DEGREE PROJECT

Integration of an Electric Propulsion High Voltage Unit into the FLP2 SmallSat Testbench

Thomas Walford

Space Engineering, master's level (120 credits)
2018

Luleå University of Technology
Department of Computer Science, Electrical and Space Engineering

LULEÅ UNIVERSITY OF TECHNOLOGY

Integration of an Electric Propulsion High Voltage Unit into the FLP2 SmallSat Testbench

Thomas C. Walford

Master Thesis
2017

Luleå University of Technology
Department of Computer Science, Electrical and Space Engineering
Julius-Maximilians-Universität Würzburg
Institut für Informatik

Abstract

This paper is based around the test bench for the Flexible LEO Platform, a proposed small satellite platform for use in Low Earth Orbit. The main focus of the paper is the integration of a High Voltage Unit into this test bench. The High Voltage Unit is used to power the Electric Propulsion Subsystem that is being developed at Airbus DS and is likely to be used in the platform. This integration focusses primarily on the data link between the on-board computer and the Power Control and Distribution Unit which supplies power to the High Voltage Unit, but no study on an on-board data link is complete without looking at the link from the satellite to ground.

The paper will mainly build on two pieces of prior work completed in the development of this platform. The first updated data chain between the ground and the on-board computer from packet level to frame level. The second began the electrical integration of the Power Control and Distribution Unit into the satellite test bench.

The key areas addressed in the paper are firstly, the implementation of code within the on-board software to command the Power Control and Distribution Unit (both simulated and real) through telecommand and identify the corresponding telemetry that is produced. Secondly, confirmation of a working data chain from the Ground Station to the High Voltage Unit of the Electric Propulsion Subsystem, via the Power Control and Distribution Unit, and if the chain is incomplete, linking them together fully. Thirdly, the robust test of this data chain to ensure that both the simulated and real engine can be controlled. Finally, a detailed documentation of the result along with the software developed and the production of a user manual.

Preface

Of course I would like to thank my family and friends who have supported me throughout my SpaceMaster studies. Many of them believed in me when I did not fully believe in myself and without them this would not have been possible.

Additionally I would like to thank my colleagues during my time at Airbus DS, Friedrichshafen. Without their patience, expertise and support, this thesis would be considerably less thorough and my time there would have been a much diminished experience. I would particularly like to thank Jens Eickhoff and Waj Chintalapati at Airbus, their combined knowledge of the satellite was invaluable.

Tom Walford

Co-funded by the
Erasmus+ Programme
of the European Union

Declaration

I, Thomas Walford, hereby declare that this thesis is my own work and that, to the best of my knowledge and belief, it contains no material previously published or written by another author nor material which to a substantial extent has been accepted for the award of any other degree or diploma of a university or other institute of a higher education, except where due acknowledgment has been made in the text.

Würzburg, 03.11.2017 Thomas Walford

Contents

CHAPTER 1

Context

1.1 Introduction

The primary goal of this paper is to look at the command and control of an electric propulsion subsystem within a SmallSat and the way in which the High Voltage Unit (HVU) is integrated into the satellite. The expanded view of this task must therefore begin at the command and control of the satellite in general. Prior work on the Flexible LEO Platform (FLP) was able to design and integrate a simulated CCSDS board. The CCSDS board is central in the data link between the On-board Computer (OBC) and ground. A study on the simulation and integration of the CCSDS board therefore provides a strong overview of this link. This knowledge is therefore vital to understand the command and control of the electric propulsion subsystem and will constitute the first section of this paper.

The second area covered in this paper is the integration of a HVU that is required to power the propulsion system of the FLP. As the satellite is not yet in the production phase, some of these parts are simulated and some are real. Therefore, this thesis must also look at how to link these simulated or real parts together and how to alter any prior On-board Software (OBSW) to ensure that the commands given by the OBC reach the HVU.

These interconnected tasks focus upon the data chain between the OBC and a system or subsystem (the ground station and HVU respectively). Within the paper, the author will begin with a brief description of the FLP, the Future Low-cost Platform (FLP) and relevant background information required to understand the tasks completed. It will then provide a more detailed description of the two data links. Within each the paper will describe final solutions created and explain how any problems encountered were overcome. The author will then describe the OBSW and suggest how it might be altered to upgrade the Attitude Control System (ACS) to an Attitude and Orbital Control System (AOCS)

in the most robust and least disruptive way. At the end of the paper, the author will summarise the lessons that can be taken from this experience.

1.2 The Future Low Cost Platform

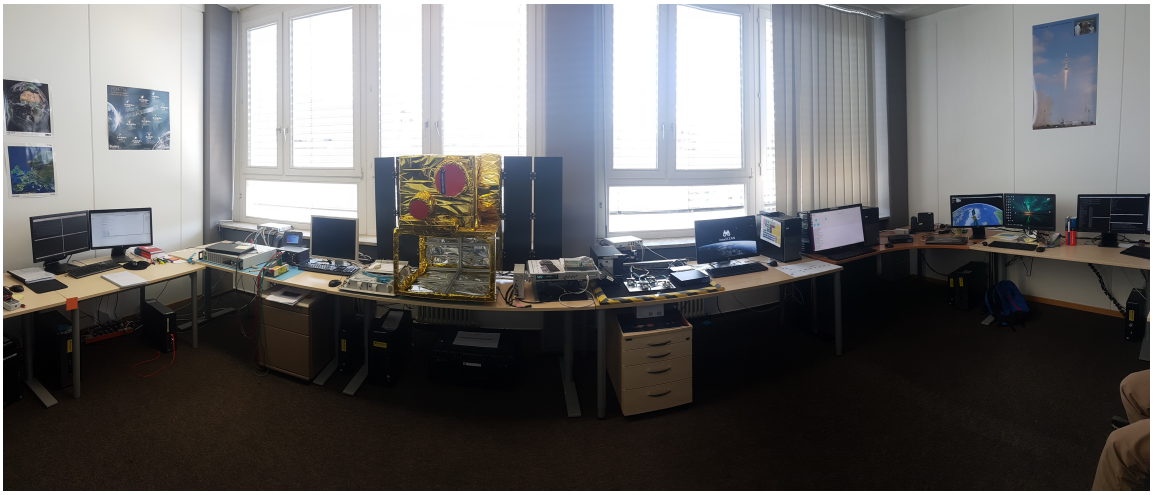

Figure 1.1: The FLP test bench in Friedrichshafen

1.2.1 Introduction of the Flexible LEO Platform

The FLP, as shown in figure 1.2, is a new Smallsat platform currently in development by the Future Programs department of Airbus DS, Friedrichshafen. In 2009, Universität Stuttgart began the development of a satellite named the “Flying Laptop”.[5] As Airbus DS became more involved, it was decided that the platform would be called the "Future Low-cost Platform", shortened to FLP to match the name of the satellite. The FLP satellite has since been launched from Baikonur, Kazakhstan. During the design process two test benches were made to work on the OBC, one in Stuttgart and one in Friedrichshafen, to allow support by Airbus DS to be accurate. As the project moved forward and less support was required from Airbus DS, the company began work on the next generation of the FLP. This platform was then renamed to "Flexible LEO Platform" as it moved past the early development phase. The project has five main unique selling points that will be covered below.

Dual Core Processor

The first is that the OBC has a dual core processor. This was chosen with the intention of using one core for the platform and one core for the payload. This allows for a reduction in the number of computing boards on the satellite and therefore correspondingly reduces the amount of redundant boards necessary. Additionally this means that less individual instances of the Real-Time Executive for Multiprocessor Systems (RTEMS) Operating System (OS) are required and this frees up valuable space within the memory of the satellite, reducing the complexity of intra-satellite communication.

Ethernet Based Payloads

The second unique selling point is the use of a Space Wire RMAP Responder (SRR) to allow the payload to be connected to the OBC via an Ethernet cable rather than the more common Space Wire (SpW) connection. This allows for significantly reduced complexity within the building and testing of the payload as it is able to use the same interface from the start of the build to the final operation.

Combined Data and Power Management Infrastructure

The third major innovation is the use of the combined data and power management infrastructure that was developed by Jens Eickhoff at Airbus DS.[4] This merges the on-board computing of the satellite with the Power Control Unit (PCU) and therefore reduces the number of redundant processors on board. In addition to the obvious benefit of a reduction of weight, it also improves reliability (due to the smaller number of parts that can potentially fail), reduces the cost of components and minimises the overall complexity of the system. This reduction in cost comes from the reduced weight and complexity throughout the satellite. Additionally, this system reduces the total computing power required, further reducing the cost and the power demands on the satellite. These reductions in power demand allow the platform to weigh less or to support a more powerful payload. This architecture is therefore a vital innovation that allows for more affordable satellites.

Wall Mounted Electronics

The next innovation is the use of wall mounting for the OBC and other important electronic equipment. This was originally intended as a space saving method when it was decided to include a propulsion system that required a large tank of propellant. However, in addition to being an effective space saver, it also saved a large amount of mass. When the same ideas are applied to other components in the satellite, they could significantly

reduce the size and mass of the platform area of the satellite, resulting in reduced costs for placing the satellite into orbit or the ability to increase the size and weight of the payload.

Re-usability

The final innovation is the the re-usability of the satellite. The original satellite has at the time of writing already been placed in orbit. By reusing much of the original design and software, the cost of designing and building the second generation of this platform has been reduced. The saved money has in part been used to fund other innovative parts of the satellite, but also to lower the cost to a potential investor and therefore reduce the expense of important scientific missions.

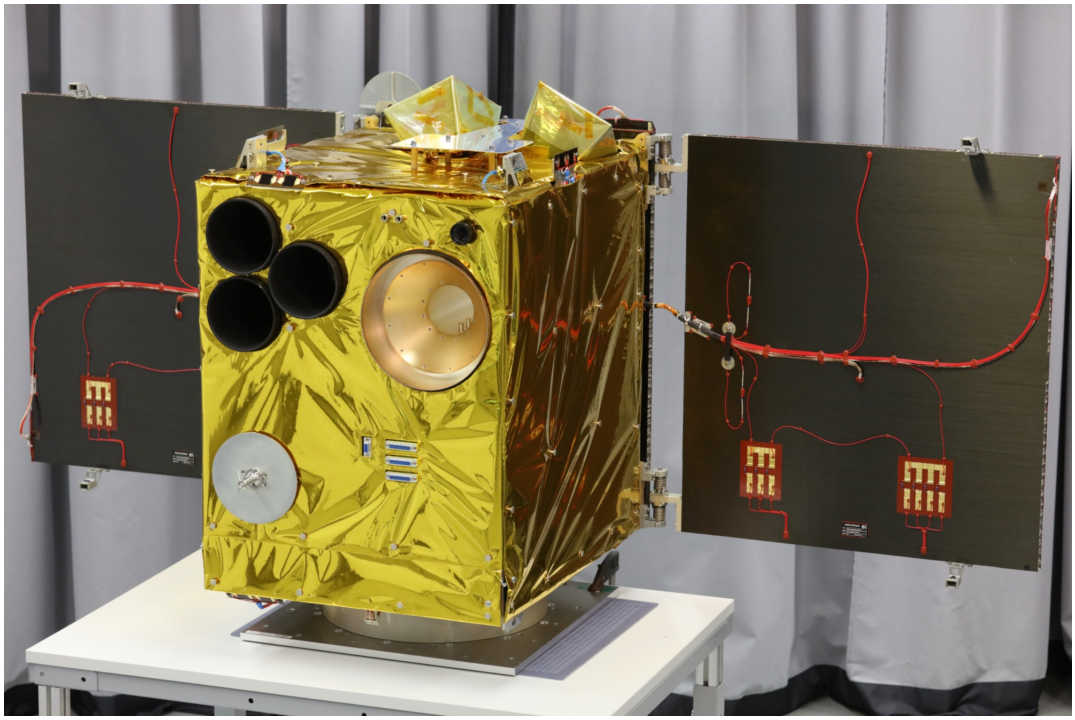

Figure 1.2: The completed Flying Laptop. ©IRS, University of Stuttgart

1.2.2 Other important features

In addition, the platform includes other important features which, while they are not unique, are vital to allow the platform to fully service any payload it has on board. The

first of these is that the platform has full redundancy. This ensures that any single failure within the platform does not affect the ability of the platform to perform. Obviously, in order for this measure to be effective, all of the components within the platform are also cross-coupled to allow a different route for the data or power to get around any failed component.[16]

The platform will be fully CCSDS and PUS compliant. The Consultative Committee for Space Data Systems (CCSDS) is an organisation founded to ensure that communications between missions in space and the ground are standardised.[13] In part this was achieved by introducing the Packet Utilisation Standard (PUS) which all CCSDS compliant communications follow. This allows ground stations built by any member space organisation to successfully communicate with any CCSDS compliant satellite. By following these standards, the FLP will be able to communicate with the majority of commercial ground stations in the world.

A third important feature is the radiation hardness of the processor board. Despite being a new innovation, it has been extensively tested to ensure that it is resistant to a large total radiation dose and is robust to any Single Event Upsets (SEU) and latchups. This ensures that the processor is able to withstand the harsh environment of space.

All of these additional features ensure that the FLP is able to fly expensive payloads as it is a reliable and cost-efficient platform that is open to commercial space flight. It must also be noted that as many of the innovations and features of the FLP follow a semi-open design (many of the guiding principles behind the satellite come in the form of published papers), the satellite is highly suitable for cooperation and collaboration between the payload designer and platform provider. This provides a new business model where the building of a satellite is a project based around partners rather than a customer-provider model.

1.3 The test bench

Satellite simulation and testing is a long process and comes in many stages. Naturally, there is a point at which certain core and complicated parts of the satellite system, such as the OBC, need to be purchased, whilst other parts, such as the solar panels, are not necessary to satellite development at that stage. This is the stage that the FLP is currently at. The core components are then assembled into a test bench where they can all interact together.

By purchasing and integrating parts into a single test bench the software engineers can work directly with hardware similar to the equipment that will later be flown. It also

allows them to modularly build up the connections that would be in the real satellite. The test bench is therefore very important for testing the various links, simulations and real components.

For satellites built by large teams, or teams spread out geographically, there can be two or more test benches. For the FLP, as shown in figure 1.3, Stuttgart built a test bench at their university and, as Airbus got more involved, a second test bench was built at Friedrichshafen that mimicked the original. As FLP moved to its second generation, the test bench at Airbus began to be expanded and differ more heavily from the original and its current development is shown in figure 1.1. At the time of writing it takes up the physical space of two offices, can have up to six people working on it at any one time and contains the OBC, Power Control and Distribution Unit (PCDU), routers, SRRs, a model of the satellite, example wall mountings for the electronics and a simulated payload. Additional equipment is also in the lab to power these components, connect through to other projects or support the work of employees of the various subsystems. This is shown in figure 1.1.

Figure 1.3: The FLP Testbench in Stuttgart. ©IRS, University of Stuttgart

CHAPTER 2

Radio Frequency Communications

2.1 Communicating with Ground

The ability for the satellite to communicate with the ground station is one of the most basic tasks that all satellites must perform. It is the link through which all data, both mission related and housekeeping, is passed. Obviously, it is impossible to physically link a satellite with the ground, and therefore all satellites must communicate through free space. Whilst in recent times much research time and resources have been dedicated to developing the field of free space optical communications between satellites and ground, this is still a very young field of study.[14] Almost all satellites rely in some way on the use of Radio Frequency (RF) communications.[7]

Almost all of the larger scale and commercial satellites are able to communicate with commercial ground stations. While some small scale satellites are developed that do not use commercial ground stations, the most common examples being the academically funded cubesats, this choice comes with a myriad of drawbacks. At the most basic level, not including the ability to communicate with a commercial ground station increases the likelihood of the ground to satellite link not working in some way. Therefore, despite the fact that including this capability increases the cost of the satellite in several different ways, most commercial satellites include this capability in order to reduce this mission ending risk. This will be more closely looked at in the next section.

All commercial, and most custom, ground stations use a similar system, that uses encoding and decoding to ensure security and maximise transmission speed.[12] As they are so similar, and to save space, only the commercial system will be described. This uses the PUS that was described at the start of this thesis, though a custom ground station could use a different standard. Looking from the direction of satellite to ground, this

link begins at the OBC. The OBC usually contains a buffer that holds telemetry packets of various types that need to be downlinked to ground. This data is then passed to the CCSDS board that places the data into a Channel Access Data Unit (CADU).[13]

The CADUs are then further encoded, usually using Attached Synchronisation Markers, Reed-Solomon coding, pseudo-randomizers and convolutional coding, though other encodings are possible. This signal is then sent to the antenna, at which point it is transmitted to Earth using radio frequencies. At ground, a second antenna (often much larger and with higher gain) receives the signal.

At this point all of the encoding is reversed and the CADUs directed into the ground station computer terminal. The terminal then extracts the packets and they are displayed or saved in a database as necessary. This data can then be used in the decision making process, or sent to the end user of the data gained by the payload. This chain is then performed in reverse using the same antennas when commanding the satellite.

2.2 Commercial and Amateur Radio Frequencies

If the mission chooses not to include the ability to communicate with a commercial ground station, it must still be able to communicate with ground. This means that the satellite must communicate with either a custom built ground station, which must be built and paid for using mission funds, or an arrangement must be made with a pre-existing custom ground station. For the purposes of ground to satellite communication, there are two broad categories of communication frequencies, amateur radio frequencies and licensed. The amateur radio frequencies cost nothing and are unregulated. This means that if another user decides to use the same frequency to transmit signals near to the ground station, it will increase the amount of noise and therefore reduce the ability for the ground station to receive data from the satellite. By contrast, the licensed radio frequencies require a licence, which costs money and comes with many legal and technical requirements. The major benefit is the relative guarantee that the frequency is not in use when the satellite communicates with ground and therefore a much lower Signal to Noise Ratio (SNR). A lower SNR allows a faster data transmission speed for the same amount of power used by an antenna.

Any new component in a satellite, or in this case satellite to ground link, needs to be tested. With a custom ground station, this is technically difficult as it must perform several functions simultaneously, such as pointing to the satellite, receiving a signal, decoding and failure detection. As the antennas are physically large (typically several meters tall and long at the minimum), it is difficult to create a simulated signal that moves across the antenna's field of view. This means that it is difficult to test all the

functions of the antenna simultaneously, which significantly increases the chance of link failure. This problem is reduced when using commercial ground systems as the antennas have already been tested on satellites in orbit, which effectively mitigates the risk of link failure from the ground station.

In order to use a commercial ground station, the owner of the satellite must pay for the service. Over the months and years that a satellite runs this can amount to a significant cost, and this makes it attractive for some missions to use alternative forms of satellite to ground communication. Furthermore, when using commercial ground stations, there are many communication protocols which must be used. While many can be programmed manually, this represents extra cost and time. Alternatively, parts such as the CCSDS board, which automatically encodes and decodes these protocols within the satellite, can be purchased, but again this increases cost. Finally, the right to use the licensed radio frequencies used in commercial ground stations must be purchased at additional cost and as these frequencies can vary between countries, this can quickly become a large sum of money. Ultimately, a cost to benefit analysis must be completed with regard to the mission of the satellite, which will determine the choice of using amateur or licensed radio frequencies.

The Flying Laptop therefore chose to implement both systems. The link to a commercial ground station was used for the first three weeks that the satellite was in orbit, and a custom ground station based on an antenna built atop a building in Universität Stuttgart was designed for the rest of the mission lifetime. This was done because the University wanted to gain experience in the entire satellite and ground system, and therefore the building of a ground station was necessary. It was also done to reduce the ongoing costs of maintaining the satellite to ground station link over the long term. However, it was decided that the risk of the ground station to satellite link failing was too high, especially considering they were flying three payloads that were funded by external customers. It was therefore necessary to purchase short term licenses for the use of commercial ground stations. The link between ground and the FLP satellite could therefore be guaranteed to work while the satellite functionality was tested and if the amateur radio link failed for any reason or at any time, a backup ground to satellite link was available.

2.3 Error Checking and Security

There are six main encodings, modulations and signal alterations that occur to a signal and each is applied iteratively to the previous encodings.

- Reed-Solomon Coding
- Pseudo-Randomization

- Bit-Synchronous Output
- Non-Return-to-Zero Level Modulation (NRZ-L)
- Multiplexing
- Command Link Transmission Unit (CLTU)s and CADUs

The purpose of these encodings vary, but the first main reason for using them is to build error correction into the signal that is sent. As satellite communication is free space communication across large distances, and the satellite has relatively small amounts of power available for increasing the signal strength, the chance of an error in the transmission of a message is large compared to wired communication. Without error correction, such as Reed-Solomon encoding, any mistake in a CLTU or CADU would result in that message having to be resent. With error correction, these mistakes can sometimes be found and corrected without wasting valuable contact time to resend these messages.[15]

The second reason is security. As the signals cover large distances, and the ACS of a satellite is not perfect, the area in which a signal can be observed on the ground is often several miles wide. As the data is often valuable, and to ensure no other individuals or organisations can gain control of an expensive piece of hardware, security measures are needed. Depending on the exact nature of the satellite, the security coding can be very complex or relatively simple. This is a trade off between increased security and decreased data transmission speed for the satellite. Almost all satellites however will use some form of security coding such as Pseudo-Randomizers to ensure the satellite is secure.[19]

The third reason is to improve frame synchronisation. This ensures that if there is a bit slip or other event that forces the transmitter and receiver out of synchronisation it can be quickly restored. This is important as an error in the synchronisation results in all messages received not being correctly deciphered until the synchronisation is restored.

NRZ-L modulation is used to increase the band width of the signal, which is a vital improvement in satellite to ground communications. Having the on and off state of the bits in the message further apart effectively increases the signal strength without a significant increase in the maximum power output of the antenna. It also smooths the current and voltage profile being sent to the antenna.

As CLTUs and CADUs will be discussed extensively throughout the rest of this thesis, the last signal alteration looked at is multiplexing. This is the act of sending several messages via the same medium simultaneously. In the case of free space communications, this most commonly refers to polarization multiplexing. This uses the orthogonal polarization of the electromagnetic wave to create separate channels. As both polarizations can send separate messages, this increases the rate of data flow which again is vital to

ensure the maximum data can be transmitted in the short amount of time that a satellite is visible to the ground station.

Under normal operation, the CCSDS board is responsible for error checking and security with the signal that is sent. This is not done in the simulated CCSDS board as these components are not present in the ground system. This is only valid whilst the OBC communicates with the computer that houses the ground station via wires and not antenna. At this point, the CCSDS board simulation will either have to be upgraded or, more likely, replaced with the hardware version.

2.4 PUS

Nr	Description
1.1	Telecommand Acceptance Report - Success
1.2	Telecommand Acceptance Report - Failure
1.3	Telecommand Execution Started Report - Success
1.4	Telecommand Execution Started Report - Failure
1.5	Telecommand Execution Progress Report - Success
1.6	Telecommand Execution Progress Report - Failure
1.7	Telecommand Execution Completion Report - Success
1.8	Telecommand Execution Completion Report - Failure

Table 2.1: PUS Service 1 Functions and Sub-services

The PUS was implemented by the European Space Agency (ESA) in the early 90s in order to standardise the method by which packets were sent and received in Telemetry (TM) and Telecommand (TC) with satellites. The PUS sets up a set of services and functions that are regularly used in satellite TM and TC. By using the PUS these services can then be utilised for a broad range of functions when communicating with a satellite. Each service is assigned a number and many have several numbered sub-services or functions that relate closely to the service. This allows for fast implementation into software (SW) and the ability for controllers to quickly look up a service number. An example of these is PUS service 1: Acknowledge Service shown in table 2.1. This shows the 8 different TM reports that the PUS defines. These are used whenever a TC is sent to the satellite to ensure that the Ground Station (GS) user knows that the TC has been received and processed by the satellite. The PUS services are shown in table 2.2.[2] These services do not necessarily need to be implemented for every satellite and notably for the

FLP several services, such as Services 4, 18 and 19, are only partially used and others, such as 2, 8 and 9, are not used at all. Additionally, the FLP has several additional TC and TM requirements and therefore uses service 20x to define additional services.[5]

Ser	Service Description
1	Acknowledge Service
2	Device Commanding Service
3	Housekeeping Service
5	Event Reporting Service
6	Memory Management Service
8	Function Service
9	Time Management Service
11	On-board Operations Scheduling Service
12	On-board Monitoring Service
15	On-board Storage and Retrieval Service
17	Test Service
18	Event-Action Service
20x	Self-defined services for mode commanding

Table 2.2: The Standard PUS Services

So far, these services have been described in a general form. When they are implemented into code, they naturally form a bit string. This bit string is a set of binary data and depending on the protocol that is used, this can be translated to form a set of data. One translation is American Standard Code for Information Interchange (ASCII) encoding, in this system an eight bit string is used to form a single printable character such as letters. When this is used in the context of the PUS protocols, the individual bits of the bit string form a series of flags.

Rather than a system where eight bits represents a single character, the bits that make up the headers and trailers of the PUS protocol are individually assigned as a specific flag. These flags can be single, for a value that is either off or on, such as if the frame contains a Cyclic Redundancy Check (CRC), or grouped for numbers, such as the satellite ID or number. Due to the flexibility of this system the information required by both ground and satellite can be significantly compressed. This ensures that the number of bits that do not directly relate to data is reduced and therefore more data can be transmitted in the same amount of time. The downside to this is that the computer and developer must know exactly what each bit means in order to ensure that the encoding makes sense. The next section will look deeper into the specifics of this system.

CHAPTER 3

Data Transfer to Ground

3.1 Data transfer to ground

This section details how the simulation and integration of the CCSDS board was completed between the OBSW and ground station. This is then related to the the command and control of the HVU. The first step in understanding this link in the data chain, the CCSDS standards and how compliant satellites communicate with ground, has already been broadly covered in the preceding chapter. This section will more closely detail the exact structure of an individual message. This will be done in such a way that it could be any standard PUS message to the OBC.

To start, the author will describe how the Central Checkout System 5^{th} Generation (CCS5), the ground station used by the FLP, sends telecommands and receives telemetry from the satellite and, therefore, give an understanding of how a satellite receives communications. The RF signal from ground is received by the antenna of the satellite. Immediately after, the signal is processed by Field Programmable Gate Arrays (FPGAs). The processed signal is then passed into the CCSDS board. From there, the data within the signal is processed before being sent via SpW to the OBC.

It is beyond the scope of this thesis to look closely at the signal, however this was briefly discussed in the previous section. Therefore, this section will look at the form that the data arrives and leaves the satellite, and how this differs with the data that is received and produced by the OBC. This will be covered first as it makes up the basis for everything else completed in this part of the project.

3.1.1 Packets

As data moves from the OBC to the individual components within the satellite (such as the magnetometers or propulsion system) and vice versa, it will pass through routers and I/O boards. These routers and I/O boards send the data to the correct component within the satellite but require instructions as to where each section of data needs to go. To do this, the raw data being sent is placed within a "packet" which consists of a "header" that precedes the data and a "trailer" that follows it. These have many uses, including but not limited to, error control and detailing the address of the system the data is intended for.[13]

The exact form of the packet depends on the protocol used. In the case of the OBC to ground link, this protocol is the PUS. In this protocol, a packet header, 6 bytes long, and a data field header, 4 bytes long, are placed in front of the data being sent from the OBC. These headers contain important information that various components use, such as the Application Identification (APID) which defines the specific process within the satellite that is communicating with the OBC, and the packet length flags, which define the length of the packet so that the various routers and I/O boards know where the end of the packet is. The frame trailer in this protocol is simply packet error control. This error control is in the form of a 2 byte CRC that ensures that any errors within the data are found and the packet resent. The data to be received, which is now surrounded by the headers and error control, is called a packet and forms the basic form of communications within the satellite.

The exact form of a PUS packet depends on whether it is TC or TM. It is relatively rare for a protocol to use different packet forms for outgoing and incoming data. This could be due to the limited power and antenna size of the satellite causing TC to have a higher chance of containing errors. Additionally, the amount of data being sent to ground is far higher than the reverse, and this is reflected in the telemetry protocol which is designed to contain a larger amount of data.

Unfortunately, when sending information either between the ground and the satellite or between satellite systems, the packet system is still not enough as it fails to include other vital information such as which satellite is to be receiving or sending data or information for the routers in intra satellite communication. Intra-satellite communications therefore use the Remote Memory Access Protocol (RMAP) which is, as stated above, the protocol for any SpW based communications. This requires a specific protocol as the SpW is a fully duplex, point to-point serial link which means that messages travel along the same wire in both directions simultaneously. This simultaneous data flow requires more information than is provided by a packet and therefore the packets are placed into frames. The process RMAP uses was not studied for this thesis and therefore will not be discussed in this paper. Similarly, the extra requirements for ground to satellite commu-

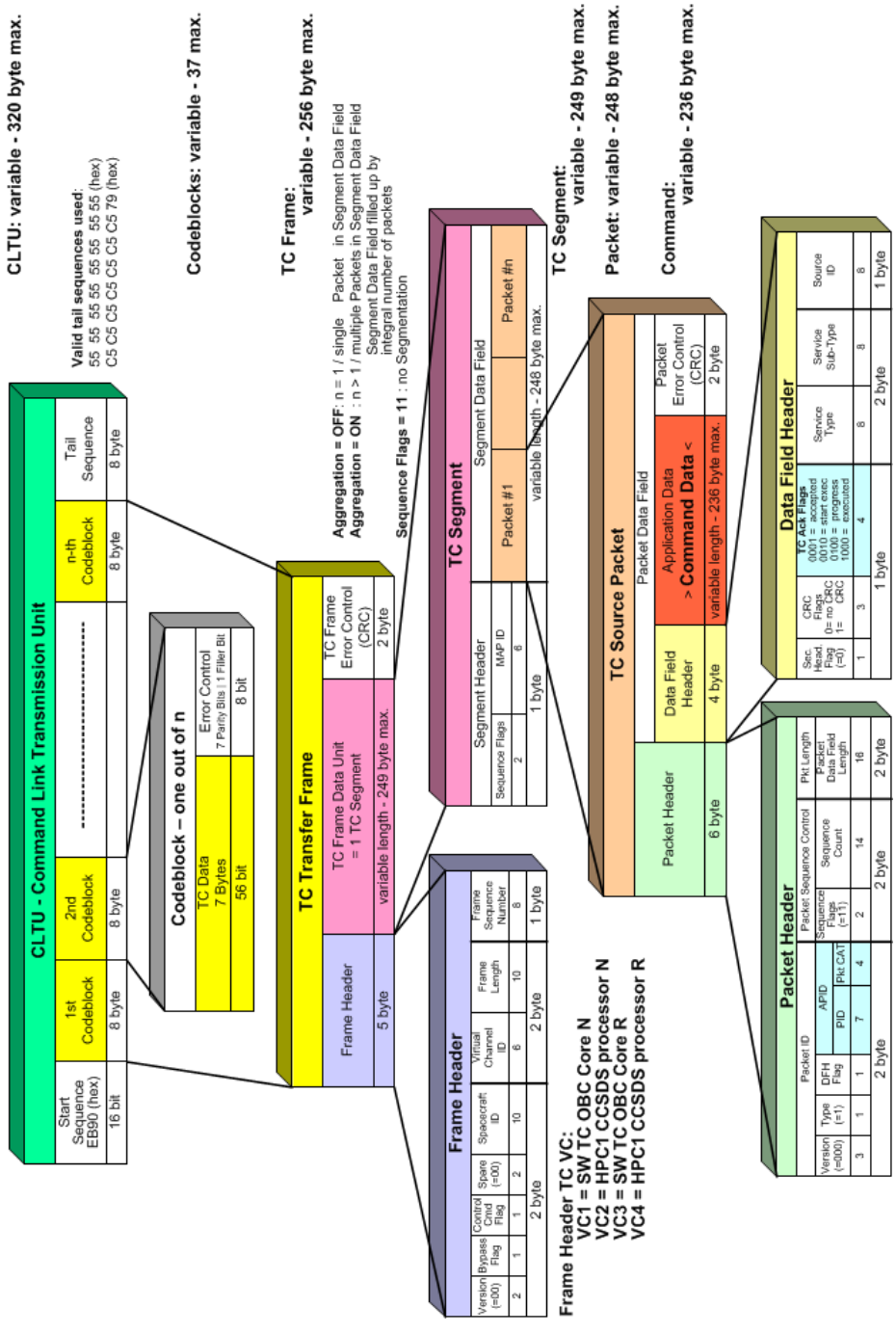

Figure 3.1: The Structure of a CLTU.

nications mean that packets sent must also be placed into frames using the PUS protocol.

3.1.2 Frames, CLTUs and CADUs

A frame is the extra data that surrounds the core data to be transmitted and aids the process of networking by ensuring that data is delivered to the correct place at the correct time. The most common analogy is a picture frame (the data frame) that contains and maintains the structure of the painting (the data). Therefore, it could be considered that the packet already contains a frame (the packet header and data frame header), however, within the PUS protocol this nomenclature is not used. Within the PUS, two different protocols are used, one for the uplink and one for the downlink of data. For uplink, the following frame-packet hierarchy is used:

- data is contained in a TC source packet,
- the TC source packet is contained within a TC segment,
- the TC segments are contained within a TC frame,
- the TC frame is contained within a CLTU.

For downlink, the following frame-packet hierarchy is used:

- data is contained in a TM source packet,
- the TM source packets are contained within a TM frame,
- the TM frame is contained within a CADU.

The various levels of frames stated above contain many important pieces of information that define the spacecraft ID, the order that various frames and packets should go in, error control and many other important flags. However, none of the information above the TM packet is given by the OBC and it cannot understand any data provided above the TC frame. Therefore, between the antenna of the satellite and the OBC there is a component called the CCSDS board that is responsible for decoding any incoming messages from the CLTU level to the TC frame level and encoding any outgoing messages from the TM frame level to the CADU level. This will be explained in the next subsection.

3.1.3 The CCSDS board

As stated above, the CCSDS board is required to encode/decode data from the CLTU/CADU level to the frame level. In a complete satellite, this operation would be performed by a CCSDS board that was bought and integrated into the satellite. The FLP is currently

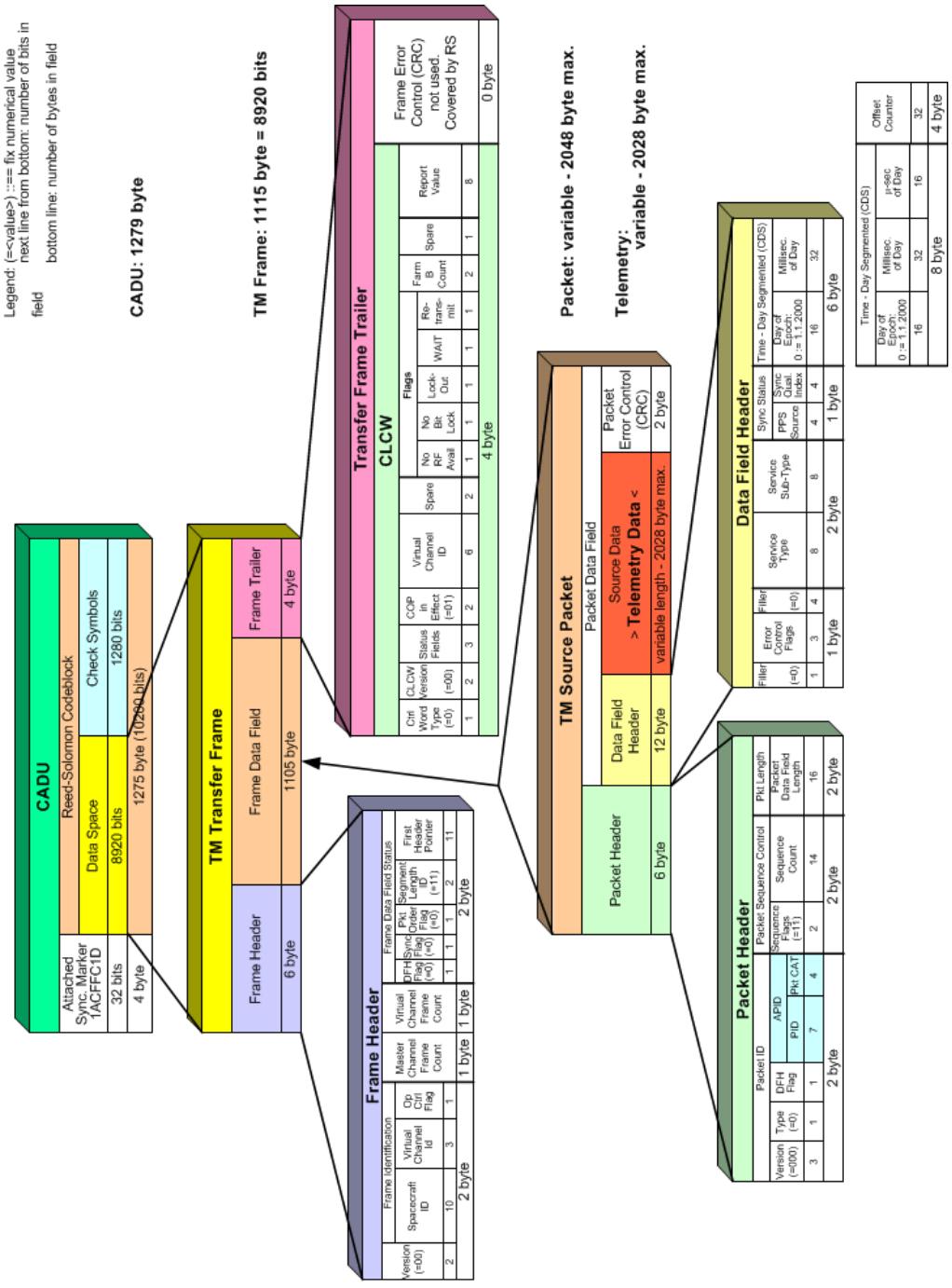

Figure 3.2: The Structure of a CADU.

under construction however, and prior to the integration of the CCSDS board the OBC received information via an Ethernet wire that provided the data at the packet level.[18]

The satellite is still in development phase C/D which verifies the the components and integrates them into the system, specifically, at the time of writing, the project is in the hybrid verification of infrastructure step.[3] This means that each component is in the process of being simulated, tested with the overall project and then integrated. Therefore, the CCSDS board was simulated in simTG before connecting it to the CCS5 and the OBC. As this is the infrastructure that was available when the HVU was integrated, this is what the author used when commanding and controlling the PCDU.

3.1.4 The CCS5

The CCS5 is a piece of software designed and maintained by Terma to be the central system for creating and sending commands to the OBC and then later for receiving telemetry. It comes with many built in functions and is designed to streamline the process of building a ground station as it is already compliant with ESA Mission Control Infrastructure. One its features is the ability to write commands, and FLP already has a database of useful commands from the original Flying Laptop. These commands include a large number of flags which determine the nature of the command, including whether the command will be sent as a packet or CLTU. When the OBC of the FLP was communicating with the ground station via Ethernet cable, these flags were set to packet level. The internal settings within the CCS5 software were also set to work at packet level. Therefore all these settings and flags had to be found and changed before the CCS5 was able to communicate at CLTU level.

3.1.5 Types of protocol

Within the umbrella of the PUS there are many additional specific protocols. These often only make very minor adjustments to the protocol, such as an alteration of the error control, and are often chosen as drivers for the CCSDS board and CCS5. In the broadest sense, the CCS5 uses two main groups of these protocols, these groups are the Mission Control System (MCS) and Electrical Ground Support Equipment (EGSE) protocols. The EGSE protocols are mainly for the early testing phase and verification that communications between the OBC and ground work. Examples include Eco Data Exchange Network (EDEN) and Command-and-Control (C&C), the main point of similarity is that they both work at the packet level. As the project progresses and communications need to be at a higher frame level the command and control will be switched over to one of the MCS protocols such as Network Information Service (NIS) or Space Link Extension (SLE) in order to send data within frames. For the FLP the NIS protocol was

chosen, however this may be changed further on in the process.[17]

3.2 SimTG

When simulating the CCSDS board, a program designed by Astrium called simTG was used. SimTG has two base models that the user can choose from. These are synchronous and asynchronous models. Both systems have a constructor, an initialisation function and a destructor, which perform the obvious functions. They also have a step function which is where the main code is called and executed. The main difference between the two models is the method by which the step functions are called. When defining how a model is to be used, the designer of a synchronous model must define a frequency that the models step function is called. Then when the simulation is started, after the models have been initiated, the model will cycle through the step functions of any models running at the specified frequency. On the other hand, the step function of an asynchronous model will only be called if there is a change to one of the inputs of the model.

Additionally, simTG allows the user to design a system including its inputs, outputs and primary functions of the model in a graphical way rather than building the model in code from scratch. Once these inputs, outputs and functions have been decided upon and written into the model, the model is built and all of the basic code is generated automatically by simTG. This means that the user can focus on building the relevant code for the model and linking it together.[10]

3.3 The new ground to On-Board Computer link

In the fully operational FLP, the CCSDS board would be directly linked to the OBC via SpW. However, as the satellite is not complete and the CCSDS board has not yet been purchased or integrated, this route is not able to be used. Currently, the outgoing data from the OBC travels via SpW, and therefore using RMAP as a protocol, to a router and then into the SRR. The SRR then wraps the whole frame (including RMAP) inside a Transmission Control Protocol (TCP) frame. This then allows the data to be sent via Ethernet cable to the desktop computers.

When the desktop receives these frames, they are still wrapped in the RMAP frame and therefore unable to be processed by any simTG models, which are not expecting any specific model. It was therefore necessary for the first FLP satellite to produce a program to recognise and remove the RMAP protocol. The program that completes this is called IOBroker. As the TCP frame was removed upon entering the port, the socket

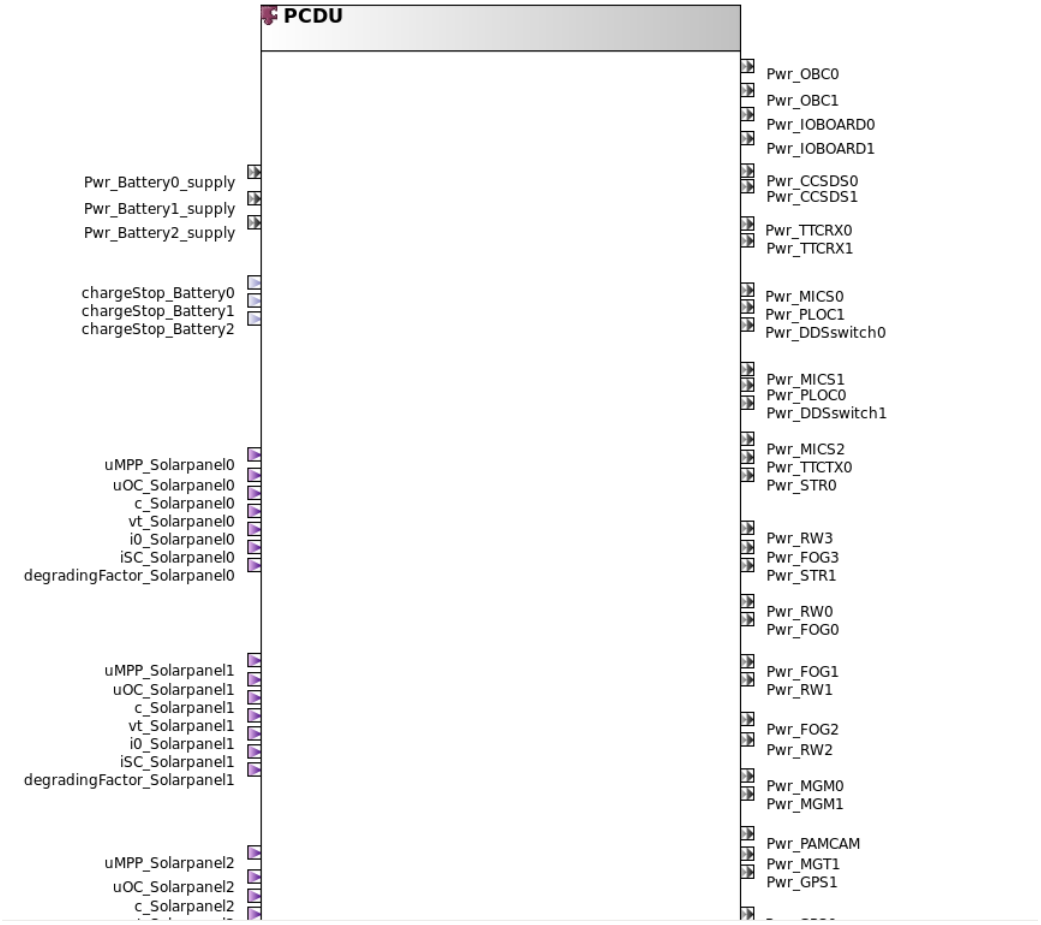

Figure 3.3: The PCDU Model within SimTG

is attached to IOBroker, which strips the RMAP frame away and leaves the raw data packets that the models can understand. This process is of course done in reverse when the simTG models send data to the OBC. In normal operation, the removal and addition of the RMAP frames from the OBC would be done within the IO board or the individual components, depending on the design choices of the satellite.

3.4 The Model of the CCSDS Board

Therefore, the full model of the CCSDS board has two main parts. The first is a socket program that includes three ports, one for TC, one for the TM and one for the monitoring information received from the OBC. These three ports were therefore defined as

the ports that the CCS5 connects to. Importantly, this socket must be a non-blocking socket as otherwise the entire simulation of the spacecraft would stop functioning while it waits for the socket to receive a message. With that in mind, the main set up of the socket program was placed within the initiation function of the model so it can be completed whenever the program is run. The model then uses an if statement and the select function to check if there is any data on the line from the CCS5. In the case that there is data, it calls the relevant function (TC, TM or mon) to read the data from the port and complete any relevant actions such as placing that data on the buffer.

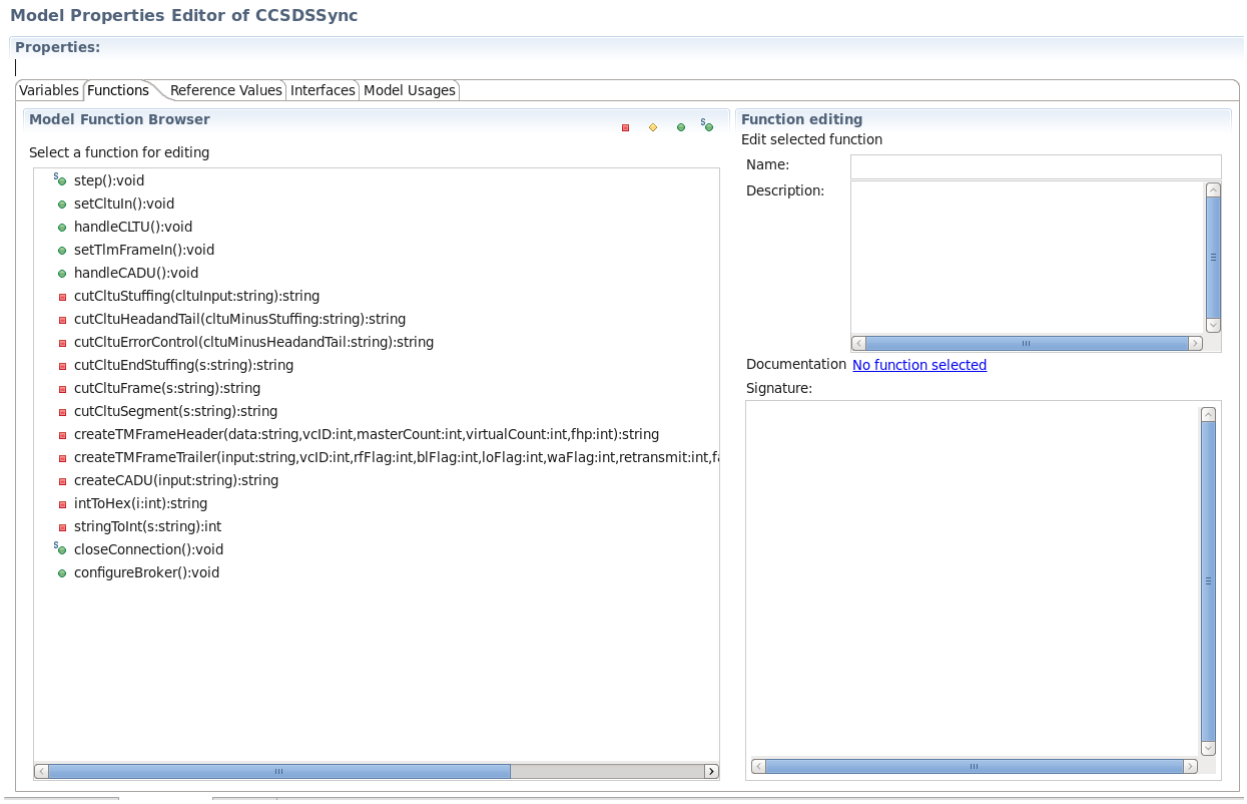

Figure 3.4: The Main Functions of the CCSDS Board shown in SimTG

The second main part of the model is the ring buffers. This was originally implemented within the IOBroker program, which also simulates the I/O board, as a method of storing instructions moving to and from the OBC. This class comes in useful for the CCSDS board model as well. It was therefore decided to use the class four times within the model. The first use is when the socket program writes the information on the TC socket onto the buffer "io2ccsds4". This information is then read within the CCSDS model. When the

CLTU is decoded in the model, the information is placed on the "ccsds2broker" buffer in frame form. This information is then read by the IOBroker and sent on to the OBC via the SRR. When the OBC needs to send telemetry, it sends the information via the SRR to the IOBroker program. IOBroker then places the packets sent onto the "broker2ccsds" buffer. This is then read by the CCSDS board model. After the data has been encoded into a CADU, it is written onto the "ccsds2io4". This system of using ringbuffers ensures that no data is lost between the models and components while also reducing the complexity of the model by storing and sending data in a similar way between the different parts.

Inside the model of the CCSDS board, the step function calls two functions, setCltuIn() and setTlmFrameIn(). It requires two functions as on the TC side the CLTUs need to be decoded and on the TM side CADUs need to be encoded. Both of these things need to happen simultaneously. In both cases, the model checks to see if there is any data within the input buffer before proceeding with any encoding or decoding. This ensures that if there is nothing for the model to do it does not waste processing power or risk becoming stuck in a loop. From there, the code begins to differ.

The handleCLTU() function reads all of the messages within the input buffer and places it into array. Writing the data in hexadecimal form, a CLTU always begins with the header "eb90" and ends with the tail "c579". The function then counts the number of headers and the number of tails. If these numbers are equal to one another and to the number of messages received from the OBC, the model can be confident of the number of CLTUs. From there it finds the position within the array of the first header and tail and copies everything between these positions into a new array. This array is transformed into a string to allow easier data handling within the simTG defined functions. After the first CLTU has been processed, the program cycles through any remaining CLTUs in a similar manner until there are none left, when it exits the loop.

To process the CLTUs, the string is passed through several different functions, each removes a specific part of the CLTU from the string, starting with any stuffing before the header or after the tail. From there it removes the header and the tail sequence. Next, it removes all the error control at the CLTU level, before cutting any stuffing placed at the end of the frame to ensure the CLTU was the correct length. This staged approach was used partly to aid in the debugging process, but more importantly to allow anyone who works on the model after the completion of the project to quickly and easily leave parts of CLTU in the output message or add more by seeing short examples as a model for their changes. The string left after the CLTU has been cut down is the frame that the OBC is able to read. This is then converted into a byte array and written onto the output buffer. From there the program cycles until all CLTUs have been decoded and written onto the output buffer.

The handleCADU() function works in a similar way initially, copying all the packets sent from the OBC to the input buffer into an array. As the data field of packets within a CADU is always 1105 bytes and the amount of data recieved from the OBC is variable the simulation must process an unknown number of bytes into a fixed size of buffer. Therefore, a buffer of 1105 bytes is created and all elements set to 0. Then, if the total data field is more than 1105 bytes, the first 1105 bytes are cut from the original array and pasted to the new array. This is then processed as detailed below. The process of cutting, pasting and processing is then cycled until the amount of data left in the orignal array is less than 1105 bytes. Then, any remaining data is cut and pasted, and as the new array is always reset to zero at the start of each cycle, any remaining space appears as zeros. This is the format the ground station would expect.

The new array containing a single CADU of data is then converted to a string, the frame header and trailer are computed and added before the header of the CADU is added to the front of the string. Based on discussions with the designers of the CCS5, it was assumed that the last part of the CADU, the check symbols, are unnecessary for the CCS5 to understand the CADU and therefore have not been incorporated into the model. However, due to the modular design, if this becomes necessary at a later stage it would not be a lengthy or complex task to build a function to do this and add it to the model. After the CADU is constructed, it is converted into a byte array and written onto the output buffer as before.

The final point of complexity is due to the variable size of packets. This means that as the size of the CADU is set, a packet being cut in half is a regular occurrence. The solution is already detailed in the PUS protocol. Within the frame header, one of the fields is the "first header pointer". This states the position of the first byte of the header of the first packet. The CCS5 can then use these to attach any cut off data to the correct packet. In order for the CCSDS board simulation to do this, a simple loop that checks each byte in sequence and identifies the packet header was used. The loop breaks and returns the position of the first packet after it finds it. This is then included within the frame header.

3.4.1 Connecting to IOBroker

The connection through the IOBroker to the OBC is required to wrap the telecommands that are sent to the OBC into an RMAP packet and unwrap the telemetry.[8] Within the overall final structure of the lab set-up of the FLP, the IOBroker will be unnecessary as any component that has an ethernet output will be routed through a smaller version of an SRR that at the time of writing is under development. However, currently as many of these components are simulated on a computer and the computers in the lab do not have

a SpW port, all telecommands sent to or from the OBC must be encoded to or decoded from RMAP packets via the IOBroker.

As this program was primarily designed to test the simulation of the platform components such as the PCDU and Magnetometers (MGMs), it begins by setting up a series of interfaces that an IO board would require to complete its function. The CCSDS board was not planned to be part of this, and therefore was not included. One avenue that was explored was to create a new interface for this purpose. This caused many problems which are beyond the scope of this thesis, but it is important to know it used the same interface and a work around was created for this. The program then continues on to set up any memory buffers into which any data entering or leaving the IOBroker from the components side is placed.

After the initial set up, the programme opens up a connection port that waits for a simTG model to connect to it. This connection is completed when the simTG model places specific commands on the input buffer of IOBroker. IOBroker has several modes that it can use during input or output for different protocols such as ethernet or SpW. The input given when the connection is completed determines this protocol. From this point the focus will be on a single mode, SpW RMAP mode as it is the mode used to communicate with the OBC.

The program then links through the SRR to the OBC. Once the link is complete the program goes into a while loop. The function inside the while loop waits for the OBC to send a message to the IOBroker. When the message called a “WriteCommand” from the OBC is received, it will either be telemetry to be sent to ground, or a handshake for the CCSDS board, either way the packet is immediately forwarded on. The message is then checked inside the CCSDS board, discarded if it is a handshake and processed and sent to ground if it is telemetry.

Whilst this process happens in the CCSDS board, the IOBroker immediately sends a reply to the OBC to confirm the receipt of the "WriteCommand". When the OBC receives the reply, it quickly sends a "ReadCommand". This command then requests any telecommand data that the CCSDS board has. This causes IOBroker to check the buffer containing processed telecommands. If it contains any messages, the IOBroker then pushes these messages down a data line in the form of a "ReadReply". If no messages exist, a "ReadReply" is still sent to the OBC to notify it of this.

At the time of writing, the OBC messages were performed by a dummy program as the main code was not in working order, however this dummy programme used much of the original code and was contained on the physical OBC. As the connection between the IOBroker and OBC is run through a socket, and the socket programs should be the

same in the final software, any issues in the final connection are likely be minimal.

CHAPTER 4

Data Transfer to the PCDU

4.1 Overview of the Link

The main focus of the thesis is the command and control of the propulsion system. The previous section looked at how the CCS5 ground station was linked to the OBC via the simulated CCSDS board, IOBroker, the SRR and routers. This section will cover how the OBC is linked to the Electric Propulsion System (EPS) via the PCDU and a HVU.

It is important to note at this point that the full chain from OBC to EPS is impossible to test at one time. The OBC is housed in the room that contains the FLP test bench. In a completely separate building, there is a laboratory that houses a large vacuum chamber. The only place that the EPS can be tested is inside this chamber because turning it on outside of vacuum conditions could damage it or be a safety concern. Obviously, in order to test the full chain both of these parts need to be linked. However, as the OBC is run via a desktop in the FLP testbench and is linked to CCS5, simTG and several other components of the testbench that are not required in the understanding of this thesis, moving it is prohibitively complicated due to the networking issues involved. Obviously, the vacuum chamber cannot be moved either and therefore it is impossible to utilise a single link from OBC to EPS at this point in testing.

Fortunately however, it is possible to move both the PCDU and HVU, and more importantly, it is possible to command the PCDU via another method other than the OBC. In order to effectively test the PCDU, a piece of software is provided by the producer of the PCDU, Vectronics, that can command and control the PCDU from another computer via a Universal Asynchronous Receiver-Transmitter (UART) cable. This computer was therefore chosen to be a laptop so it can be physically moved. This means that as the laptop commands and controls the PCDU in the same way as the OBC, it is possible to split the chain into two overlapping parts.

The first part starts with the laptop, and links through the PCDU to the HVU and eventually to the EPS. The second links from the OBC through the PCDU to the HVU. As these two chains are overlapping, they can be carefully tested and it is safe to assume that if the outputs from the HVU are the same, the entire chain would work.

At the time of writing, there is no customer lined up for the FLP so many components have not been finalised. This is most true of the propulsion system as the original FLP did not have such a system, and the precise system used will heavily depend on the mission planned. Therefore, to provide context on the design decisions available to the project and customers, this section will begin with a brief description of some of the main types of propulsion systems currently available and in development. It will then give a brief overview of a PCDU and its functions before detailing the specifics of how this relates to the propulsion system and the setbacks that occured when connecting these components.

It is important to note that the PCDU used in the testbench is not a flight model and its specific design will change dependant on the requirements of the payload. The author will detail the PCDU to EPS chain via the HVU and the setbacks that occurred when connecting these components then look at how to command and control the PCDU both directly to test the PCDU via a laptop and to connect it to the OBSW. It will then detail how this new connection affected the OBSW and ground station before finally briefly looking at the two data chains tested.

4.2 The Electric Propulsion Unit

As there are not enough particles in space to produce a practical amount of friction for a spacecraft to propel itself with, any propulsion forces must be made using fuel that originates on board the satellite at launch. There are several broad categories of these propulsion systems which will be briefly described below. The main idea of the propulsion system is to expel particles, usually as gas or plasma from an exhaust. This provides a force on the satellite equal and opposite to the force on the gas. The main component shared by all thrusters, aside from a fuel tank, is the exhaust. This can be shaped in many different ways, and the different shapes each have advantages and disadvantages.

The most obvious variable is in the amount of force a similar amount of gas can provide with the most important variable being the shape of the exhaust plume. In space, if the emitted gas remains too close to the satellite, it can be caught in the gravity well of the satellite and be attracted back. This gas can then coat the outside of the satellite, reducing the efficiency of solar panels and ruining the optical surfaces of instruments. Therefore, the plume shape is designed to minimise this.

4.2.1 Cold Gas Propulsion System

The simplest form of propulsion in space is the use of cold gas. At its most basic, a large container of pressurised gas is placed inside the satellite. This is connected to at least one valve, that leads to the exhaust of the thruster. When propulsion is required, the valve is opened and some of the gas is forced out through pressure. The choice of gas, amount of propellent and size of container all play a part in the exact thrust provided by the thruster.

More complicated still is that as more gas is released, there is less gas in the container and therefore less pressure. This leads to a lower force being provided as the mission goes on. This must therefore be accounted for in how long the valve is opened. For a more active control of the thrust generated, the ability to vary the amount of gas released is required. For simple control, this can be done either by having multiple valves that lead to the same exhaust for a simple control. Alternatively, variable valves for a more complex control can be used. Most cold gas propulsion systems also have multiple exhausts for redundancy and additional control of the direction of thrust.[1]

4.2.2 Hot Gas Propulsion System

A hot gas propulsion system adds additional complexity. In this the system multiple types of gas, often in multiple containers, are mixed. As they leave the main fuel tank, they are burnt. This provides the gas additional thermal energy and therefore provides additional thrust in comparison to a cold gas system. There are many different hot gas propulsion systems that provide varying amounts of extra thrust. The main difference is in the choice of propellants and the choices need to balance the amount of extra thrust with the risks and additional costs that the different mixtures create. If the propellants are very corrosive it can be more dangerous to install them on the satellite or if the two propellants are highly volatile this could cause explosion risks to the satellite.

With the addition of extra thrust, it seems obvious that hot gas propulsion is superior to cold gas, especially when the propellants can be chosen to minimise risk if desired. There is, however, a major drawback to this system, added complexity. In order to successfully ignite the fuel, additional components are required. These vary considerably depending on the propellant used and the exact design, but often include heaters for the combustion chamber, injectors and a method of ignition unless hypergolic propellant is used. Hypergolic propellants are a combination of propellants that spontaneously ignite when they come into contact with each other. While this reduces the complexity of the system, hypergolic propellants are often highly toxic or corrosive, making them difficult to handle.[6]

4.2.3 Electric Propulsion System

The final type of propulsion system we will look at is the EPS. This is still a young technology and is still being tested extensively. However, as it uses less fuel than the other two systems, it is likely to be used in the future, especially for station keeping in satellites. Although it is the most complex system in terms of command and control, it is the system that will be integrated into the FLP. It seems likely to be the preferred propulsion in the future and it provides the project with the opportunity to manage this level of complexity while future-proofing the satellite.

The principle behind an EPS is that by placing an electric charge on a gaseous propellant, it is possible to use the coulomb force to accelerate the gas. This causes the gas to leave the thruster with very high exhaust speeds and therefore an EPS has a much higher specific impulse than other types of propulsion systems. By utilising this high specific impulse, it is possible for the satellite operate for significantly longer than its counterparts whilst using the same amount of fuel. Its main limitation is the amount of power that can be supplied to accelerating the ions. As satellites rely on solar power, an EPS typically has a much lower thrust than chemical propulsion systems. This means that while the technology cannot be used in rocketry, it can provide a small thrust over long periods of time.

Despite all systems being different, the main steps that most systems share start by releasing a small amount of gas via valves. This gas is then ionised in some way. In the case of the High Efficiency Multistage Plasma Thruster (HEMPT) that Airbus is currently developing, this is done by passing the propellant through a stream of highly energetic electrons. From there it is passed into a an accelerator. This works by passing the ions between two highly magnetic plates. The ions are often then ejected from the thruster, though occasionally some thrusters use nozzles to shape the plume or better utilise the thrust. The acceleration uses electromagnets, and as the ionisation of the propellant uses a lot of power it is easy to see why an EPS uses more power than the other two propulsion systems we have discussed.[11]

4.2.4 Propulsion System Selection

The choice of a hot gas propulsion system, a cold gas propulsion system or an EPS heavily depends on the mission. Some of the factors that affect this decision are:

- Budget available,
- Complexity tolerance of a design team,

- Design and build time available,
- Reliability requirements,
- Expected time scale of the mission,
- Mass and space available on the satellite,
- Type of orbital manoeuvres expected of the satellite,
- Contact frequency between ground and satellite when in orbit,
- Satellite orbit.

All of these affect the decision of which system to use. Below, the author will give two examples of possible missions and suggest which system they might use and why.

Example 1 - A high budget, military, earth observation satellite in Low Earth Orbit (LEO).

In this example, the primary consideration is likely to be reliability. Naturally, a high budget satellite failing in orbit it is a much larger issue than a low budget satellite. It is therefore more likely that the propulsion system will be chosen to ensure that it does not fail. Immediately, this suggests that an EPS would not be used as they are more complex and less tested than the other two systems.

The next consideration relates to the orbit and mission. As it is in LEO, the propulsion system is likely to be used to extend the lifetime of the satellite, and as its mission is to not to change the orbit there is no requirement for a high specific impulse. These two factors point towards the use of a cold gas system as there is no major requirement for a hot gas system. As cold gas systems tend to be cheaper and less complex than hot gas systems, it is more regular to use cold gas unless the mission requires the high specific impulses provided by hot gas.

The final consideration is the length of the mission and the budget. Military satellites often have low lifetimes as their orbits tend to be very low in order to get the best possible photographs of an area. The final choice between hot and cold gas would depend on how much longer the mission must last for after it would naturally burn up in orbit. The longer the mission needs to last, the more likely it is that hot gas would be used as the additional force provided for the same amount of gas becomes vital to keeping the mass of the satellite within reasonable limits.

In this case, the author assumes that the satellite does not need to be in orbit long enough for a hot gas propulsion system to become necessary. Therefore, to reduce mass,

complexity and expenditure whilst increasing the reliability of the propulsion system, a cold gas system is chosen.

Example 2 - A low budget, scientific satellite in Medium Earth Orbit (MEO).

In this example, the primary concern is the orbit. In MEO, a propulsion system is not necessary to maintain the orbit of the satellite as there are not enough particles to slow the satellite significantly over the lifetime of a mission. Despite this, almost all satellites in MEO still use propulsion systems. There are three main reasons for this. The first is if the mission requires changes in the orbit of the satellite. The second is to avoid other satellites or debris whilst in orbit. The third possible reason is to de-orbit at the end of the mission, though it is possible to do this without a propulsion system.[9]

In this example, the author assumes that due to the scientific nature of the mission the satellite will need to perform orbital manoeuvres both to change the orbit and avoid objects in space. While object avoidance can be performed with low specific impulses, this requires greater awareness of the surroundings and faster responses from ground than avoidance performed by high specific impulses. Orbital manoeuvres also become highly complex without a high specific impulse. Both of these factors heavily indicate that a hot gas system would be used.

Scientific missions are often under less time pressure during the design and building phase than commercial satellites, and therefore a higher complexity can be accepted. The low budget of this mission will also cause a low mass to be very desirable, and these two factors together also make the hot gas system very attractive.

Therefore, assuming the customer has the capital required to sponsor a hot gas system, this is the system that is most likely in this case.

Summary of Propulsion Systems

In both the cases above, small changes to assumptions made could completely change the choice made at the end. Additionally, the author has made the implicit assumption that these satellites are not designed to test new components on board and have no financial incentive to do so. This incentive is not uncommon as new systems need to be tested and can be bought at reduced cost or become their own "payload". The FLP currently has no customer. Consequently it becomes prudent to plan for the most complex system possible and scale back the complexity later if the customer does not require it.

It was therefore decided that at this early stage of development that an EPS would be

integrated as it has the most complex command and control out of these three systems. This means a potential customer can select any system and without concern that additional problems will arise in the platform later. The EPS system used is a HEMPT that operates in the micronewton range. It has been designed at Airbus DS Friedrichshafen and was previously simulated in SimTG. It is shown in figure 4.1.

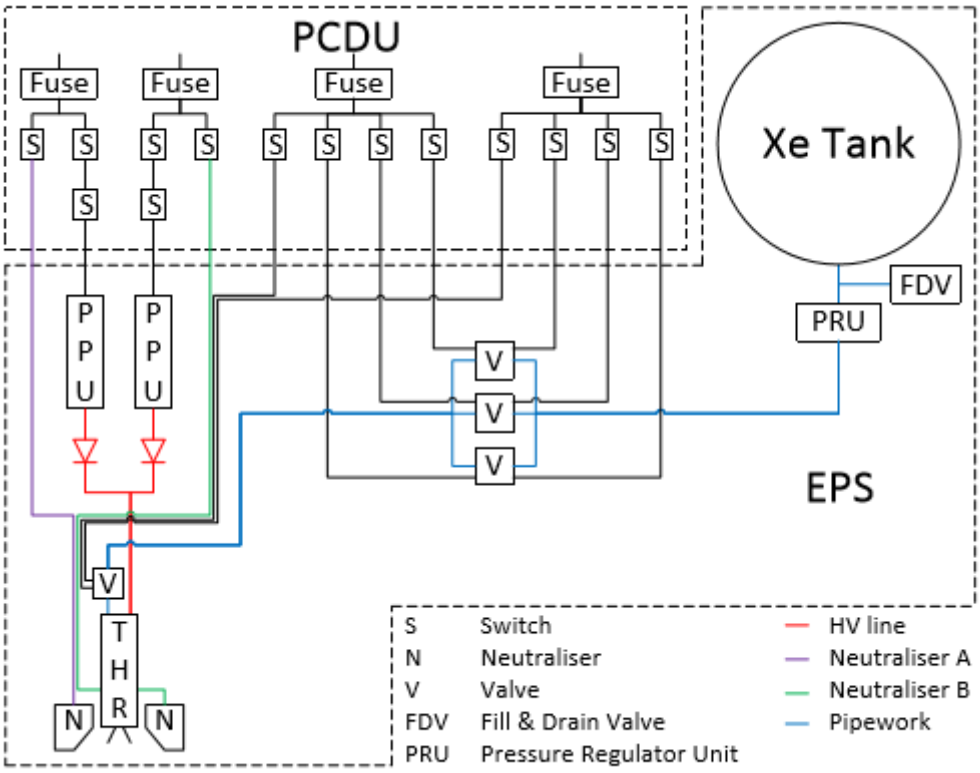

Figure 4.1: The System Architecture of the EPS.

4.3 The Power Control and Distribution Unit

The PCDU is one of the most important parts of the Power Supply Subsystem of the satellite. The remaining parts of this subsystem typically include the solar arrays, batteries and a power-bus. The FLP utilises all of these, and specifically uses an unregulated power bus in its current form. The author will focus on the PCDU as it is all that is needed for this paper. The PCDU is primarily used to regulate, control and distribute

power between all the other subsystems of the satellite. It does this by monitoring the power-bus and in many ways its most important function is the protection of this bus. It performs these functions through a series of fuses, switches and relays. It is important to note here that as the fuses on a satellite in orbit cannot be replaced, any references to fuses in this paper and its primary sources refer to Latch-Up current Limiters (LCLs). Additionally, the FLP utilises Combined Data and Power Management Infrastructure (CDPI) and therefore any system failure handling and reconfiguration is also included in this system.

While on ground, and at this early stage in the testing process, the batteries and solar arrays have not been purchased and therefore, without these components the PCDU is powered using a power supply with variable current, potential difference and power limits. This supplies the power required by the PCDU, however unlike the flight case, the power to the PCDU will be very stable. This is not an important issue for this thesis, but will naturally need to be checked at a later point in the project.

Additional to the myriad of sensors required to properly regulate the power bus, the PCDU in the testbench has 26 fuses, 76 switches and 2 relays. These are wired to 13 D-Sub sockets of varying numbers of pins that provide all the power supply inputs and outputs for the PCDU including but not limited to those connections required for the solar panels, batteries, the OBC, all parts of the ACS and the heaters. At the time of writing, there were no dedicated pins within the PCDU for an EPS as this had not been confirmed when the test bench PCDU was purchased, though the flight model will be altered to include this.

The PCDU is designed in such a way that if the input voltages ever exceed a specified maximum, in the case of the PCDU used on the test bench 29 Volt, the entire PCDU shuts down to prevent any damage through over-voltage. The pins are grouped onto individual switches that provide power to a single component. If the total current leaving through a single switch exceeds a the maximum current value, a value that has a default amount but can be changed at will by the OBC between prior set limits, then the switch is automatically closed, again to prevent damage. All components, and therefore switches, are grouped onto a series of fuses and if the total current moving through any of these fuses exceeds a value set for each fuse individually, the fuse breaks and must be reset. Once again, this is there to protect any sensitive electronics on board the satellite from electrical damage.

All of these events have different reactions that are determined by Failure Detection, Isolation and Recovery (FDIR) and the response varies from waiting a short time, checking any sensors along that chain and attempting to turn the switch back on to a full shut down of different systems and the satellite moving into safe mode until contact with

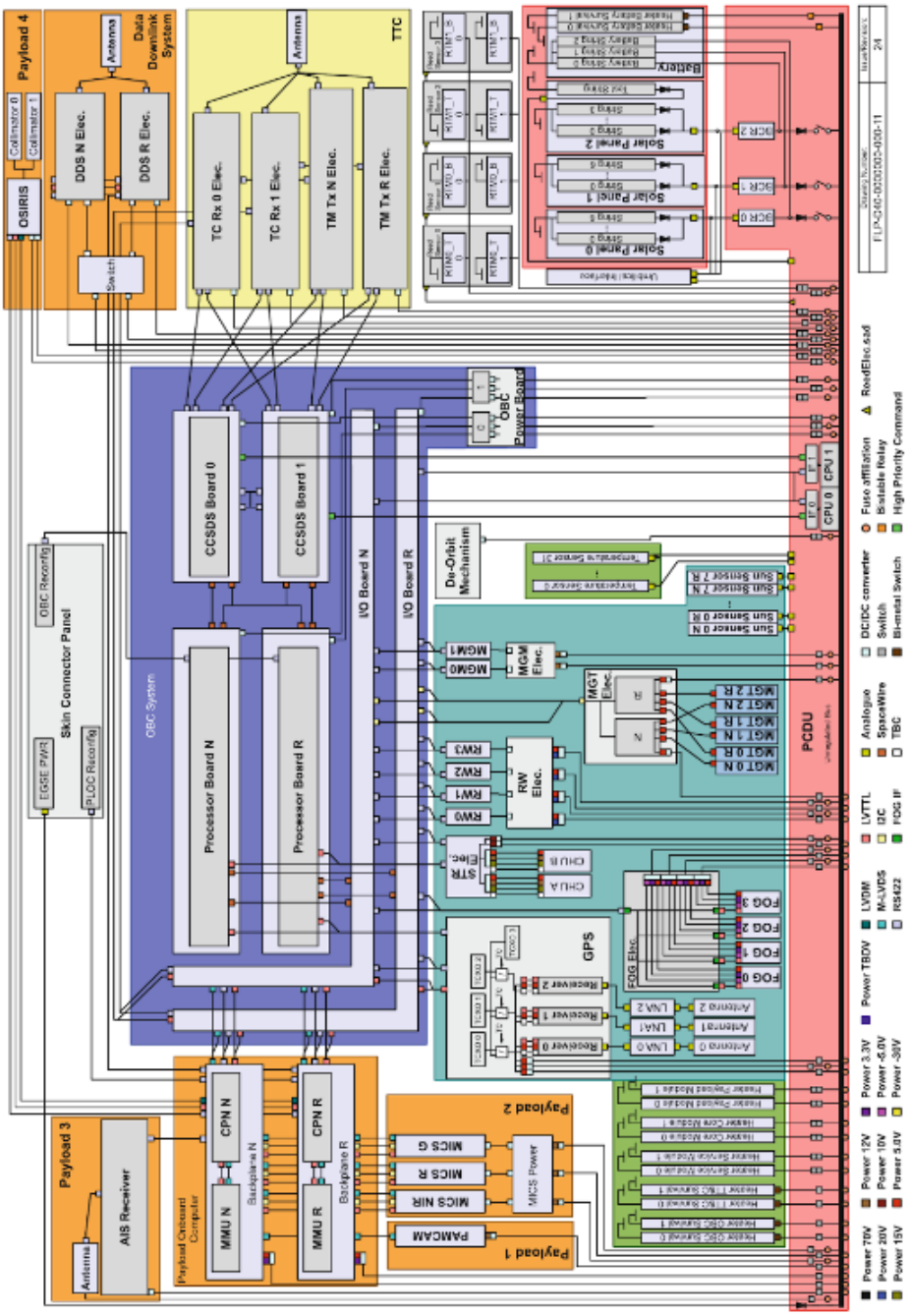

Figure 4.2: The Electrical Diagram of the PCDU

the ground is established, a full investigation is undertaken and the satellite instructed to begin operating normally again, with or without any changes. Naturally, the severity of the response depends on the nature of the event, and the relative importance of the

systems that triggered the event.

As the PCDU powers all systems on the satellite, including the OBC, it is in the relatively unique position within the satellite of having to be able to operate, and if necessary restart itself, without any guidance from the OBC. This means that if it ever loses contact with the OBC or needs to restart either component for any reason, it needs to have the ability to do so autonomously, and in the worst case instruct the entire satellite to move into safe mode if it cannot regain contact. This means that it requires its own computing and a highly robust FDIR process to ensure that the satellite is safe.

The Modes of the PCDU

There are six basic modes that the PCDU can be placed in, each is used for different situations, has different defaults for the switches, fuses, relays and any other options, and has the ability to move to specified other modes. This system is used primarily to protect the power bus, in the implementation of various FDIR events and in the early testing stages of the satellite in orbit.

The PCDU always begins in the same mode, "None", when it is turned on. This means that all switches are off, and is never used in normal operation. As soon as the satellite is attached the the launcher, it is placed into "Shutdown mode" which is also known as "Launch Mode". This is because when the satellite is launched, the usual requirement is that as few components as possible are powered. Therefore, in this mode, the only circuits that are active are the detector of separation from the launcher and any battery charge control boards. In this phase of the launch, the batteries are charged via an umbilical cord from the launcher. This mode can also be triggered by a power failure on-board the satellite.

Upon separation from the launcher, or a total recovery from a power failure, the first subsystem that is booted is the PCDU. This boot causes the PCDU to move into "Safe Mode", however it does not go there directly. In order to ensure that no damage to critical satellite systems occur, there are three intermediate steps. The first intermediate step is "Recovery Mode 0" and this can only occur if the satellite has separated from the launcher and the voltage received through the main switch is at least 21.5 Volt. From there, the PCDU attempts to move into "Recovery Mode 1", which can only happen if the State of Charge (SoC) is at least 45%. Next, the PCDU moves to "Mini Ops Mode", which is blocked if the OBC and Telemetry, Tracking and Command (TTC) survival heaters have not been receiving power for at least 5500 second. So far, we have been talking about the modes of the PCDU, which is a subsystem. However there are also system modes within the entire satellite. One is the "Safe" Mode" and it if it is triggered, it requires that all subsystems, such as the Power Supply Subsystem (PSS), are in the

correct, specific mode. "Mini Ops Mode" is the subsystem mode within the PSS that is required when the satellite system is in "Safe Mode".

Each of these modes iteratively powers more heaters, components and subsystems after checking the appropriate sensors to ensure no part of the satellite is damaged. When the system is finally in "Mini Ops Mode", the heaters for the essential systems are all in operation, the OBC is powered, as is the AOCS and any appropriate components. However, usually the propulsion system will not be operated in this state and only the base circuitry will be powered and as the payload will not be in operation it is not powered either. The TTC subsystem will by default will be turned off, though unlike the payload, the OBC can be manually turned on in this situation in the event of the satellite making a pass over the ground station.

The final mode is "Idle Mode". Hopefully, this is the mode that the satellite will spend the majority of its time as this is the mode used when the satellite is operating normally. This allows all subsystems to be under the direct command of the controllers and device handlers in the OBSW and therefore be turned on or off as is necessary for the situation the satellite is in. The main situations are completing the mission, which often requires turning the satellite to point towards a target, communicating with ground or moving the satellite into a new orbit.

4.4 The Data Link to the Propulsion System

4.4.1 Controlling the Power Control and Distribution Unit

The testing of the PCDU can be completed using software housed on a laptop. This provides a Graphical User Interface (GUI) that displays all the potential commands a user can give to the PCDU in addition to boxes that display any received telemetry. The other main display from the GUI is a section that allows the user to view the hexadecimal version of all messages set to, or received from, the PCDU. This allows the user to accurately view how the PCDU sends and receives messages.

Prior to any commands being sent or received, upon opening the command and control software, the user must first connect to the PCDU. This is done using the first tab, which is automatically shown upon opening. Inside the "default interface" box is a button, to "Update Device List", when this is clicked, the drop down menu shows the device "VAYNYJ9H", which is the name of the PCDU. When this is selected, the user can connect to the PCDU and command and control can begin.

When the PCDU is turned on, it starts in the "None" mode. Therefore, after the tab

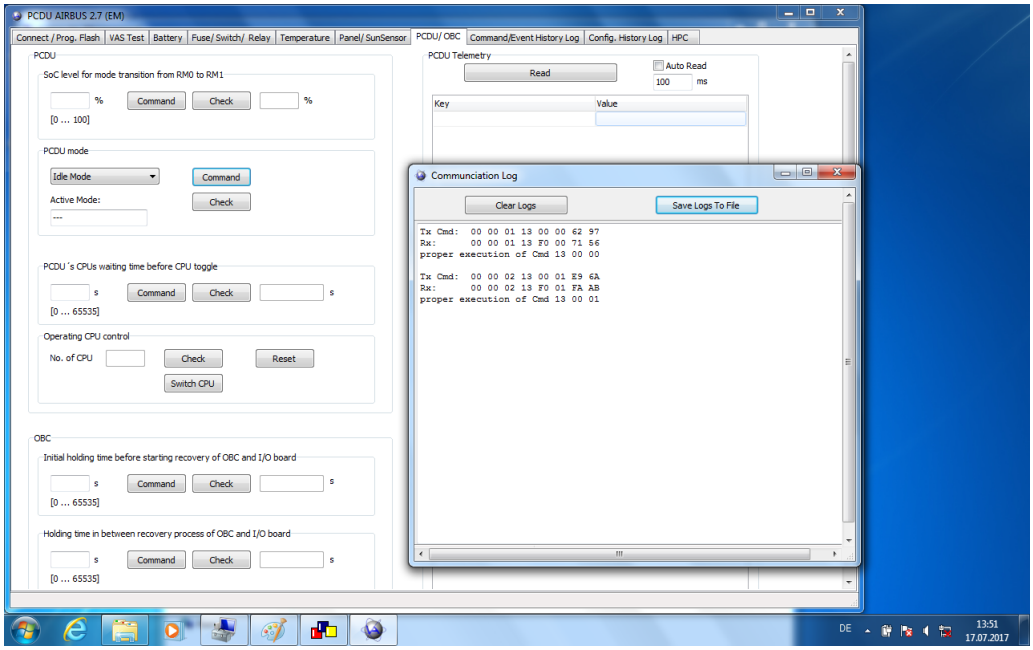

Figure 4.3: Screen Shot of the PCDU control GUI software

within the GUI is changed to "PCDU/OBC", the first command is usually to command the mode to change into "Mini Ops Mode". While it is possible to move the satellite into "Recovery Mode 0" and "Recovery Mode 1" manually prior to "Mini Ops Mode", this is done automatically and is therefore an unnecessary complication in the command and control of the PCDU. From there, "Idle Mode" is commanded to allow full control of the PCDU. It is possible to command the PCDU straight from "None" to "Idle Mode", but as this instruction would not be sent by the OBSW, best practice is to complete this step manually. Usually, the modes are checked between commands to ensure that the PCDU is operating as expected.

There are several other tabs that can be used to command and study the various parts of the PCDU, however, as only one is useful in the context of this thesis, the only tab to be discussed will be the "Fuse/Switch/Relay" tab. This tab houses the command and control of the inputs and outputs of each of the pins that connect to the PCDU. By selecting the appropriate fuse, switch or relay and clicking the appropriate command button, all switches fuses and relays can be commanded to turn on or off. Additionally, the maximum current through any switch or fuse can be set (within specified limits), the currents can be checked and the status (on or off) can be checked. This gives complete control over which components receive power and displays the current status of all components as required.

4.4.2 The Connection to the High Voltage Unit and Electric Propulsion System

As there were no dedicated pins for the EPS, and it required potential differences of several hundred volts to successfully ionise and accelerate the ions, intermediate steps were required. The maximum voltage allowed from the PCDU is 28 V, therefore to reach the potential differences that the EPS use, a HVU is required. This HVU was designed as part of an internship within the department that designed the HEMPT, and primarily consists of step up transformers that in total steps up the input potential difference by a factor of 10. Additionally, a series of Metal Oxide Semiconductor Field Effect Transistors (MOSFETs) are used to reduce the noise on the wire leading into the transformers. These reduce any dangerous instabilities in the current and potential difference across the load. This is necessary as if the current or potential difference fluctuate too much, especially when the HVU is switched on or off, it can damage both the EPS and the power bus. While it is possible to regulate this using the PCDU, it will cause many FDIR events and cause unnecessary disruption to the satellite, often at critical times such as orbital manoeuvres. It is therefore highly preferable to reduce these fluctuations, and this is possible by using the MOSFETs and other circuitry within the HVU.

The final main thing to note about the HVU is that it also incorporated a heater. This heater would not be used in normal operations, however, as the full EPS could not be used during the all various tests done on the HVU, it was crucial to include a load in the HVU that could dissipate the high voltages and power provided by the HVU throughout the test process. Most crucially for this thesis the full link from OBC to EPS cannot be tested simultaneously. This is because the test bench for the FLP is in a different building to the lab that houses the vacuum chamber that the EPS must be tested inside and without extensive re-networking it impossible to move the OBC. As both the HVU and the OBC can be physically moved, it is therefore necessary to have the option of routing the output of the transformers to a dummy load to allow the link from the OBC to be tested. The heater inside the HVU is this dummy load and during the test phase the outputs from the transformers are attached to the heater unless the link to the EPS is being specifically tested.

The next intermediate step is to utilise pins on the PCDU for purposes different to that which they were originally intended. Based upon the studies done on the HVU, it was determined that a voltage of close 30 V was required, additionally a current of up to 3 A was expected. As this is a large amount of power, at 90 W, this must be tested using pins that could handle that amount of current. Careful study of the Operation Manual for the PCDU showed that there were groups of pins that were suitable. All of the pins on the satellite give an unregulated voltage of 28 V, so that parameter was not a problem. The pins that serviced the "Bimetal + Heater OBC Survival 0 / Bimetal +

Heater TT&C Survival 0" and "Bimetal + Heater OBC Survival 1 / Bimetal + Heater TT&C Survival 1" had a maximum allowed current of 5 A. This placed it well within the parameters required for the HVU. It was therefore decided that the HVU would be tested using the pins for "Bimetal + Heater OBC Survival 1 / Bimetal + Heater TT&C Survival 1". This meant the focus is on fuse number 20 and switch number 51. This component lead onto the J6 connector, with the positive pins being numbers 6, 28 and 29, and the negative pins being 48, 49 and 50.

The first challenge faced when connecting the HVU to the PCDU was the testing of the switches inside the PCDU that would provide the potential difference that would be stepped up by the HVU. The switch is connected to 6 pins inside the J6 connector. This is a 62 pin sub-D female connector, and at the time the thesis started, we did not have a plug that matched the socket, or a way to access the individual pins/wires. For the long term solution it was decided that a breakout board for the 62 sub-D should be purchased, from there it is relatively easy to identify the relevant pins and wire them through to a banana plug. However it would take 2 weeks to arrive and until this was tested, the work connecting the PCDU to the HVU could not continue. A short term solution was decided upon and a 62 sub-D found in a workshop. From there, wires were soldered to the correct pins on one side, banana sockets on the other and insulated. This meant that a multimeter could be used to test the potential difference between the pins. This solution worked and the pins transmitted a signal to the correct plugs.

Figure 4.4: Banana Plug to sub-D Connector

Further testing presented a second issue. The next step after checking the connections in this new adapter was to check the outputs from the PCDU using a multimeter. When the switches were on, the expected result was returned. This result was a 28 V difference between each of the positive pins its negative counterpart. However, when the switch was off an unexpected result occured. Even when the switch was supposedly off there was still a 9 V difference. This was not documented in the operation document and therefore had to be investigated. It was theorised that the most likely issue was the lack of a load across the pins. This would mean that all energy would be dissipated within the multimeter and as it did not have an infinite impedance, this meant it drew a current and gave a false reading.

The obvious solution was to place a load across the pins. However, the amount of power leaving the pins was variable and the maximum was large enough that there were concerns that it would damage the PCDU in some way. It was thus decided to use the HVU as the load while implementing a slow and careful approach, testing each link in turn to ensure that no damage was done to either component. The HVU was chosen as this component as it was designed to handle the maximum outputs of the PCDU and was therefore the best option to check that the issue was caused by a missing load.

Testing the Full Chain

Figure 4.5: The OBC to HVU test chain.

Prior to the test, all appropriate staff were informed of the test to ensure that there would not be any unforeseen issues. Additionally, as the test was to utilise high volt-

ages, staff members trained in this had to be present. When this was arranged, the PCDU, laptop, power supply and any breakout boxes and connecting cables were moved in a modular fashion, and in some cases using specifically designed casing and antistatic equipment, to the laboratory containing the EPS and HVU. The author will focus on the PCDU as it is all that is needed for this paper. Instead, a single detailed description of how to appropriately set up this test assuming all the necessary pre-checks have been performed will be given. If further detail is required, the original set up of the PCDU was performed by another intern at Airbus and a set up manual is housed there.

- If the HVU has not already been tested, the switching patterns of the MOSFETs should be checked, and an external power supply should ensure that the HVU is fully operational to a value of 30 V.
- After the PCDU has been tested with no load (as detailed below), the HVU should be connected to the outputs of the pins.
- The restriction values of the power supply to the PCDU are checked to be 28 V, 3.5 A and 100 W. This is the maximum expected for this test, however if several other systems are running simultaneously, this may be too low.
- The output power of the power supply is turned on, and 20 s are waited to allow the PCDU to complete its boot up procedures.
- The command and control software on the laptop is used to connect to the PCDU and command it into "Idle Mode" via "Mini Ops Mode".
- The switch "Heater OBC & TTC Survival 1" is selected, as is fuse 20. For the remainder of this section, any mention of the fuse or switch refers to these.
- The current flowing through the fuse is checked, and as no power is moving through the switch, it should be less than 30 mA.
- Moving the PCDU into "Idle Mode" sets the switch to on. Therefore, it is commanded to turn off.
- The maximum output current of the switch is checked. If the output is not expected to have a load, this should be set to 50 mA. If testing of the HVU or EPS is happening, it should be set to 3000 mA.
- In the initial testing, the output voltage of the pins will be checked when the switch is on and off.
- It is this point that the PCDU should be turned off completely as detailed at the end of this section, the three positive pins on the PCDU connected in parallel to

the positive input of the HVU and the negative pins connected in a similar fashion to the negative input of the HVU, the apparatus switched off as detailed at the end of this section and the previous steps repeated.

- The switch should be turned off at this point. Then, the switch that power MOSFETs should be turned on. This MOSFETs switch is a physically separate switch that must be changed manually.
- The outputs of the pins can now be checked. This should show a value of less than 0.3 V and the output current of the power supply should be approximately 0.13 A.
- The PCDU can now be turned on, the voltage should change to approximately 28 V and the output current of the power supply should read approximately 2.32 A.
- This shows normal behaviour of the PCDU and its link to the HVU. It is this point that the entire apparatus should be turned off completely as detailed at the end of this section, the three positive pins on the PCDU connected in parallel to the positive input of the HVU and the negative pins connected in a similar fashion to the negative input of the HVU and the previous steps repeated.
- After this link has been checked using the dummy load, the entire apparatus should be turned off completely as detailed at the end of this section, the HVU connected to the EPS thruster using banana wires and the previous steps repeated.
- Under normal operation, the control system of the EPS will then show that the thruster is being successfully powered.
- To turn off the apparatus, first the switch must be turned off. Then the mode changed into "Mini Ops Mode", then "Shut Down Mode". Then, the GUI can be commanded to disconnect from the PCDU. From there, the power supply output can be switched off, as can the MOSFETs. Finally, the power supply can be switched off safely.

By completing the above steps, the author and team that studied the EPS were able to successfully show that the individual components, PCDU, HVU and EPS all worked. Additionally, and more importantly, it was shown that they could each be integrated into a single chain, which was operated via the command and control of the PCDU. As this is how a propulsion system would normally be operated, this showed that the chain could be successfully integrated into a satellite system. Therefore, the test was a complete success for all participants.

Further checks of the HVU

After the main testing of the chain, it was decided to make further checks of the HVU and PCDU, primarily to check the current profile of the HVU as it was turned on, but also

to check the manner in which the PCDU error control worked. This was based around manipulating the maximum current allowed through the switch by the PCDU at different points in the use of the HVU to see whether the switch was automatically turned off. This was done without the connection to the EPS to protect it from sudden changes in current.

Under normal and continuous operation, the HVU was expected to draw just under 2.2 A of current from the PCDU. However, for a short time directly after a switch is turned on, the component often draws far in excess of the normal operational current. This could be a problem on board a satellite for many safety reasons in addition to the possibility of the satellite being unable to provide enough power for this current spike. It was therefore necessary to ensure that the HVU did not produce this large current spike.

To test this, first the HVU was fully powered on using a maximum current of 3000 mA to ensure the link was still working as expected. Then, the switch was turned off, and the maximum current reduced to 1500 mA. When the switch was commanded to turn on, it did not. This was the response that was expected, as the HVU attempts to draw a larger current than this. This test showed that the PCDU worked correctly by automatically turning the switch off when the current drawn exceeded the maximum current allowed.

The next step was to test to see if the PCDU would turn the switch off while the HVU is running when the maximum current was reduced below the output current. To do this, the maximum current was increased to 3000 mA, the switch turned on and when it was shown that the HVU was operating, the maximum current reduced back to 1500 mA. After this command was given, the HVU immediately stopped operating and when the switch was checked, it was off. This test showed that the PCDU worked as expected and turned the switch off when the maximum current was reduced below the level of the output current.

The next step was to find the current at normal operation. This was completed by setting the maximum current to 3000 mA, turning the switch on and incrementally commanding the maximum current to be reduced by 100 mA each step. When the switch turned off, the normal operating current was the previous step. The switch turned off at 2100 mA, and therefore the normal operating maximum current was shown to be 2200 mA, which was as expected.

The final step was therefore to show that there was no major spike in the current upon the HVU being turned on. To do this, the switch was turned off, the hvu set to start when the switch was opened. The maximum current was then set to 2200 mA and the switch turned on. In this test, the switch turned on as normal and the HVU operated as normal. This showed that when the switch is opened, there is not a large spike in current. This is crucially important as it means that the EPS can be safely switched on without

worrying about ensuring there is extra power available on start up or the potential for an electrical failure on board the satellite. This meant the entire test was a complete success.

4.5 The data link to the On-Board Software

4.5.1 Using the MIB database

The final connections in the chain were to the OBC. These connections were the connection between the OBSW and the PCDU, and the connection between the OBSW and the ground. As has been previously detailed, the OBC was first connected to the ground station. As this was already implemented, the main task on this side was to ensure that this link worked.

Originally, the only function that had been tested was a PUS ping test. This very simple test is one of the most important. The test uses PUS service 17 and simply sends a CLTU up to the OBSW, from there, the OBSW sends back a reply confirming the receipt of the PUS ping command. This is used to show that the satellite is able to receive telecommands and send telemetry. Therefore its primary use is in the early phase of in-orbit testing to ensure that communication with the satellite is viable.

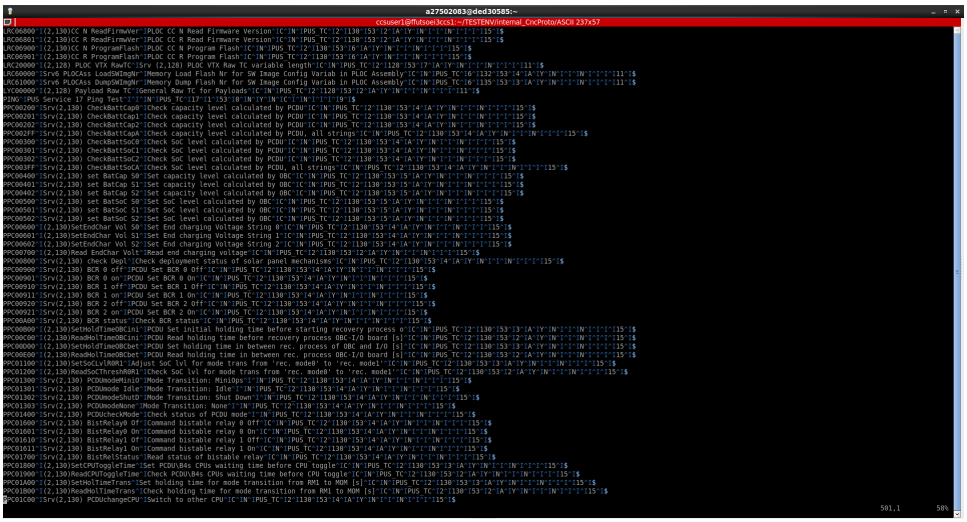

Figure 4.6: A Screen Shot of the MIB Database.

Obviously this basic test would not serve a purpose in the command and control of

the PCDU and thus the author needed to expand the test to ensure other forms of PUS service were received. First, the author began to look into the Mission Information Base (MIB) database. As this was the second generation of a platform that was already in orbit, an extensive set of telecommands and telemetry had already been decided upon and implemented. These are stored in the MIB, are accessed by the CCS5 and are shown in figure 4.6. It was therefore more efficient to use these existing commands than design and code an entirely new set, especially as these commands were tailored to the OBSW used. The most important commands in this were:

- Check mode
- Command mode
- Check fuse status
- Command fuse on
- Command fuse off
- Check switch status
- Check current through switch
- Set switch maximum current
- Command switch on
- Command switch off

As each fuse and switch is assigned a number in both the software and hardware, these commands are targeted at a single switch. This is done within the command sent from ground and the switch number is selected in hexadecimal. Alternatively, by selecting the hexadecimal 0xFF in the "Check" commands, the status of all switches or fuses is returned rather than a single component.

In order to check these, the author spent time familiarising themselves with the OBSW. As this is flight software, it is far beyond the scope of this paper to go into the detail of how it works. The broad overview is that each subsystem of the satellite has a controller, device assembly and a series of device handlers. The device handlers are linked to a single specific device (commonly an actuator or sensor), for instance a single reaction wheel within the AOCS, and their function is the command and control of this device. They receive instructions from the device assembly. The controller acts as the decision maker for the individual subsystem, taking any relevant sensor data from a datapool, receiving telecommands via the device assembly and using this information to decide what each device needs to do and when. The device assembly acts as the connection between

the controller, devices and, command and control. In this function it receives telemetry from each device handler and places it into the datapool, relays the necessary commands from the controller to the relevant device handlers and most importantly receives any telecommands from the ground and sends them into the controller.

Therefore, in order to ensure that the relevant telecommands were being received by the OBSW it was not necessary to view their effects on the device in question. A much simpler solution was to place a line of code into the device assembly that triggered when the relevant command was received. In this way it was demonstrated early on that the OBSW was receiving the correct telecommands and this initial connection had been established.

4.5.2 Altering the On-Board Software to create a UART connection

The second major connection from the OBSW was much less simple to show. There are two reasons for this. The first is that the power supply subsystem controller in the OBSW is not currently operational. This is because it does not receive any telemetry from the device handlers because the devices are not currently connected to the OBC. Normally a simulated version of these would be implemented, but this had not been completed at the time the author was working on this. The second was that in the original FLP the PCDU had been connected to the OBC via SpW and an IO board. However, in the second generation of the FLP this connection was to be completed directly between the OBC and PCDU via UART. This was decided as it would drastically reduce the number of connections between these two vital components and therefore reduce weight and more importantly reduce the complexity of the FDIR protocols in the case of one of these components failing. As the original FLP had not used UART for any form of command and control, this connection needed to be built from scratch.

These two issues in conjunction meant that instead of simply plugging the PCDU into the OBC and sending commands to it, the OBSW needed to be altered. These alterations came in two forms, first the creation of a UART port within the software. The second is a method of receiving telecommands from the ground, sending the appropriate command to the PCDU, then receiving its response and sending the appropriate telemetry back to ground. Fortunately these two issues could be dealt with in a single class, thereby reducing the alterations to the flight code and therefore ensuring minimum disruption when the controller was able to be used once more.

To start, a deliberate decision was made to ignore the commands from ground. This was because the set up to send them from this location would mean that every test of

the code would take approximately 5 minutes as there were several sockets that had to be opened and many individual pieces of code ran in a specific order every time a single component was changed and rebuilt. This was an inefficient use of time so an example command for the PCDU was created in the OBSW. This would significantly reduce the time for each individual test.

The next major step was within the hardware.h file of the OBSW. This file was designed to connect the software to the hardware of the OBC, particularly the ports. To complete this, the documentation from Gaisler for the GR712RC (the name of the OBC) was heavily used. This document specified that there were a total of six UART ports on the OBC. Two were already in use in the form of a "minicom" and "picocom". Picocom was used as the debugging mechanism for the platform and all information that was to go to the terminal was printed along this line. Minicom provided the same function for the payload.

This had already been done in the original FLP and are a common way of ensuring the OBC works as expected. They are therefore easy to configure and this had already been completed in the main OBSW. The remaining four UARTs were not in use, and it was to one of these that the PCDU was to be connected. As this port would be sending and receiving data in both directions, unlike the minicom and picocom ports that had already been set up, it was necessary to define them within the OBSW in a different way.

To complete this task, as previously stated, the documentation from Gaisler was used. In conjunction with the comments within the OBSW, eventually the correct defines, includes and names were created to theoretically allow a connection through the third UART port. This was not enough to complete the connection however. This gave the port a name and provided the software infrastructure for its access, but did not establish a connection.

Within the OBSW, a new class was created for a UART port that was designed to mimic the class that handled the SpW ports. This was done for two reasons, first to maintain continuity across the OBSW that reduces the number of bugs and aids in future changes to the program. The second reason was to reduce the amount of changes to the OBSW when connecting it the PCDU. As much of the functionality of the PCDU had already been programmed in, it made much more sense to imitate that style rather than attempt to rewrite the code already in place.

To start, this class was written as a common socket programme. To start this class seemed like a success. The OBC was connected via UART to the PCDU and the example telecommand sent down the line. This command was sent correctly and received correctly, as evidenced by the nine bytes telemetry being sent back along the UART and

being received by the OBSW. However, as the command sent was the "check mode" command (deliberately chosen to reduce the risk of damage to the PCDU), the telemetry was expected to be eleven bytes long in total.

Much like the PUS protocol, the PCDU has a set of standard commands and responses, and these are embedded within a protocol. This protocol requires an eight byte telecommand header, followed by any telecommand data. The first two bytes state the length of the data that is to be sent after the header (usually zero). The third is a command count, this is a simple counter that iterates as a command is sent to ensure that commands are not lost and match telemetry to a specific command. The fourth is a command ID which identifies the command as part of a group of commands (such as check the status of a switch). The fifth and sixth are the command parameter that provides extra detail on the command (such as which switch's status needs to be checked). The last two bytes of the header are the CRC of the command, and usually there is no data that moves with a command.

The telemetry follows a similar format with an eight byte echo followed by the data and a two byte CRC. One major difference is that the fifth byte is the execution flag. In this, if the command was executed the byte value in hexadecimal is "0xF0", if it failed to execute it is "0x0F".

Closer inspection of the telemetry showed the first eight bytes of the nine that were received matched the echo structure that was expected. The telemetry from the example command showed that the command was executed and the length of the data was one byte. Since there is always 2 bytes of CRC following the data, the total size of the telemetry packet should have been 11 bytes. This was good news as it meant that the telecommands were correctly received, but showed a flaw in the returning data.

It was eventually realised that the ninth byte, which had previously been assumed to be random, was in fact the last byte of the CRC. This was then checked using several different telecommands to verify that the issue was that the ninth and tenth bytes were being cut from the telemetry. This indicated a serious error within the hardware, the OBSW or the socket program itself.

After looking closely at the documentation, it was decided that the hardware was not at fault as it was both sending and receiving data and the eight byte First In First Out buffer (FIFO) in the hardware was more than sufficient to transfer the small amount of data received. From there the general properties of a UART were investigated.

The basics of a UART are that as a byte of data is received, it immediately writes that data onto a small FIFO. A flag is then shown to the processor that tells it there is

data inside the buffer. The processor will take this data from the buffer and store it in Random Access Memory (RAM) as soon as possible. As data can be written very quickly onto the buffer and the processor can take time to read the data, if the buffer fills up to a point where data loss is a danger, the buffer sends an interrupt flag to the processor. This flag makes removing data from the buffer the highest priority. This removes the possibility of lost data.

In the testing of the socket program, data was lost. Obviously this means there is an issue with either the FIFO or interrupt flag. A minor subsection of the Gaisler documentation stated a way to read and write from the UART sockets. As this did not match the socket program, it was postulated that when the FIFO was filling up, the interrupt was not being generated and therefore data was lost. Thus, the socket program was rewritten to be in line with the documentation.

After some teething issues, based around the naming conventions of both Gaisler and the OBSW, the new socket was completed and successfully sent the correct telecommands and telemetry. This strongly implies that the hypothesis was correct. This meant the telemetry was being both sent and received correctly. From there, the final stage could be completed.

4.5.3 Testing the link

The final step was to link the communication being received from ground with the communication being sent to the PCDU within the OBSW. At this point the example telecommands were removed from the code, and the internal connections between the datapool of the CCSDS and PCDU controller set up. These changes meant that after the OBSW processed the CLTUs, they were sent to the PSS assembly. From there the PSS assembly placed them into a datapool for the PSS controller to process.[10]

As the controller was not receiving enough inputs from the sensors, it was unable to make decisions. Therefore a short bypass was set up in parallel to the controller. This bypass sent the command straight to the now connected PCDU via the socket. From there, the PCDU responded to the command and sent back its telemetry. This went through the bypass again back into the PSS assembly. From there it was processed, sent into the simulated CCSDS, encoded into a CADU and sent to ground.

This process was then tested by successfully commanding the HVU from the ground. This process will not be described as it was very similar to the test of the link between PCDU to the EPS using the laptop and is therefore repetitive. The important things to note are that the ground was able to use the telecommands within the MIB to successfully

send the following commands and receive their telemetry:

- Check mode
- Command mode
- Check fuse status
- Command fuse on
- Command fuse off
- Check switch status
- Check current through switch
- Set switch maximum current
- Command switch on
- Command switch off

These are the commands directly relevant to the control of the propulsion system and therefore the test was a complete success. This shows that the EPS can be turned on, turned off and regulated by defining how much current enters the system. For a fully operational EPS the valves and sensors must also be considered. This will be looked into more detail in the final section, in which the author will conclude the paper and detail future next steps for the project.

CHAPTER 5

Conclusion

5.1 Conclusion

In conclusion, this paper has looked at the command and control of the propulsion system of a satellite from ground. By necessity, this has given insight into the many areas of the satellite listed below:

- how a ground station is developed,
- how satellites are given commands,
- how satellites send telemetry,
- how the hardware of OBC functions,
- how the OBSW is developed and its internal structure,
- how a satellite subsystem is controlled,
- how the PCDU functions,
- how a propulsion system is chosen.
- how an EPS operates and is controlled.

In broad strokes this encapsulates the operation of the entire satellite from ground, with a particular focus on the power subsystem.

The FLP satellite platform has been outlined and its strengths detailed and a thorough examination of RF communications with a satellite has been undertaken. A detailed look at a CCSDS board and its protocols has been given alongside a brief introduction to the simulation of satellite subsystems and components.

The main three groups of propulsion systems have been described and analysed for their strengths and weaknesses. The purpose and functionality of a PCDU has been outlined along with a more thorough look at how its command and control structure works.

A broad outline OBSW, in particular the way in which subsystems are handled, has been given before a detailed investigation into how new connections are formed to a subsystem and how that relates to the command and control structures already in place. Finally, a series of tests have been completed to show how an EPS can be integrated into pre-existing satellite architecture. This began by using a laptop with software to command a PCDU to command an EPS inside a vacuum chamber and culminated with the command and control of the HVU within the FLP testbench from the CCS5 ground station.

5.2 Moving Forward

This thesis has not gone into serious depth on the mechanisms behind the AOCS as they are beyond the scope of the thesis. Project managers work in close cooperation with the mission control and orbital analysts to decide the direction a satellite should point in and any changes in orbit that the satellite needs to undertake. Control engineers can spend large parts of their career developing the control loops that allow the satellite to move from the position that it starts in to the orbit and attitude desired by the project manager.

Eventually, all this comes down to two parts of a satellite. The first is the command and control subsystem that receives the plans of the project manager for the desired orbit and attitude of the satellite at each moment in time. The second is the subsystem controllers, in this case the AOCS controller. This uses the sensors within the satellite and the control loops of the engineers to determine the speed and acceleration of each reaction wheel and how the propulsion system is to be used. In this, the AOCS controller works in close cooperation with the PSS to ensure that the satellite has enough power to complete the orbital manoeuvres necessary.

The EPS has a series of valves, and dependant on the subsystem methods of accelerating the gas further, to utilise in selecting an amount of thrust that is generated. Additionally it can select an amount of time over which the thrust is generated. Therefore, the two primary parameters that can be selected to control an orbital manoeuvre are which valves are open and the length they are open for. For some systems it is also possible to choose whether to further accelerate the gas using accelerators or combustion chambers, but it is highly unusual to choose not to accelerate the gas if the option is there.

Other factors come into play but these are not able to be controlled by the AOCS

system, though they still need to be accounted for if a manoeuvre is to be successful. An example of this is the pressure inside the fuel tank. This decreases every time the propulsion system is used, and as the pressure decreases, the force generated by an opened valve also decreases. This means that for the same amount of acceleration, the valve needs to be opened for longer. Another example in an EPS is the electrical power supplied to the ion accelerator. In an unregulated power bus, or dependant on the current state of the batteries and solar arrays, this can vary. As it varies, it changes the additional acceleration provided to the gas being expelled. Again, this is a change to the force provided by the satellite that cannot be controlled.

As the satellite does not have a customer, naturally, the satellite design is not finalised. At the very least this means that the control loops cannot be developed. However, the original FLP did not have a propulsion system. Therefore, the OBSW does not have an AOCS, instead using a simpler ACS. It also does not have the device handlers, device assemblies or controllers required for a propulsion system. These systems are yet to be developed, and it is possible to do so now.

Therefore, there are three main recommendations the author provides for the continued development of the FLP, and in particular the subsystems focussed upon in this thesis.

5.2.1 Testing the CCSDS Board

The first is the full testing of the CCSDS. Currently, while it has been established that the model is capable sending telecommands and receiving telemetry, it has yet to be confirmed that it does this correctly every time. Therefore, before it is deemed a success, several tests need to be completed. The first is large quantities of telecommands and telemetry being sent through it simultaneously. This will ensure that the functions work correctly and are efficient enough to deal with a large volume of traffic. The second is for a large proportion of the MIB database to be sent as individual telecommands to the OBC. This will verify that the CCSDS is accurate for the telecommands used in the prior mission.

5.2.2 Integrating the Power Control and Distribution Unit

The second recommendation is the full integration of the PCDU. Currently the basic functions have been tested and shown to be robust. However, at this point it is not integrated with its controller in the OBSW. For complete confidence in its full integration this needs to be completed. It would also be a major step to change the power supply of the OBC from being powered by the mains to being powered by the PCDU.

This is a complex task as the PCDU will have to autonomously go through the start up process that determines that many parameters and inputs required for booting the OBC are met. Normally this requires checking the SoC of the batteries amongst other checks, naturally while the PCDU is powered from the mains this is not possible and will have to be bypassed or a relevant input will have to be provided. The final option in the integration of the PCDU is the testing of both functionality and FDIR events. This has been completed within the parameters of an EPS, but it is by no means a completed task for the PSS.

5.2.3 Creating an Attitude and Orbital Control Subsystem within the On-Board Software

The final, and perhaps the most complex, recommendation is the development of the device handlers, assemblies and controllers for the EPS. This goes hand in hand with the expansion of the ACS into a full AOCS and the links between the assemblies, controllers and datapools of three systems; the PSS, the AOCS and the EPS.

The first main task is the development of the device handlers, subsystem assembly and controller of the EPS. This will need to be done in line with the other subsystems within the OBSW, and in a general way to accommodate the different types of potential propulsion systems that could be used. Therefore, the author suggests the implementation of a "valves" class in the same way that the PCDU has a class for "switches" and "fuses". This will allow the number of valves to be changed easily and the values they provide to be easily changed. The only sensors universal to the propulsion systems is the pressure of the gas within the fuel tank and their temperature. These will need to be developed into device handlers. Dependant on time, it is recommended to select a specific propulsion system, such as the HEMPT that has already been modelled, and base all other aspects on this.

The next main task within this is the expansion of the ACS into an AOCS. This will require the creation of additional variables and connecting them to the datapool. Some of this data needs to be linked to the data provided by the sensors of the newly created propulsion subsystem. The controller could be given a very basic control loop that utilises the information from the sensors available and creates a simple plan for the propulsion system to carry out, stating the time that the propulsion system will be on for and the times that the valves will be open for. Naturally to start this should just be printed to the console, but it is important for this functionality to be developed as some propulsion systems require time to prepare components of the system in advance of the thruster being fired. The EPS needs time to turn on the ioniser and accelerator and most hot gas propulsion systems must heat the ignition chambers before the gas is released.

From there this information must be passed to the propulsion system controllers so that it may perform its function correctly. The propulsion system will also have to communicate with the PSS as the switches to power the heaters, ionisers and accelerators are housed there. Even using a cold gas system, it is likely that the valves are a binary switch that need power to be opened or closed.

The final consideration of these changes is to the modes of the entire satellite. When a large scale orbital manoeuvre is performed, at a minimum it requires a lot of power and a small margin of error in the direction the satellite points during the manoeuvre. The direction the satellite points in is required to orient the thrusters in a specific direction to change the momentum of the satellite. This can move the solar panels out of alignment with the sun and reduce power, prevents all missions in which pointing is important and almost guarantees that any ground station cannot be contacted. Additionally, in order to stabilise the direction of the thrust, it is likely that the reaction wheels will be used extensively along with any relevant sensors. These factors mean that the mission will have to be switched off, as will any non-relevant systems. The magnitude of the changes to the overall system of the satellite makes it clear that a new mode should be created, or an existing mode altered, to ensure that all necessary checks and procedures are made before the thrusters fire. This minimises the risk of damage to the satellite and maximises the chance of a successful orbital manoeuvre.

Bibliography

[1] Charles D. Brown. *Spacecraft Propulsion.* AIAA education series, 1996.

[2] ESA-ESTEC Requirements & Standards Division. *Ground systems and operations - Telemetry and telecommand packet utilization.* ESA Publications Division, 2003.

[3] Jens Eickhoff. *Simulating Spacecraft Systems.* Springer, Heidelberg, New York, Dordrecht, London, 2009.

[4] Jens Eickhoff. *A Combined Data and Power Management Infrastructure.* Springer, Heidelberg, New York, Dordrecht, London, 2015.

[5] Jens Eickhoff. *The FLP Microsatellite Platform.* Springer, Heidelberg, New York, Dordrecht, London, 2015.

[6] Peter Erichsen. Performance evaluation of spacecraft propulsion systems in relation to mission impulse requirements. *Second European Spacecraft Propulsion Conference*, 1997.

[7] Dan Williams et al. Rf and optical communications: A comparison of high data rate returns from deep space in the 2020 timeframe. 2007.

[8] Parkes et al. Spacewire, links, nodes, routers and networks. *European Cooperation for Space Data Standardization*, 1, 2003.

[9] R. Janovsky et al. End-of-life de-orbiting strategies for satellites. *DGLR Jahrbuch 2002*, 2002.

[10] Steffen Gaisser. Modelling of a power supply system for small satellite simulation. 2016.

[11] Jan Knippschild. Simulation model of an electric propulsion subsystem for small satellites. 2017.

[12] Bryan Klofas Kyle Colton. Supporting the flock: Building a ground station network for autonomy and reliability. *Small Satellite Conference*, 2016.

[13] S. Habine P.Sinander. Telecommand and telemerty components for today and tomorrow. *First ESA Workshop on Tracking, Telemetry and Command Systems*, 1998.

[14] Marc Sans and Zoran Sodnik. Design of the esa optical ground station for participation in llcd. *International Conference on Space Optical Systems and Applications*, 2012.

[15] Vijay K. Bhargava Stephen B. Wicker. *Reed-Solomon Codes and Their Applications*. Wiley-IEEE Press, 1994.

[16] Kozo Takahashi. Reliability and availability of redundant satellite orbit systems. *IEEE Transactions on Aerospace and Electronic Systems*, AES-18(10):258 – 267, 1982.

[17] Kristy Westphal. https://www.symantec.com/connect/articles/nfs-and-nis-security, 2001.

[18] Daixun Zheng, Tanya Vladimirova, Prof Sir, and Martin Sweeting. A ccsds-based communication system for a single chip on-board computer, 2015.

[19] Bartosz Zoltak. Vmpc-r cryptographically secure pseudo-random number generator alternative to rc4.