Correction of faults and errors induced by cosmic radiation on operating system level in the NUTS StudSat project

Kjell Arne Ødegaard

NORWEGIAN UNIVERSITY OF SCIENCE AND TECHNOLOGY
DEPARTMENT OF ENGINEERING CYBERNETICS

July 1, 2013

Problem Description

The objective of the prestudy is to examine and evaluate the NTNU Test Satellite and propose reliability solutions for a low cost, consumer component based system running on the satellite. The focus is to alleviate the work needed for future implementation on the satellite. This is done by conducting a complete study of the satellite and its systems. In addition, the satellite's operating environment is examined in order to define possible error conditions. Once the error conditions are defined, possible solutions will be found in a reliability literature study. The intention is to lay the ground work for future implementations of a reliable system on the NTNU Test Satellite.

Supervisor: Associate Professor Amund Skavhaug, Department of Engineering Cybernetics, NTNU.

Abstract

The NTNU Test Satellite (NUTS) project aims to build and launch a double CubeSat within 2014. With the use of CubeSats, there is a growing use of low-cost space platforms for research and even commercial use. By investigating what is possible to achieve with low cost components in terms of reliability, two important things can be achieved. The cost of using space platforms for research can be reduced, while at the same time decreasing the number of non-functional satellites (i.e. problematic space junk).

The focus of this work has been to explore the use of consumer components in space and the impact of the harsh vacuum and radiation environment on these components. Most of the work has been looking into intermediate radiation faults in processors and memory and how to best mitigate them. Some approximations regarding long term radiation effects have also been included. It is important to note that this is mainly an exploratory paper. Much of the practical work concerning fault insertion and simulations on the system is planned for the next semester.

This paper provides a description of the problems with consumer electronic components in a space environment and what can be done to mitigate said problems. It also presents a shorthand on what effects that causes these problems and a rundown on how they will affect the finished system. Data from NASA is used in order to approximate the expected fault intensity in the satellite's components.

Finally, a number of possible solutions to the presented problems are presented and evaluated along with the error model assumptions. Master-slave operation on critical modules will ensure system continuity. Checkpointing should provide a system in a restorable, safe state in the event of a restart. Error detection and correction in storage will ensure correction of errors in stored data and provide a place to store critical runtime variables. In addition two levels of watch dog timers, program memory integrity check and a periodic reset to clear any undetected errors, both soft and hard, is planned.

The resulting design is expected to be able to mask even frequent resets from the system's operation. In addition the error correction and detection will prevent the expected hundreds of errors per day from accumulating in memory and affecting the system. To simulate a radiation environment for testing, fault injection via JTAG will be used. This allows for repeatable test runs and focus on specific parts of the system.

Contents

List of Figures

List of Tables

List of abbreviations

ADCS	Attitude Determination and Control System
COTS	Consumer Of The Shelf
CRC	Cyclic Redundancy Check
CSP	Cubesat Space Protocol
EDAC	Error Detection And Correction
ELDR	Enhanced Low Dose Radiation
EPS	Electrical Power System
FSP	Finite State Process
LEO	Low Earth Orbit
LET	Linear Energy Transfer
LTSA	Labelled Transition System Analyser
NUTS	NTNU Test Satellite
OBC	On Board Computer
OTP	One-Time Programmable
RTC	Real Time Clock
SEE	Single Event Effect
SEP	Single Event Phenomena
WDT	Watch Dog Timer

Chapter 1

Overview

1.1 Introduction

The gateway to space for research institutions and commercial actors has traditionally been associated with a very high cost. Recent year's development of small, inexpensive satellites known as pico and nano satellites looks likely to change this by considerably lowering the price point of satellite construction and launch. The launch costs are kept low by piggy backing onto other launches, and the satellites themselves often use Consumer Of The Shelf (COTS) electronic components.

The use of consumer solutions allows for fast development with modern tools and enables the designers to get full advantage of the economy of scale with cheap and plentiful components and development tools. Due to the typically shorter lifespan of these satellites compared to traditional endeavors, it is possible to use newer more innovative and untested components and design without running big financial risks. This is interesting as it allows for development and advancement in an otherwise conservative industry.

With this new development, research institutions and commercial actors can build and launch a satellite for experiments or surveys in a much shorter time frame and on a smaller budget. This allows for a number of interesting and novel uses of space and the development we see in the industry today is very exciting.

An interesting development along these lines has been the introduction of the CubeSat platform. To help universities worldwide perform space research the CubeSat platform was developed in 1999 by, among others, California Polytechnic State University and Stanford University. The CubeSat programs goal is to provide practical, cost-effective and reliable launch opportunities for small satellites and their payloads through a standardized platform [18].

Due to cost concerns, CubeSats typically use Consumer Of The Shelf, or COTS, components. A number of different factors that will be detailed further on in this paper make these components vulnerable to the environment in space and in this paper we explore measures to alleviate the impact of these factors to the reliability, availability and survivability of the design.

1.2 Problem

The goal of this paper is a COTS system with equally high reliability as a system customly equipped and designed for a space environment. Reliability is a common goal for space designs and is not a new design criterion. What is new, however, is the emergence of the low cost CubeSat platform.

One of the main challenges for space applications is the hard radiation operating conditions [8] [10]. To solve these challenges, radiation hardened electronic components and fault tolerant hardware has been used is space systems for a number of years to either ensure error free operation or to mask the errors from the system. In the context of a CubeSat, however, the challenges of high reliability system design shifts. It is still desirable with a high reliability system but the budgetary constraints are much stricter than for commercial or government designs.

In addition to being considerably more expensive, radiation hardened components traditionally lag behind their non-hardened equivalents in performance. This means that one gets a less capable system at a higher price point. At the same time it is not very important with a high availability design since the system does not control critical applications, but rather performs data collection tasks. This means that the on line redundant backup components can be omitted as long as we ensure that the system does not malfunction critically (i.e. fail). By using software methods, combined with redundancy for the most important subsystems, it is therefore possible to get higher performance, more flexibility and lower price, all without hot standby redundant backup components.

This paper aims to investigate low-cost methods to increase mission lifetime of small COTS based satellites. When considering the different reliability measurements it is important not to impact the performance of the rest of the system to an unacceptable degree. By mitigating the effects of Single Event Phenomena (SEP) occurrences in non-hardened components, it is possible to ensure higher up-time and increased mission lifespan. This further promotes safe operation and increases the likelihood of not loosing mission critical or payload data. Student satellites do not have access to the established solutions because of budget constraints, and have to rely on smart solutions and COTS hardware to have a usable system in extreme conditions.

1.2.1 Related Work

NTNU has been involved in satellite projects with other Norwegian universities for a number of years through the NCUBE project. Two satellites were constructed in this program, NCUBE-1 and NCUBE-2. NCUBE-2 launched in 2005, but radio communication with the satellite was never established. An error during antennae deployment is suspected to be the cause of this. NCUBE-1 was launched in 2006, but due to problems with the rocket the launch had to be aborted and the satellite crashed. The work on NUTS started again in 2010 with a new specification for a double CubeSat. Since then there have been a lot of work on reliability and the occurrences and migrations of SEP. Some interesting work is also done on soft errors and the impact they have on control systems [6].

1.3 Scope and Disposition

1.3.1 Scope

The NUTS project was started in September 2010 with the goal to manufacture and launch a double CubeSat by 2014. There have been a number of published articles and papers concerning the reliability of the system as the project have evolved. The most noticeable ones are the work done on the backplane by Dewald [11], the Electrical Power System (EPS) by Jacobsen [13] and the On Board Computer (OBC) by Holmstrøm [12].

The purpose of this work is to make a study of useful techniques for implementing a reliable system by using the modules that have previously been developed. The approach has been to study reliability theory and find methods that can be used to mitigate the specific problems encountered by using COTS components in a space environment. The previous work has focused on reliability for certain aspects of the system, but has not provided software and methods to incorporate this into a redundant system. The work done in this study has been done in two parts. The first part has been a study of the problems and effects that the satellite will encounter and what can be done to fix them. The second part, the part planned for the next semester, is to implement some or all of these solutions and simulate faults in order to get a measure of the severity and the ability to recover from these faults.

1.3.2 Report Disposition

This paper will begin with a basic introduction on the topics of space, radiation, electronics and redundancy. The reason for this is that the paper is meant to be an introduction for the multidisciplinary project team which might not all be equally versed in the previously mentioned subjects.

The paper continues with a presentation of the current design in NUTS. It touches on the limitations, challenges, requirements and the balancing of reliability versus availability before continuing with the current design and its possibilities and limitations with regards to the use of COTS components in space. The intention is to simulate the efficiency of the proposed solutions by injecting faults. This work has had to be postponed until the rest of the satellite is at a more finished stage.

The paper concludes with suggested solutions for the presented problems and gives a presentation on the work done to study the reliability of the system.

1.4 Theory

To put the expected problems into context, a brief overview of the relevant theory is presented. First some relevant definitions from reliability theory are presented, followed by precise definitions of reliability and availability. Finally, a summary of the problems that electronic components in space can encounter and how this affects the expected lifetime of the components is presented.

1.4.1 Fault Detection and Manifestation

There are many kinds of faults that can occur in a system. In reliability theory one differentiates between fault, failure and error. The definitions below are from [3, p. 22]. In figure 1.1, borrowed from [3], the cause and consequence of different faults are further detailed.

Failure occurs when the delivered service deviates from the specified service, failures are caused by errors.

Error is the manifestation of a fault within a program or data structure; errors can occur some distance from the fault sites.

Fault is an incorrect state of hardware or software resulting from failures of components, physical interference from the environment, operator error, or incorrect design.

Permanent describes a failure or fault that is continuous and stable; in hardware, permanent failures reflect an irreversible physical change. (The word hard is used interchangeably with permanent.)

Intermittent describes a fault that is only occasionally present due to unstable hardware or varying hardware or software states (for example, as a function of load or activity).

Transient describes a fault resulting from temporary environmental conditions. (The word soft is used interchangeably with transient.)

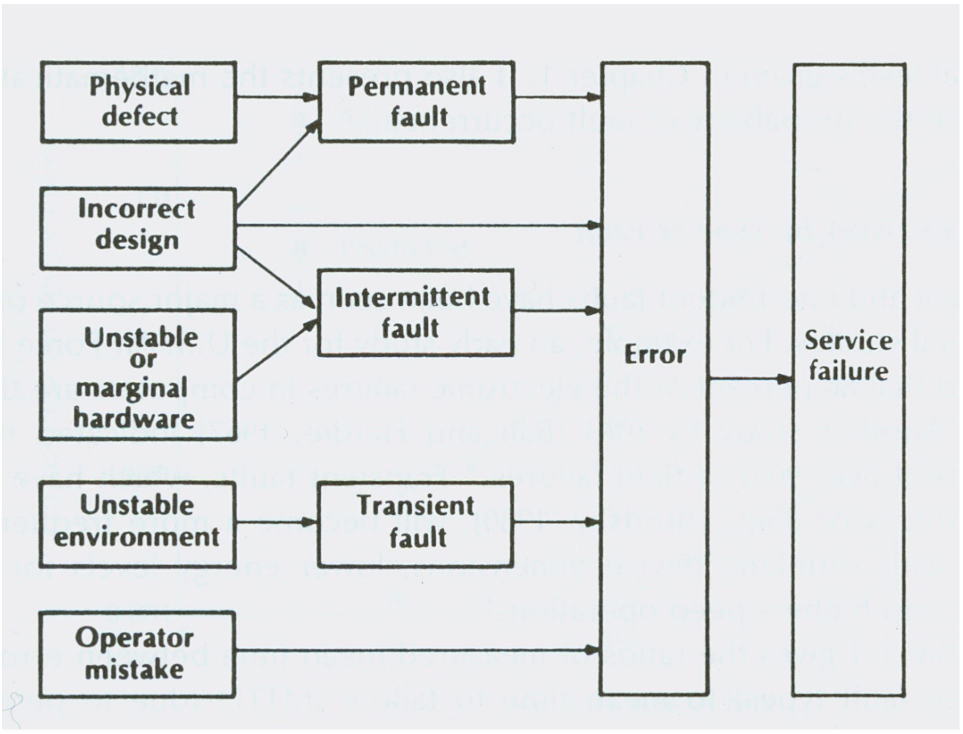

Figure 1.1: Sources of errors and service faults

1.4.2 Reliability and Availability

Reliability and availability have precise definitions in the literature. The following definitions are form [3]. The reliability of a system as a function of time, $R(t)$, is the conditional probability that the system has survived the interval $[0, t]$, given that the system was operational at time $t = 0$. Reliability is used to describe systems in which repair cannot take place (as in satellite computers), systems in which the computer is serving a critical function and cannot be lost even for the duration of a repair (as in flight computers on aircraft), or systems in which the repair is prohibitively expensive. In general, it is more difficult to build a highly reliable computing system than a highly available system because of the more stringent requirements imposed by the reliability definition. An even more stringent definition than $R(t)$, sometimes used in aerospace applications, is the maximum number of failures anywhere in the system that the system, can tolerate and still function correctly [3, p. 4].

The availability of a system as a function of time, $A(t)$, is the probability that the system is operational at the instant of time, t. If the limit of this function exists as t goes to infinity, it expresses the expected fraction of time that the system is available to perform useful computations. Activities such as preventive maintenance and repair reduce the time that the system is available to the user. Availability is typically used as a figure of merit in systems in which service can be delayed or denied for short periods without serious consequences [3, p. 4].

1.4.3 Single Event Phenomena, SEP

Electronic components are affected by cosmic radiation and the collective term for the different failure mode occurrences is Single Event Phenomena (SEP). When a charged cosmic particle hits the components the resulting collision deposits energy in the component. This is known as a Linear Energy Transfer (LET) and is defined as the linear density of energy deposited in material by a charged particle of ionizing radiation traveling through it.

This energy can alter the charge of the transistors and capacitors that are internal in the electric components. If the charge is altered sufficiently, the voltage level of the transistor or capacitor can change, and this results in the stored digital value changes. This is known as a soft error or a bit-flip.

If the cosmic ray hits specific parts of the electronic components with enough energy, more severe failures may occur. There are several types of failures that require physical intervention. The Single Event Latchup (SEL) is triggered when heavy ions, protons or neutrons hits at a susceptible point in the component structure and it may cause catastrophic thermal runaway [10]. It is only recoverable through power cycle and is strongly temperature dependent with the threshold for SEL decreasing at higher temperatures. Modern devices may have many different SEL paths and a proper characterization of a latchup is a difficult problem. It is also worth noting that modern devices may have both high and low current SELs, something that complicates the characterization further [10]. There are also destructive Single Event Effects (SEE) such as the Single Event Gate Rupture and the Single Event Burnout, but these are permanent and outside the scope of this work.

There are several types of SEP that has to be evaluated when designing a space mission.

Table 1.1: Single Event Phenomena

Name	**Effect**
Single Event Transient, SET	Soft intermittent fault Propagating through circuit
Single Event Upset, SEU	Soft transient fault State change on latch or memory
Single Event Latchup, SEL	Apparent short circuit Can be mitigated with power cycling Can cause destructive thermal runaway
Single Event Gate Rupture, SEGR	Permanent failure
Single Event Burnout, SEB	Permanent failure

In this paper the main focus is on the two most common, namely SET and SEU, both of which are considered soft errors [1], and the occurrence of SELs. These three failure modes are the only ones that can be fixed without a component replacement, something that is outside the scope of this work.

The work does not consider more severe conditions such as strong electromagnetic pulse (EMP) (which could disable the entire system) or sun storms which would effectively overwhelm the COTS components in the student satellite.

When cosmic rays from the space environment hit the electronic components they can deposit electrical charge in the n-doped material sections of the semiconductor material. The physical effects of cosmic rays in the form of heavy ions and protons on the electronic components are shown in figure 1.2. This shows that heavy ions are more damaging to the components, something that also intuitively makes sense as they can hold more charge and energy than the protons.

The SEU threshold Linear Energy Transfer (LET) is described at the energy level per amount of material of the radiation that will trigger SEU events. For COTS components this is typically 5 $MeV/mg/cm^2$ [8]. The expected SEU error rate for COTS in Low Earth Orbit is 10^{-5} $error/bit-day$ [8]. This might sound like an uninterestingly small amount, but with 128kB of RAM it amounts to over 10 errors per day. If these errors are

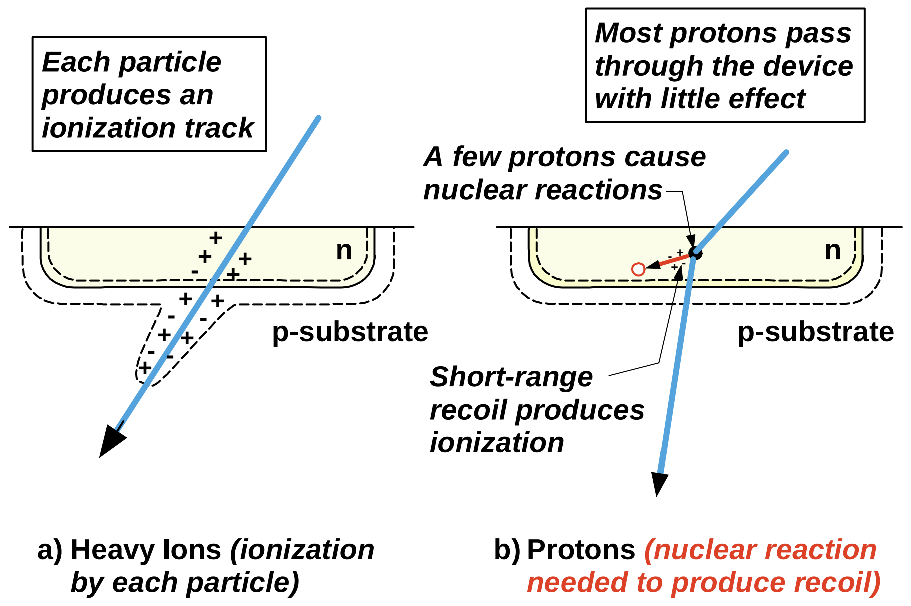
Each particle produces an ionization track
Most protons pass through the device with little effect
A few protons cause nuclear reactions
n
p-substrate
n
p-substrate
Short-range recoil produces ionization
a) Heavy Ions (ionization by each particle)
b) Protons (nuclear reaction needed to produce recoil)

Figure 1.2: Mechanisms for Heavy Ion and Proton SEU effects

allowed to accumulate it can be negative for the satellite's reliability

1.4.4 Total Radiation Dose

The total ionizing dose is the combined damage of the semiconductor lattice that is caused in electronic components exposed to ionizing radiation over time.

The total radiation dose of the system in not a major concern as it will mostly affect mission lifetime. There is some concern, however, due to new effects such as Enhanced Low Dose Radiation (ELDR) sensitivity and subtle failure modes in complex parts. The mission also uses sensitive technologies with internal charge pumps such as flash memories [10].

For satellites in inclinations between 20 and 85 degrees in Low Earth Orbit (LEO), both the northern and southern hemisphere, the typical dose rates are 1000-10000 rad(Si)/year [8]. As NUTS will have an even higher orbital inclination a conservative assumption would be at least 10000 rad(Si)/year due to the higher radiation levels close to the Earth's magnetic poles. In combination with the information presented in table 1.2 and [10] we see that some components may fail within the stipulated mission lifetime of 3 to 6 months. Linear IC's,mixed signal IC's and flash memories are the most sensitive components and this should be kept in mind if there are any unexplained failures two to five months within the mission.

Table 1.2: Typical total dose failures levels for various technologies

Technology	**Failure level [Krad(Si)]**
Linear IC's	2 - 50
Mixed-signal IC's	2 - 30
Flash Memories	5 - 15
DRAMs	15 - 50
Microprocessors	15 - 70

Chapter 2

NUTS

NTNU has for a number of years had a student satellite. The main challenge for such small and low-budget systems is to accommodate a certain amount of features while working with a limited budget. In this chapter the limitations and current challenges in the NUTS project will be presented first. The system requirements, alongside the reliability and availability demands for the system, will be presented before the current design is described.

2.1 Limitations and Challenges in the NUTS project

As previously mentioned, some of the main challenges with NUTS are the design constraints in the form of a limited budget. In addition, there are standard satellite constraints such as power use, total weight and volume.

Weight and volume does not directly limit the design and functionality of the electronics, but due to the requirements of the payload, antenna and ADCS system, the electronics might have to be designed with these physical specifications in mind. With respect to the electronics there is almost no weight limitation for a double (2 Unit) [17] CubeSat such as NUTS. If we were to fill up the entire $10cm * 10cm * 20cm = 2000cm^3$ of the satellite with solid aluminum if would amount to only 5400g. This is twice the allowed weight as specified by [17]. The satellite can therefore be relatively massive, and the weight of the electronics module is therefore not a design criteria.

The power limitation, on the other hand, is another matter altogether. The available power from the satellite's solar panel array is not very large initially, and the communication downlink is capable of using all the power there is to spare. The satellite will most likely have more data than available transmit power. The power consumption of the on board electronics should therefore be a strict design criterion, and limited as much as possible. The ADCS uses coils to reorient the satellite in the Earth's magnetic field. The largest power draw will occur during the detumbling phase right after launch, and it will most likely be a major contributor during normal operations.

The last and perhaps most important major subsystem within the satellite is the payload. It uses a large amount of power so it will have to be planned how often and for how long it is allowed to operate.

The main challenge in NUTS is the economical one and the more that can be done with resources already available the better. The theory and methods presented to solve these problems are not new in themselves, but when applied to the relatively new CubeSat platform, the different focus and design challenges of the project makes for demanding work.

2.2 Requirements

This section presents the requirements that are relevant in a reliability context. There is no exhaustive system specification at this point, but a complete overview of the specified requirements to date can be reviewed at [12].

Table 2.1: General Requirements

Description
The satellite must execute a one-time initialization sequence on first boot up
The satellite must be able to create and store commands programmatically
It must be possible to initiate a full or partial satellite system reset from the ground station
It must be able to set the current time in the satellite, from the ground station

Table 2.2: Reliability Requirements

Description
Only uncorrupted commands shall be executed
A failing program must not affect the core functionality of the system
Execution of less-important tasks shall not affect the timeliness of higher-prioritized tasks
A frozen system program shall not render the satellite useless

It is desirable to have a highly reliable system. In order to operate correctly the satellite has to have rules of conduct when communication to the ground station malfunctions.

Table 2.3: Autonomous Requirements

Description
Self-repairing to the greatest possible degree
Absolute measurement of time in order to initiate self-repairing correctly
Correctness determination algorithms in order to initiate self-repairing correctly

When talking about self-repairing in this context it is primarily regarding the radio. If the radio should malfunction in orbit the satellite is inoperable. The module could then be power cycled or, if all else fails, be reprogrammed. This is hazardous in a radiated environment, but without a radio the satellite is defective.

The satellite modules should have some measure of time in order to have timeouts on critical systems. To study the need for time awareness a possible use case in the ADCS module can be reviewed. If the satellite ends in an unfortunate position or with to fast spin after launch the antennae will not be able to communicate with the ground segment. Therefore the satellite needs to be able to be able to initiate detumbling autonomously, especially considering the likely antennae problems on NCUBE-2. It is also possible that the EPS could have some problems and reset intermittently or periodically and this period might very well be shorter than the timeout chosen for the ADCS. Some notion of real time passed since initial boot in the system therefore needs to be implemented.

2.3 Reliability vs. Availability

This chapter builds on them terms described in in section 1.4.2.

In the context of NUTS it is not important to have a highly available system. The payload is not dependent of being online at all times, only in shorter bursts to take a series of photos. As long as it is capable of obtaining a set of pictures and processes it within a certain time frame it is able to complete its mission. In other words it is sufficient with an adequate average availability. If the satellite is online at average 22 hours per day it would translate to an availability of 0.92, but this is probably in the lower bound of what could be accepted.

The same argument can be used with regards to reliability. The satellite spends most of its time in eclipse from the ground station and it is not paramount that the satellite is online at all times. Even if the satellite should malfunction during communication there are still multiple other passes on the same day. As long as payload data is received most of the time it does not matter if some minutes or even hours pass without data, as long as the satellite continues to be operational.

Table 2.4: Design requirements and drawbacks

Problem	**Solution**	
	Reliability	**Availability**
Freeze/crash	Low power watchdog	Online backup module
Result check	Recalculation	Result checking logic circuits
Continuity of operation	Saving of current state	Seamless switchover
SEP	Checkpointing	Replication of logic circuits
Drawbacks	Must be very robust	Higher power consumption Extra circuitry

As can be seen from the assessment in table 2.4, the design of a high availability system adds a number of strict requirements for the architecture of the system. The design effort could therefore favorably be shifted towards a system with high reliability and survivability without the extra overhead of a highly available system.

2.4 Current Design

A number of design choices have already been made for the project. This limits the possibilities, but at the same time it allows us to focus more thoroughly on the problems as possibilities of the platform at hand. In order to study what can be done in terms of a reliable system, and get an overview of potential problems, it can be helpful to examine the current satellite modules and components. Below follows a brief overview of some of the major hardware and software modules in the project, their possibilities and limitations.

2.4.1 Frame

The construction of the satellite's frame is done with carbon fiber which is novel in space craft design and therefore interesting research. Positive characteristics includes high strength and low weight, something that should make it an ideal candidate for applications where much of the cost is related to the amount of mass launched into orbit. The carbon fiber has some negative characteristics which can prove to degrade the performance of the frame in a reliability context. Compared to aluminum the carbon fiber frame has poor heat and electrical charge conduction and this might lead to a harmful rise in temperature and accumulation of electrical charge in the electronic components.

2.4.2 Backplane

There has been done a lot of work to ensure that the functionality of the backplane is realized by using only discrete logic. This allows for a more complete state space analysis and ensures that is it possible to account for all of the backplane's states. When all states are accounted for it is possible to guarantee that the backplane does not enter a deadlock.

Each module also has redundant power supply and a redundant system bus connection. In addition to this it is possible for one of the two master modules to power cycle or shut down individual modules in the system.

In the event of a high current SEL, the current limiting power supplies will automatically cycle power without requiring outside intervention. The backplane also contains a WDT for a system wide reset.

In the backplane the two master modules also have access to the programming pins of the MCUs. This means that they have the possibility to reprogram each other in the case of a critical malfunction. This measure should, however, only be used as a last resort due to the possibility of corruption while programming. It could prove useful if a module should critically malfunction, but should only be used as a last resort.

The master modules of the backplane also have the possibility to communicate independent of the satellite's i2c bus. This is useful because the bus and the overlying protocol are quite complex. As a result of this it could prove difficult to guarantee schedulability of the prioritized messages between the master modules in the event of a bus malfunction. The use of a separate bus with no contention for heartbeat and other status messages is a valuable feature.

The backplane contains a lot of useful functionality for configuring the satellite in a system reliability capacity and will be central as the work progresses. For further information on the capabilities of the backplane please refer to [11].

2.4.3 Electrical Power System, EPS

The EPS does not contain complex components and it is therefore not much that can be done from the perspective of the command module to increase the reliability. For more information on the EPS refer to [13].

2.4.4 On Board Computer, OBC

The MCU on the OBC is an AT32UC3-A3 with 16Mb extra SRAM and 16GB NAND flash for image processing and storage [12]. The MCU itself has a WDT and a Real-Time Clock (RTC) timer which will both be useful in a reliability context. In addition there is a lot of flash that can be used for checkpointing.

2.4.5 Radio

The chosen MCU for the radio module is AT32UC3-A3. In addition to the MCU, the radio module have two VHF radios for communication and beacon for simpler status messages. The most interesting component concerning reliability is the MCU as it is the same one that is used in the OBC. This means that it will have the same capabilities with respect to reliability, such as WDTs etc., as the OBC. It will also be placed in a master slot on the backplane with full access to the backplane logic and resources.

2.4.6 Attitude Determination and Control System, ADCS

The ADCS system is very important for the satellite. It will deploy a magnetometer and a star camera to determine the attitude and spin rate of the satellite and coils to orient the satellite in the Earth's magnetic field. This subsystem can use a lot of battery power and from a reliability standpoint it is important to ensure that it does not operate unnecessarily.

2.4.7 Payload

The primary payload is a camera for atmospheric studies. The Department of Physics wishes to study gravity waves in the upper atmosphere. In fluid dynamics, gravity waves are waves generated in a fluid medium or at the interface between two media (e.g., the atmosphere and the ocean) which has the restoring force of gravity or buoyancy [15]. The collected data is going to be used to improve current atmospheric models used for weather prediction. The mission will be considered a success if one series of pictures is completed and downloaded.

A secondary payload in the form of a wireless bus is also planned. In the event that the camera is not ready for the scheduled launch or malfunctions while in orbit, the satellite will still be able to contribute research data. A wireless system bus has a number of favorable characteristics, and being able to investigate them would be very beneficial. The possibility to use wireless buses would decrease the weight of a satellite while at the same time solving troublesome wiring configurations. Some satellites use rotary configurations in fuel systems or for scientific experiments. With a wireless bus problematic and heavy components like rotary connectors can be eliminated. Even though the wireless bus may not have high enough availability or reliability for critical subsystems, they can be beneficial for scientific instruments, and for NUTS the redundant bus would be a valuable reliability feature.

2.4.8 Operating System

The satellite will deploy the FreeRTOS operating system running on OBC and Radio, possibly on the ADCS and Payload as well. FreeRTOS is an open source real time OS that is module based and easy to customize to different configurations. It is very light weight, support threads and tasks and can be configured to have a POSIX simulator. The addition of an operating system is considered a good thing in a reliability context and for NUTS. There are several advantages:

Scheduling ensures the proper execution of high priority system tasks. This is useful to properly ensure that a few resource hungry tasks do not obstruct the rest of the system.

Memory protection is very useful for a system with many different tasks. It ensures that it is not damaging for the system if a task should run out of memory or if there is some faulty memory management in one of the processes. In NUTS there is also some very memory intensive applications such as video processing.

Communication stack allows for the lower level inter process and inter module communication to be abstracted away from the rest of the development.

File system stack allows for easy storage and retrieval of data. By using an OS it is possible to port the well tested and well performing YAFFS2 file system for flash storage.

The reasons detailed above allow the teams working on the specific applications not to be burdened with system level programming and should supply a better developing environment.

2.4.9 Cubesat Space Protocol, CSP

CSP was devised at Aalborg University in 2008. It is a small network-layer delivery protocol designed for CubeSat missions [16]. This is an important addition to the project as it allows the use of an advanced protocol for module and ground station communication, without incurring much of a development burden. The CSP protocol implements drivers (layer 1), MAC interfaces (layer 2), network router (layer 3) and a reliable datagram protocol (RDP) in the transport layer (layer 4) [16]. It is planned to be used at both the ground and space segment.

As part of system wide reliability, CSP plays a big role. The protocol is the fundament for communication between the satellite's different modules and subsystems, and will be used to determine the correct operation of said modules. It would be beneficial to be able to determine the correct functionality of the protocol. The work that has been done regarding the reliability of CSP can be found in section 4.2

Chapter 3

Suggested Solutions

The majority of the work in this paper has been to study the satellite and its systems and suggest solutions to the problems that are likely to be encountered. The Error Detection And Correction (EDAC) module will ensure that the SEPs that occur in memory are detected and corrected. Checkpointing is a proven technique to ensure that the system does not lose data or context while recovering from failures. Master-Slave functionality allows for a spare control computer in case the main crashes. The Watch Dog Timer (WDT) ensures that the system does not deadlock while interfacing with other system components. A periodic reset protects against any undetected failures that linger in the system. The ability to disable faulty modules safeguards against a malfunctioning module affecting the rest of the system. Finally, the ability to perform an integrity check on the program memory ensures that possible errors can be detected and restored. A more detailed overview follows below.

3.1 Error Detection and Correction, EDAC

It would be advisable to have a system task that is in charge of secure storage of variables. Due to the random nature of the expected faults it would be difficult to determine if the data variables are safe to operate on. To guard against this it would be advisable to store the variables multiple times in order to be able to do a majority voting on the correctness or have an error correcting algorithm such as Bose-Chaudhuri-Hocquenghem (BHC) [4, p. 155] codes to correct the faults in run-time. This would add a large amount of complexity to the individual tasks running on the satellite. At the same time it is something that would be advisable for most of the tasks, due to the expected number of SEUs that will affect the system memory and accumulate over time.

A specialized secure storage system task could ease the programming burden for the rest of the designers by removing the sometimes complex algorithms from the smaller programs. A module based design is also favorable in programming because of the increased ease of maintaining and ensuring the correctness of smaller modules. This point applies even more for reliable systems [3, p. 202].

3.2 Checkpointing

Checkpointing is a proven solution in software system redundancy. It works by storing the system state that is necessary for continued execution and completion of the process, at specific points during process execution [3, p. 214]. This enables the system to roll back in the case of an error or initialize quickly and without losing critical data in the event of a system restart. An important feature to ensure is the ability to roll back multiple instances in the case of some unforeseen fault being present in the restored system.

Power cycling of faulty modules is already implemented in the backplane. The modules of the satellite must therefore tolerate a sudden reset without losing any significant amount of work. There are also some events such as antennae deployment and detumbling that should only be executed once and including these events in the saved system state will provide a simple measure of ensuring progress for the satellite.

3.3 Master-Slave

The design of the backplane allows for two master modules that can control the backplane logic and communicate independently of the system bus. This allows for master-slave functionality in the control of the system. This enables one module to take over operations if the other one should fail. With the assumption that it is unlikely for both modules to fail at the same time, this should ensure the continued operation of the satellite.

An added complexity to the traditional master-slave setup is that the modules do not have the same capabilities. The radio module, for instance, is the only module that can communicate with the ground segment and the OBC is the only module with sufficient RAM and storage to perform image processing from the payload. The logical conclusion of this is that the system is not able to operate normally if one of the modules fail, it and should instead focus on power-cycling or repair of the faulty module whenever possible.

When this is implemented as tasks running on the subsequent modules there are a number of things to keep in mind.

Simple code is more analyzable since it has fewer states. A small amount of code is also less likely to experience SEP.

Independent software should be developed to interact as little as possible with the rest of the system. By limiting the interaction with the rest of the system's software, the possibility of being trapped in a deadlock or waiting for an unavailable resource decreases.

Independent communication from the system bus should be possible. A high level reliable transmit protocol is complex to analyze and also has numerous points of failure. It can also be difficult to ensure its real time capabilities.

3.4 Watch Dog Timer, WDT

A very useful and often employed concept in reliable computing is the watchdog timer (WDT). The basic functionality of the watchdog timer is fairly simple. The running program must periodically reset the WDT and if this fails the system will restart. The WDT should be employed every time the system performs and input, output or waits for an internal module. It is also possible that the system enters an unrecoverable state in other sections of the code if a SEP, as described in section 1.4.3, occurs. A WDT also protects against weaknesses in the system design and ensures that the system will not freeze in the event of an untested software bug [3, p. 130].

All of the MCUs used in NUTS have internal WDT modules that should be used. In addition it is also possible to use an external chip with WDT functionality. This chip is located on the backplane and will provide a full system reset. Given the use of WDTs on each module and the possibility of losing data in the entire satellite, the WDT on the backplane should only be activated if both of the control computers fail to respond.

3.5 Periodic Reset

When designing the EDAC service it is difficult to guarantee full coverage of the error detection. Some errors may be left undetected and the EDAC service might be overwhelmed in periods with high radiation intensity (e.g. when passing through the South Atlantic Anomaly). Another consideration is that the EDAC service is not meant to be on a system wide level due to the overhead of the implementation, especially for multiple-error-correcting codes [3, p. 147].

In modern components it may also be difficult to properly detect SEL events, since both high and low current SELs can occur. The high current SEL is caught by the backplane which triggers an automatic reset on excessive current consumption, but no such mechanisms exist for low current SELs is the design.

The possibility for undetected memory corruption and low current SELs is a real concern as they are both difficult to determine properly. To solve this problem a periodic power cycle of all the modules is suggested.

3.6 Disable Faulty Modules

Given the possibility for failure in software or hardware modules it should be possible to disable the faulty modules without compromising the rest of the system. The main difficulty with this is to construct algorithms to determine what modules that are not operating correctly. Generally speaking the modules can fail in two ways, either a silent failure where the module becomes unresponsive, or what is known as a babbling-idiot failure [7]. A babbling-idiot fault typically occurs when a node occupies the bus and transmits high-priority messages at erroneous time instants so frequently to cause additional delay in the communication of properly operating nodes. The worst-case scenario happens when a node keeps the bus continuously busy, thus inhibiting every communication between the other nodes [7].

Fault determination algorithms can also fail themselves. This is mitigated with the presence of two control modules and the assumption that only one will critically fail at a time. This is the same number of control computers that NASA originally had on the space shuttle engine control [9]. It is considered safe because of the low probability of radiation hitting the same part of the logic in two separate modules. Once the faulty module is determined it can be disabled via the backplane logic.

3.7 Program Integrity Check

In addition to secure storage there should be a periodic subroutine that performs a Cyclic Redundancy Check (CRC) on system flash in order to determine if there has been corruption of the program flash. This can be done alternately by two or more different but identical functions placed in different parts of the flash to mitigate the risk of an error in the CRC function itself. The function should be possible to implement in very little code, so this is a feasible solution.

It should be possible to store copies of the main program in the large flash data-bank available. If this data is maintained by an error-checking subroutine and checked before flashing other modules it should be reasonably safe. In case of failure of part of the flash-bank, the programs could be stored multiple times on different parts of the flash.

One could argument that self repairing code is better than a boot-loader that fetches a securely stored version of the program from One-Time Programmable (OTP). The boot-loader could be corrupted by a SEP. While unlikely, this still constitutes a single point of failure. A possible workaround could be not to have the boot-loader in protected flash space. This would be dangerous for obvious reasons, but it would also allow for the execution of error correcting code on the boot-loader itself. These considerations do of course constitute a sufficiently robust error correcting code, but that should be within the scope of this work. It would be advisable to execute error detecting and correcting code upon restart or power-up of the system. It could be it is possible to implement this as a boot-loader with the added ability to check itself and to load error free code from a secure location. Error correcting code also reduces the problem with SEP while loading the code from the secure backup and while reprogramming or boot-loading the device. Although this is very unlikely given the expected rate of SEPs it would still need to be considered as a possibility, and one would need to make a decision regarding the possibility of SEPs taking place and possible countermeasures.

Last but not least the number of detected radiation induced errors should be logged. This is useful both for future designs and to be able to judge the reliability of the satellite.

3.8 Testing

The most realistic test would be to expose the system to a radiation environment and measure how the system holds up under real stress. While this might be desirable for the finished system it is not very useful when testing specific algorithms or sub modules in the system. The reason for this is that it is very difficult to control which module is to be tested and next to impossible to replicate the exact error conditions in order to determine the severity of the fault.

Another alternative is to simulate random error occurrence via JTAG in the software running on the board. This is somewhat better because the efficiency of the error correcting code can be determined directly since the number of inserted faults is known. Arguments against this testing regime are the lack of realistic errors. Latchup, for instance, is hard to simulate in software.

With these considerations in mind, the preferred testing method is to simulate errors with JTAG injection of faults during runtime. This is the most economically viable option while at the same time allowing for repeatable test runs and focus on specific parts of the system.

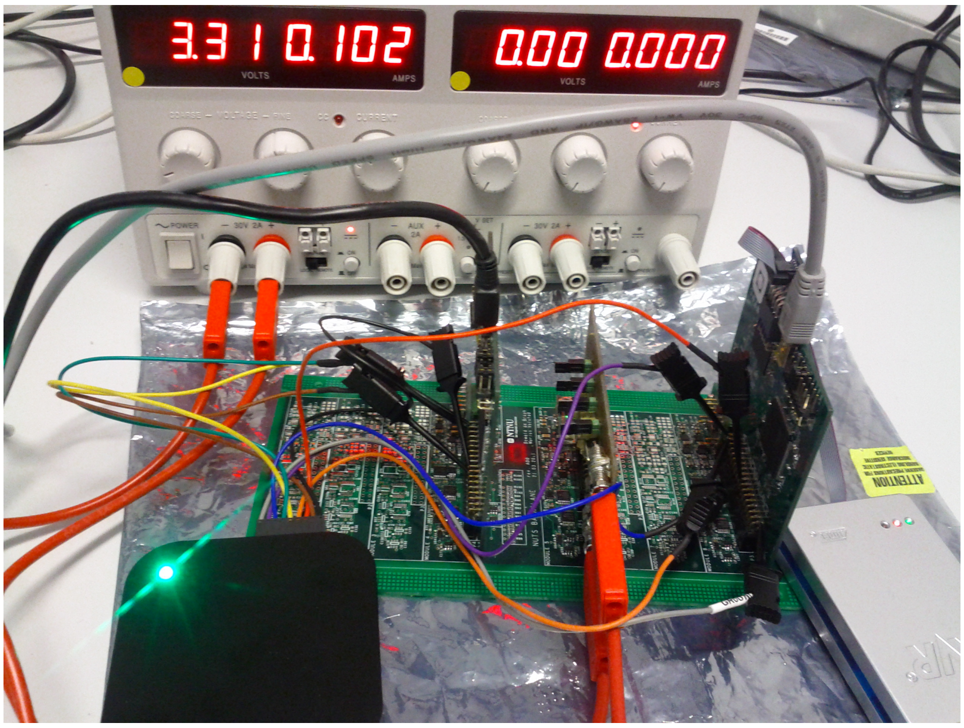

Figure 3.1: Test setup

Chapter 4

Work

In addition to the literary study and review of possible solutions for the satellite, there have been conducted more detailed studies on a couple of subjects. A presentation of this work is given in this chapter.

4.1 Reliable module

With MCUs and other electronic components continuously falling in price and increasing in performance, the possibility of incorporating redundant hardware modules into a number of areas have opened up. The purpose of this work is to study how high reliability a control system or reliable co-processor using COTS components can achieve.

While this reliable processor or co-processor setup could be useful for NUTS, the project is in such a late state that it is not desirable to add extra features in form of added hardware. In addition there already exists one backup pair of processors which makes this modification more fitting for the next generation of satellite. This work is not interesting work for the satellite project, but an interesting subject to look into on its own.

4.1.1 Specification

Multiple features were considered for the specification. A brief overview of the considered features and their rationale is presented.

Mature Components

To make the module more resistant against radiation effects one could use older and more better tested logic families. When the transistors get smaller the charge required for SEP is much less. This is, however, a bit outside the scope of this paper. The goal is to have the highest possible reliability in space environments for modern and low-cost hardware. In addition, older process technologies use additional power, something that is not ideal for CubeSats. If the reliable co-processor is going to be use in other environments where radiation is not a big issue, this point also becomes moot.

Verification Logic

The required verification that the results of the MCUs are equal is planned to be implemented in discrete logic. This would also ensure that the MCUs cannot pull the pins high and low at the same time as this can cause problems if the pins are not adequately protected.

Shielding

Shielding of the module is also possible. Shielding of COTS components is, however, a bit of a futile effort, especially against SEP events [8]. If shielding is done if would primarily affect total dose and therefor the lifetime of the CubeSat. The shielding would be heavy and have a complicated mechanical design with the withstand the mechanical forces the satellite would be subjected to during launch. It is therefore of the authors opinion that it would be better to use radiation tolerant or even radiation hard components than to deploy shielding. In most designs some shielding would be provided by the frame aluminum frame of the CubeSat. NUTS employs a carbon fiber frame and it could therefore be interesting to explore the effects and number of errors in shielded vs. unshielded cards, but this would require a laboratory with radiation equipment.

Power Cycle

The module could also have the capability to power cycle the processors individually. The module could then survive SEL events by power cycling the affected processor. To detect a malfunction a rudimentary WDT could be implemented on one of the communication pins between the processors.

If a processor fails to demonstrate liveliness by toggling a pin, the other MCU could perform a reset.

4.1.2 Method

The resulting schematics and board was made to be easy to assemble. This is the reason for choosing large, easy to solder, through hole components as can be seen in figure 4.2. The respective MCUs, oscillators and LEDs were already available on the institutes' lab, something that simplified the sourcing of components.

The ATMega328 was chosen as the MCU on the board. It was selected because of previous experience with the MCU family, something that eased the design burden substantially. This is also a mature product line and there is a lot of previous work available in the public domain.

Some time was spent trying to incorporate comparison logic on the output of the board. This work had to be abandoned in part because of the choice to use the relatively large through hole components, and in part because of the more complex routing.

4.1.3 Results

In figure 4.1 and 4.2 the schematics and board layout for the reliable module is presented.

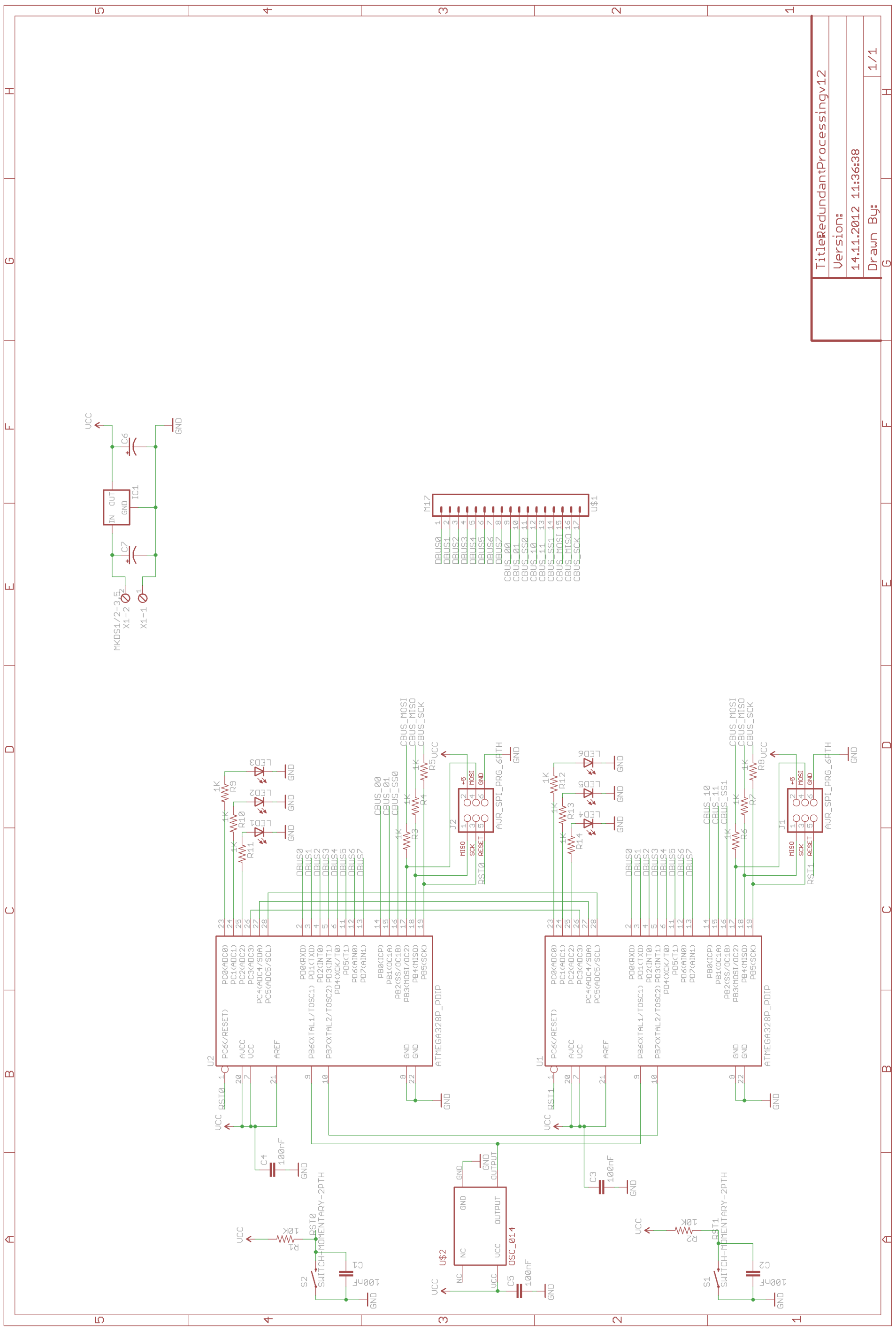

Figure 4.1: Reliable Board Schematics

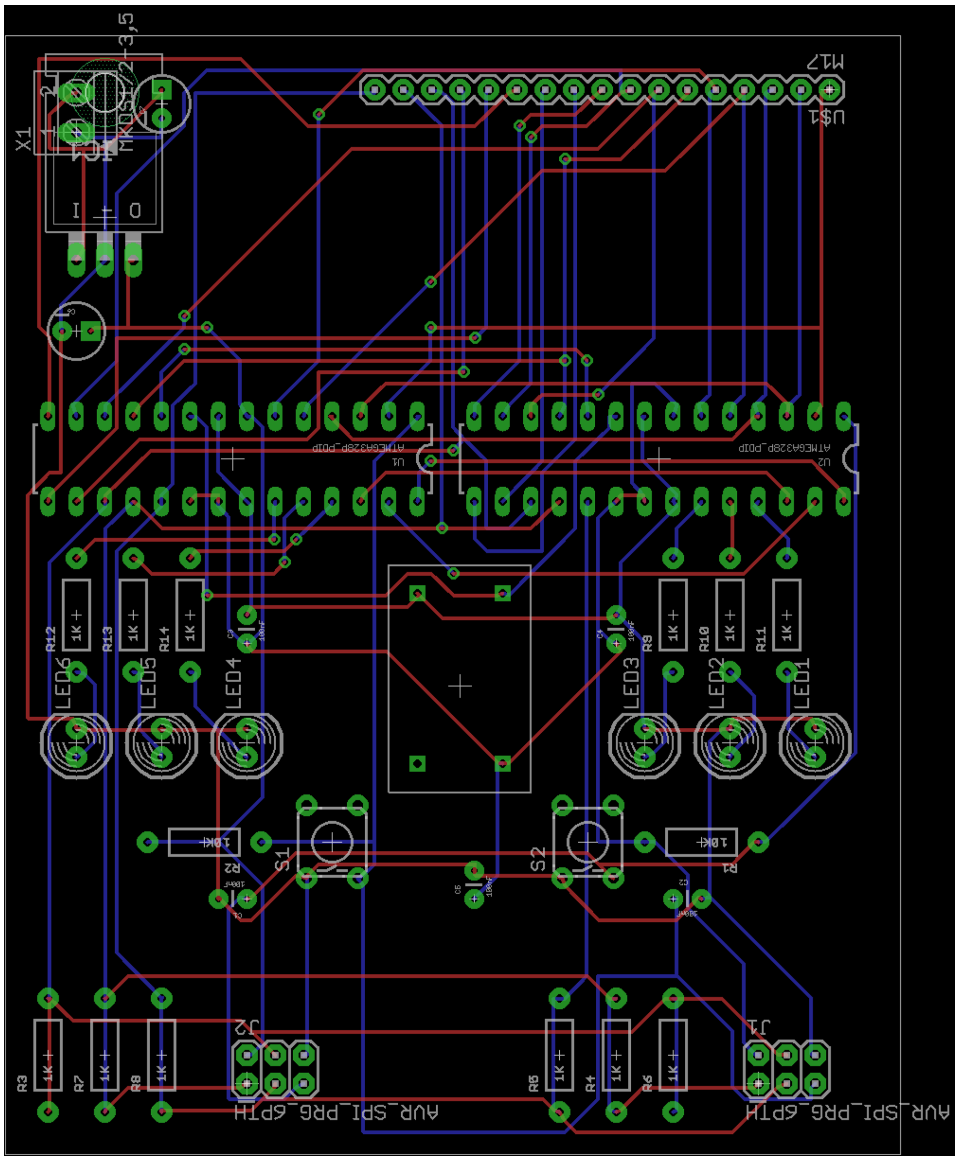

Figure 4.2: Reliable Board Layout

4.1.4 Discussion and Conclusion

There still remains a lot of work on the reliable module, but valuable lessons were learned in this first version. The work regarding choice and sourcing of components is time consuming and should be concluded early in the design process. After this the layout and routing should be started. This can be tedious work, especially for smaller boards. And added complexity in the reliable module was the shared clock line, and it is considered a good idea for a future board to have clock generation with multiple outputs.

The biggest result of this work has been the valuable experience gained during the design process, and evaluating solutions for future designs. A smaller board using more advanced components is planned for future work. This module will not be a co-processor but rather the main processor in a system. Due to the considerable work and board space required to have comparison logic in the original design, it will feature two processors running back-to-back with one verifying the results of the other.

4.2 System analysis of CSP with FSP and LTSA

4.2.1 Introduction

In order to determine to what degree it is possible to trust the Cubesat Space Protocol (CSP) network it was planned to analyze the protocol formally. CSP will be the main method of communication between the modules and also to some extent for internal communication between tasks running on the same module. CSP allows creating sockets for inter process and network communication similar to the ones used on Linux. This message based communication is preferable in a real time and reliability context due to the reduction of complexity in the program.

The use of a message based communication between tasks would require a reliable medium of communication. CSP is a relatively new protocol and although it has been deployed on a satellite it cannot yet be considered mature. In order to determine if this protocol is useful it is planned to device a process algebraic model of CSP and test this representation. A suitable approach for this is a Finite State Process (FSP) representation in the Labelled Transition System Analyser (LTSA) Java tool.

4.2.2 Implications

If CSP turns out to be reliable this would be useful for the further reliability design in the system. Barring failure in hardware the protocol itself could be said to be reliable and could be used safely for inter task communication.

The bus could still malfunction, but since there are two physically separate buses with their own bus repeaters on the backplane, these failures can be assumed to be quite rare. The expected number of SEUs in LEO are $10^{-5} error/bit-day$ which translates to roughly 10 errors per day in internal SRAM and 160 errors per day in external SRAM under normal conditions.

4.2.3 Documentation on CSP

The Cubesat Space Protocol implements a Reliable Datagram Protocol (RDP) in accordance with RFC-908. Further documentation can be obtained from [16].

4.2.4 Method

A process algebraic model of CSP was made with the LTSA tool and with help from supporting literature [2] [5]. The bulk of this work was to accurately translate the protocol from documentation and source code into a viable FSP model.

In order to create a system model there has to be some assumptions on which components will fail and how these failures will manifest themselves. This is called the system's error model. For the transmission line the error model was fairly simple.

There can be an error at the most every third transmission, and if an error occurs the two next transmissions will be error free. In Communicating Sequential Processes (a superset of FSP) notation this translates to:

$$\begin{aligned} E_0 &= left?x \rightarrow (right!x \rightarrow E_0 \sqcap right!(1-x) \rightarrow E_2) \\ E_{n+1} &= left?x \rightarrow right!x \rightarrow E_n \quad \text{for} \quad n = 0,1 \end{aligned}$$

The theory and rationale behind this choice is available in [5, Sec. 5.1].

This might seem a bit simplistic, especially since errors often comes in bursts. A more complex error model could be constructed, but the solution would still be to retransmit lost transmissions until contact in achieved. The increased number of retransmits would require more power, something which is a limited resource on the satellite. With this in mind is seems more reasonable to limit the number of retransmits and wait until the ground station initiates another transmission.

4.2.5 Model Code

```
//Bounded buffer for CSP, limited to 5 packages
//error is bitwise error in the message

BUFFER_SEND(N=5) = COUNTS[0],
COUNTS[i:0..N] =
        (when (i<N) put_send->COUNTS[i+1]
        |when (i>0) get_send->COUNTS[i-1]
        ).
```

```
APPLICATION1 =
         (put_send -> send_complete -> APPLICATION1
         |put_send -> send_fail -> APPLICATION1).

SEND_CALL = (get_send -> send -> ack -> send -> ack
         ->send -> ack -> send_complete -> SEND_CALL).

BUS (N=2) = BUS[0],
BUS[i:0..N] =
         (when (i==0) send -> recv -> BUS[0]
                  | error -> BUS[2]
         |when (i>0) send -> recv -> BUS[i-1]
         ).

BUFFER_RECV(N=5) = COUNTR[0],
COUNTR[i:0..N] =
         (when (i<N) put_recv->COUNTR[i+1]
         |when (i>0) get_recv->COUNTR[i-1]
         ).

RECV_CALL =
         (recv -> ack -> put_recv -> recv_complete ->
                 RECV_CALL
         |error -> recv -> ack -> put_recv ->
                 recv_complete -> RECV_CALL).

APPLICATION2 =
         (recv_complete -> get_recv -> APPLICATION2).

||CSP = (APPLICATION1 || BUFFER_SEND(5) || SEND_CALL ||
         BUS ||
         RECV_CALL || BUFFER_RECV(5) || APPLICATION2).
```

4.2.6 Results

The result of this study of CSP shows that the implemented model of the protocol is deadlock and livelock free. Beyond that, the LTSA tool does not provide any insights or analysis of the system.

4.2.7 Discussion

The model was based on the system described in the documentation and on the provided source code. In the process of building a model based on an already implemented system it is difficult to make any guarantee that the process algebra that have been created from the source code is an accurate description of the system.

The protocol is also open source and this is why NUTS is able to use it in the first place. In order to be able to make a definitive statement of the reliability of the implemented code one would need to deploy some sort of automatic translation tool. Otherwise it would prove very time consuming to maintain a correct model of CSP.

The two other teams working with the implementation of CSP in the project ran into some problems. They fixed these problems by making changes to the code for the radio and the satellite bus in order to have a functional system. The result of these changes is that the implemented process model is no longer valid, and the drawn conclusions regarding the validity and reliability of the protocol has to be re-evaluated.

4.2.8 Conclusion

The implemented model of CSP was deadlock and livelock free with the provided error model, and the specification of the protocol seems to be robust from this standpoint.

However, due to changes in the implemented protocol from various groups in the project, the model used in the verification is no longer valid. It is difficult to get an accurate representation of the implemented system in the process model and the extra work needed is not deemed a reasonable endeavor.

Formal verification methods are most useful in the specification phase and the techniques used on CSP might prove useful in the specification of the planned reliability measures for the satellite.

The final word on the conclusion is that the model that was used when designing the specification for the protocol is valid.

Chapter 5

Conclusion

During this project, the focus of the work has been to study the satellite to date and present possible solutions in order to implement a reliable overall system. The bulk of the work has been to understand the satellite's systems and reason which solutions that are most fitting to solve the expected problems.

The presented suggested solutions describe the different problems that are expected to affect the satellite. Further, it details how these problems can be solved with the constraint of using the already developed satellite systems.

The work done in this project has been a more thorough examination of some of these solutions. There is some work left, but a lot of valuable experiences regarding formal system specification and PCB design and fabrication have been gained.

The future work will be to implement the suggested solutions on the satellite's systems. At the time of writing there are only two modules at such an advanced stage that they are physically implemented and ready for testing. The OBC and backplane, while not in their final revision, are ready for code development and testing. The EPS design is finished but has not yet been implemented in hardware. Finally, the radio, ADCS and payload modules are under planning. The future work is to implement the suggested solutions as the other modules of the satellite approach completion.

Bibliography

[1] Wiley J. Larson and James R. Wertz, *Space Mission Analysis and Design.* Addison Wesley, Massachusetts, 2nd Edition, 1994.

[2] Jeff Magee and Jeff Kramer, *Concurrency, State Models & Java Programs.* Addison Wesley, Massachusetts, 2nd Edition, 1994.

[3] Daniel P. Siewiorek and Robert S. Swarz, *Reliable Computer Systems, Design and Evaluation.* Digital Press, Burlington, 2nd Edition, 1992.

[4] Hazarathaiah Malepati, *Digital Media Processing, DSP Algorithms Using C.* Newnes, Burlington, 2010.

[5] Andrew William Roscoe, *The Theory and Practice Of Concurrency*, Prentice Hall (Pearson), 2005.

[6] Olof Hannius and Johan Karlsson, *Impact of Soft Errors in a Jet Engine Controller*, Computer Safety, Reliability, and Security Lecture Notes in Computer Science, Volume 7612, Springer, 2012.

[7] Giuseppe Buja, Juan R. Pimentel and Alberto Zuccollo, *Overcoming Babbling-Idiot Failures in CAN Networks: A Simple and Effective Bus Guardian Solution for the FlexCAN Architecture*, IEEE TRANSACTIONS ON INDUSTRIAL INFORMATICS, VOL. 3, NO. 3, AUGUST 2007.

[8] *Space Radiation Effects on Electronic Components in Low-Earth Orbit*, PRACTICE NO. PD-ED-1258, JPL NASA, APRIL 1996.

[9] *Computers in the Space Shuttle Avionics System*, Computers in Spaceflight: The NASA Experience, NASA, June 6, 1986, Retreived December 8, 2011.

[10] Sammy Kayali, *Space Radiation Effects on Microelectronics*, JPL NASA

[11] Dewald De Bruyn, *Power Distribution and Conditioning for a Small Student Satellite*, Trondheim, June 2011.

[12] Dan Erik Holmstrøm, *The Internal Data Bus in a Student Satellite*, Trondheim, June 2012.

[13] Lars Erik Jacobsen, *Electrical Power System of the NTNU Test Satellite*, Trondheim, June 2012.

[14] Marianne Bakken, *Signal Processing for Communicating Gravity Wave Images from the NTNU Test Satellite*, Trondheim, July 2012.

[15] Gravity wave, `http://en.wikipedia.org/wiki/Gravity_wave` Wikipedia.

[16] libcsp/doc, `https://github.com/GomSpace/libcsp/tree/master/doc`

[17] Cubesat Specification, `http://www.cubesat.org/images/developers/cds_rev12.pdf`

[18] CubeSat mission statement, `http://cubesat.org/index.php/about-us/mission-statement`