorithms ranging from elementary functions like sum, sine, cosine, and complex arithmetic, to more sophisticated functions like matrix inverse, matrix eigenvalues, and quaternion algebra. Simulink is an add-on product to Matlab which provides interactive, graphical environment for modeling, simulating and analyzing of dynamic system. It enables rapid construction of virtual prototypes to explore design

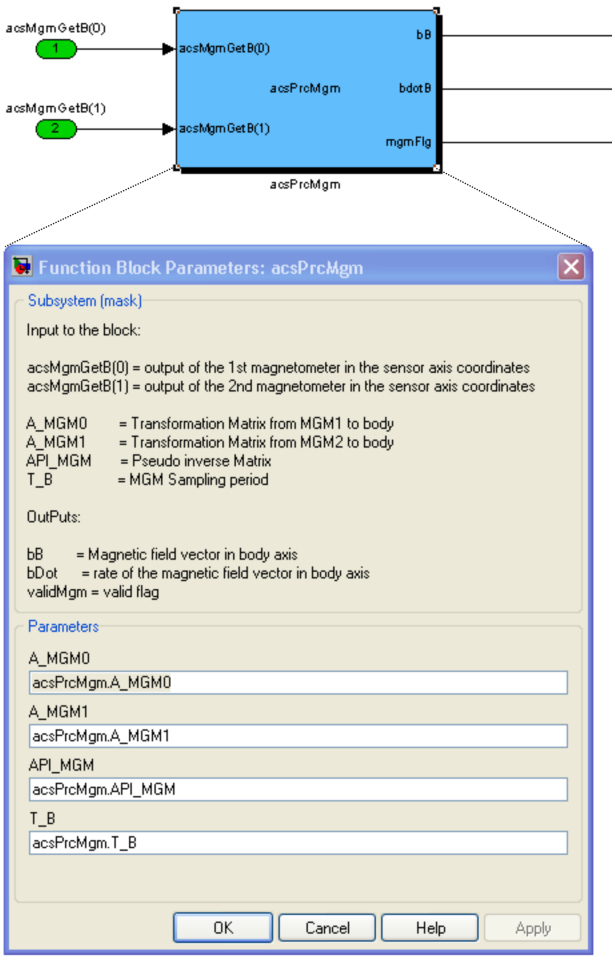


Figure 6.2: Screen shot of acsPrcMgm

concepts at any level of detail with minimal effort.

Fast prototyping with minimal effort was the reason to select Matlab/Simulink as a development environment. The on-board algorithm part was implemented in Simulink using Embedded Matlab Function blocks (EML) instead of using conventional Simulink blocks because it was more close to how it have to be implemented later on. The Embedded Matlab Function block allows you to add Matlab functions to Simulink models for deployment to embedded processors. This capability is useful for coding algorithms that are better stated in the textual language of Matlab than in the graphical language of Simulink. This block provides optimizations for generating efficient, production-quality C code for embedded applications.

All the functions described in Chapter 5 are implemented in parallel as independent function as shown in the Figure 5.1. Also the parallel blocks of the Simulink provide

apparently a similar structure as it has to be implemented on the FPGA.

The constants used in these algorithms are stored as structures in a separate data file. These constants are passed to the respective functions as parameters instead of passing them as inputs. For example the function `acsPrcMgm` is shown as a screen shot in Figure 6.2.

As it is shown in this figure, the constants used in each function are stored as a structure having the same name as of the function. So for the function `acsPrcMgm`, the name of the structure is also `acsPrcMgm` and `acsPrcMgm.A_MGM0` is the alignment matrix represented as A_{MGM0} in equation 5.1.

The math library is also implemented as external Matlab functions saved as m-files in one folder called `acsMathLib`. For the Matlab version 7.5.0 (R2007b) and above, it is possible to call external Matlab functions inside the Embedded Matlab Functions (EML blocks). Since Matlab contains a number of built-in mathematical functions which could be used to save the development time, therefore this library only contains very few mathematical functions related to the quaternion algebra. The rest of the mathematical functions are used from the Matlab's built-in libraries.

6.3 Implementation of the algorithms as a hardware

Implementation of ACS algorithms into the hardware was carried out using Handel-C. Handel-C is a programming language and Hardware Description Language (HDL) for compiling algorithms into hardware design for the FPGA. Further details about Handel-C are provided in the following section. The functions developed in the Simulink EML blocks environment were manually ported into Handel-C using the same function name with the same inputs and outputs.

6.4 Handel-C

Handel-C is a powerful language superset of ANSI-C adding simple constructs that make it possible for designers using the DK Design Suite to accelerate complex algorithms directly in optimized FPGA logic. DK design suite a product of Agility Design Solutions Inc. provides the design environment of Handel-C. The hardware design that DK produces is generated directly from the Handel-C source program. There is no intermediate

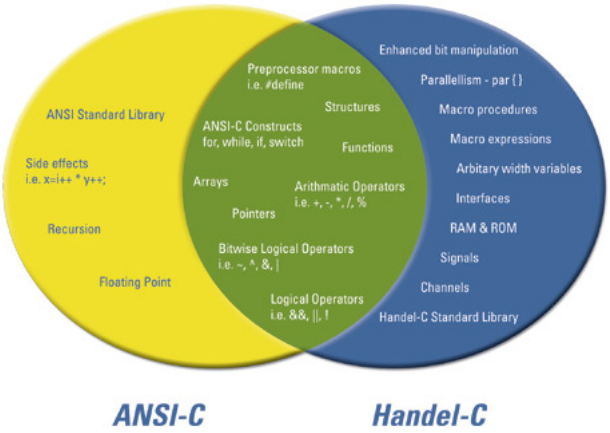


Figure 6.3: Comparison of Handel-C with ANSI-C [1]

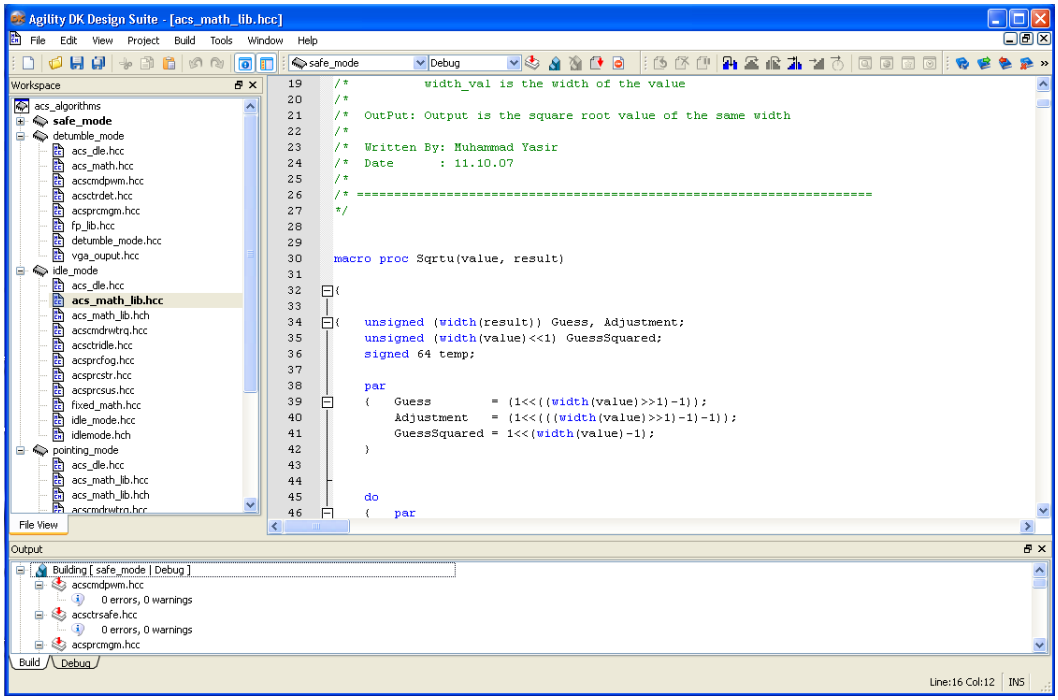


Figure 6.4: DK Design Suite Screen Shot

'interpreting' layer as exists in assembly language when targeting general-purpose microprocessors. The logic gates that make up the final Handel-C circuit are the assembly instructions of the Handel-C system.

Handel-C uses much of the syntax of ANSI-C with the addition of inherent parallelism. Sequential programs could also be written using Handel-C but parallel constructs are preferred to gain maximum benefit in performance from the target hardware. Although Handel-C has many things common with the ANSI-C but there are also some differences which are shown in the Figure 6.3. Use of fixed-point instead of floating point arithmetic is one of these differences. Although use of floating point in Handel-C is possible but it is not used in the implementation of ACS algorithms for the FLP since it is supported through external libraries which consume a significant amount of hardware on FPGA.

6.5 DK design suite

DK design suite simplifies the path from the algorithm to the implementation by generating hardware design using Handel-C and place and route tool from the FPGA manufacturer. DK Design Suite makes it easy to compile C code onto an FPGA and iterate on the implementation to improve performance and/or area. A screen shot of DK design suite is presented in the Figure 6.4.

The power of DK Design Suite comes from the Handel-C development environment enabling the developer to direct the C synthesis for uncompromised Quality of Results while offering enhanced productivity. DK Design Suite maintains a single source for simulation and implementation giving the developer a fast, automated solution from algorithm change to new implementation.

The DK Design Suite provides a complete design flow for implementing high-level language algorithms in RTL or as EDIF netlists. Algorithms can be written directly in Handel-C, or ported from ANSI-C or C++. ANSI-C and C++ code could be co-simulated with Handel-C to support a modular porting process. Handel-C code can be compiled to VHDL or Verilog for RTL synthesis, or directly to EDIF to target FPGAs. The DK design flow is shown in the Figure 6.5.

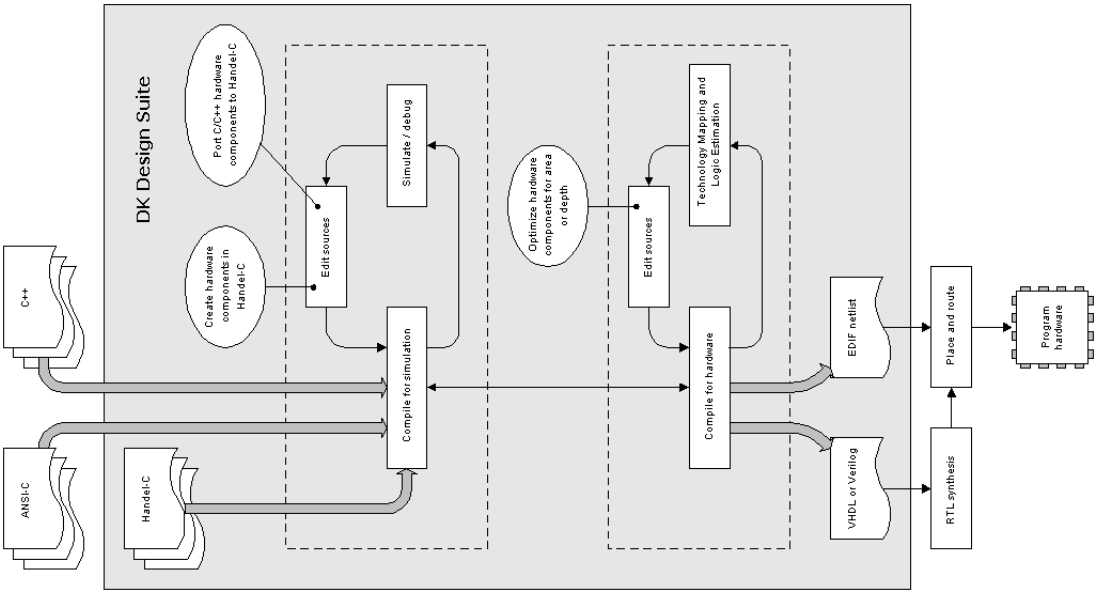


Figure 6.5: DK Design Flow [2]

6.6 Implementation strategy

The implementation of the function is carried out in Handel-C in a similar way as they are implemented in the Matlab/Simulink environment. The name of the functions explained in Chapter 5 are kept the same and the name of the variables, the calculation procedure and loop structures are also left unchanged during porting of the code into the hardware.

The functions are replaced with the macros with the same name and with the same inputs and outputs. The main difference between a macro and a function in Handel-C is that function uses a shared hardware so its hardware is built once and this is linked to every function call while macro built a new copy of hardware every time it is called. The implementation was started with the less complex and comparatively small functions so in a first attempt only algorithms for the Detumble mode were implemented followed by the algorithms of the Safe mode and later the complex algorithms were implemented which required complex matrix/quaternion algebra, and trigonometric functions. The structure of ACS algorithms from the Matlab/Simulink model is kept the same. The code was capable of being directly translated into Handel-C and it was converted line by line into Handel-C with the modifications. Some of these modifications are listed below.

Conversion of floating point arithmetic into fixed point arithmetic

The main modification was to redefine all the variables as integers instead of double precision floating numbers. Matlab uses double precision floating numbers which are stored in a 64-bit word, with 52 bits for mantissa, 11 bits for exponent and 1 bit for the sign of the number. It was not possible to use this structure in Handel-C due to hardware area constraints as floating-point arithmetic is more complex than integer or fixed-point arithmatic and tends to require more hardware. So all the variables were redefined as integers. The length of the variables was estimated/approximated for the first implementation and it was refined later after extensive testing of the algorithms for example the word length to represent quaternion was set to 32 bit as it directly affects the accuracy in the pointing modes and the word length to represent the sun vector in the body axis is set to 16 bits as only a coarse sun pointing is required during the FLP operations.

Matrix operation

The beauty of a Matlab is its ease with which it handles the matrices. The most basic Matlab data structure is the matrix. Additionally Matlab is equipped with a number of built-in libraries to support various matrix operations. While in Handel-C, although it supports arrays used in the same way as in ANSI-C, however, there are implications resulting from the way arrays are implemented in hardware. An array can be seen as a collection of variables which can all be accessed in parallel, with elements either specified explicitly or indexed by a variable. Explicit access to individual array elements is efficient, but indexing through an array can generate significant amounts of hardware, particularly if it is done from more than one point in the code. But for the ACS algorithms the use of arrays could not be avoided so to overcome this problem and to obtain an efficient and optimized code at some places the array was replaced with the RAM simply by adding `ram` keyword at the start of the array declaration:

```
static ram signed int 24 A_MGM0[3][3]
```

This will create more efficient structure in the hardware, but will now be limited to a single access per clock cycle which is a performance vs area trade-off.

Parallel statements

Handel-C parallelism is true parallelism, not the time-sliced parallelism familiar from general-purpose computers. When instructed to execute two instructions in parallel, those two instructions will be executed at exactly the same instant in time by two separate pieces of hardware. So instructions which are not inter-dependent were executed with the parallel construct in the Handel-C for example

```
par { x = 1;
      y = 2;
      z = 3;
    }
```

The three statements will be executed in a same clock cycle in the hardware. This is not the time-sliced pseudo parallelism of a conventional microprocessor implementation but rather three specific pieces of hardware built just to perform these three assignments.

Mathematical functions

As stated above, implementation in the Matlab uses most of the supporting mathematical functions within the built-in libraries of Matlab. But for the implementation in the Handel-C a challenging task was to create this library. As the mathematical function consumes a significant amount of hardware look up tables (LUTs) so the goal was to built the mathematical functions which consume less amount of hardware and at the same time they do not affect the overall performance of the ACS algorithms.

Many different schemes are used to optimize the mathematical functions aiming to produce circuits which are small and run at higher clock rate. Sometimes to select performance over area or vice versa is a trade off. The first implementation was the worst in that respect but that was because the code was merely changed to conformance with the Handel-C language standard. Lot of optimization was carried out to bring the mathematical functions within the performance and the area bounds. Optimization of the algorithms is discussed in detail in the following section.

Implementation results

The implementation of the ACS algorithms were first carried out with the testing hardware. RC10[1] based on Xlinix Spartan-III FPGA was used as a target device. The results

[1]Further details about RC10 board are given in the Section 8.1

of the implementation are listed in the Table 6.1. These results were not achieved during the first attempt of implementation rather an ample amount of optimization was required to achieve these results. The discussion about the optimization process is provided in the next section of this chapter. The results of Nadir/Target pointing mode are not listed in Table 6.1 because these modes still require more hardware than what is offered by RC10. However the algorithms are already ported into Handel-C and the relevant functions are tested individually.

Table 6.1: Implementation results

Mode	No. of equivalent gates	No. of LUTs	No. of Flipflops
Detumble	223,970	3,300 (12%)	2,527 (9%)
Safe	506,274	4,758 (17%)	3,863 (14%)
Idle	555,553	12,942 (48%)	10,698 (40%)
Inertial Pointing	638,001	19,403 (72%)	13,102 (49%)

6.7 Optimization of the code

Optimization of the code was the most challenging task in this regard because it involves optimization in two different and sometimes inter-conflicting dimensions which could be written as:

- ◇ Optimization of overall clock cycle efficiency
- ◇ Optimization of hardware resources (i.e. system gates, area of the logic, LUTs etc)

Sometimes a mathematical operation which could be conducted in a single clock cycle has to be divided into several clock cycles to extricate the huge amount of hardware this operation is utilizing. Similarly shared hardware could be used for the mathematical functions which are called in sequence but at the same time sometimes using routing functions to the shared hardware resource consumes more hardware than building a new copy of a same hardware and it also improves the clock cycle efficiency. The optimizations done for the ACS algorithms are listed in the following sub-sections.

6.7.1 Use of Macro proc vs. function

Use of macro proc instead of a function always give better performance with respect the clock cycle efficiency however it utilizes some extra space in the hardware. Another difference between a macro proc and a function in Handel-C is the number of copies of hardware that result. Placing a block of frequently used code in a function means that one copy of the code will exist in the hardware, and every time the function is called this single copy of the code will be used. A macro proc, however, builds a fresh copy of the code every time it is called. This means that if the code block needs to be called several times in parallel, a single function can not be used, as multiple copies of the code are required. Functions employ call-by-value on their parameters, whereas macro effectively employ call-by-reference.

All the ACS functions which have to called only once are implemented as macro proc instead of a function, for example all the functions listed in the Chapter 5 like processing functions (`acsPrcMgm`, `acsPrcSuS`, `acsPrcGPS` etc), navigation function (`acsNav`), and control functions (`acsCtrINT`, `acsCtrSafe` etc) are implemented as macro procs. Similarly the mathematical functions which are not frequently used are also implemented as macro procs.

6.7.2 Use of shared hardware

Mathematical functions which are particularly large (in hardware terms) block of code and are used infrequently but in several places, are implemented as shared hardware. For example mathematical functions like multiplication of a matrix, calculating a norm of a single dimension vector, various quaternion algebra functions like quaternion multiplication, quaternion to DCM and DCM to quaternion, cross multiplication of vectors etc are implemented as functions and are represented as a shared hardware which could be called from different parts of the algorithms.

There are also shared expressions in Handel-C which are used in the ACS design. Shared expressions are evaluated in a single clock cycle, where the expression is assigned to a variable. Functions and macro proc may involve control logic, and may take many cycles. Example of one such shared expression is written below

```
shared expr fixed_add_0(a,b)=a + b;
```

This shared expression `fixed_add_0` takes the inputs `a` and `b` and returns the output which is the sum of both the inputs. One thing which should be considered in the use of

shared hardware is that multiple sequential calls to a single shared hardware will result in complex circuitry at the entry and exit points of that particular hardware, leading to even bigger overall circuits.

6.7.3 Implementing basic-level arithmetic functions

The arithmetic operators provided in Handel-C are listed in Table 6.2. Usually these

Table 6.2: Arithmetic operators in Handel-C

Operator	Meaning
+	Addition
-	Subtraction
*	Multiplication
/	Division
%	Modulo arithmetic

arithmetic operations are conducted in a single clock cycle. The amount of hardware required division of any integer length greater than 16 bit is prohibitively large. Similarly the amount of hardware required for conducting multiplication in a single clock step is also significantly large. The first design which was implemented was using arithmetic functions provided with the Handel-C. Soon after few functions are ported into Handel-C it was unavoidable to make a new divide function and for multiplication shared multipliers were used. But with the increase of the size of the code, it became impossible to keep the area of the logic within the bounds of flight hardware. To solve this problem, the code for the multiplication and division was separately written and implemented in a hardware.

Implementing multiplier

The simplest method of Implementing a multiplier is adding a series of partial products as shown in Figure 6.6. It is based upon an adder-accumulator, along with a partial product generator and a hardware shifter. This is relatively slow, because adding N partial products requires N clock cycles. This could be made fast by detecting the position of ones in a binary number and then by shifting to that position directly. This could save significant number of cycles and now the total number of cycles would be equal to total number of ones in a binary number. The position of right most one in a binary number

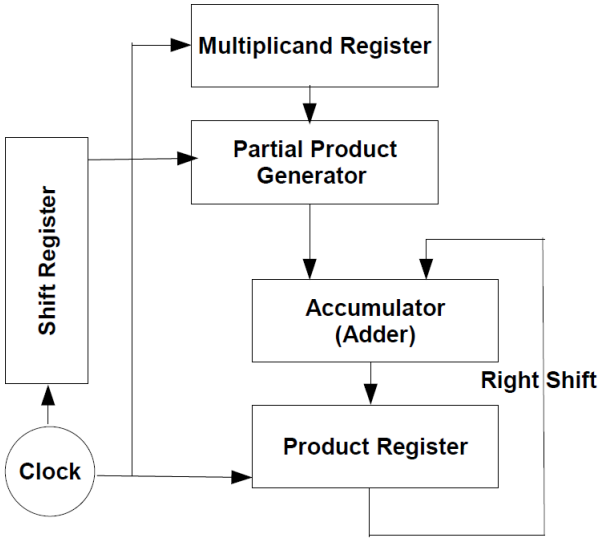


Figure 6.6: Simple adder-accumulator multiplier

could be calculated by the expression `rmo` which is a function of a standard library of Handel-C.

Implementing divider

The divider was implemented using long division method. It uses two inputs `a1` and `a2` and the `Result` is the integer part of $a1/a2$. Although not very useful in itself, the body of the program could be run in parallel with some other task, giving Handel-C programs access to division which is not provided as one of the standard arithmetic operators. The code for division is listed below

```
macro proc myDivide (a1,a2,Result)
{
        unsigned (width(a1)) a, mult, result;
        unsigned (((width(a1))<<1) - 1) c;
        unsigned int 1 getSign1, getSign2;

        getSign1 = sign(a1);
        getSign2 = sign(a2);

if (a2 == 0)

        Result = 0;
```

```
else{
        a = (unsigned) abs(a1);
        c = (unsigned) abs(a2) @ 0;
        mult = 1<<((width(a))-1);
        result = 0;
        while (mult!=0)
        {
        if ((0 @ a) >= c)
        par     {a -= c <- width(a);
                 result |= mult;
                }
        par     {c = c >> 1;
                 mult = mult >> 1;
        }
   }

Result =  (signed) result;
Result =  (getSign1 == getSign2)  ? Result : -Result;}}
```

6.7.4 Use of hardware multipliers

There are `18 bit x 18 bit` hardware multipliers available both in the testing boards and the flight hardware. These multipliers are identified by the DK design suite for a particular hardware and are used automatically to improve the performance. DK design suite uses these multipliers during its internal optimization if required. However DK design suite could be forced to use these multipliers by using `xilinxmult(Result,a,b)` provided by the manufacturer of the hardware in the platform development toolkit.

These multipliers are only used for the multiplications of the integers whose word length is either 18 or less than that. The result is a 36 bit integer. Due to this limitation it was not possible to use the hardware multipliers because most of the calculations in the ACS algorithm involve integers of higher bit length. To overcome this problem, multiplication of `32 bit x 32 bit` was divided into 4 multiplications of `16 bit x 16 bit` which were then combined to get a result. In this way four `18 bit x 18 bit` hardware multipliers were used to calculate single `32 bit x 32 bit` multiplication. This code could not be used for `36 bit x 36 bit` multiplication of signed variables as Handel-C stores

one bit for the sign and therefore this method is only useful for multiplication lower than `36 bits`. The code for this function could be listed as:

```
int 64 FixMultLong (int 32 a, int 32 b)
{
        int 64 Result;
        int 18 b1, b2;
        int 18 a1, a2;
        int 36 Res1, Res2, Res3, Res4;
        unsigned  getSign1, getSign2;

        par {   getSign1 = sign(a);
                getSign2 = sign(b);
                a = abs(a);
                b = abs(b);
            }
        par {   a1 = 0@a[15:0];
                a2 = 0@a[31:16];

                b1 = 0@b[15:0];
                b2 = 0@b[31:16];
            }
        par {   Res1 = (0@a1)*<=/@b1);
                Res2 = (0@a2)*<=/@b1);
                Res3 = (0@a1)*<=/@b2);
                Res4 = (0@a2)*<=/@b2);
            }
        Result = (0@Res1)+((0@Res2)<<16)+((0@Res3)<<16)+((0@Res4)<<32);
        Result =  (getSign1 == getSign2)  ? Result : -Result;
            return Result;
}
```

6.7.5 Client-Server architecture

When an operation or device driver is particularly complex or requires significant resources when implemented in hardware, it may not be efficient to use it repeatedly in different locations in a Handel-C program. A client-server architecture puts all the complexity into a "server" process which runs indefinitely, and provides a "client" API (Application

Programming Interface) through which you can gain access to the resources of the server. The end result is similar to using a function in Handel-C, but allows more control, as you can specify an API and devise methods of handling multiple simultaneous requests for access to the resource.

The basic arithmetic functions of multiplication, division, addition and subtraction along with the function of matrix multiplication are realized as server process as these were very frequently used in the ACS algorithms. The inputs and the outputs in the server are passed using structures. As the server will take few cycles to calculate the result and store it in the data structure, a delay must be inserted in the client macro to allow for this.

6.7.6 Use of simple and effective mathematical algorithms

The efficiency and the performance of algorithm could be improved using simple but effective mathematical algorithms. Many algorithms were replaced with the improved algorithms producing similar results. few such examples are given below.

Quaternion multiplication and cross product algorithm

The algorithms of quaternion multiplication, cross product were replaced with the matrix multiplication which is realized by the client-server architecture.

Square root calculation algorithm

The algorithm to calculate the square root was first realized using Newton's method but later it was replaced with the Fast-Integer square root method. The flow chart of this method is shown in the Figure 6.7. This method uses the binary nature of the input. Each digit in a binary number represents a power of two. By successively rotating through each bit, or power of two and testing the result against the desired value, i.e. squaring the guess and checking if it is greater than the original argument, the approximate root gets closer and closer to the actual value. In conjunction with this, the value is achieved quickly.

Trigonometric algorithm

To calculate sine and cosine, there were three possible solutions. First was to use look up tables, second was to use polynomial approximation (Maclaurin Series) and the third possibility was to use the CORDIC (Coordinate Rotation Digital Computer). Compared

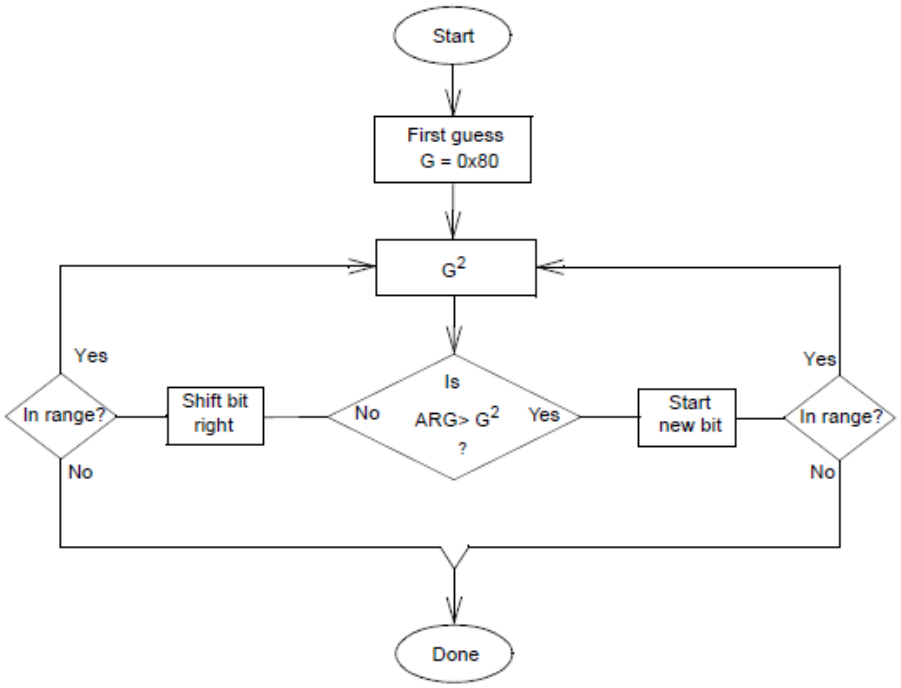


Figure 6.7: Fast Integer Square Root Flowchart

to the other approaches, CORDIC was selected as it is hardware efficient algorithm which does not involve any multiplication or division. The CORDIC algorithm provides an iterative method of performing vector rotations by arbitrary angles using only shifts and adds. The algorithm was first published by Volder [43]. Basic idea of CORDIC algorithm is explained in the Appendix C.

Multiplication/division with a constant

Another simplification which could save a significant amount of hardware is to replace the constant multiplication or division with shift and add procedures. Following example will show how multiplication with a constant 10 is realized in Handel-C.

```
macro proc MultBy10 (a, Result)
{
        Result = ((a<<2)+a)<<1;
}
```

In this example the input a which has to be multiplied with a constant 10 is shifted to the left by 2 which is equal to as if it is multiplied with 4 (2^2), after this the input is added

Table 6.3: Optimization results of idle mode

Parameter	First Attempt	After Optimization
Equivalent system gates	1,505,318	555,553
Equivalent Look Up Tables (LUTs)	24381	13,341
Percentage of FPGA resources (Testing Hardware)	90%	48%
Overall clock cycles	1300	1820
Time elapsed per ACS cycle	108.0 μs	151.20 μs
Percentage of available ACS cycle time	0.108%	0.15%

to the product which will make it equals to the input multiplied with 5. Finally one left shift (2^1) will result in the output which is equal to the input multiplied with 10.

6.8 Optimization results

The implementation of the algorithms were conducted step by step by porting each function into Handel-C environment. In a first attempt algorithms for detumble mode and safe mode were ported into Handel-C and tested. Since algorithms for detumble mode and safe mode are not very complex so no optimization was required at this stage. The situation become very complex later with the inclusion of more functions from the Idle mode. It was not possible to test the Idle mode with the testing hardware board (RC10 board). The process of optimization was started at this point and in a first attempt the Idle mode was demonstrated with the testing FPGA board and after that with the extensive optimization it was reduced to a significant extent of the hardware. The optimization results of the idle mode are presented in the Table 6.3.

A significant amount of hardware was extricated as a result of the optimization however the overall clock cycles required to perform idle mode calculations were increased from 1300 to 1820 clock cycles. Since one advantage of using FPGAs is their high clock rate due to which this increase will not effect the overall design.

CHAPTER 7

Simulation Environment

An essential requirement in order to develop and verify various attitude control algorithms is the availability of space environment model. Such a tool provides necessary infrastructure at the preliminary stage to examine and validate the algorithms prior to hardware implementation.

A simulator was set up to develop and verify attitude control algorithms for the FLP. The simulator was developed in Matlab/Simulink environment. The top level diagram of the simulator is shown in the Figure 7.1. The structure of the simulator is kept close to the overall structure of ACS algorithms. The yellow blocks in the top level diagram represents the space environment along with the ACS hardware. The blocks in the blue color contain all the functions that have to be implemented on the FPGA-based OBC and the blocks in the green color show the user interface which is required to initialize the inputs and monitor the output of the simulator.

A fabulous support provided by EADS Astrium in a form of Astrium STARS library tool box must be acknowledged here. This tool box contains several key functions which not only helped but also accelerated the development of the space environment.

Each block shown in the top level digram is discussed briefly in this chapter to provide an overview of the simulator. The blocks and sub-blocks of this simulator are well documented in [44] and [45]. The main blocks of the simulator are

- ◇ Space environment

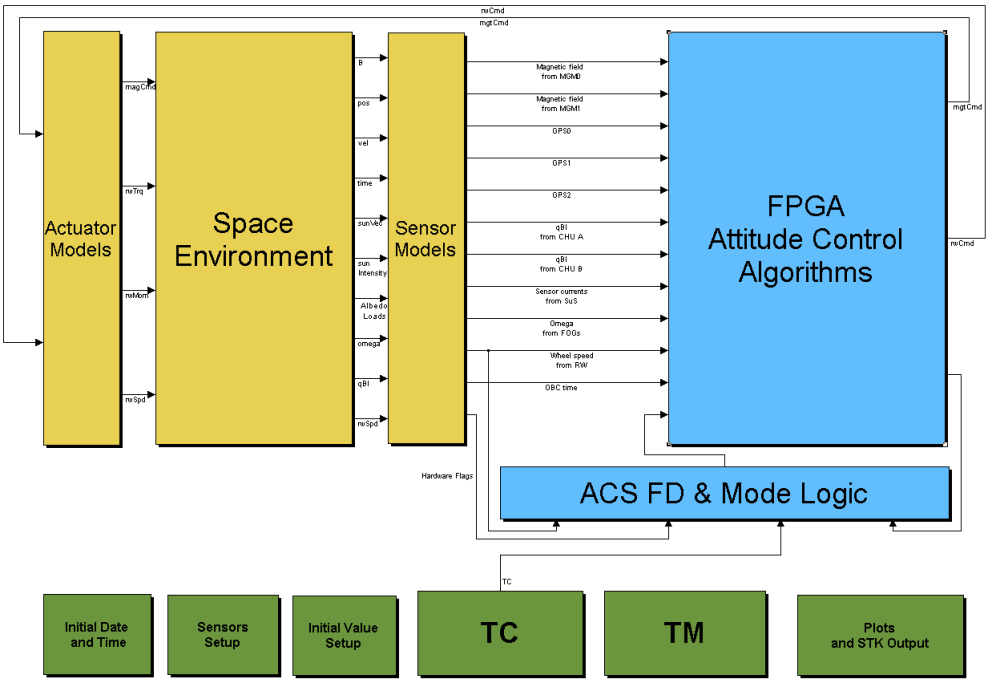


Figure 7.1: Top level diagram of the simulator

- ◇ Sensor models
- ◇ Actuator models
- ◇ User interface
- ◇ ACS algorithms
- ◇ ACS FD and mode logic

7.1 Space environment

This block emulates the space environment for the simulation. The core of this block is the orbit generator which connects all the other blocks. The main blocks in space environment are shown in the Figure 7.2 and are listed below.

1. Epoch clock
2. Orbit propagator
3. Spacecraft dynamics and attitude block
4. Earth's magnetic field model

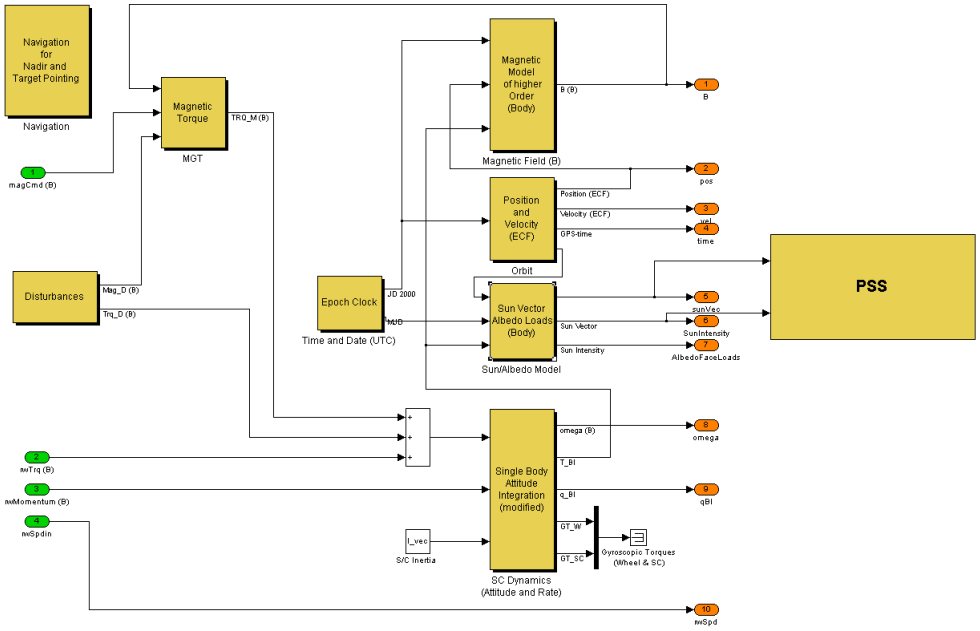


Figure 7.2: Space environment

5. Sun vector and eclipse flag generator
6. Modeling albedo effect
7. Power subsystem (battery) model

7.1.1 Epoch clock

Time and date information is required in several simulation blocks. The magnetic field model only requires the year information while for the sun vector and orbit calculations more accurate information of the time is required. The Julian Date (JD) is converted into Modified Julian Date (MJD) and Julian Date 2000 (JD2000) using following equations.

$$\begin{aligned} JD &= 367(yr) - floor\left[\frac{7}{4}\left(yr + floor\left(\frac{mo+9}{12}\right)\right)\right] + floor\left[275\left(\frac{mo}{9}\right)\right] \\ &\quad +d + 1721013.5 + \frac{1}{24}(\frac{1}{60}(\frac{s}{60} + min) + h) \end{aligned} \tag{7.1}$$

JD2000 and MJD are then obtained by subtraction of the following offsets depending on their reference epochs (1.1.2000, 12:00:00 UT for JD2000 and 17.11.1858, 00:00:00 UT for MJD)

$$JD2000 = JD - 2451545 \tag{7.2}$$

$$MJD = JD - 2400000.5 \tag{7.3}$$

7.1.2 Orbit propagator

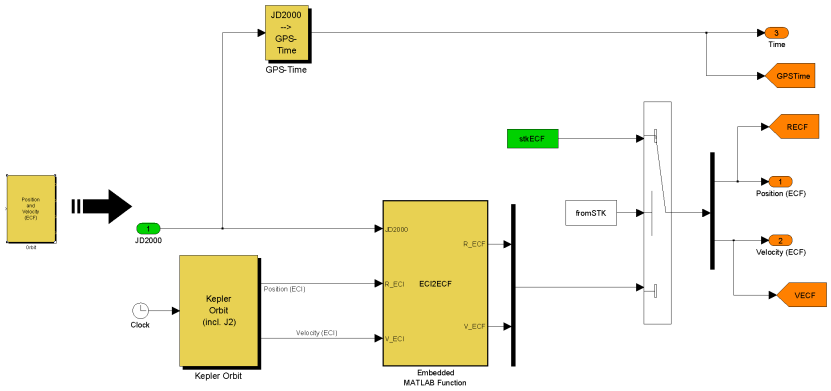


Figure 7.3: Orbit propagator

The block is shown from inside in the Figure 7.3. The core of this block is the `Kepler Orbit (Incl.J2)` as shown in this figure. This is used for the propagation of the orbit and it uses a s-function called `KeplerJ2Orbit_sf` which is developed by EADS Astrium. The output of this block is position and velocity in ECI coordinate frame which is transformed to ECF coordinate frame by the transformation which is carried out in a embedded Matlab function.

7.1.3 Spacecraft dynamics and attitude block

The Simulink implementation of this block is shown in the Figure 7.4. Angular velocity and attitude quaternion are formed as a solution of the kinematic and dynamical equations. Further details about modeling the kinematic and dynamical equation could be found in the Appendix. The inputs to this block are wheel momentum h_w, outer and inner torques on the satellite body and the inertia matrix.

7.1.4 Earth's magnetic field model

The implementation of this block is shown in the Figure 7.5. The magnetic field information model is based on a modified version of the `Magnetic Field of higher Order`[1] block provided by EADS Astrium. Using the position in the ECF frame and the year of simulation, an S-function calculates the components of the magnetic field vector in ECF coordinates. With the help of JD2000 the transformation matrix from ECF to ECI

[1]The magnetic field model uses is an IGRF model of order 10

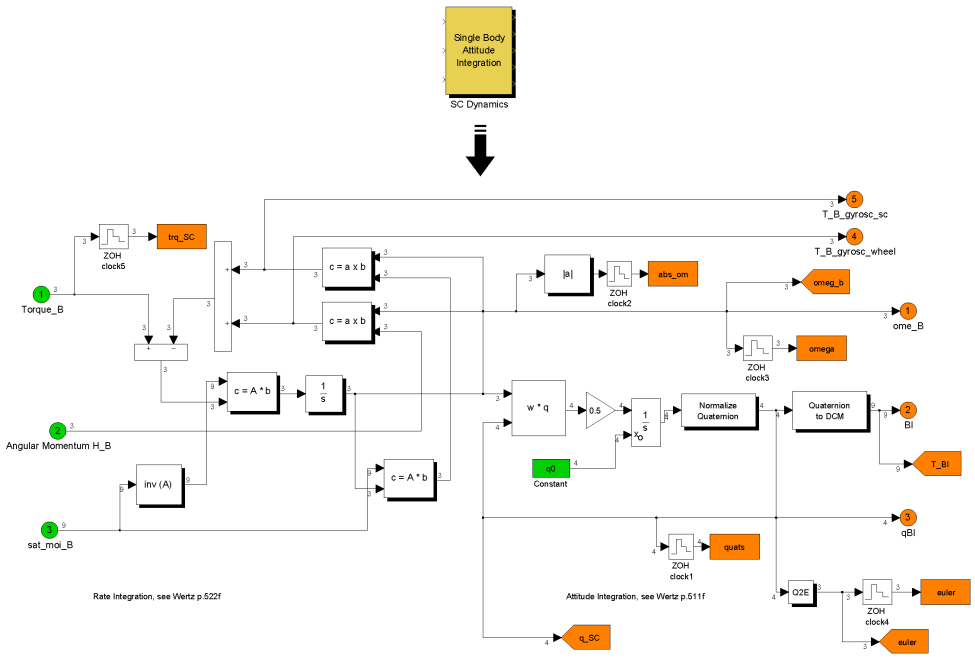


Figure 7.4: S/C dynamics

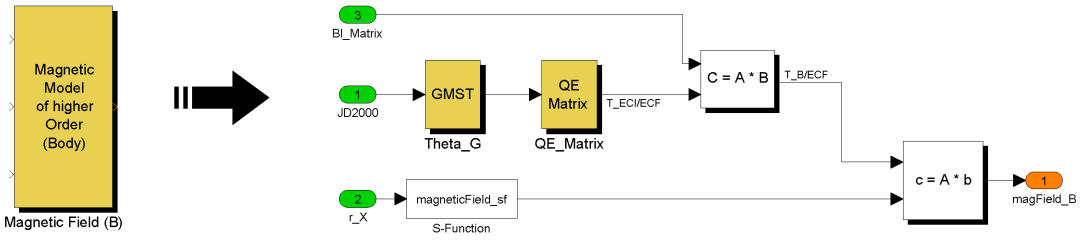


Figure 7.5: Magnetic field model

frame is computed and multiplied by the transformation matrix between body and inertial frame. Finally a description in body coordinates is achieved after another multiplication.

7.1.5 Sun vector and eclipse flag generator

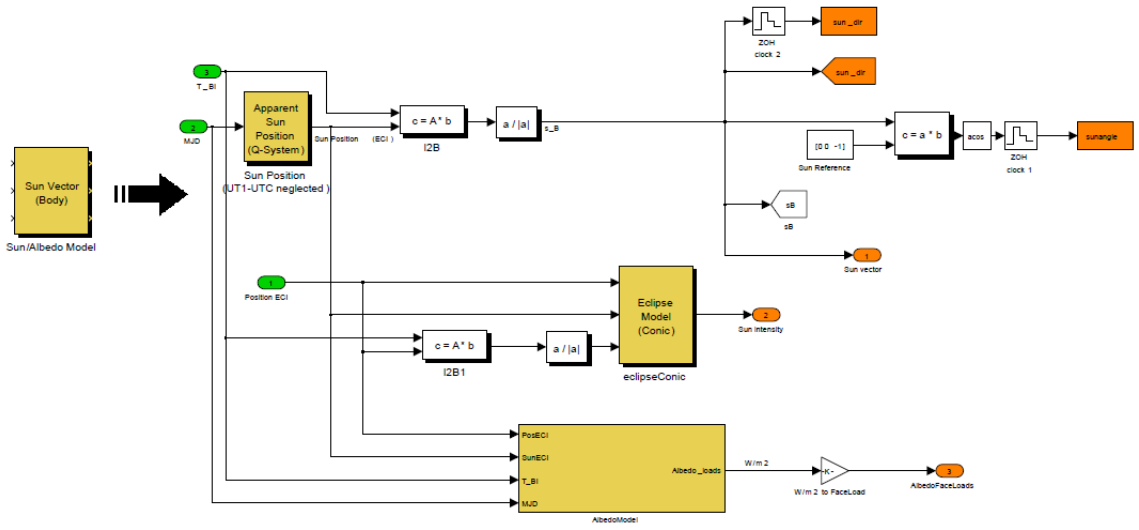


Figure 7.6: Sun vector and eclipse flag generator

The implementation details of this block are shown in the Figure 7.6. The inputs to the block are transformation matrix from ECI frame to body frame, MJD and position of spacecraft in ECI frame. This block contains two main functions, `Apparent Sun Position (Q-System)` and `Eclipse Model (Conic)`. Both of these functions are provided by EADS Astrium. The output of the `Apparent Sun Position (Q-System)` function is the apparent sun position in ECI frame and to obtain the sun position in reference to the spacecraft this output is multiplied with the transformation matrix from inertial to the body frame coordinate system.

7.1.6 Modeling albedo effect

Again the core block to calculate the Earth's albedo effect `AlbedoIR` is provided very generously by EADS Astrium. The implementation details are shown in the Figure 7.7. This block also generates the infrared loads which are not utilized during the simulation. This block is called in the simulation only if the `AlbedoEnable` flag is switched to 1.

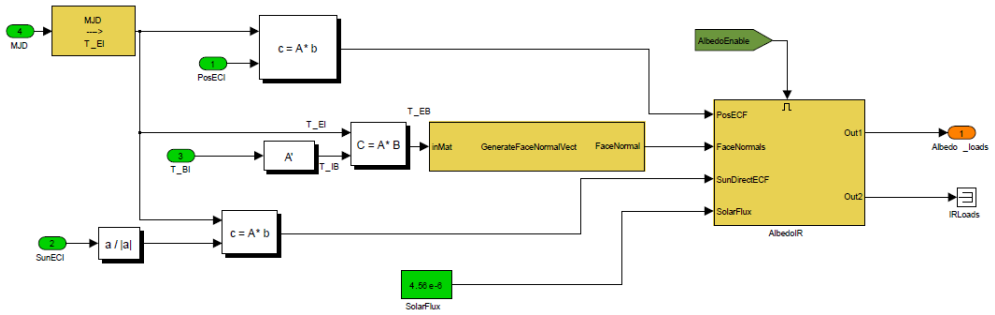


Figure 7.7: Modeling albedo effect

7.1.7 Power subsystem (battery) model

This block consists of three main parts which include `Power Generation`, `Power Consumption`, and `Battery`. The power generation is subjected to the solar panel configuration and the direction in which these solar panels are pointing. The middle solar panels is always available but the two side solar panels are closed during launch and will be opened only once the satellite is in orbit.

The power consumption is realized using an Matlab embedded function. This model does not check hardware on/off status but it uses predefined values of power consumption for each mode and it outputs the value corresponding to the current control mode.

The battery model is also realized with the Matlab embedded function which uses initial state-of-charge, power consumption and power generation as inputs and this function calculates the battery current state-of-charge.

7.2 Sensor models

The output from the space environment is treated as a real behavior. This output is corrupted with the noise and time delays, and is also converted into the respective sensor frame of reference. Finally this information is sampled at different frequencies respectively to model the sensors. To keep this chapter concise the details for every sensor, its noise modeling and transport delay is not presented here but this information is well documented in the technical note [46]. The top level diagram of the sensor model is shown in the Figure 7.9.

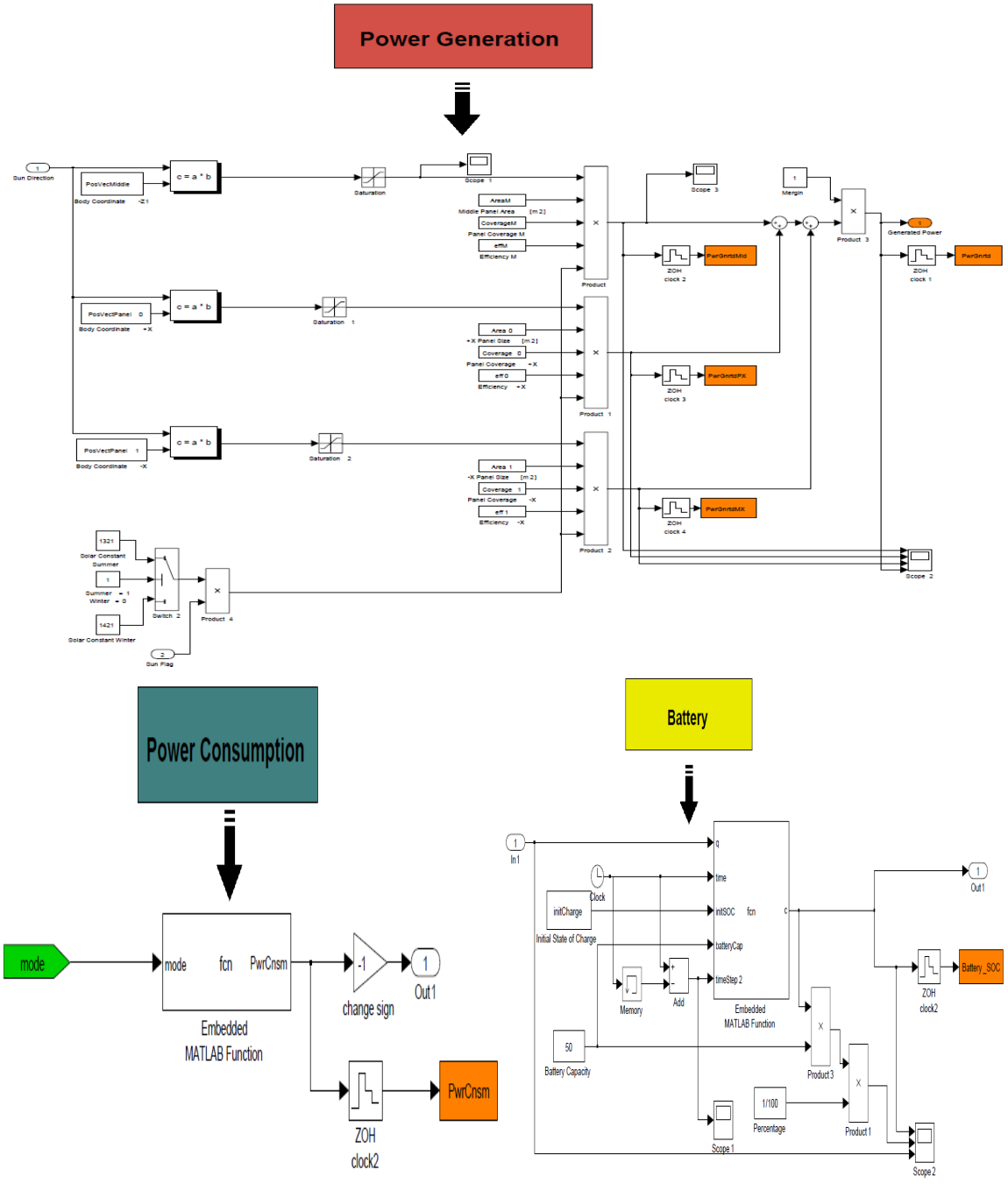


Figure 7.8: Power subsystem

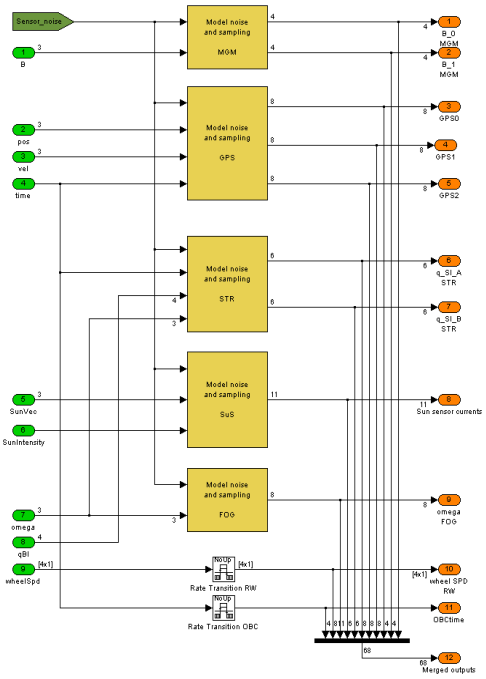


Figure 7.9: Sensor models

7.3 Actuator models

Actuator models consists of model of magnetic torquers and model of reaction wheels. The input to these models are the commands generated by the control algorithms and the output of these blocks is the torque and angular momentum which is fed to the attitude dynamics blocks to close the loop.

The reaction wheel model was provided by Rockwell Collins, however some modification to this model were made. Due to a non-disclosure agreement this model could not be presented here.

The MGTs are operated in a bang-bang configuration. The torque applied on the satellite body is approximated simply by the cross product of commanded magnetic dipole moment and the magnetic field vector. The MGT alignment matrix transforms the signal from the sensor to body coordinates.

7.4 User interface

The blocks of the user interface which is required to configure the simulator are shown in the Figure 7.10. The basic information regarding simulation conditions is provided in

Figure 7.10: User Interface Blocks

first three blocks from the left i.e. `Initial Date`, `Sensors Setup` and `Initial Value Setup`. The expanded view of these blocks is shown in the Figure 7.11.

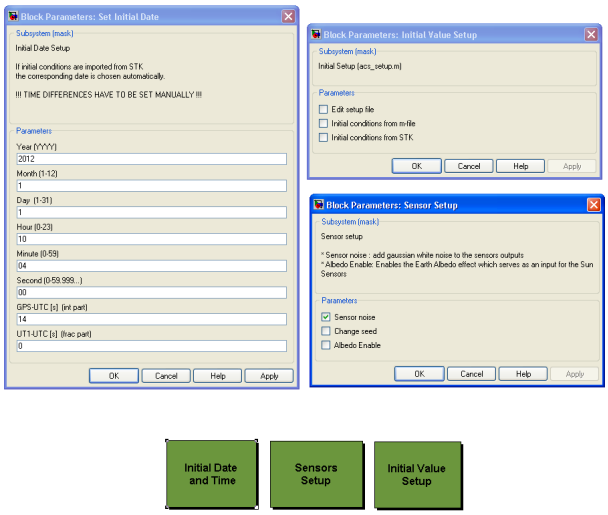


Figure 7.11: Expanded view of the basic configuration blocks

Before running the simulation, all necessary data like initial attitude, position, disturbances, inertia properties or sensor information (alignments, sampling times, noise properties) have to be loaded to the Matlab workspace. This data could be loaded either by directly checking the checkbox of `Initial conditions from m-file` shown in Figure 7.11 or this could be initiated directly from the STK[1]. The Matlab m-file `acs_setup` contains all the initial conditions.

The initial date could be changed from the `Initial Date` block. Its format is common Gregorian date and time (year, month, day, hour, minute, second). The correct date is automatically chosen, if initial conditions from STK are used.

The usage of `Sensors Setup` block is to enable or disable the noise of the sensors. If disabled, the sensors output is still generated at the specified sample times. To avoid having exactly the same noise in every simulation, the noise seed can be changed using the second checkbox. Simulating the space environment with the Earth's albedo effect

[1]Satellite Tool Kit `http://www.stk.com/`

makes the simulation slow due to extensive calculations involved in simulating this effect. Therefore there is provision to avoid these calculation where it is not significant to save the simulation time.

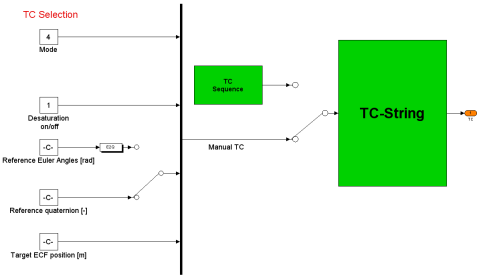


Figure 7.12: Expanded view of the TC block

`TC` block is expanded in the Figure 7.12. The `TC` block is used to select an attitude mode with appropriate reference attitudes or targets at the beginning of the simulation and for mode changes during the simulation (double-clicking the blocks on the left in Figure 7.12 pauses the simulation and allows changes). The switches can be used in the same way. TC-sequence opens the file `acs_sequence.m` and allows scheduled TC commands.

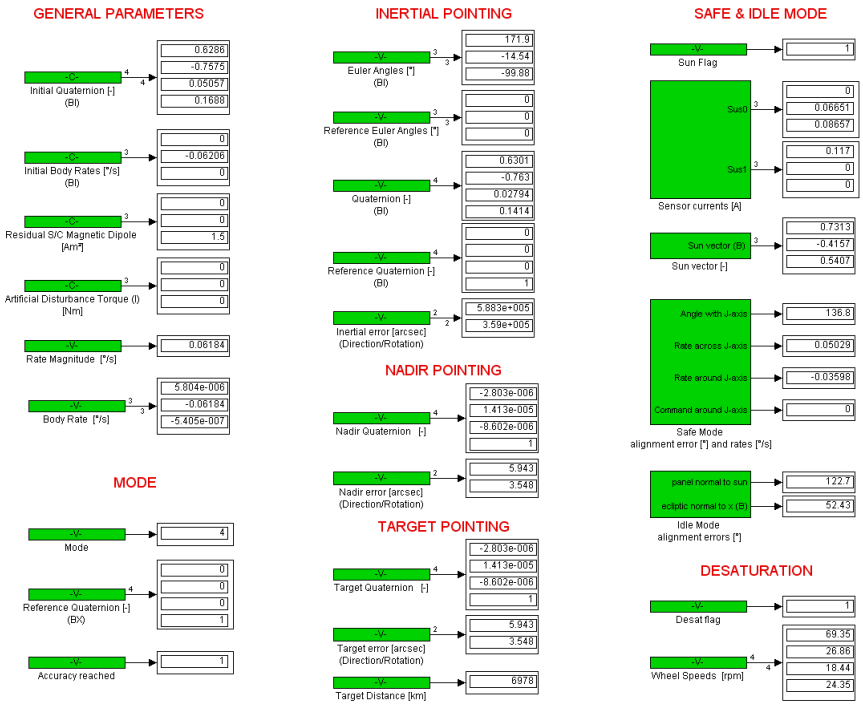


Figure 7.13: Screen shot of the TM block

`TM` block is used to display all the data during the simulation. A screenshot of the `TM` block is presented in the Figure 7.13. Finally the `Standard Plots and STK Output file` block is used to plot the output. STK output file for the visualization could also be generated using this block.

The main blocks of `FPGA Attitude Control Algorithms` and `ACS FD & Mode Logic` are not discussed here as necessary details of these blocks is already presented in the Chapter 5 and Chapter 6 respectively.

CHAPTER 8

FPGA-in-the-Loop Testing Interface

Use of FPGA as an on-board computer leads to the requirement of implementing the attitude control algorithms in the hardware which emphasizes the need of its testing in the loop with the simulator. Extensive testing of ACS algorithms was carried out during the development stage while running the simulation in Matlab/Simulink environment. In a next step the on-board algorithms were ported into Handel-C environment to make them ready to be synthesized on FPGA for testing and verification. At this step it was required to test and compare these embedded algorithms with the previously obtained results. Two different simulation environments are used to conduct the verification/testing. The first simulation environment is based on Matlab/Simulink environment and is already explained in Chapter 7. The second one is a Model-Based Development and Verification Environment (MDVE). A brief introduction of MDVE is presented in the subsequent sections of this chapter.

This chapter presents the details of the testing interface realized to conduct the verification of the embedded algorithms. This chapter also gives a brief introduction of the FPGA qualification boards which are utilized during this testing.

8.1 FPGA hardware used for the testing

Two FPGA-based development boards RC10 and RC240 from Agility were used to conduct this testing. These boards connects very high density FPGA devices with an exten-

sive array of peripherals. These boards provide a plethora of peripheral connectivity to support testing of algorithms in a real-time environment.

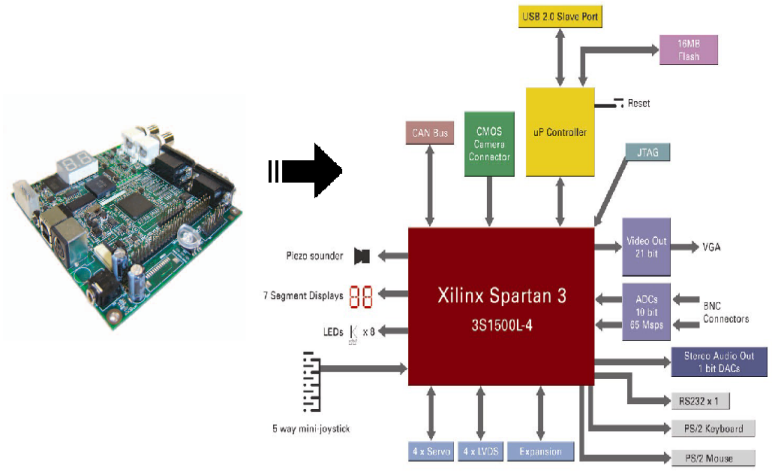


Figure 8.1: RC10 development board

The RC10 board as shown in the Figure 8.1 is an advanced desktop environment specifically designed to be easy and rewarding to use for training, evaluation, rapid prototyping and development based on Handel-C to hardware methodologies. The board is based on Xilinx Spartan-III FPGA and a wide range of I/O to suit a range of potential applications. However the software support available for the board enhances its capabilities beyond the raw hardware, making possible an easy way to test and verify the algorithms and conduct Hardware-in-the-loop testing.

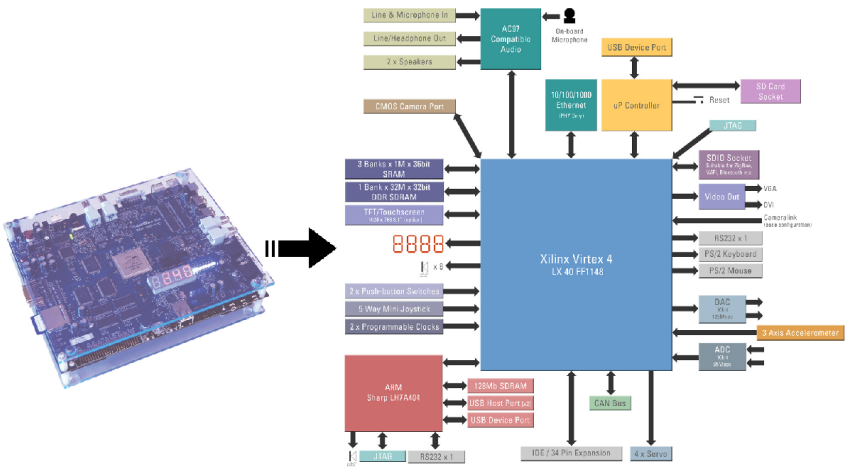


Figure 8.2: RC240 development board

The RC240 board is shown in the Figure 8.2. The RC240 includes a 4M Gate Xilinx Virtex-4 LX40 FPGA with direct access to 12M bytes of pipelined ZBT SRAM and 128M bytes of DDR SDRAM. It is a high performance development and evaluation board. The RC10 board is limited with its hardware resources but RC240 provides sufficient hardware to test all different subsystems together emulating one of the central processing node (CPN) of the OBC.

8.2 Matlab testing interface

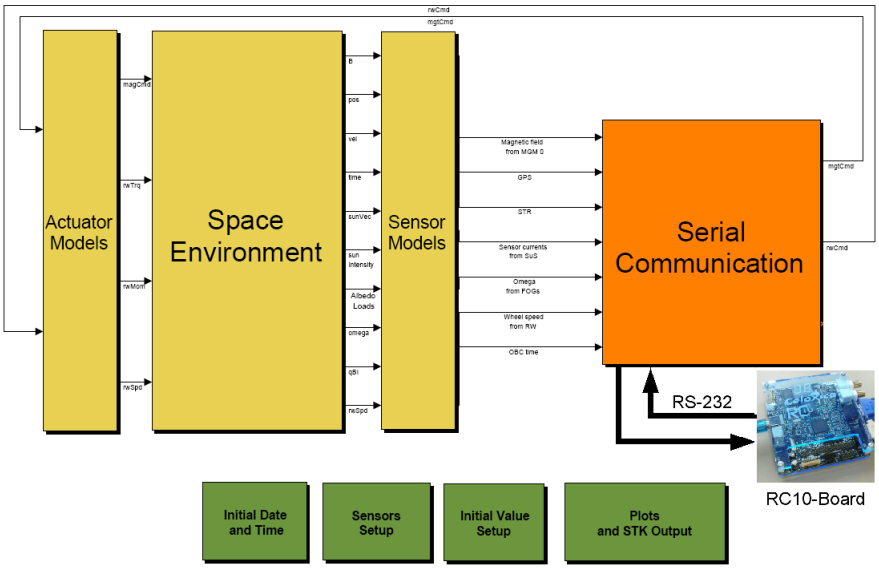


Figure 8.3: Top level diagram of testing algorithms with Simulink

Matlab simulation environment as discussed already in the Chapter 7 is used for conducting this testing. The attitude control algorithms were implemented on RC10 development platform therefore the on-board attitude control algorithm part of the Matlab simulation environment was replaced with the serial communication block as shown in the Figure 8.3. The sensor outputs in the simulation environment are communicated to the RC10 board using RS232 serial communication. The RC10 board uses this information to calculate the respective actuator commands which are then communicated back to the simulation environment through serial port. These blocks are further expanded in the Figure 8.4. The serial communication as shown in this figure is further divided into three functions.

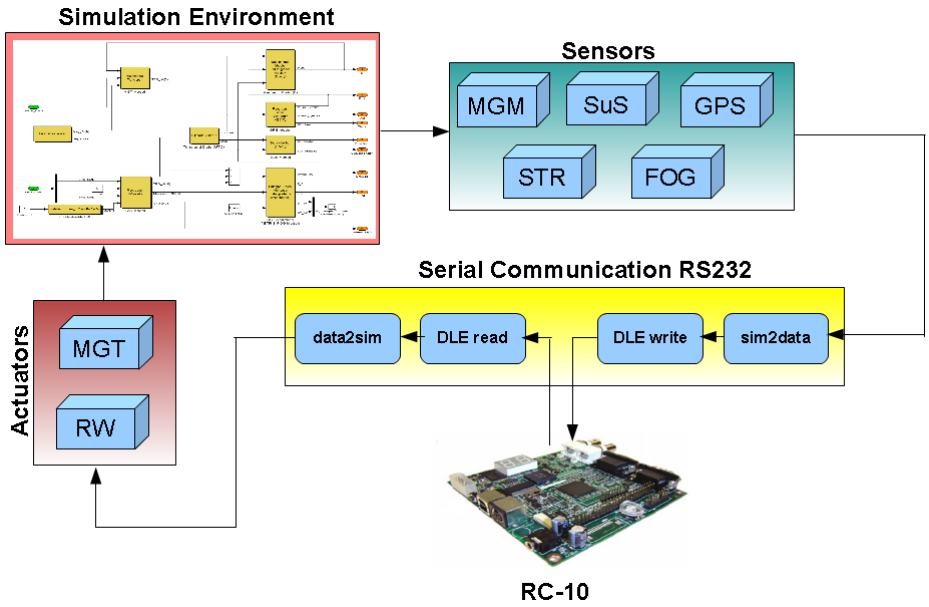


Figure 8.4: Flow diagram of testing algorithm with Matlab

sim2data

The first function is `sim2data` which is implemented in the m-file (`sim2data.m`). The incoming information from the sensor blocks is in Matlab double format which is converted into sensor data format in this block. For example the output of the magnetometer contains the information of the earth's magnetic field in three axis. The information of each axis is scaled and converted into 2 bytes arranged in little endian format such that each bit is representing $5e^{-9}$ Tesla. Further details about each variables, its scaling and its length is provided in [47].

This function additionally arrange the data which has to be communicated via serial port in a data transfer protocol. The data transfer protocol is shown in the Figure 8.5 and explained in the Table 8.1. The packet starts with the standard STX (0x02) byte and next two bytes contains the data length. The next byte specifies the command to the ACS. The command is 1 when data is transferred from simulator to FPGA and it is 2 when data is transmitted from FPGA to simulator. This byte is followed by n-bytes of data, one byte is reserved for the checksum which is followed by the standard ETX (0x03) byte. The length of the data would be constant during flight mode but it is variable and different for each mode depending upon the hardware which is included for that specific mode. The maximum data length is 186 bytes which is explained in detail in [48].

Figure 8.5: Data transfer protocol

Table 8.1: Data transfer protocol

Field	Data size (Bytes)	Type	Description
STX	1	unsigned	Start of the packet
DTL	2	unsigned	Data length
CMD	1	unsigned	Command to the ACS
Data	186 (max)	signed int	Sensor/actuators data
Chksum	1	unsigned	Check sum byte
ETX	1	unsigned	End of the packet

DLE

The DLE (Data Link Escape) is a protocol consists of DLE-read and DLE-write which are combined together and implemented in the m-file (`dle.m`). This function first collects to data packet which need to be transmitted through serial port and it replaces all the control characters with the predefined characters to avoid misinterpretation of the data. For example the STX and ETX byte need to be avoided during the data packet otherwise it could create confusions so similar bytes in the data field need to be masked with the DLE byte (0x10) followed by a predefined byte. The CR (Carriage Return) byte (0x0D) and DLE byte (0x10) are also required to be masked to avoid confusions. The details are summarized in the Table 8.2.

Table 8.2: Control character replacement in DLE protocol

Control character	Value	Replaced value
STX	0x02	0x10 0x42
CR	0x0D	0x10 0x4D
DLE	0x10	0x10 0x10
ETX	0x03	0x10 0x43

After masking the data in DLE protocol, this data is transmitted through serial port using RS232 standard with the baud rate of 115200. The next step of this function is to

read the data from the serial port which contains the commands for the actuators. The data coming from FPGA is also masked with the DLE protocol so first it is unmasked and then this data is transmitted to the next function of sim2data.

data2sim

This function is implemented in m-file (`data2sim.m`). The input to this function is the data unmasked by `dle.m`. This data contains the commands to the reaction wheel and magnetic torquers along with some TM messages. Each reaction wheel is commanded by a two byte torque command arranged in a little endian format. Each bit in this command represents $2e^{-7}$ Nm torque command.

The commands to magnetic torquer are embedded in six bits contained in one byte. Two bits control each magnetic torquer, the first bit specifies its on/off status and the second bit specifies the direction of the current.

8.3 Model-based development and verification environment

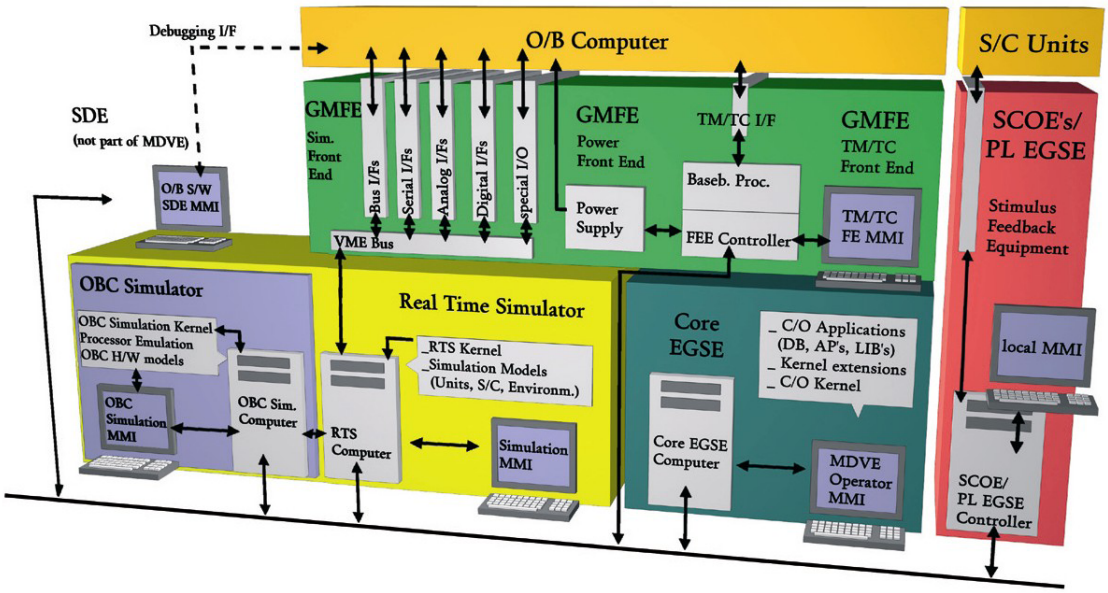


Figure 8.6: Model based development and verification environment

The MDVE simulation environment provides a tool infrastructure allowing Spacecraft

models ranging from early pure virtual simulations via hybrid testbenches up to full FlatSat configurations to be setup. This largely minimizes cost for hardware models, and in parallel provides risk mitigation through stepwise verification of on-board software, on-board hardware, flight procedures etc. first by utilization of simulation testbenches and later by hardware-in-the-loop configurations. A more detailed explanation of MDVE technology and its configurations can be found in [49].

MDVE will be used in various project phases for a number of design, development and verification tasks in different working environments. The most important MDVE standard configurations are:

1. Development SVF (DEV-SVF)
2. Software Verification Facility (SVF)
3. System Testbed (STB)
4. Extended Real-Time Testbed (FlatSat)
5. Spacecraft simulator for the mission control center.

The core element of the MDVE is an On-board Computer Simulator (OBC-Simulator) and a Real-Time Simulator (RTS) modeling the remaining equipment units of the spacecraft plus space environment conditions (Figure 8.6). In dedicated simulation setups both of these simulators (synchronized to each other) are commanded via a control console in most cases a Core EGSE. The functional behavior of the satellite system and all interactions of the on-board equipments are modeled on the basis of the system and equipment specifications.

A major benefit of model-based system development utilizing MDVE is the possibility for early simulated satellite mission operations, using real on-board software in the virtual satellite. By the use of the model based concept each flight hardware unit can be realized by an equivalent software model prior to the availability of the real hardware. This represents an outstanding support to system design qualification and performance verification.

8.4 MDVE testing interface

RC240 board was used to carry out testing with MDVE. The algorithms were implemented on RC240 board and simulator of MDVE was used to transmit the simulated sensor signals

and receive actuator commands through Simulator Interface Connection (SIC). Some more information could also be consulted in [50; 51].

The goal of the SIC is to direct all satellite hardware interfaces of the on-board computer FPGA board through one interface to the Real-time Simulator (RTS). Figure 8.7 roughly illustrates this principle. The Simulator Front-End (SIM_FE) as part of the on-

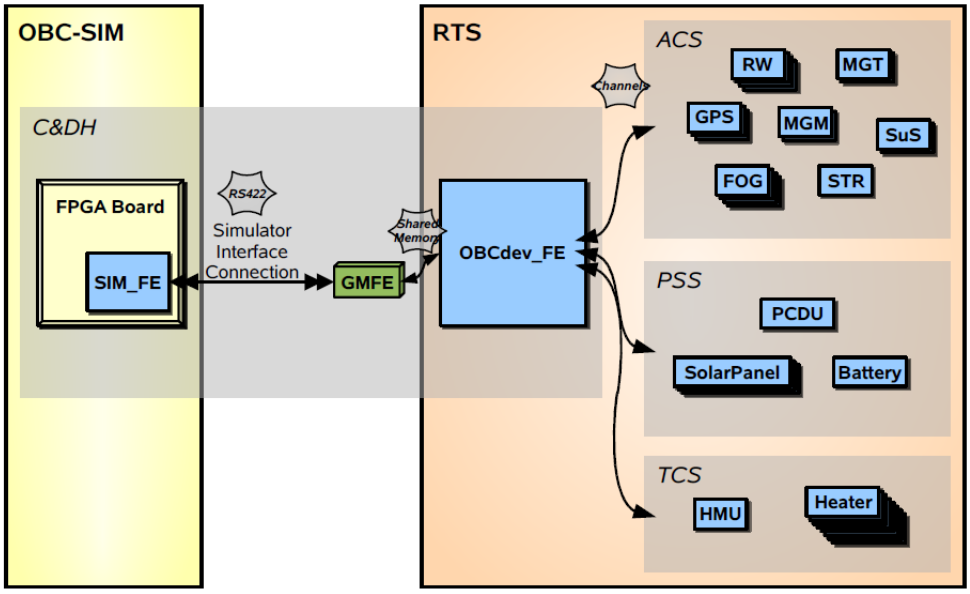


Figure 8.7: Simulator Interface with OBC

board computer FPGA-Board provides communication functionality between the OBC-Simulator (OBC-SIM) and the Real-time Simulator. After external data reception it provides these data to the dedicated internal control algorithms and wraps all new packages to send into Simulator Transfer Frames (STF) for immediate transmission to the Real-time Simulator.

The Development On-board Computer Front-end (OBCdev_FE) in general is a standardized Equipment Model and is scheduled by the scheduling table principle. Via an Generic Modular Front-end (GMFE) interface card and its specially high level driver it provides communication functionality with the Simulator Front-end of the FPGA-Board, which is part of the OBC-Simulator. Basically it transmits and receives Simulator Transfer Frames from the on-board computer FPGA board. The Simulator Transfer Frames are created or interpreted and forwarded to the corresponding Equipment Unit of the Real-time Simulator.

The simulator Interface Connection (SIC) was initially a RS232 connection which was upgraded to RS422 making it possible to take advantage of the maximum port speed of

921600 baud. In this way the communication take place with the SIM_FE instead of satellite hardware peripherals. The SIM_FE encapsulates the outgoing communication into a standardized simulator transfer frame and passes it to the PSL (Platform support library for RC240) which then sends it via the RS422 interface to the simulator. The structure of the data frame is almost similar to the one used for the Matlab simulation and similarly it is also masked with the DLE protocol.

CHAPTER 9

Results and Conclusions

9.1 Simulation results

The simulations are conducted for Orbit-01 (please for orbit definitions see Table 2.4 and Table 2.5). The satellite inertia was approximated from the CAD model as

$$I = \begin{bmatrix} 7.066197 & 0.471470 & 0.129597 \\ 0.471470 & 6.950219 & -0.209866 \\ 0.129597 & -0.209866 & 8.555828 \end{bmatrix} \tag{9.1}$$

To keep this chapter concise and demonstrate the functioning of ACS within the desired limits, only a selection out of wide variety of simulation results is presented. For further details and more simulation results please refer to [52], [53], and [54].

9.1.1 Detumble Mode

In order to verify that the detumble control law is able to reduce the over all kinetic energy of the satellite, a worst case tumble scenario is presented where the satellite initially has body rates $[\omega_x \quad \omega_y \quad \omega_z] = [8 \quad -7 \quad 1]^\circ/s$, the norm is 10.67 °/s. Most of the angular velocity is placed on the body x-axis and y-axis as most of the separation mechanisms are likely to induce angular velocities around these axes. The simulation results presented in the Figure 9.1 shows that the the satellite took a little more than one orbit to bring

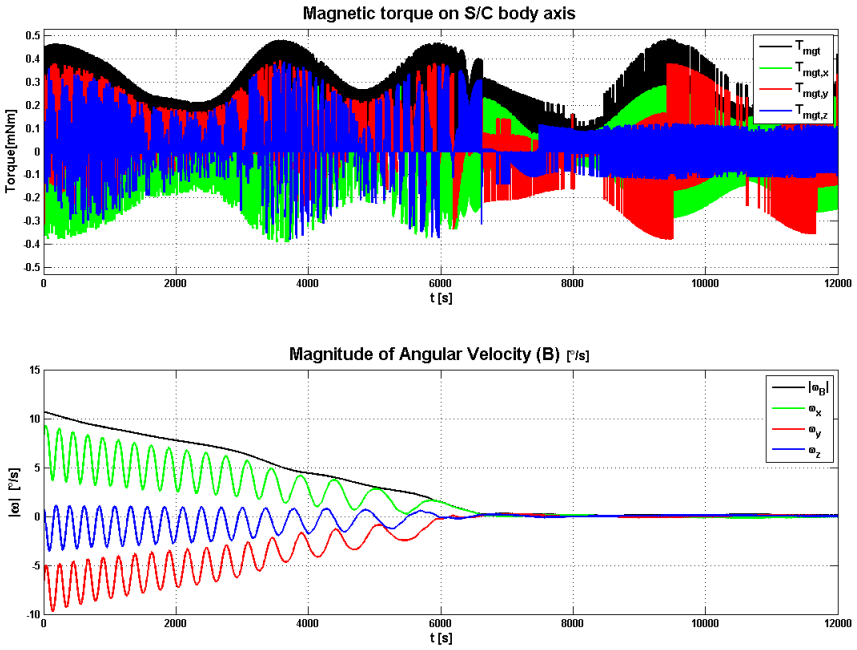


Figure 9.1: Detumble Mode

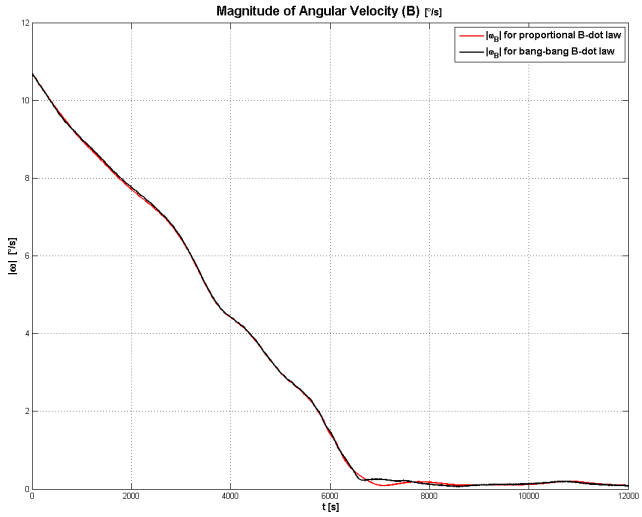


Figure 9.2: Comparison b/w proportional B-dot controller and bang-bang B-dot controller

the angular velocity from 10.67 °/s to under 1 °/s. Figure 9.2 shows the difference in the performance of proportional B-dot controller and bang-bang B-dot controller.

Detumble mode was also simulated with the FPGA-in-the-loop after implementing the detumble control algorithms in the RC10 FPGA evaluation board. After implementation first it was tested with the Matlab/Simulink simulator and the results are presented in the Figure 9.18, after this the same algorithms were tested also with the MDVE simulator and the results are presented in the Figure 9.19.

The results with the Matlab/Simulink simulator are almost identical to the one presented in Figure 9.1, the only difference is the use of fixed point instead of a floating point. However the results with the MDVE simulator are a bit different due to the use of different simulator. The main difference in this simulation is because MDVE also uses residual magnetic torques as a disturbance and such a disturbance is not modeled in Matlab/Simulink simulator.

9.1.2 Safe Mode

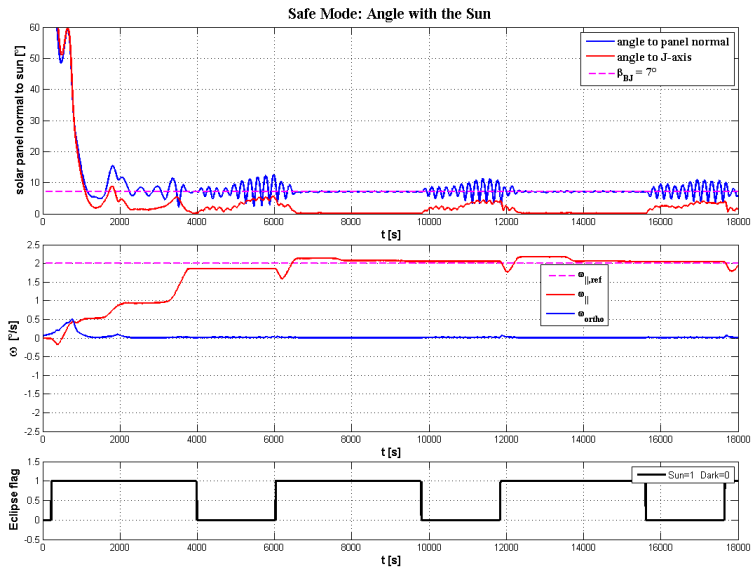


Figure 9.3: Safe Mode without the Earth's Albedo Effect

The satellite is placed in safe mode automatically after the detumble mode when the total body rates decreases below 0.5 °/s. The satellite is put in the safe mode during the sun phase of the orbit. The simulations are conducted with and with the Earth's albedo effect. The results are shown in the Figure 9.3 and Figure 9.4.

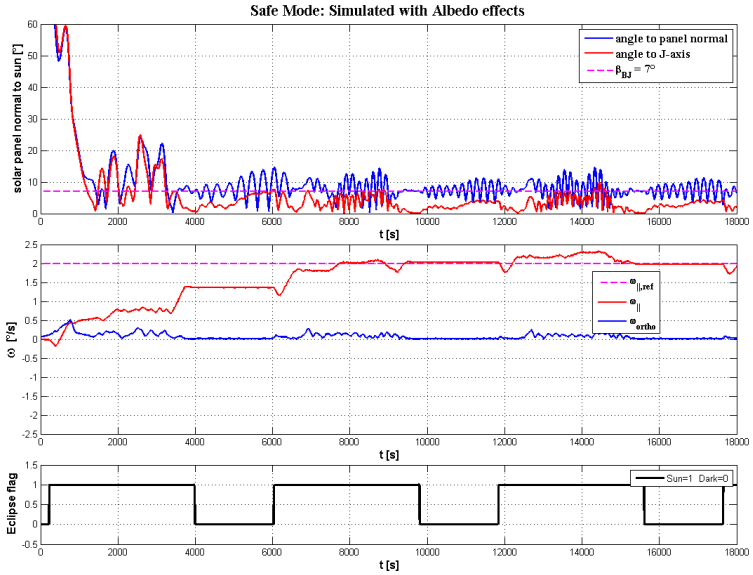


Figure 9.4: Safe Mode with the Earth's Albedo Effect

The results with the included albedo effect shows that albedo effect is minimum just before satellite is going into eclipse phase or coming out of the eclipse phase and this is because of the solar reflection angle which is not affecting the satellite. The results without albedo effect shows that the satellite acquires the desired attitude in less than 1500 s and it takes about one complete orbit to spin-up the satellite but with the inclusion of albedo's effect the time to spin-up the satellite increases to a little more than one orbit.

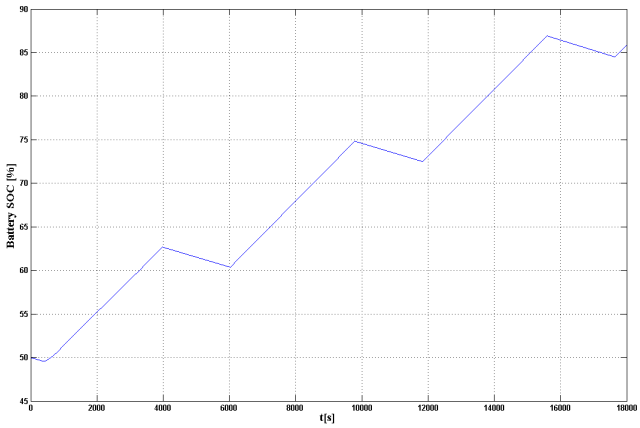


Figure 9.5: Safe mode: Battery SOC

In Figure 9.5 the battery State of Charge (SOC) is plotted against time which is an important indicator of the safe mode performance. The most important requirement of the safe mode is to keep the satellite alive by keeping the SOC's value increasing.

9.1.3 Idle Mode

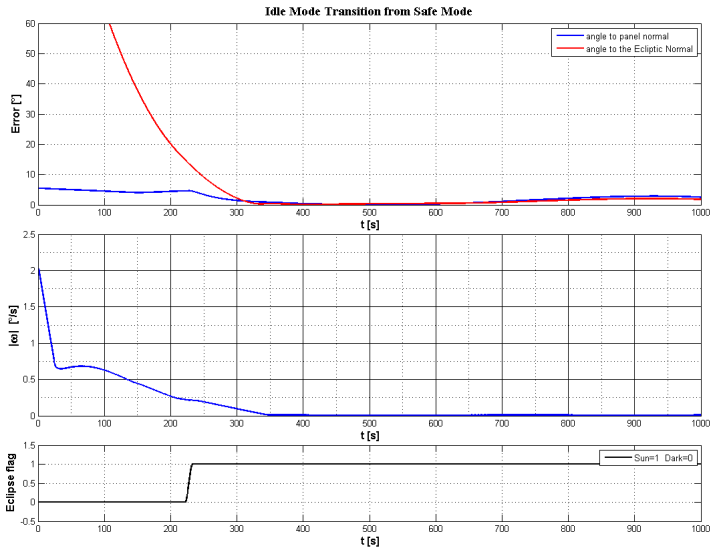
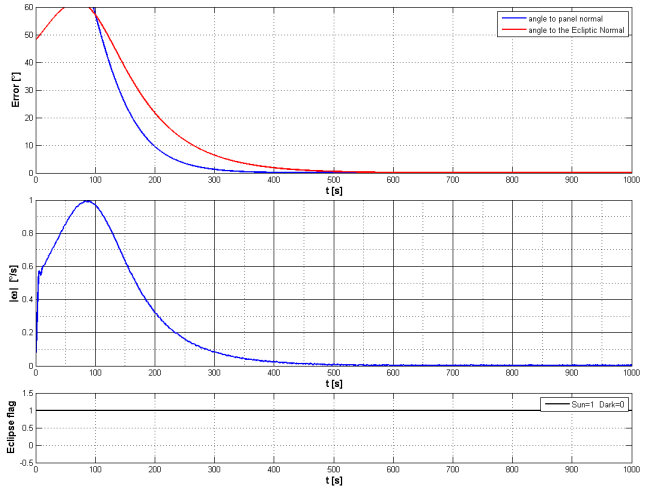


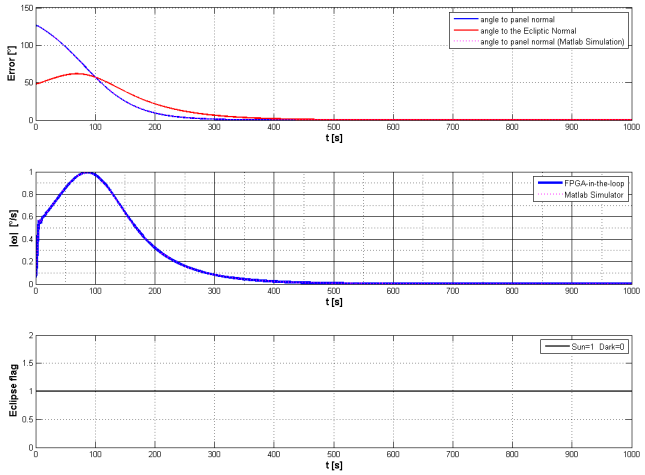
Figure 9.6: Idle mode (Transition from Safe mode)

The plot in Figure 9.6 shows the simulation results of idle mode which is initiated just after the safe mode. The initial turning rate of the body is around 2 °/s which is due to the spin induced in the safe mode. The satellite's solar panel is facing toward the Sun at an angle of about 7° which is the angle between satellite's major principal axis and solar panel normal. As the transition was made during the eclipse so there was no valid information of the sun vector available, therefore satellite started acquiring desired attitude after the start of the sun phase and it took about 300 s to reduce the body rates which were increased due to the spinning induced by the safe mode.

In Figure 9.7, the transition from the inertial mode is shown which is started with almost zero body rates and from a fixed inertial attitude. These results are also compared with the results obtained during FPGA-in-the-loop simulation. The results are very much identical and there is a very slight difference due to fixed-point conversion in the FPGA implementation.



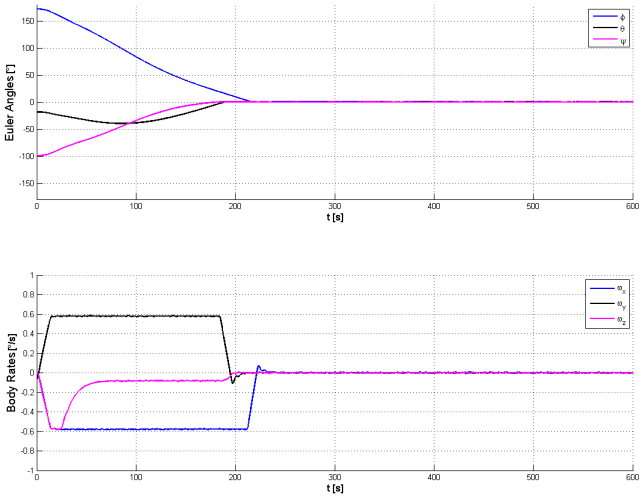
(a) Matlab/Simulink Simulation



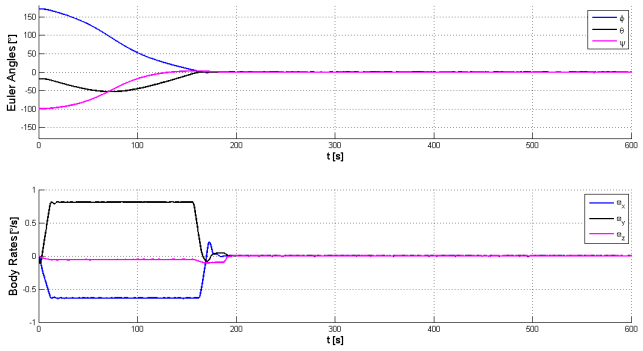
(b) FPGA-in-the-loop simulation

Figure 9.7: Idle mode (Transition from Inertial Pointing Mode)

9.1.4 Inertial Pointing Mode



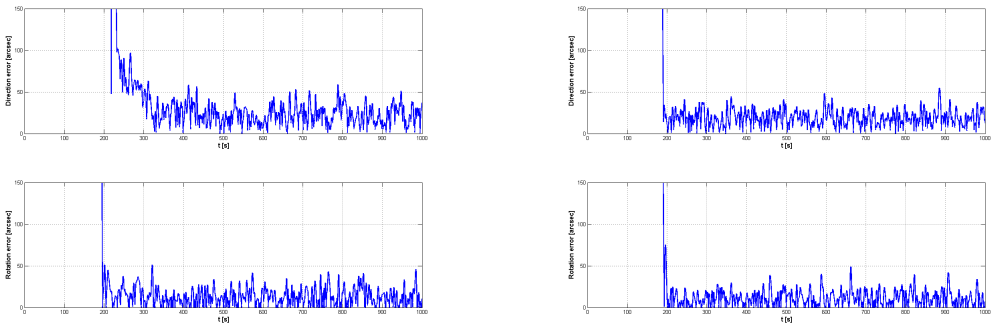
(a) Matlab/Simulink Simulation



(b) FPGA-in-the-loop simulation

Figure 9.8: Inertial Pointing Mode: Euler Angles and Body Rates

The simulation results of the inertial pointing mode are shown in the Figure 9.8 Figure 9.9 and Figure 9.10. For better demonstration a slew maneuver is conducted. The slew maneuver starts from inertially fixed attitude with the initial euler angles of 172°, -18.6° and -99.2° respectively. The maximum allowed body rate is 1 °/s. The slew maneuver takes around 200 s to bring the satellite within the accuracy bounds. This algorithms were implemented also on FPGA and results from the FPGA-in-the-loop simulations were also obtained. Both of these results are compared in the Figure 9.8, Figure 9.9 and Figure 9.10. The results of the FPGA-in-the-loop simulation are similar to the Matlab



(a) Matlab/Simulink Simulation (b) FPGA-in-the-loop simulation

Figure 9.9: Inertial Pointing Mode: Pointing Error

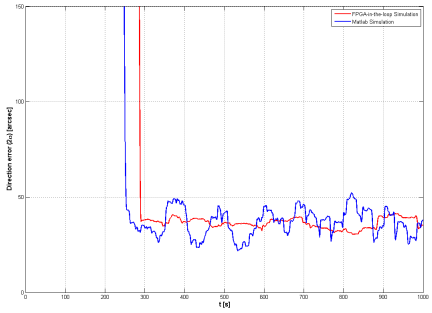


Figure 9.10: Inertial Pointing Mode: Pointing error (2σ)

simulation. FPGA-in-the-loop simulation is taking a bit more time to acquire the required attitude but in terms of accuracy it achieved similar results.

9.1.5 Nadir Pointing Mode

The simulation results of the nadir pointing mode are shown in the Figure 9.11 and Figure 9.12. The pointing error is contained within the bounds of 150″ with the peak 2σ value of 61″ as shown in the Figure 9.13. The slew starts from a close nadir orientation and in 170 s satellite acquires nadir direction. Also it can be seen from the results that the pitching angular velocity ω_y is constant at around -0.0062°/s which corresponds to the orbital rate for a 600 km orbit.

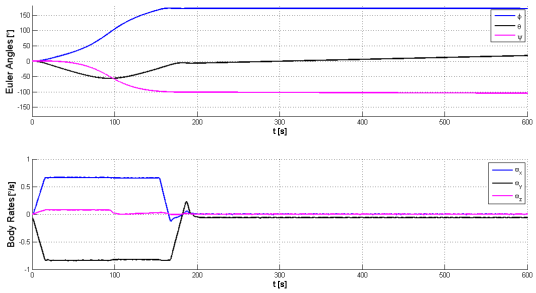


Figure 9.11: Nadir pointing Mode (Euler Angles and Body Rates)

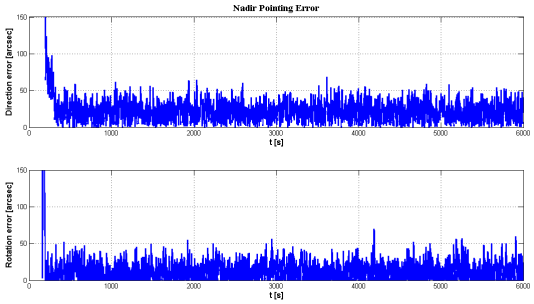


Figure 9.12: Nadir pointing Mode (Error)

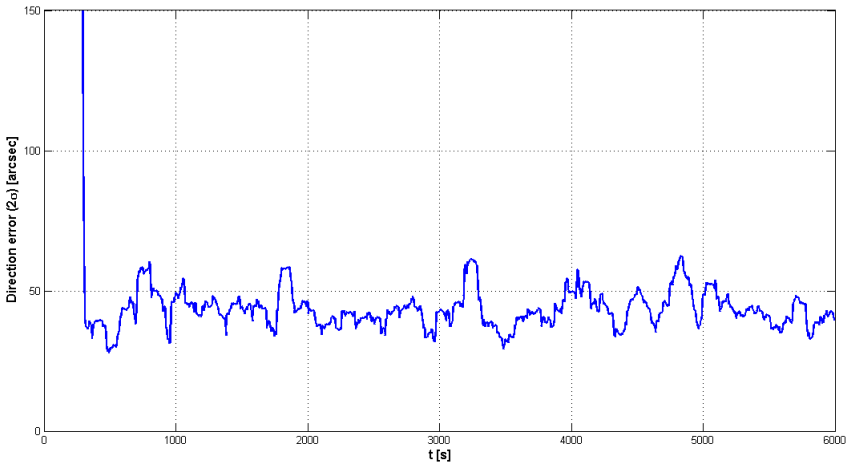


Figure 9.13: Nadir pointing mode: Pointing error (2σ)

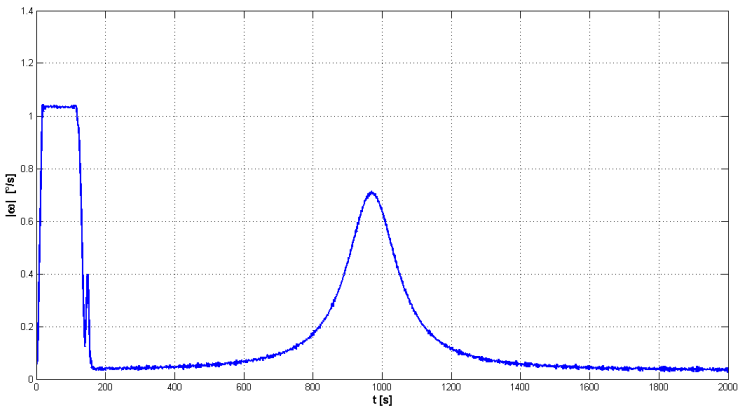


Figure 9.14: Absolute angular velocity during Target pointing slew maneuver

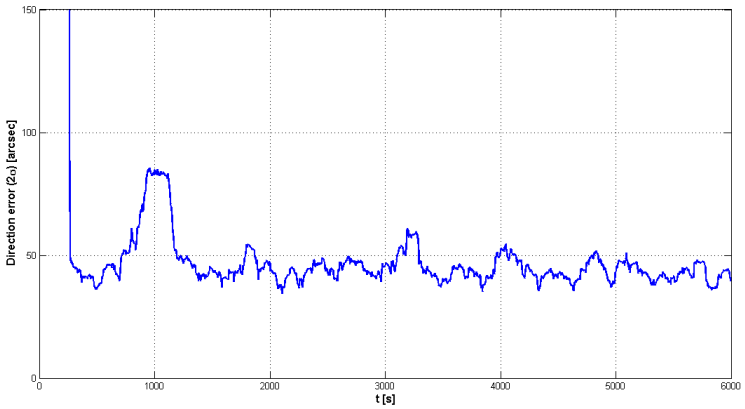


Figure 9.15: Target pointing mode: Pointing error (2σ)

9.1.6 Target Pointing Mode

The simulation results of the target pointing mode are shown in the Figure 9.16 and Figure 9.17. The body rate during the slew maneuver are shown in the Figure 9.14. The peak pointing error 2σ is about $81''$ at the time of peak slew rate.

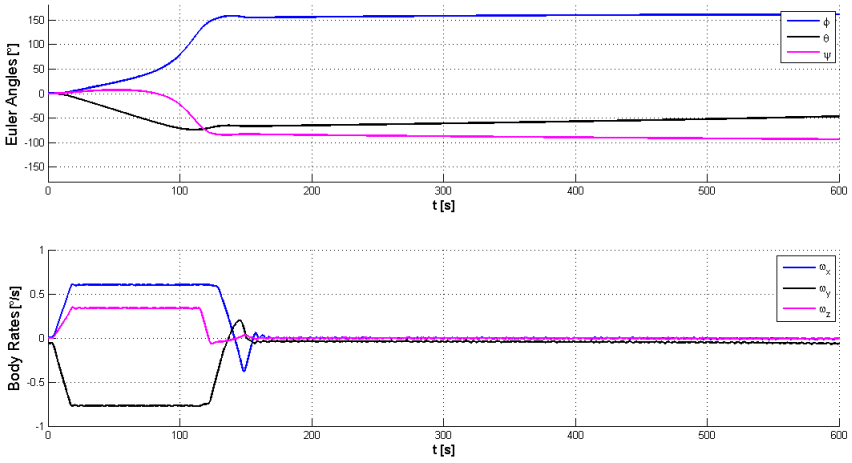


Figure 9.16: Target Pointing Mode (Euler Angles and Body Rates)

9.2 Comparison with MDVE

The results were also compared with MDVE for validation and verification. MDVE is a simulation environment which is developed by entirely different team in an entirely

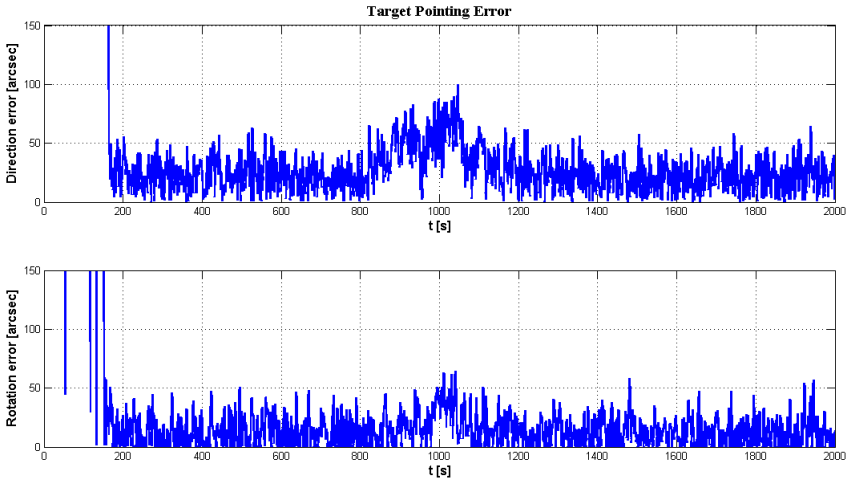


Figure 9.17: Target Pointing Mode (Error)

different environment. MDVE even runs in different operating system. The algorithms used for orbit propagation used in MDVE are also different from what are used in Matlab based simulation environment.

The results which were obtained were similar proving the validity of the algorithms. The comparison of the results is presented here for Detumble mode, Safe mode and for Idle mode. The results of the simulation are compared with the Matlab simulation and are presented in the Figure 9.20, Figure 9.21 and Figure 9.22. One thing which is important to note that in these results the inertia matrix is not the same as it is used in the previous results. Initially in the development phase of ACS the inertia matrix from the BIRD satellite was used for the simulation which is another small satellite of similar configuration. The inertia matrix in the MDVE simulator is not updated at the time of simulation so the same inertia matrix is used in the Matlab simulator to present meaningful comparison. The results obtained by MDVE are similar to the Matlab results but as there are differences in the simulator so both the results are not the same. MDVE also used residual magnetic torques as a disturbance as it is shown in the Figure 9.21 where in the eclipse phase still there are induced torques which acct as a disturbance.

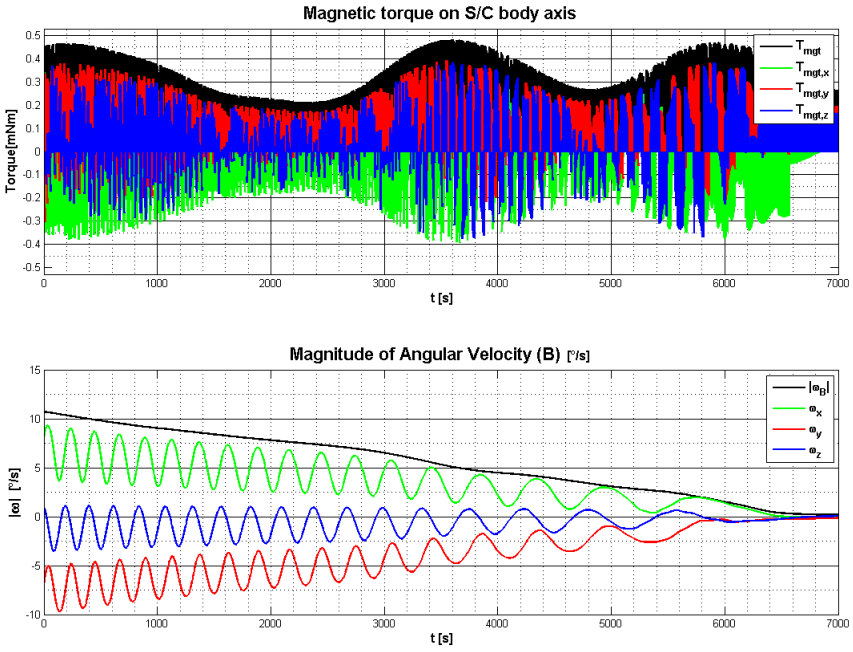


Figure 9.18: Detumble Mode- FPGA-in-the-Loop (Matlab)

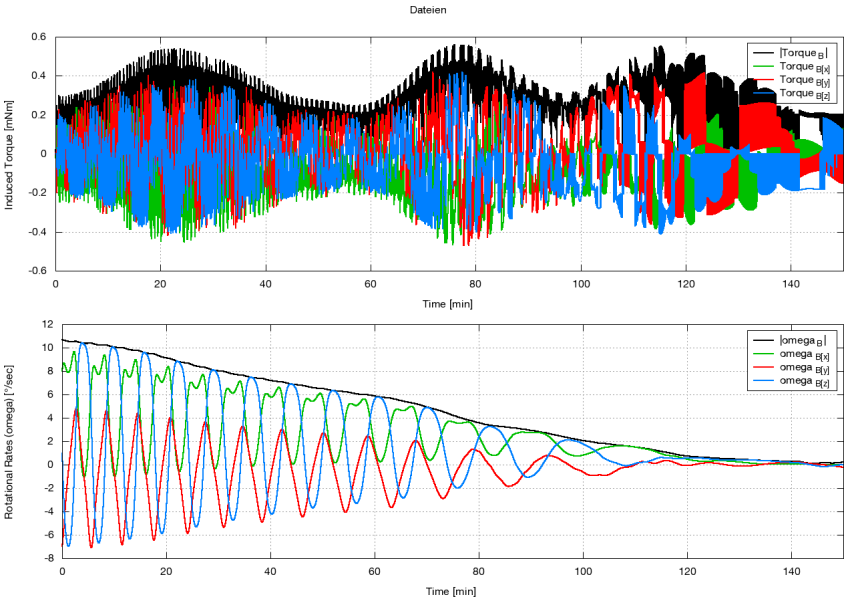
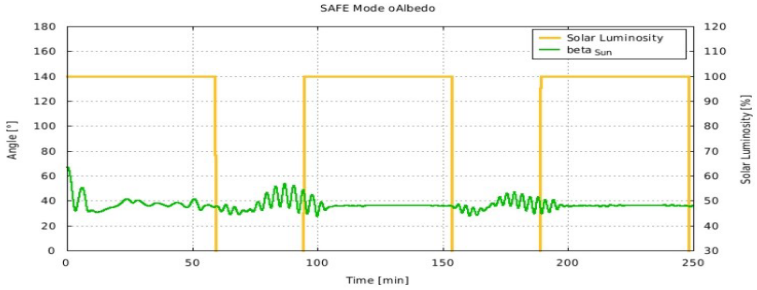
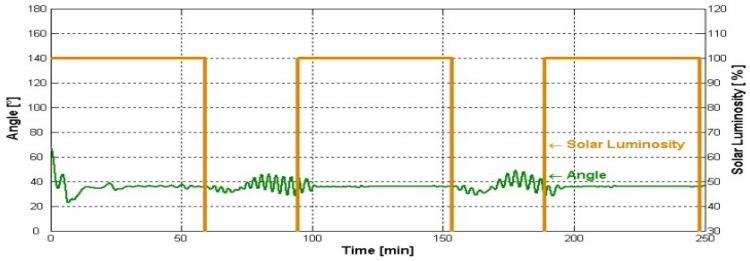


Figure 9.19: Detumble Mode- FPGA-in-the-Loop (MDVE)

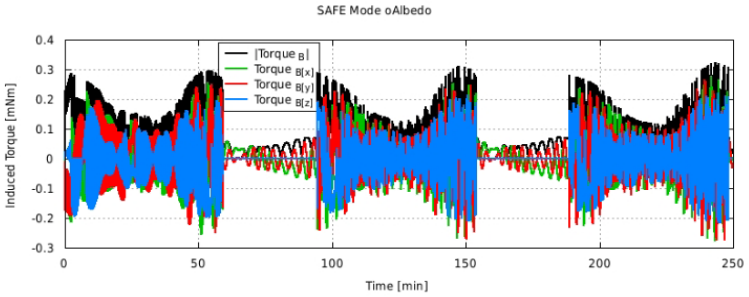


(a) Angle between solar panel normal and sun vector (MDVE)

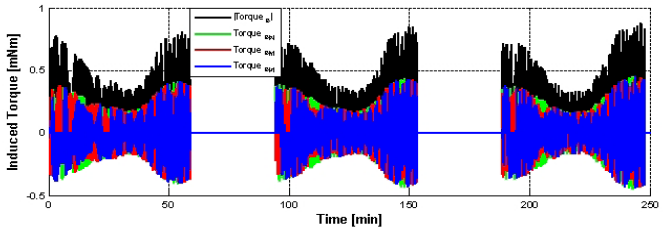


(b) Angle between solar panel normal and sun vector (Matlab Simulator)

Figure 9.20: Safe mode: Beta angle

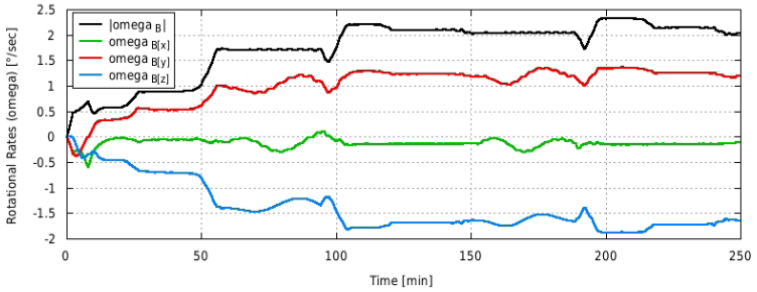


(a) Induced Torques on Satellite (MDVE)

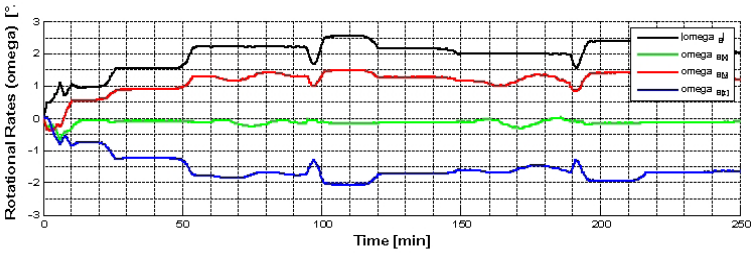


(b) Induced Torques on Satellite (Matlab Simulator)

Figure 9.21: Safe mode: Induced torques



(a) Rotational Rates (MDVE)



(b) Rotational Rates (Matlab Simulator)

Figure 9.22: Safe mode: Body rates

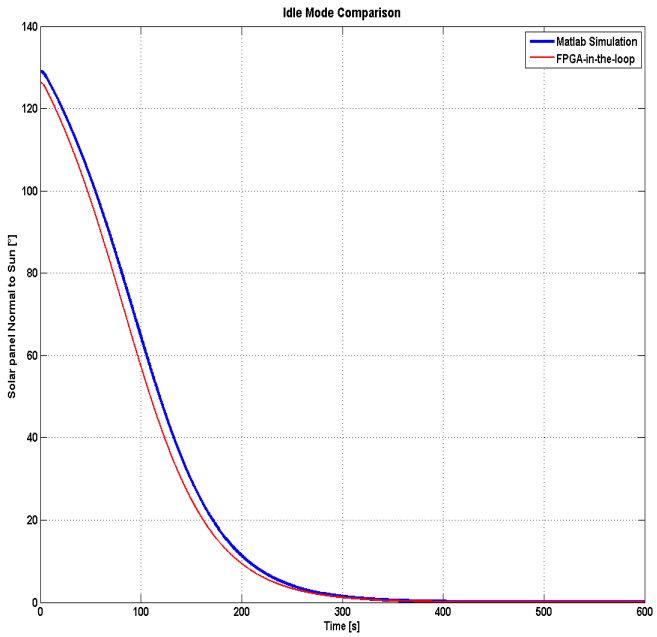


Figure 9.23: Idle mode: Comparison with MDVE

9.3 Conclusion and Outlook

This research presented in this thesis, attempts to lay essential foundations for developing, implementing and testing FPGA specific attitude control system. Apart from developing attitude control system, a new system of spacecraft computation is used for the attitude control algorithms which makes this research to be used as basis for the similar future programs. The main contributions of this thesis are summarized below.

9.3.1 Contributions

Following are the contributions made by this research.

- ◇ The overall structure of ACS algorithms was developed in accordance with the requirements and the constraints. The on-board structure of the ACS algorithms, its hierarchy in a system-level, its command authority based on the inputs from fault detection, system FDIR, and TC/TM was laid down.
- ◇ The ACS algorithms were developed and implemented in Matlab based environment, where these algorithms were evaluated by thorough testing. A clear concept of different modes of the satellite was established based on the various requirements listed in the Chapter 2.

    * The detumble mode was tested against different possible orbit insertion angular velocities. Different algorithms were evaluated in this regard and finally based on simplicity, performance and its portability into the hardware bang-bang B-dot law was used for reducing the kinetic energy of the satellite in detumble mode.
    * Due to very strict reliability constraints, it was very difficult to obtain a simple and effective safe mode control law. A number of different algorithms were candidate for the selection but after a detailed study of all these algorithms safe mode was developed by using magnetic torquing along with magnetometers and sun sensors. The battery SOC was monitored during the simulations and it was assured that this algorithms not only provides safety to the payload but also keeps the satellite alive by increasing the battery SOC.
    * The concept of idle mode was developed to give the satellite a platform to generate solar power while it requires power or it is waiting for some commands.

  - The accuracy of the pointing modes was a prime requirement. Pointing accuracy of $150''$ (2σ) was required by all three pointing modes which was achieved during this design. The pointing knowledge is directly obtained from the star tracker which is well below one pixel ($7''$ (2σ))of the multi-spectral payload instrument. All these requirements were not only met but exceeded in this design which was a challenge for a FPGA based attitude control system.

- The algorithms were ported into the Handel-C environment. The floating point design was also converted into fixed point during this. Handel-C was used to create a netlist to target the FPGA hardware. The evaluation board RC10 based on Xilinx Spartan-III was used at this point to qualify the algorithms. Extensive amount of optimization was required at this stage to contain the ACS algorithms within the bounds of limited hardware. This optimization work could serve as a basis for the similar future projects.
- A mathematical library containing several key mathematical functions like cordic algorithms and quaternion algebra was also created in Handel-C which is not only helpful for the implementation of ACS algorithms but it is also helpful for other subsystem as well as other similar projects of this institute.
- Already existing Matlab/Simulink based satellite simulator was improved by adding some important features into that such as effect of the Earth's albedo, eclipse flag generator and simulation of the power supply system. The effect of the Earth's albedo helped in understanding the safe mode and its better implementation.
- Testing interface with the Matlab simulator was developed and FPGA-in-the-loop testing was conducted to verify the working and to qualify the algorithms. In this testing the simulator was behaving as satellite generating space environment which is the input for the sensor models which then communicate with the FPGA-based OBC to control them through actuators which were also modeled in the Matlab/Simulink environment. Many modifications in the algorithms and in the type of the variables used are resulted during this testing. The established simulation environment could further be utilized for the development and verification of the on-board algorithms of the FPGA-based OBC as well as this could be further used in the Hardware-in-the-loop tests by using the satellite hardware components.
- To gain more confidence on the already obtained results, the ACS algorithms were also tested with the MDVE simulator. This testing not only increased the confidence

level but it also provided a basis for the hardware-in-the-loop testing using the real hardware instead of using their models.

- ◇ Most of the ACS hardware was selected/procured already except for the sun sensor. After a detailed study of available sun sensors and their technology, sun sensor was developed in house with the cooperation of Azur Space Solar Power GmbH and EADS Astrium Ottobrunn. The placement of the sun sensor in the satellite was decided after a number of simulations
- ◇ Different conditions of the fault detection and failure were identified regarding ACS which helped in developing a firm concept of FDIR.

In a nutshell this research proves the possibility of using ACS algorithms as an embedded hardware logic to meet the challenging requirements of the accuracy. With the development of ACS algorithms, these are implemented and tested on the hardware. The hardware built mathematical libraries have potential to be used in the future projects as well. The work carried out in the optimization during implementation of the algorithms will open the way and will lay the first stone in the small satellite program carried out at this institute. The testing interfaces will provide a ready infra structure for the hardware-in-the-loop simulations.

9.3.2 Outlook

There are some suggestions for improvements in future for this work.

- ◇ Real sensors could be included in the Hardware-in-the-loop simulations. There are options to include some of the real sensors while simulating with MDVE. Similarly Matlab based environment could also be used to include some of the real sensors in the loop. Inclusion of real sensor will include the real noise in the loop which is always helpful in analyzing the results.
- ◇ Simulations with air bearing table are also recommended to gain more confidence. Detumble mode and Safe mode could be simulated and verified using air bearing table. The use of air bearing table is helpful in dynamical analysis using real actuators.
- ◇ Filtering techniques are avoided in the whole algorithm design due to their computational overloads. Kalman filters are always very helpful in improving accuracy but

the only problem related with Kalman filters is that yet it is not possible to implement a seven state Kalman filter on FPGA. So some research could be conducted in this field to implement a Kalman filter on a limited hardware resources.

- ◇ Pipeline architecture in the implementation of algorithms on the hardware could be very useful and this architecture could be used to free some more space and improve the over all performance.

At the time of writing the FLP is still under development at IRS. The whole satellite system design is converging gradually to the design freeze stage, so small changes during this phase until the whole system reaches design freeze stage are likely to happen. With these changes the overall attitude control system may get refined or extended but the it is expected that this work will provide the underlying basis and will contribute in a highly exciting, ambitious satellite mission.

APPENDIX A

Reference Systems

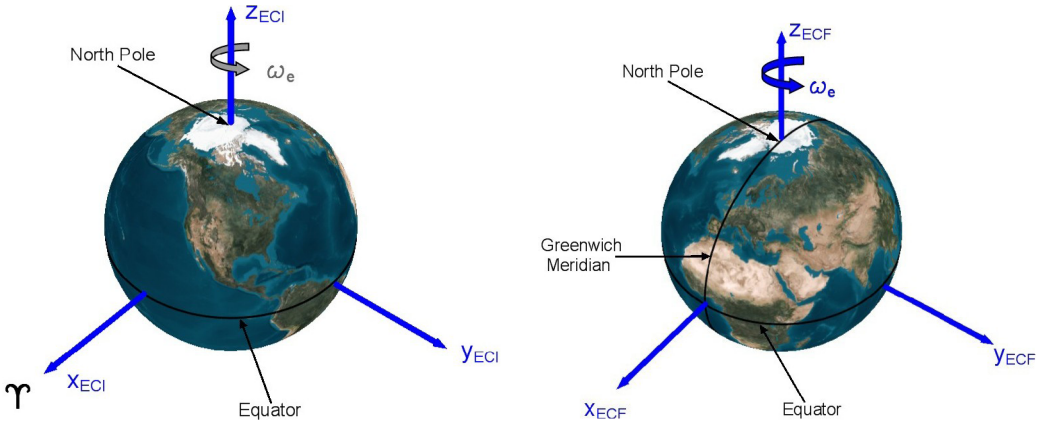
A.1 Reference coordinate systems

Following reference frames are used in this report.

A.1.1 Earth centered inertial coordinate system

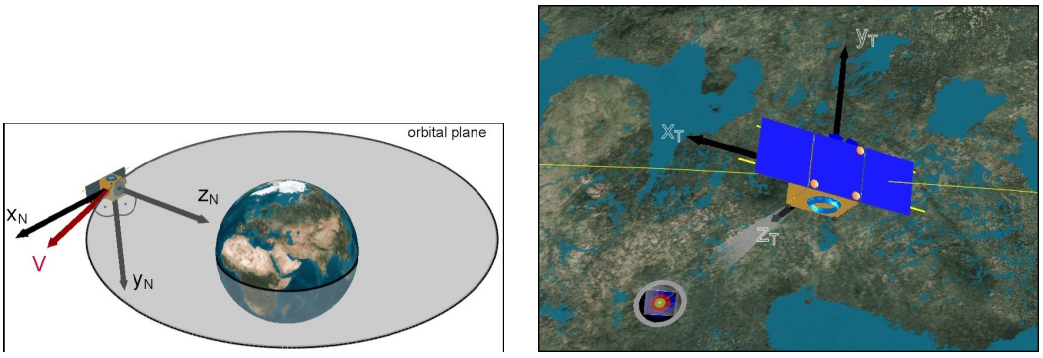
The Earth Centered Inertial (ECI) coordinate system is a right handed, orthogonal system having its origin located at the center of mass of the Earth. The z-axis is aligned with the rotational axis of the Earth and x-axis lies in the equatorial plane and pointed toward the vernal equinox (Υ). The y-axis completes the right handed coordinate system. This reference system is shown in the Figure A.1a.

Since the rotation axis of the Earth is not fixed in inertial space due to nutation, precession, and the polar motion, therefore the coordinate system presented above is not an inertial reference system in real. To make this system fixed in inertial space, time instance is required to be defined. This time instance is 12:00:00 1^{st} January 2000 and orientation of ECI frame particular to this time instance is usually called as J2000 reference system.



(a) ECI coordinate system (b) ECF coordinate system

Figure A.1: Earth's centered coordinate systems



(a) Nadir coordinate system (b) Target coordinate system

Figure A.2: Satellite's centered coordinate systems

A.1.2 Earth centered fixed coordinate system

The Earth Centered Fixed (ECF) coordinate system is another coordinate system whose origin is at center of mass of the Earth. The z-axis is identical to the rotational axis of the Earth, while x-axis points toward the intersection of Greenwich meridian with the equator. The y-axis completes the right handed system. This coordinate system is shown in the Figure A.1b.

A.1.3 Nadir coordinate system

The origin of nadir coordinate system lies at the center of mass of the satellite. The z-axis of nadir coordinate system points toward the nadir direction which is the center of the mass of the Earth. The y-axis is perpendicular to the orbital plane and is opposite to the orbit normal direction as shown in the Figure A.2a. The x-axis completes the right hand system.

A.1.4 Target coordinate system

The origin of target coordinate system is located at the center of mass of the satellite. The z-axis points toward the fixed target on the surface of the Earth. The x-axis is kept in the orbital plane in such a way that no yaw is allowed with respect to the orbital plane. The y-axis completes the right handed system. Nadir coordinate system could be one special type of target coordinate system in which the point of interest or the target is located at the center of mass of the Earth instead of having this point on the surface of the Earth. This coordinate system is depicted in the Figure A.2b.

A.1.5 Body fixed coordinate system

This coordinate system is fixed with the body of the FLP. The origin is the center of mass of the satellite. The z-axis points outward from the boresight of the payload cameras. The x-axis points toward the solar panels as shown in the Figure A.3. Y-axis completes the right handed system.

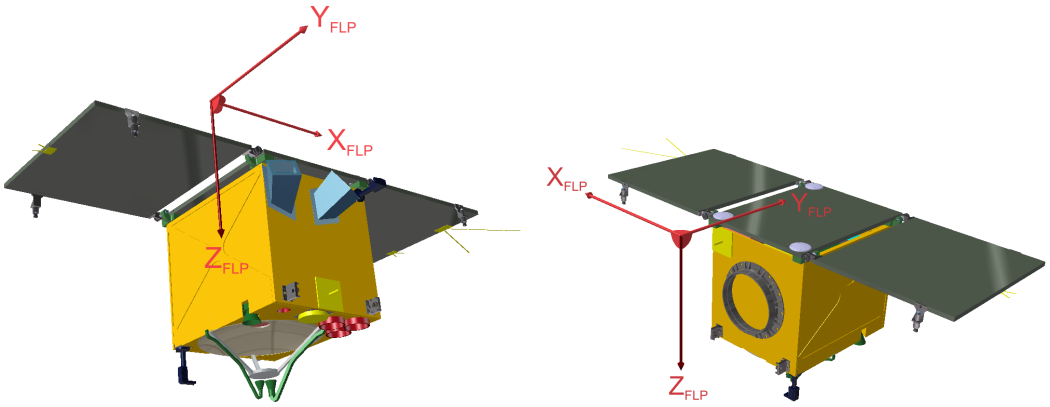


Figure A.3: Satellite's body coordinate systems

A.2 Transformation between ECI and ECF coordinate system

Ideally this transformation could be obtained by a single rotation of ECF around its z-axis but due to the motion of the Earth's rotational axis in the inertial space, this transformation requires more complex computations. This transformation could be defined as

$$T_{FI} = \Theta NP\Pi \tag{A.1}$$

where Θ = rotation about the true pole so that the X-axis of the true equator and equinox frame aligns with the Greenwich meridian.

N = rotation for nutation of the mean equator and equinox of date to the true equator and equinox of date

P = rotation for precession of the mean equator and equinox of J2000 to the mean equator and equinox of date

Π = rotation for polar motion.

The Earths spin axis is not fixed in the Earth but has small oscillations known as polar motion or wobble. Hence the coordinate system fixed on the Earth will have small variations in inertial space due to this motion. To eliminate these variations z-axis is defined along the Conventional Terrestrial Pole (CTP). The CTP is an arbitrary reference and corresponds to the mean position of the true pole between 1900 and 1905. The correction involves rotations through two smaller polar motion angles about the y-axis and x-axis respectively. The effect of the polar motion is not significant and could be avoided for simplifying the problem. This effect was not considered to simplify the calculations.

The equation A.1 could be simplified as

$$T_{FI} = \Theta matNP \tag{A.2}$$

where $matNP = NP$

The Earth's rotation angle Θ is explained in the section 5.4.

A.2.1 Precession matrix

The transformation matrix to eliminate the effects of precession could be defined with three accumulated precession angles ζ_a, θ_a, and z_a defined by Lieske, et al. [55]. These three angles are defined as:

$$\zeta_a = 2306.2181''t + 0.30188''t^2 + 0.017998''t^3 \tag{A.3}$$

$$\theta_a = 2004.3109''t - 0.42665''t^2 - 0.041833''t^3 \tag{A.4}$$

$$z_a = 2306.2181''t + 1.09468''t^2 + 0.018203''t^3 \tag{A.5}$$

where t could be defined as

$$t = \frac{[TT - J2000](days)}{36525} \tag{A.6}$$

TT is the Terrestrial Time which is also known as Terrestrial Dynamical Time (TDT). This epoch corresponds to a Julian date of 2451545.0. Now the precession matrix P could be defined as:

$$\begin{bmatrix} \cos\zeta_a \cos\theta_a \cos z_a - \sin\zeta_a \sin z_a & -\sin\zeta_a \sin z_a & -\sin\theta_a \cos z_a \\ \cos\zeta_a \cos\theta_a \sin z_a + \sin\zeta_a \cos z_a & -\sin\zeta_a \cos\theta_a \sin z_a + \cos\zeta_a \cos z_a & -\sin\theta_a \sin z_a \\ \cos\zeta_a \sin\theta_a & -\sin zeta_a \sin\theta_a & \cos\theta_a \end{bmatrix} \tag{A.7}$$

A.2.2 Nutation matrix

This transformation could also be explained with the help of three angles: the mean obliquity of the ecliptic ϵ_m, the nutation in longitude $\Delta\psi$ and nutation in obliquity $\Delta\epsilon$. To simplify the notation, the first and last of these quantities are added to yield the true obliquity of the ecliptic$\epsilon_t = \epsilon_m + \Delta\epsilon$. Mean obliquity of the ecliptic ϵ_m could be defined as:

$$\epsilon_m = 84381.448'' - 46.8150''t - 0.00059''t^2 + 0.001813''t^3 \tag{A.8}$$

The nutation matrix N could now be defined as:

$$\begin{bmatrix} \cos\Delta\psi & -\cos\epsilon_m \sin\Delta\psi & -\sin\epsilon_m \sin\Delta\psi \\ \cos\epsilon_t \sin\Delta\psi & \cos\epsilon_m \cos\epsilon_t \cos\Delta\psi + \sin\epsilon_m \sin\epsilon_t & \sin\epsilon_m \cos\epsilon_t \cos\Delta\psi - \cos\epsilon_m \sin\epsilon_t \\ \sin\epsilon_t \sin\Delta\psi & \cos\epsilon_m \sin\epsilon_t \cos\Delta\psi - \sin\epsilon_m \cos\epsilon_t & \sin\epsilon_m \sin\epsilon_t \cos\Delta\psi + \cos\epsilon_m \cos\epsilon_t \end{bmatrix} \tag{A.9}$$

APPENDIX B

Fundamentals

B.1 Equation of motion

The motion of a satellite as a rigid body could be divided into translational motion and rotational motion. The translational motion is the motion of center of mass of a satellite and the rotational motion is the motion about center of mass. Position and velocity are the outputs of the translational equations of motion and are required to evaluate the target/nadir coordinate system. The orbit propagator block inside GPS handels this.

The rotational motion could be simplified by considering FLP as a rigid body and with a constant moment of inertia. This motion could be described by the dynamic and kinematic equations of motion.

B.1.1 Dynamic equations of motion

The dynamics of a satellite is driven by the external torques applied to the body. This could be written in the form of equation as:

$$T = T_{dist} + T_{mgt} = \dot{h} + \omega \times h = I\dot{\omega} + \omega \times (I\omega + h_{RW}) + \dot{h}_{RW} \tag{B.1}$$

where h is the angular momentum vector and h_{RW} is the angular momentum of the reaction wheels. I is the time invariant inertia matrix and T is the external torque which is a sum of disturbance torque T_{dist} and torque generated by magnetic torquers T_{mgt}.

B.1.2 Kinematic equations of motion

There are many different ways to write the time dependent relationship between the angular rate and the attitude. Quaternions are used in this work for number of reasons to establish this relationship which could be written as:

$$\dot{\underline{q}} = \frac{1}{2}\Omega(\omega)\underline{q} \tag{B.2}$$

Where $\Omega(\omega)$ is the skew-symmetric matrix and could be written as:

$$\Omega(\omega) = \begin{bmatrix} 0 & \omega_z & -\omega_y & \omega_x \\ -\omega_z & 0 & \omega_x & \omega_y \\ \omega_y & -\omega_x & 0 & \omega_z \\ -\omega_x & -\omega_y & -\omega_z & 0 \end{bmatrix} \tag{B.3}$$

So this could be written as:

$$\begin{bmatrix} \dot{q_1} \\ \dot{q_2} \\ \dot{q_3} \\ \dot{q_4} \end{bmatrix} = \begin{bmatrix} 0 & \omega_z & -\omega_y & \omega_x \\ -\omega_z & 0 & \omega_x & \omega_y \\ \omega_y & -\omega_x & 0 & \omega_z \\ -\omega_x & -\omega_y & -\omega_z & 0 \end{bmatrix} . \begin{bmatrix} q_1 \\ q_2 \\ q_3 \\ q_4 \end{bmatrix} \tag{B.4}$$

B.2 Quaternions

Quaternions are most commonly used is describing attitude while developing on-board attitude control algorithms. A quaternion can be considered as a generalized complex number built as a direct sum of a real number q_4 and a three dimensional vector $\underline{q}_{1-3}$

$$\underline{q} = \underline{q}_{1-3} + q_4 = q_1 i + q_2 j + q_3 k + q_4 \tag{B.1}$$

One of the most important property of the quaternions is that the sum of squares of these four parameters is always unity

$$q_1^2 + q_2^2 + q_3^2 + q_4^2 = 1 \tag{B.2}$$

Due to this singularity could be avoided by using quaternions. Another prime advantage of using quaternions to represent attitude is their linear behavior which provides ease of integration. Some of the properties of quaternions are given below.

B.2.1 Transformation between different coordinate systems

The transformation from coordinate system 1 to coordinate system 2 could be given as $\underline{q}_{21}$ and two consecutive transformations could be represented by a quaternion multiplication written as:

$$\underline{q}_{21} = \underline{q}_{23}\underline{q}_{31} \tag{B.3}$$

B.2.2 Error quaternion

Since quaternions are used in the pointing modes of control for the feedback so to obtain the error quaternion, the reference quaternion $\underline{q}_{ref}$ is multiplied with the attitude quaternion $\underline{q}$ which could be written as:

$$\underline{q}_{Error} = \underline{q}_{ref}\underline{q} \tag{B.4}$$

B.2.3 Convert quaternion to DCM

The direction cosine matrix could be obtained from quaternions as given in the equation below

$$T_{BA} = \begin{bmatrix} 2(q_1^2 + q_2^2) - 1 & 2(q_1q_2 + q_3q_4) & 2(q_1q_3 - q_2q_4) \\ 2(q_1q_2 - q_3q_4) & 2(q_2^2 + q_4^2) - 1 & 2(q_2q_3 + q_1q_4) \\ 2(q_1q_3 + q_2q_4) & 2(q_2q_3 - q_1q_4) & 2(q_3^2 + q_4^2) - 1 \end{bmatrix} \tag{B.5}$$

B.2.4 Convert DCM to quaternion

Figure B.1 shows the flow diagram of conversion of DCM into a quaternion.

B.2.5 Quaternion to Angular Rate

Figure B.2 shows the flow diagram of conversion of quaternions to angular rates.

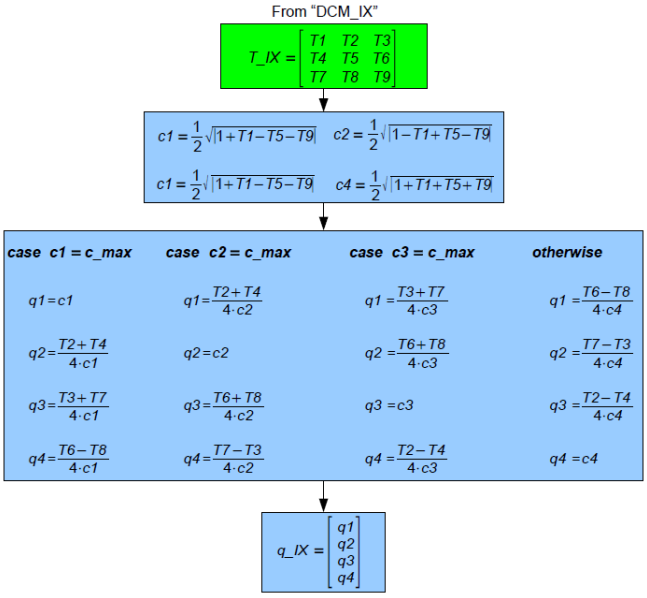


Figure B.1: DCM to Quaternion

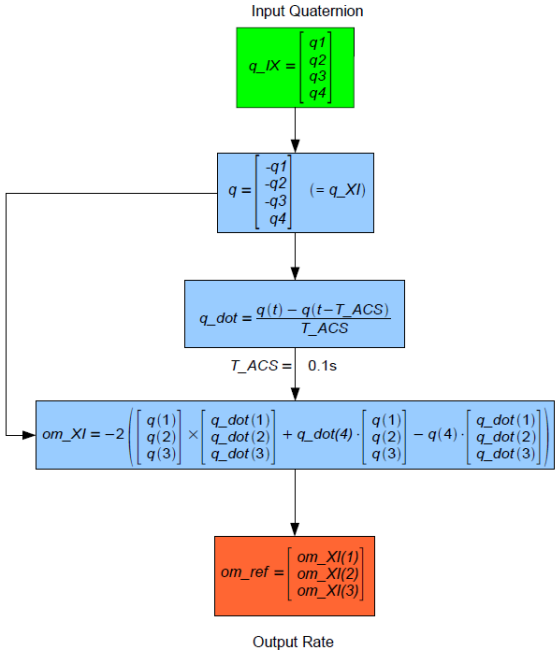


Figure B.2: Quaternion to Angular Rates

APPENDIX C

CORDIC Theory

CORDIC stands for Coordinate Rotation Digital Computer and this is an algorithm introduced by Jack E. Volder [43] in 1959 to compute the trigonometric functions. This is an iterative algorithm which uses vector rotations instead of multiplication to compute the trigonometric functions. [43; 56; 57]

Vector rotation could also be used for polar to rectangle and from rectangle to polar conversions. The CORDIC algorithm provides an iterative method of performing vector rotations by arbitrary angles using only shifts and adds. As there is no multiplication involved in this algorithm, therefore it is hardware efficient solution. The only disadvantage is the delay involved in this iterative process but the cost of this delay compared to the hardware resources utilized is negligible is most of the cases. The algorithm is derived from the general rotation transformation. Which rotates a vector in a Cartesian plane by the angle ϕ.

$$\overline{x} = x\cos\phi - y\sin\phi \tag{C.1}$$

$$\overline{y} = y\cos\phi + x\sin\phi \tag{C.2}$$

This can be rearranged as:

$$\overline{x} = \cos\phi.[x - y\tan phi] \tag{C.3}$$

$$\overline{y} = \cos\phi.[y + x\tan phi] \tag{C.4}$$

In the above equations if the rotation angles are restricted in such a way that $\tan\phi = \pm 2^{-i}$, the multiplication by the tangent term is reduced to simple shift operation. Arbitrary angles of rotation are obtainable by performing a series of successively smaller elementary rotations. If the decision at each iteration i is which direction to rotate rather than whether to rotate or not to rotate, then the $\cos\delta_i$ term become a constant since $\cos\delta_i = \cos -\delta_i$. The iterative rotation can now be expressed as

$$x_{i+1} = K_i[x_i - y_i.d.2^{-i}] \tag{C.5}$$

$$y_{i+1} = K_i[y_i + x_i.d.2^{-i}] \tag{C.6}$$

where:

$$K_i = \cos(tan^{-1}(2^{-i})) = 1/\sqrt{1+2^{-i}} \;\; and \;\; d_i = \pm 1 \tag{C.7}$$

Removing the scale constant from the iterative equations yields a shift-add algorithm for vector rotation. The product of a K_i can be applied elsewhere in the system or treated as part of a system processing gain. That product approached 0.6073 as the number of iteration goes to infinity. Therefore, the rotation algorithm has a gain A_n of approximately 1.647. The exact gain depends on the number of iterations and obey the following relation.

$$A_n = \prod_n \sqrt{1+2^{-2i}} \tag{C.8}$$

The angle of a composite rotation is uniquely defined by the sequence of the directions of the elementary rotations. That sequence can be represented by a decision vector. The set of all possible decision vector is an angular measurement system based on binary arctangents. Conversions between this angular system and any other can be accomplished using a look up table. A better conversion method uses an additional adder-subtractor that accumulated the elementary rotation angle at each iteration. The elementary angles can be expressed in any convinient angular unit. Those angular values are supplied by a small look up table or are hardwired depending on the implementation. The angle accumulator adds a third difference euqtion to the CORDIC algorithm.

$$z_{i+1} = z_i - d_i \arctan(2^{-i}) \tag{C.9}$$

References

[1] HTTP://AGILITYDS.COM/PRODUCTS/C_BASED_PRODUCTS/DK_DESIGN_SUITE/HANDEL-C.ASPX. "Agility Design Solutions Inc." last visited August, 2009. xv, 107

[2] Inc., A.D.S. "Agility Documentation; DK Design Suite DK 5.0." 2008. xv, 109

[3] Laufer, R. and Röser, H. "Lunar Mission BW1-An Academic Low-Cost Small Lunar Exploration Satellite." International Astronautical Federation, paper, IAC-07-A3.1.A03, 2007. 3

[4] Grillmayer, G., Lengowski, M., Walz, S., Röser, H., Huber, F., and Schoenermark, M. "Flying Laptop- Micro-Satellite of the university of Stuttgart for Earth Observation and Technology demonstration." International Aerospace Congress, Vancouver, volume IAC-04-IAA.4.11.P.08, 2004. 4

[5] Grillmayer, G., Falke, A., and Röser, H. "Technology Demonstration with the Micro-Satellite Flying Laptop." 5th IAA Symposium on Small Satellites for Earth Observation, Berlin, volume pp 419-427, Apr 2005. 4

[6] Huber, F., Behr, P. Röser, H., and Pletner, S. "FPGA-based on-Board Computer System for the Flying laptop Micro-Satellite." Proceedings of the data System in Aerospace Conference, SP-638 ESA, Naples, Apr 2007. 4

[7] FLP-Team. "FLP-Mission and System Design Overview." IRS Internal Report, April 2009. 5, 6

[8] Wertz, J.R. Spacecraft Attitude Determination and Control. Kluwer Academic Publishers, ISBN 90-2777-0959-9, 1978, reprinted 2000. 14

[9] Yasir, M., Grillmayer, G., and Röser, P.D.H.P. "Safe mode control of micro-satellite Flying Laptop." 12th IEEE-INMIC Conference, Karachi, volume 978-1-4244-2824-3/08 IEEE, December, 2008. 19

[10] Böhringer, F. and Grillmayer, G. "Orbit definitions for simulation purposes." FLP-SP-00000-001-IRS, IRS Internal Report, September 2006. 28

[11] Belgian Institute, o.S.A. "SPENVIS." `http://www.spenvis.oma.be/`, results updated in November 2007. 28

[12] Yasir, M. "Radiation environment." FLP-TN-00000-004-IRS, IRS Internal Report, June 2006. 29

[13] Haase, T. and Michael, C. "Magnetometer Manual." ZARM Technik, IRS-ZAR-MAG-MM-01-0001, February 2006. 32

[14] Saile, D., Waidmann, M., Grillmayer, G., and Yasir, M. "ACS component orientation and position." FLP-TN-00000-002-IRS, IRS Internal Report, Feb 2008. 33, 38, 44, 49, 59

[15] Waidmann, M. and Grillmayer, G. "Magnetometer Interface Description." FLP-RP-02800-001-IRS, IRS Internal Report, Feb 2006. 36

[16] AZURSPACE. "Properties of Cell Type: 3G-27%." Data_Sheet_3G27, IRS Internal Report, April 2007. 36

[17] Yasir, M. "Sun Sensor system specifications and design." FLP-TN-07520-001-IRS, IRS Internal Report, Nov 2007. 40

[18] Montenbruck, O. and Markgraf, M. "User's manual for the Phoenix GPS receiver." Version, 1.7a, GTN-MAN-0120, DLR, Sep 2006. 42, 43, 47

[19] Waidmann, C. and Grillmayer, G. "GENIUS GPS system design and development." FLP-TN-02900-001-IRS, IRS Internal Report, May 2006. 42, 44, 47

[20] GmBH, L. "μFORS User Manual." 140650-2000-311, Rev B, July 2006. 48, 53

[21] Waidmann, C. and Grillmayer, G. "Rate Sensor development and Design." FLP-TN-03200-001-IRS, IRS Internal Report, Nov 2006. 53

[22] Jgensen, J. and μASC Team. "μ-Advanced Stellar Compass, General Information." IRS-DTU-PRP-3000, Issue 1.2, IRS Internal Report, June 2004. 54, 55

[23] Saile, D. and Grillmayer, G. "Star Tracker Operations Manual." FLP-MA-03100-001-IRS, IRS Internal Report, Mar 2006. 55

[24] Matthews, O. and Offterdinger, P. "Flying Laptop Magnetic Torquer MT6-2, Operation Manual." ZARM Technik, FL-MA-ZAR-001, November 2006. 59

[25] Saile, D. and Grillmayer, G. "MGT Power Electronics." FLP-TN-02600-001-IRS, IRS Internal Report, Nov 2006. 61

[26] Stücklin, S. and Stapler, W. "Technical description, Reaction Wheel Assembly, RSI 01-528 with integrated wheel drive electronics." 15175-338, Issue 02, TELDIX, Oct 2000. 62, 64

[27] Waidmann, C. and Grillmayer, G. "Reaction Wheel Operations Manual." FLP-MA-02500-001-IRS, IRS Internal Report, Feb 2006. 64

[28] Inc, X. "Available from `http://www.xilinx.com/`." last visited August, 2009. 66

[29] Inc, A. "Available from `http://www.actel.com/`." last visited August, 2009. 66

[30] Rabaey, J. and Chandrakasan, B. Digital Integrated Circuits - a design perspective. Prentice Hall, ISBN 0-13-120764-4, 2003. 66

[31] Maxfield, C. The design Warrior's Guide to FPGAs. Elsevier, ISBN 0750676043, 2004. 66

[32] Winetraub, Y., Bitan, S., dd, U., and Heller, D.A.B. "Attitude Determination - Advanced Sun Sensors for Pico-satellites." Handasaim School, Tel-Aviv University, Israel `http://www.agi.com/downloads/corporate/partners/edu/advancedSunSensorProject.pdf`, last visited July, 2009. 82

[33] Montenbruck, O. and Gill, E. Satellite Orbits. Springer, ISBN 13 978-3-540-67280-7, 2005. 91, 97

[34] Ruppin, T.C. "Simulation of the CASsat Satellite and Environment.", 2004. 91

[35] Flatley, T.W., Morgenstern, W., Reth, A., and Bauer, F. "A B-Dot Acquisition Controller for the RADARSAT Spacecraft." Code 712/Guidance, Navigation and Control Branch, Goddard Space Flight Center, Greenbelt, Maryland, 20904, USA http://ntrs.nasa.gov/archive/nasa/casi.ntrs.nasa.gov/19970017192_1997026214.pdf, last visited July, 2009. 94

[36] Jan, Y. and Chiou, J. "Attitude control system for ROCSAT-3 microsatellite a conceptual design." Acta Astronautica, volume 56, pp. 439-452, 2005. 96

[37] Lee, S., Ahn, H., and Yong, K. "Three-axis attitude determination using incomplete vector observations." Acta Astronautica, volume 65, pp. 1089-1093, 2009. 96

[38] Santoni, F. and Bolotti, F. "Attitude determination of small spinning spacecrafts using three axis magnetometer and solar panels data." Aerospace Conference Proceedings, 2000 IEEE, volume 7, pp. 127-133, 2000. 96

[39] Shuster, M.D. and Oh, S.D. "Three-axis attitude determination from vector observations." Journal of Guidance and Control, vol. 4, Jan.-Feb. 1981, p. 70-77., volume 4, February 1981:pp. 70–77. 96

[40] Wie, B. Space Vehicle Dynamics and Control. AIAA Education Series, ISBN 1-56347-261-9, 1998. 98, 99

[41] Fichter, W. Advanced Spacecraft Navigation and Control. Lecture notes, IFR, Universitaet Stuttgart, 2007. 100

[42] http://www.mathworks.com/. "The Mathworks." last visited August, 2009. 104

[43] Volder, J.E. "The CORDIC Trigonometric Computing Technique." IRE Transactions on Electronic Computers, pp330-334 http://www.jacques-laporte.org/Volder_CORDIC.pdf, September 1959. 119, 171

[44] Hirth, M. and Grillmayer, G. "ACS Simulink Model Description." FLP-TN-00000-005-IRS, IRS Internal Report, Nov 2006. 121

[45] Moutaux, A., Hirth, M., and Grillmayer, G. "ACS Matlab Model Description." FLP-TN-00000-006-IRS, IRS Internal Report, May 2007. 121

[46] Moutaux, A. and Grillmayer, G. "ACS Sensor Noise Model." FLP-TN-00000-007-IRS, IRS Internal Report, May 2007. 127

[47] Wolter, V. and Hirth, M. "Definition of ACS Variables." FLP-IDD-4200-001-IRS-01, IRS Internal Report, October 2006. 136

[48] Yasir, M. and Grillmayer, G. "Data Transfer Protocol of ACS simulation in FPGA-Matlab Environment." FLP-TN-00000-009-IRS, IRS Internal Report, November 2006. 136

[49] Eickhoff, J., Falke, A., and Roeser, H.P. "Model-based design and verification State of the art from Galileo constellation down to small university satellites." Acta Astronautica, pp383-390, 2007. 139

[50] Kuwahara, T., Yasir, M., Falke, A., Ziemke, C., Eickhoff, J., and Röser, H. "Development of a Hardware-in-the-Loop Simulation Environment on a MDVE for FPGA-based OBC." Trans. Jpn. Soc. Aeronaut.Space Sci. Space Tech. Japan, volume 7, pp. Pf_1-Pf_9, 2009. 140

[51] Yasir, M., Kuwahara, T., Ziemke, C., Fritz, M., and Röser, H. "Simulation-Based Testing of Embedded Attitude Control Algorithm of a FPGA based Micro Satellite." SSC09-VII-10, 23rd Annual AIAA/USU Conference on Small Satellites, Utah, 2009. 140

[52] Hirth, M. "Auslegung des Reglermodells und der Regelschleifen für den Inertial und Nadir Pointing Mode des Kleinsatelliten Flying Laptop.", Feb. 2006. 143

[53] Hirth, M. "Entwurf des navigationssystems und Auslegung des Target-Pointing Reslers für den Kleinsatelliten Flying Laptop.", Nov. 2006. 143

[54] Moutaux, A. "Simulation of the Attitude Control System and Development of the Sensor Noise Model for the Flying Laptop Satellite.", Sep. 2006. 143

[55] Lieske, J., Lederle, T., Fricke, W., and Morando, B. "Expression for the precession quantities based upon IAU (1976) System of astronomical constants." Astronomy and Astrophysics, volume 58, pp. 1-16, 1977. 165

[56] Delosme, J. "CORDIC algorithms: theory and extensions." In Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series, volume 1152. November 1989. 171

[57] Andreka, R. "A survey of CORDIC algorithms for FPGA based computers." In ACM/SIGDA sixth international symposium on FPGA, session 9, Novel FPGA Applications. February 1998. 171

Curriculum Vitae

Personal

Name:	Muhammad Yasir
Birth:	1st of October 1976
Birth Place:	Sargodha, Pakistan
Nationality:	Pakistani
Martial Status:	Married
email:	yasir@daad-alumni.de

Education

04/2006-Present	PhD in Aerospace Engineering, Student at Institute of Space Systems, University of Stuttgart, Stuttgart, Germany
10/2001-04/2004	Masters in Aerospace Engineering, Student at Beijing University of Aeronautics and Astronautics, Beijing, China
1995-1999	BSc. in Mechanical Engineering, NWFP University of Engineering and Technology, Peshawar, Pakistan

Professional

2006-Present	Manager GNC, Pakistan Space and Upper Atmosphere Commission SUPARCO, On study leave abroad for PhD studies in Aerospace Engineering
2004-2006	Assistant Manager GNC, SUPARCO, participated in projects related to GNC including research as a member of Satellite Launch Vehicle design team
2001-2004	Assistant Manager, SUPARCO, On study leave for Masters studies
1999-2001	Training Engr. SUPARCO, training in Institute of Space Technology, Islamabad