



Norwegian University of  
Science and Technology

# NUTS Backplane Revision

Improving Reliability and Modularity of the  
NTNU Test Satellite

**Stian Haug**

Master of Science in Electronics

Submission date: July 2016

Supervisor: Bjørn B. Larsen, IET

Co-supervisor: Roger Birkeland, IET

Norwegian University of Science and Technology  
Department of Electronics and Telecommunications



# Abstract

This paper details the process of reviewing and updating the backplane circuit board of the test satellite designed at NTNU. It covers introductory theory on the subject, a complete review of the current design, proposed updates and a schematic implementation of proposed updates to the backplane system. A brief discussion and a conclusion on how the system has been successfully improved and simplified is presented.



# Contents

<b>Abstract</b>	<b>1</b>
<b>List of Figures</b>	<b>7</b>
<b>List of Tables</b>	<b>9</b>
<b>Abbreviations</b>	<b>11</b>
<b>1 Introduction</b>	<b>13</b>
1.1 Summary . . . . .	13
1.2 Summary - Norwegian . . . . .	14
1.3 Problem description . . . . .	15
1.4 NUTS . . . . .	16
<b>2 Previous development and relevant theory</b>	<b>19</b>
2.1 NUTS satellite overview . . . . .	19
2.1.1 Backplane . . . . .	20
2.1.2 OBC - On Board Controller . . . . .	20
2.1.3 ADCS - Attitude Determination and Control System .	21
2.1.4 EPS - Electronic Power System . . . . .	21
2.1.5 Antenna module . . . . .	21
2.1.6 Payload . . . . .	21
2.2 Backplane design considerations . . . . .	22
2.2.1 Electronics in space: <i>Radiation, vacuum and temper-</i> <i>atures</i> . . . . .	22
2.2.2 Fault isolation, detection and recovery . . . . .	25
2.3 BUS evaluation . . . . .	26
2.3.1 I <sup>2</sup> C bus . . . . .	26
2.3.2 CAN bus . . . . .	27

---

<b>3</b>	<b>Design proposition</b>	<b>33</b>
3.1	Proposed backplane design . . . . .	33
3.1.1	Data carrier . . . . .	33
3.1.2	Sensor network . . . . .	36
3.1.3	Power distribution . . . . .	38
3.1.4	Lab access and programming . . . . .	41
3.1.5	Backplane Control . . . . .	42
3.1.6	Module slot requirements . . . . .	45
3.1.7	Fault isolation . . . . .	47
3.1.8	Proposed design summary . . . . .	49
<b>4</b>	<b>Design implementation</b>	<b>51</b>
4.1	Backplane implementation overview . . . . .	51
4.2	Data carrier (implementation) . . . . .	51
4.3	Power distribution (implementation) . . . . .	52
4.4	Sensor network (implementation) . . . . .	54
4.5	Lab access and programming (implementation) . . . . .	57
4.6	Backplane Control (implementation) . . . . .	59
4.6.1	CAN IO expanders . . . . .	59
4.6.2	Slot state holder . . . . .	60
4.6.3	Control-logic power and power cycling . . . . .	64
4.6.4	Module programming and inter-module programming . . . . .	65
4.6.5	Control intervals and fault avoidance . . . . .	68
4.7	Module slot (implementation) . . . . .	69
4.8	Fault isolation (implementation) . . . . .	70
4.8.1	Fault isolation control . . . . .	72
4.9	Proposed testing methodology . . . . .	74
4.9.1	What to test (functionality) . . . . .	74
4.9.2	How to test (physical access) . . . . .	74
4.9.3	Test bench . . . . .	74
<b>5</b>	<b>Results</b>	<b>77</b>
5.1	Verification of power control signals. . . . .	77
5.1.1	Test A . . . . .	77
5.1.2	Test B . . . . .	79
5.1.3	Test C . . . . .	80
<b>6</b>	<b>Discussion</b>	<b>83</b>
<b>7</b>	<b>Conclusion</b>	<b>85</b>

<b>8 Bibliography</b>	<b>87</b>
<b>9 Appendix</b>	<b>89</b>
9.1 Slot state update interval . . . . .	89
9.2 Design Schematics . . . . .	91





# List of Figures

1.1	3D model of NUTS backplane based design . . . . .	16
1.2	Stacked PCB versus Backplane based PCBs . . . . .	17
2.1	Block schematic overview of the NUTS system . . . . .	19
2.2	Overview of the backplane and modules . . . . .	20
2.3	TID effect on an NMOS transistor. . . . .	23
2.4	DD effect on a bipolar junction transistor. . . . .	24
2.5	SEE on an NMOS transistor caused by cosmic rays . . . . .	25
2.6	Typical I <sup>2</sup> C bus structure . . . . .	27
2.7	Data transfer on the I <sup>2</sup> C bus . . . . .	28
2.8	The layered ISO11898 standard architecture . . . . .	28
2.9	Typical CAN bus structure . . . . .	29
2.10	Standard CAN message structure . . . . .	30
3.1	Data bus using I <sup>2</sup> C structure from current design . . . . .	35
3.2	Sensor bus using I <sup>2</sup> C from current design. . . . .	37
3.3	New sensor bus design using I <sup>2</sup> C . . . . .	39
3.4	Current power distribution in the backplane design . . . . .	40
3.5	Current backplane houskeeping system . . . . .	44
3.6	Proposed new backplane houskeeping system . . . . .	46
3.7	Current isolation towards module slots preventing fault propagation. . . . .	48
3.8	Proposed new isolation towards module slots to prevent fault propagation. . . . .	50
4.1	New backplane implementation overview . . . . .	52
4.2	New implementation of the data carrier using CAN bus . . . . .	53
4.3	Powerdistribution implementetion using E-fuses . . . . .	55
4.4	Sensornetwork power measurement implementation . . . . .	56
4.5	I <sup>2</sup> C multiplexer design . . . . .	57

---

4.6	Sensornetwork module slot I <sup>2</sup> C repeater implementation. . . .	58
4.7	High density connector for lab programming . . . . .	58
4.8	Slot controller block schematic . . . . .	61
4.9	Implementation of the slot controller CAN IO expanders . . .	62
4.10	Implementation of the slot controller state holders . . . . .	63
4.11	Housekeeping module power implementation. . . . .	66
4.12	Housekeeping power control signal implementation . . . . .	66
4.13	Module programming block schematic . . . . .	67
4.14	Inter module re-programming block schematic . . . . .	68
4.15	Layout of the backplane slot headers . . . . .	71
4.16	I <sup>2</sup> C bus repeater for module isolation . . . . .	71
4.17	JTAG bus isolation switches for module isolation . . . . .	72
4.18	Utility/reset signal fault isoaltion . . . . .	73
5.1	Breadboard test scircuit . . . . .	78

# List of Tables

4.1	Controllable parameters of the backplane. . . . .	59
4.2	Power states of the backplane logic . . . . .	65
4.3	Slot control signal fault consequences . . . . .	69
5.1	Measurements form Test A . . . . .	79
5.2	Measurements form Test B . . . . .	80
5.3	Measurements form Test C . . . . .	81



# Abbreviations

**ADCS** Attitude Determination and Control System

**CAN** Controller Area Network

**DD** Displacement Damage

**EPS** Electronic Power System

**FET** Field-Effect Transistor

**FPGA** Field-Programmable Gate Array

**I<sup>2</sup>C** Inter-Integrated Circuit

**IC** Integrated Circuit

**I/O** Input/Output

**IRQ** Interrupt Request

**ISO** International Standardization Organization

**JTAG** Joint Test Action Group

**LED** Light-Emitting Diode

**MCU** Micro Controller Unit

**MUX** Multiplexer

**OBC** OnBoard Controller

**PCB** Printed Circuit Board

**SDA** Serial Data Line

**SEE** Single Event Effects

**TID** Total Ionizing Dose

**UHF** Ultra High Frequency

**VHDCI** Very-High-Density Cable Interconnect

**VHF** Very High Frequency

# Chapter 1

## Introduction

### 1.1 Summary

The NTNU Test Satellite is double CubeSat being built at NTNU. The main payload of the satellite is a camera for earth observations. The satellite is based on a modular design with a "motherboard" style backplane where modules are placed in slots. In the current revisions of the backplane, lack of documentation and partially incomplete schematics combined with an increase in complexity over several iterations has made the task of realizing the backplane within an acceptable time-frame difficult. For data transfer the current backplane makes use of two parallel I<sup>2</sup>C buses for redundancy and a custom made arbitration bus for arbitration between the two OBCs (On-board Controllers) in the satellite. This results in a complex arbitration scheme having to be implemented in software with the potential of fault propagation over the arbitration bus between modules. A new design is proposed using CAN bus with built in arbitration with priority based on message address. Due to the robust nature of the CAN bus a single bus is considered able to replace the two I<sup>2</sup>C buses. In the current design control signals for the backplane are sent using a single wire custom serial bus implementation based on shift-registers. The proposed new design connects the control of utility functions and housekeeping on the backplane, such as module power management, to the CAN bus. A new schematic design for the backplane is presented and test methodology suggested. The design is partially assembled on breadboard and tested. Lastly a partial hardware test and verification is presented before a conclusion is reached on the reduced complexity and future ease of implementation when it comes to board layout and production.

## 1.2 Summary - Norwegian

Ved NTNU bygges det en test saltetitten basert på en dobbel CubeSat. Hovedlasten til satellitten er et kamera for atmosfæriske observasjoner. Satellitten bruker et modulært design basert på et sentralt hovedkort hvor andre moduler kan plasseres. Også kalt et bakplan. I den gjeldende versjonen av bakplanet har mangel på dokumentasjon og delvis uferdige skjemategninger kombinert med en økning i kompleksitet over flere iterasjoner resultert i et design som er vanskelig å implementere på en akseptabel måte. For overføring av data brukes to parallelle I<sup>2</sup>C buser for redundans og en spesiallaget arbitreringsbus mellom de to OBCene i satellitten. På grunn av dette må en relativt kompleks arbitrerings metodikk implementeres fysisk og i programvare til OBCene. Det er også potensiale for at en eventuell feil kan spre seg via arbitreringslogikken. Et nytt design er foreslått som tar i bruk CAN bus med innebygget arbitrering og meldingsbasert prioritet. CAN busen er en robust bus, noe som tillater oss å erstatte begge de redundante I<sup>2</sup>C busene med en enkel CAN bus. Videre er kontrollsignalene til bakplan logikken i det nåværende designet basert på en spesiallaget seriell bus laget med skiftregistre. I det nye designet foreslås det å koble bakplanets kontrollsignaler til CAN bussen. Videre presenteres et nytt skjemautlegg for bakplanet og test metodikk for et ferdig kort er foreslått. Deler av kontroll logikken og strømforsyningen på bakplanet er testet og verifisert ved delvis sammensetting og målinger på lab. Til slutt konkludered det med et design som reduserer kompleksiteten og gjør utlegg og produksjon av PCB betydelig lettere.



## 1.3 Problem description

The current focus of the NUTS project is to finish the design and build hardware for an integrated engineering model. Through this project, the student should focus on the top-level system design in general, with particular focus on the backplane and how it interconnects all other subsystems of the satellite.

The backplane is one of the principal components of the satellite. It distributes power and provides access to the databuses to the rest of the subsystems.

The backplane must be designed to be reliable, as maintenance is impossible after launch. Challenges due to the space environment, such as temperature cycles, radiation environment, failing electronic components and vacuum must be identified and discussed. In areas where mitigation of such problems is possible, solutions should be presented. Whether the solutions be implemented should be based on a cost/benefit analysis.

Key tasks for the student:

- Study and evaluate different interconnecting designs possible for NUTS, with a view on the previous proposed designs and prototypes.
- Evaluate and study different bus architectures, such as I<sup>2</sup>C vs. CAN-bus in addition to the needed backplane control logic.
- Propose a design, and produce this by designing it in Altium Designer.
- Propose test regimes to identify if the backplane fulfills its desired tasks.

In addition to the given tasks, the student is expected to participate in relevant group work. The NUTS project is a multi disciplinary project, which requires more involvement from the student than just the completion of the individual task and report.

## 1.4 NUTS

The NTNU Test Satellite is double CubeSat [1] being built at NTNU. According to the original Mission statement [2] the aim is to "design, develop, test, launch and operate a double CubeSat". This is a multidisciplinary project where student learning is in focus and through "hands-on" experience that aims to teach students skills needed in jobs post graduation. The main payload of the satellite is a camera for atmospheric observations. The original Mission statement [2] lists the following goals for the project:

- Deliver a tested satellite according to mission specifications
- Transmit a beacon signal that can be received by radio amateurs
- Confirm successful de-tumbling
- Establish two-way communication and receive full telemetry
- Test camera
- Initiate camera pinpointing
- Initiate camera sequence
- Receive a valid series of images



Figure 1.1: The figure shows a 3D model of the proposed backplane design used in NUTS.

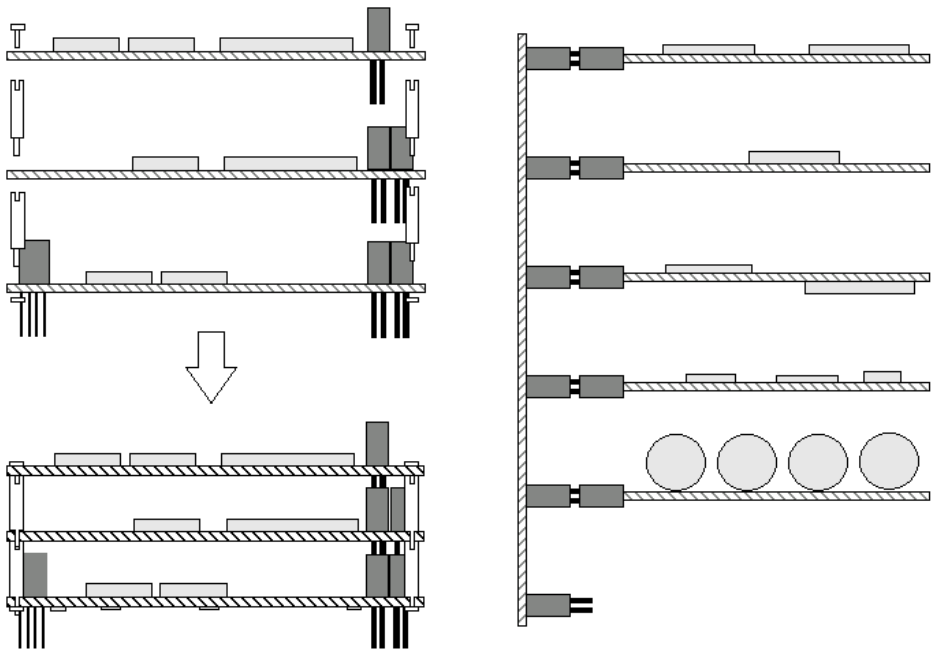


Figure 1.2: The figure shows the standard PC104 stacked card design used in most CubeSats to the left and the proposed backplane design used in NUTS to the right.

The main deviation from "standard" CubeSat designs is the choice not to use PC-104 form factor for the electronics design. Instead a backplane design is used as shown in Figure 1.1 and Figure 1.2. One of the main reasons for this decision is enhanced reliability. Using two backplane masters for redundancy both can remove other subsystems from the data bus and backplane power. The backplane essentially allows any module to talk to any other module and power management on individual module basis. The mission statement has been subject to change over the last few years.



## Chapter 2

# Previous development and relevant theory

This section will introduce the satellite and each module will be briefly described. The last part of this section will cover design considerations of particular importance to the backplane and the design decisions to be made regarding the backplane.

### 2.1 NUTS satellite overview

The NUTS satellite is a modular system with functions spread out over multiple modules as shown in figure 2.1. With a clearly defined interface between the modules this approach allows the system to be spread out over several student groups for development.

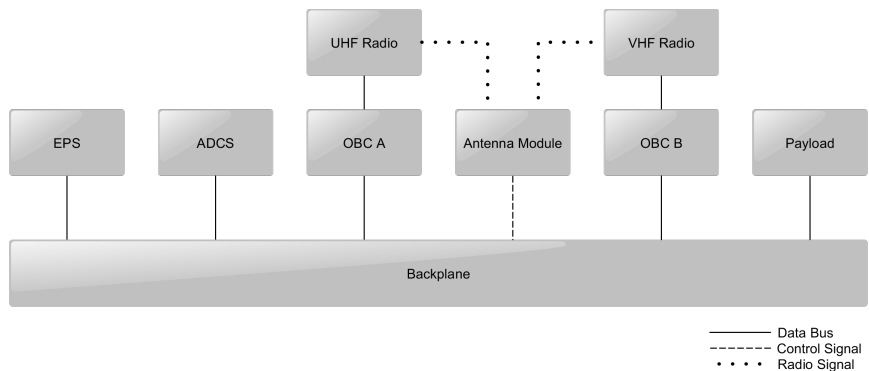


Figure 2.1: Block schematic overview of the NUTS system showing how each module relates to each other and the backplane.

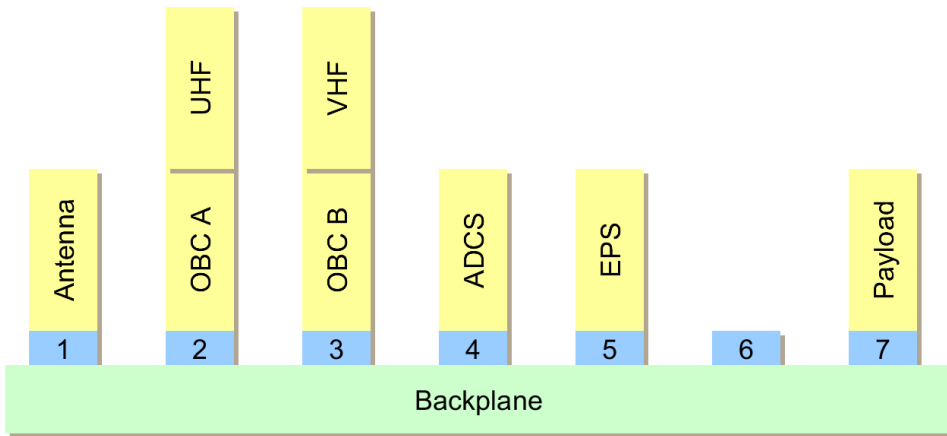


Figure 2.2: Overview of the backplane and modules of the NUTS system showing how modules are set in to slots on the backplane.

### 2.1.1 Backplane

The backplane is the "motherboard style" circuit board that other modules in the satellite slot in to as shown in figure 2.2. It is mostly transparent to the other modules and ensures a parallel configuration of the satellite modules. The principle is that any module can access any other module regardless of its placement in the backplane slot. The backplane distributes power from the EPS module to the other modules. It also ensures isolation between a module and the backplane to such an extent that faulty modules should not propagate their fault to the rest of the system. Each module has access to databuses and sensorbuses on the backplane. A selection of "housekeeping" logic states can be set on the backplane to disconnect parts of a module or a whole module from the backplane, as well as disconnecting modules from faulty power-rails in the case of a partial power failure in the backplane or on the EPS. The backplane also provides utility buses and signals allowing one module to reset another and in special cases re-program an other module via a dedicated JTAG bus between modules. When assembling the satellite in a laboratory environment the backplane provides headers for lab access allowing for programming and test of all modules while the satellite is partly or fully assembled.

### 2.1.2 OBC - On Board Controller

The NTNU Test Satellite will have two on board computers also called on-board controllers (OBC) working redundantly. The main task of the OBC

is to monitor the health of the system and execute necessary actions when particular situations demand it. The OBC periodically requests health data from software instances in other parts of the satellite and poll's sensors in the different areas of the satellite for monitoring power usage and temperatures. It also sets the system states according to battery levels. Each of the two OBC's will act as a gateway for the UHF and the VHF radios with each of the radios connected to one of the on board controllers. This will enable the satellite to communicate with the ground station in full duplex mode allowing communication in both directions simultaneously. With each of the two controllers doing either sending or receiving.

### 2.1.3 ADCS - Attitude Determination and Control System

As the satellite is detached from the launch vehicle the orientation may defer from the desired orientation. It may also have rotation along multiple axis called tumbling. In order to make use of the satellite and its payload a measure of control over the satellite is needed. The ADCS detects and affects the satellites physical orientation in space. De-tumbling is the most critical part of the ADCS system. Charging the satellite using solar panels and communication between satellite and ground station is to a degree dependent the tumbling speed of the satellite.

### 2.1.4 EPS - Electronic Power System

The Electronic Power System includes the solar panels and battery as well as the electronics required to regulate charging from the solar panels and power the satellite. It also has mechanical switches disconnecting all power from the rest of the satellite to ensure it is powered completely down during launch before it reaches orbit. Ensuring the battery can receive charge from the solar panels is the most critical part of the EPS.

### 2.1.5 Antenna module

The antenna module contain the antennas and the deployment electronics. The antennas will be released based on a timer activated on power up. Redundancy is also present allowing the OBC to control the deployment if the timer fail ensuring they are deployed once the spacecraft is in orbit.

### 2.1.6 Payload

The payload is a camera with an FPGA module able to take pictures. The pictures are then compressed to a JPEG2 format in the FPGA module

making the file size suitable for download via the radiolink to the ground station. This payload can allow for reprogramming of the FPGA in orbit potentially allowing the satellite features to be updated over time.

## 2.2 Backplane design considerations

### 2.2.1 Electronics in space: *Radiation, vacuum and temperatures*

The space environment is a harsh environment especially for electronics. Thinner atmosphere and weaker magnetic field leaves the spacecraft and its electronics more exposed to radiation in the form solar wind, solar energetic particles from flares and coronal mass ejections as well as galactic cosmic rays. In orbits ranging from 700 kilometers to 6000 kilometers the trapped particles in the Van Allen belts also become relevant.

A detailed explanation of the effects of space radiation can be found in the article "*What could go wrong? The effects of ionizing radiation on space electronics.*" by J. Scarpulla and A. Yarbrough [3, p15-19]. In this article the effects of radiation in space on electronics is divided into three categories:

- Total ionizing dose (TID) effects - *The accumulation of ionizing dose depositions over a long time.*
- Displacement damage (DD) - *The accumulation of defects in the crystal lattice of semiconductor materials caused by high energy radiation.*
- Single event effects (SEE) - *A single high energy particle creating a highly ionizing dose deposition in a sensitive region of a device.*

#### **Total ionizing dose effects (TID)**

TID is a measure of the integrated radiation dose the spacecraft acquires over a given time period. Energetic ions can break down or rearrange atomic bonds altering the material properties. After the electronics has been exposed to sufficient total-dose radiation insulation materials can become more electrically "leaky" reducing the insulating capacity. TID also affects conductive materials such as metal film in resistors changing their characteristics over time.

However, the biggest challenge when it comes to TID is the effect it has on NMOS and PMOS transistors. NMOS and PMOS transistors are used in most electronic circuits as switching elements to propagate logic



states throughout the circuit. When the gate oxide is exposed to radiation it becomes ionized by the dose it absorbs and even though the electrons are fairly mobile the so called "holes" (electron voids or positive charge carriers are often referred to as "holes") are not. Over time these "holes" become trapped and accumulate in the gate oxide as shown in figure 2.3. After enough of these positive charges are trapped in the gate oxide the positive charge builds up and eventually has the same effect as if a positive voltage was applied on the gate of the transistor. This results in NMOS transistors that are permanently in the "on" or conducting state and PMOS transistors in a permanent "off" or insulating state.

As the technology continues to scale down the sizes of transistors a result is that the gate oxide is also becoming thinner. As this happens the gate oxide is able to trap less positive charge over all and are naturally becoming more radiation resistant.

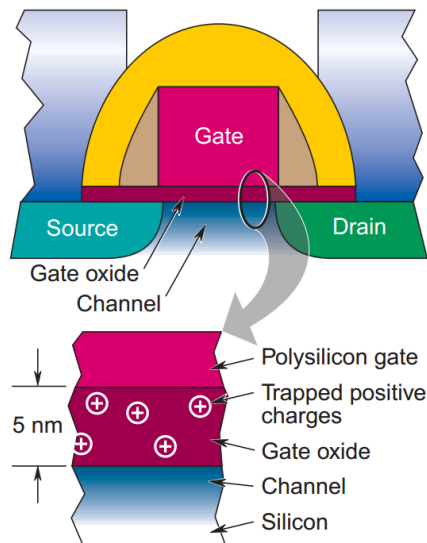


Figure 2.3: Illustration from J. Scarpulla and A. Yarbrough [3] showing the effect of TID on the gate oxide of an NMOS transistor.

### Displacement damage (DD)

Also referred to as neutron and proton damage displacement damage happens when neutron or proton that are highly energetic penetrate into the semiconductor crystal lattice such as the silicon. The intrusive proton or neutron transfers some of its energy to the silicon nucleus potentially knock-

ing the nucleus out of position. Also referred to as elastic scattering the affected and now "free" silicon atom has to disperse with the energy gained through its displacement. This can happen through ionization or by in turn displacing other atoms. An alternative is that the incident particle is absorbed in to the struck nucleus. This is called inelastic scattering and the excited nucleus needs to emit some of the absorbed energy through re-emitting the particle at a lower energy along with a gamma ray. Either way the result causes displacements in the crystal lattice. The outcome is essentially microscopic crystal imperfections called voids and clusters in the crystal lattice that can cause interference in the orderly flow of charges in the semiconductor as is shown in figure 2.4

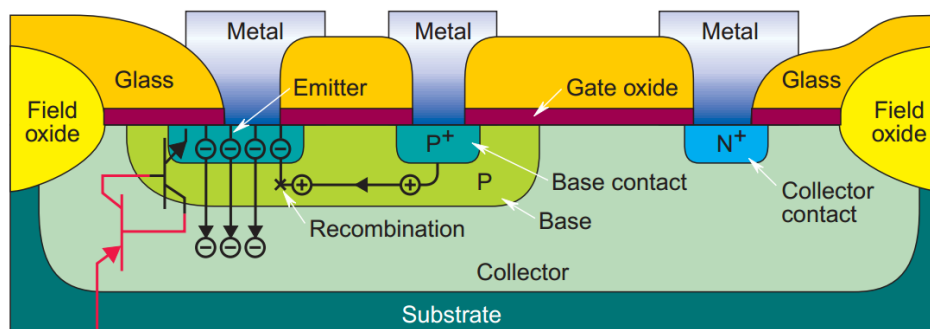


Figure 2.4: Illustration from J. Scarpulla and A. Yarbrough [3] showing the effect of DD in the crystal lattice of a bipolar junction transistor. Electrons are lost to recombination in "holes" before reaching the collector.

### Single event effects (SEE)

In space it is possible to have so called single event effects from a single particle causing trouble. Single particles from cosmic rays can pass through a space craft and its internal components and exit on the other side in a straight line. Shielding a space craft from these types of particles is highly impractical. When an energetic ion passes through a semiconductor it leaves behind a wake or column of ionized material. The column is electrically neutral and consists of an equal number of electrons and holes effectively creating a temporary "conducting wire" as shown in Figure 2.5. This disturbs the "normal" current paths and electric fields. The result can be significant current flow or temporary "shorts" as the ion "track" dissipates by recombination. This can result in single event upsets, burnouts, latchup or other undesirable events. Single event upsets can include so called "bit flips" in memory devices. As mentioned when talking about TID CMOS

scaling affects a components radiation resistance characteristics. However when it comes to SEE the scaling down of CMOS devices has a negative outcome as less energy is required to compel single event effects or the "wake" affects bigger parts of the circuit making it more likely to experience SEE's.

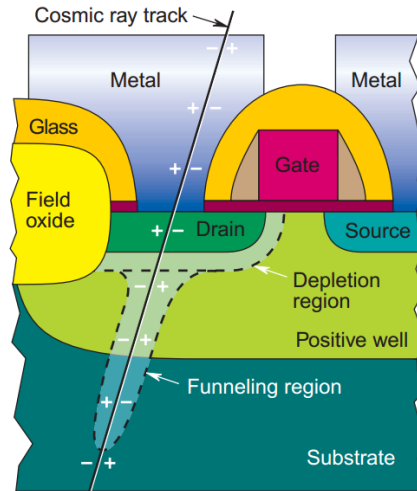


Figure 2.5: Illustration from J. Scarpulla and A. Yarbrough [3] showing the effect of SEE in NMOS transistor. The energetic particle passes through the device leaving a wake of liberated charges. This can momentarily create a short between the drain terminal and the substrate.

### 2.2.2 Fault isolation, detection and recovery

Any spacecraft orbiting earth needs some degree of autonomous operation in order for the spacecraft to be able to perform its tasks when not in direct contact with the ground station. This requires a design is dependable and can handle faults. As described in [4] chapter 15.2.1 we can achieve a dependable system by:

- Fault-avoidance - to prevent, by design, the occurrence of a fault.
- Fault-tolerance - to provide, by redundancy, the specified service in spite of faults occurring.
- Fault-removal - to remove the presence of design faults.
- Fault-forecasting - to estimate, by evaluation, the presence, creation and consequences of errors.

Faults result in errors. Errors can in turn result in failures.

## 2.3 BUS evaluation

This section will briefly look at different alternatives when it comes to choice of main data carrier bus and provide a foundation for evaluating the current bus choice against alternatives in later chapters. Especially the different recommended methods for error checking and fault tolerance are reviewed.

### 2.3.1 I<sup>2</sup>C bus

The I<sup>2</sup>C bus was originally developed by Philips Semiconductors (now NXP Semiconductors) as a simple bidirectional 2-wire bus for efficient inter-IC control as stated in the I<sup>2</sup>C-bus specification and user manual [5]. The I<sup>2</sup>C bus is a 2-wire bus requiring only a serial data line and a serial clock line. Each device connected to the bus must have a unique address often set in hardware. Figure 2.6 shows a typical application using the I<sup>2</sup>C bus. The bus speed ranges from the 100 kbit/s standard mode transfer rate all the way up to 3.4 Mbit/s in High-speed mode when supported.

#### Byte format

Bytes on the SDA line are always 8 bit long and are followed by ACK (acknowledge) bit. Any number of bytes can be transmitted in a transfer operation, however each byte must be followed by an ACK bit as shown in figure 2.7. A slave can force the master to wait for it to finish an internal operation or interrupt service by holding the SCL line LOW releasing it when it is ready.

#### Arbitration

Arbitration is only required when the I<sup>2</sup>C bus is set up as a multi-master system. A master can only start a transfer when the bus is free. However, In the event of two masters initiating a transfer at the same time each master checks that the value it puts SDA (Serial Data Line) on the bus actually is present on the bus. Bit by bit the masters check if the bus is high. If a master tries to send a HIGH, but detects that the SDA level is LOW, the master knows that it has lost the arbitration and turns off its SDA output driver. The master winning the arbitration continues to send its message and the master loosing the arbitration must restart its transmission once the bus is free. Control of the bus is only dependant on the address and data sent by competing masters this means there is no central master or order of priority on the bus.

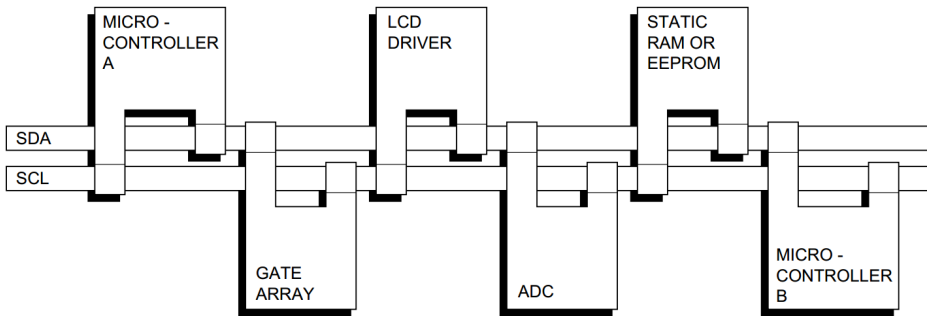


Figure 2.6: Illustration from data-sheet [5] showing a typical I<sup>2</sup>C bus implementation.

### Error Checking and Fault Confinement

- Software reset** - Software reset is an optional feature implemented in some I<sup>2</sup>C devices. A message can be sent to force the device to reset. (Only devices that does not pull SDA or SCL line after applying the supply voltage must use this function as it otherwise could block the bus.)
- Bus clear** - In the case of SCL being stuck LOW the recommended procedure is HW reset signal for devices with HW reset inputs. Alternatives for devices without HW reset signals are power cycling to activate the mandatory internal Power-On Reset circuit. In the case of the SDA line stuck LOW the master should send nine clock pulses. This should make the device holding the signal LOW release the bus. If not, HW reset or power cycle should be used.
- Software defined error check** - Checksums or error detection methods can be baked in to the higher level functionality of the messaging structure in the case of MCU to MCU communication.

#### 2.3.2 CAN bus

The CAN bus (Controller Area Network bus) was originally designed by Bosch and is an International Standardization Organization (ISO) defined serial communications bus [6]. One of the key points of the CAN standard is the high immunity to electrical interference and the ability to self-diagnose and repair data errors. The CAN communications protocol is described in

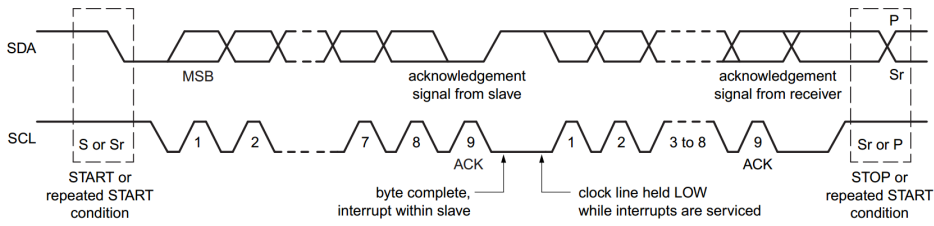


Figure 2.7: Illustration from data-sheet [5] showing data being transferred on the I<sup>2</sup>C bus.

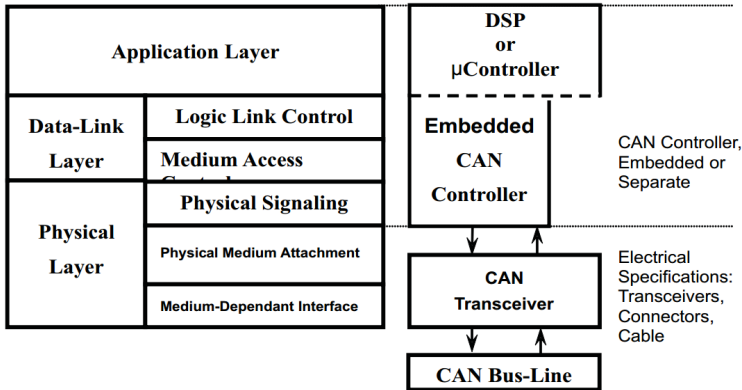


Figure 2.8: Illustration from data-sheet [6] showing the layered ISO11898 standard architecture of the CAN bus.

ISO-11898: 2003 [7] which shows that it follows the Open Systems Interconnect model which is defined in terms of layers as shown in figure 2.8. The Controller Area Network (CAN) consists of nodes connected to a bus. The bus is a two wire differential type bus and the nodes consists of a controller and an transceiver. The controller is often embedded inside a microcontroller with the transceiver being a stand alone chip. The typical layout is shown in figure 2.9. The transceiver handles the level translation and generation of the differential signal towards the bus and reads the bus differential signal providing appropriate feedback to the CAN controller.

When using the standard CAN we can send and receive messages with a structure as shown in figure 2.10 and contains the following fields:

- **SOF** - Start of frame bit marking the start of a message. Also used for synchronizing nodes on the bus.
- **Identifier** - 11bit CAN message identifier containing the message pri-

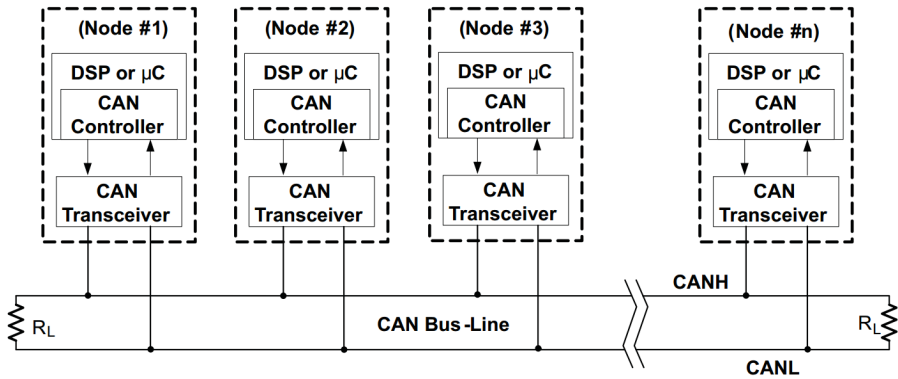


Figure 2.9: Illustration from data-sheet [6] showing the typical CAN bus structure.

ority. (Lower binary value has higher priority.)

- **RTR** - Single remote transmission request. Dominant if information is required from a node specified by the identifier in the message.
- **IDE** - Single dominant identifier extension specifying that no extension to the identifier is present. (CAN messages can be used in either extended or standard mode where the extended mode has an "extended" 29 bit identifier.)
- **r0** - Reserved bit (Not currently in use)
- **DLC** - 4bit containing the length of data transmitted.
- **Data** - The actual data transmitted. (Up to 64 bits)
- **CRC** - Cyclic redundancy check (checksum) for the data transmitted.
- **ACK** - ACK bit received by every node in the network. If one node fails to set the dominant ACK bit the message is discarded and the sender repeats the message.
- **EOF** - End of frame.
- **IFS** - Inter-frame space making time for the controller to move a correctly received frame to a message buffer area.

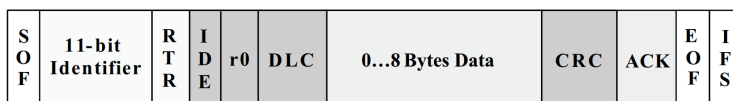


Figure 2.10: Illustration from data-sheet [6] showing message structure in a standard CAN message.

## Arbitration

In the CAN bus a Arbitration is handled in a nondestructive, bit-wise arbitration manner. Meaning that the node winning arbitration continues with the message, without the message being affected by another node. As a zero is the dominant bit in CAN messages the message that has the most consecutive zeros in the identifier wins the arbitration. The result is the lowest identifiers have the highest priority.

## Message types

The four different types of messages on a CAN bus are as follows:

- **The Data Frame** - Containing data from 0 to 8 bytes the data frame is the most common message type.
- **The Remote Frame** - Used to request data from a node. This message type contains no data.
- **The Error Frame** - A message that violates the formatting rules of a CAN message. Transmitted when a node detects an error and causes all the other nodes to send an error frame making the original transmitter repeat its message. A node can however not tie up a bus by repeatedly sending error frames.
- **The Overload Frame** - Can be used to create extra delay between messages. Transmitted by a node that becomes too busy.

A valid frame is achieved when a message is error free. An error free message has the last bit of the ending EOF field in the error-free recessive state. If a dominant bit in the EOF field is read the transmitter repeats the message.

## Error Checking and Fault Confinement

The CAN-protocol uses five different methods for error checking. Two of these methods are on the bit level and three at the message level. When an



error is detected through any of these methods an error frame is generated from the receiving node. If a faulty node hangs up a bus by continuously repeating an error, its transmitting capability is removed by its controller when an error limit is reached. Error checking is as follows:

1. **CRC** - 16bit Cyclic redundancy check containing a 15bit checksum and a 1bit delimiter.
2. **ACK** - A single acknowledge bit and an acknowledge delimiter bit.
3. **Form Check** - Checks the SOF, EOF, ACK delimiter, and the CRC delimiter bits. These must always be recessive bits. When a dominant bit is detected, an error is generated.
4. **Transmit monitoring** - As the transmitter is transmitting a message it also monitors the bus. If a data bit is written and the opposite read an error is generated. This does not happen for arbitration bits.
5. **Bit Stuffing** - Bit stuffing rule where after five consecutive bits of the same value a complement is expected. If a sixth bit is received with the same value an error is generated. Stuffed bits are removed by a receiving node's controller.

The differential structure of the bus lines and the abundant error checking is what contributes to the CAN bus being a relatively robust bus.



# Chapter 3

## Design proposition

Several iterations of the backplane has been devised previously. Each iteration has solved issues discovered in the previous designs and implemented solutions to the specific challenges encountered. This approach over several iterations has resulted in a complex design that solves a great deal of the problems. However, it has also resulted in a design where the complexity itself has become a problem with respect to lack of documentation of design decisions made previously as well as knowledge transfer for new students. In this section we will review the choice's made and challenges faced in the current design (NUTS Backplane v4.0) [8] and make decisions on what to keep, what to simplify and what to change. The chapter will outline a new design proposal.

### 3.1 Proposed backplane design

When updating the backplane design and reducing its complexity we need to do a review of the current state of the backplane. The functions of the backplane will be examined in the following sections and are divided in to the categories: data carrier, sensor network, power distribution, lab access and programming, backplane control, module slot requirements and fault isolation. The current state of the NUTS backplane is revision 4 and was designed by Ingulf Helland [8].

#### 3.1.1 Data carrier

The data carrier is the main bus where data is transported between the different modules. This bus needs to be robust as a failure in this bus will isolate each module form other modules. The bus must also allow for an

arbitration scheme that prevents any faulty modules from blocking other modules from using the bus.

### Current state

In the current design the data carrier is designed as an I<sup>2</sup>C bus with a second I<sup>2</sup>C bus as redundancy as shown in figure 3.1. The two I<sup>2</sup>C buses are routed throughout the backplane connecting to all the modules. However there is no support for two I<sup>2</sup>C buses on a lot of the modules. This is resolved by connecting both the I<sup>2</sup>C buses to the same I<sup>2</sup>C port on the module card. To ensure a module is connected to only one of the I<sup>2</sup>C buses at any given time the buses are isolated from the module side using I<sup>2</sup>C repeaters with "hot-swap support" that fulfills isolation standards. To avoid conflict between the two I<sup>2</sup>C buses only one I<sup>2</sup>C repeater IC is active at a time. Each module slot in the backplane has a set of control logic able to turn each of the repeaters on or off. The state of the control logic is set in a shift register. The shift register receives its states from one of the OBC modules through a separate custom serial bus able to clock data in to a specific module slot shift register based on its address. A custom arbitration bus and logic between the two OBC's are also present to allow for control from both OBC's based on an software defined arbitration scheme. This ensures control is maintained if one of the OBC's fail.

Then main drawback of this design is its complexity. It requires every single type of module slot to have its own logic control. It requires a custom arbitration bus as well as a custom serial bus to clock states in to the shift register of each module slot. It also requires each of the module slots to be custom designed for the module used in the slot. Especially the OBC's slots requires extra pins for the arbitration logic connections and the serial communication towards the slot shift registers. The consequence has been a design with a complexity that have been difficult in terms of knowledge transfer for new students. It also requires extra work with respect to PCB board layout as less of the component layout in each slot is identical and as a result is not repeatable across the board.

### Proposed changes

When we compare the fault and error handling capabilities of the CAN bus and the I<sup>2</sup>C bus discussed in 2.3 it is clear that the CAN bus has a more robust design. The I<sup>2</sup>C bus relies heavily on the ability to perform a hardware reset if a fault presents it self on the bus. This is not always possible without physical access to the satellite. Implementing direct control

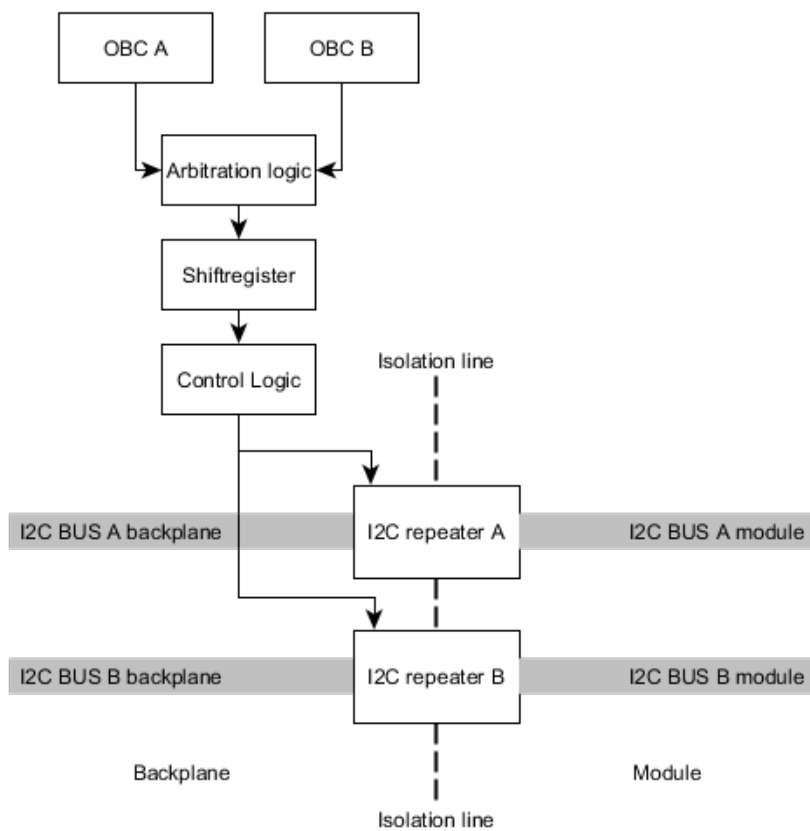


Figure 3.1: Figure showing the the redundancy functionality implemented in the current data bus based on two I<sup>2</sup>C buses.

from the OBCs to reset the I<sup>2</sup>C bus provides a new point of failure and further need for software check and control from the OBC. By choosing a CAN bus as the main data carrier with a single CAN bus replacing the two I<sup>2</sup>C buses in the current iteration of the back plane we can significantly reduce the complexity of the design. The CAN bus has built in arbitration ensuring each of the modules has priority based on it's address in the CAN network allowing us to completely remove the custom arbitration bus in the current design. By making use of a CAN protocol the overhead is reduced as error-checking is handled within the CAN controller and custom error checking does not need to be implemented on the software side of the MCUs connected to the bus. The result is an implementation that is simpler and more robust. Complexity can further be reduced when it comes to slot control signals. This is discussed in section 3.1.5.

### 3.1.2 Sensor network

The satellite needs a subsystem that enables its on-board control modules to monitor the status of the hardware in the satellite. Such a sensor systems needs to be independent of the status messages the modules might send themselves to the OBC via the main databus. This separation is important in order to allow the OBC to detect failures which the modules may not be able to report themselves. Being battery powered the satellite has a limited power reserve to draw from in order to perform its main operational functions. It is critical that the power usage of the whole system does not exceed the battery's capacity to deliver power at any given time, as well as the solar panels ability to sufficiently charge the battery. Taking this in to account the most important sensors on the backplane are sensors to determine power consumption of connected modules as well as the power consumption of the backplane itself. A common structure of sensor systems are so called sensor networks where a bus is shared between all the sensors and the sensors are accessed by a master using it's address in the network. By allowing all sensors in the satellite to connect to the same sensor network whether it is located on the backplane or on a module, they can all be accessed by any master connected to the bus.

#### Current state

The sensor network in the current design is based on a single I<sup>2</sup>C bus with sensors in the form of slaves distributed on the back plane circuit board as shown in figure 3.2. The sensor I<sup>2</sup>C bus is connected to a housekeeping module that connects the I<sup>2</sup>C sensor bus to the main I<sup>2</sup>C databus. The

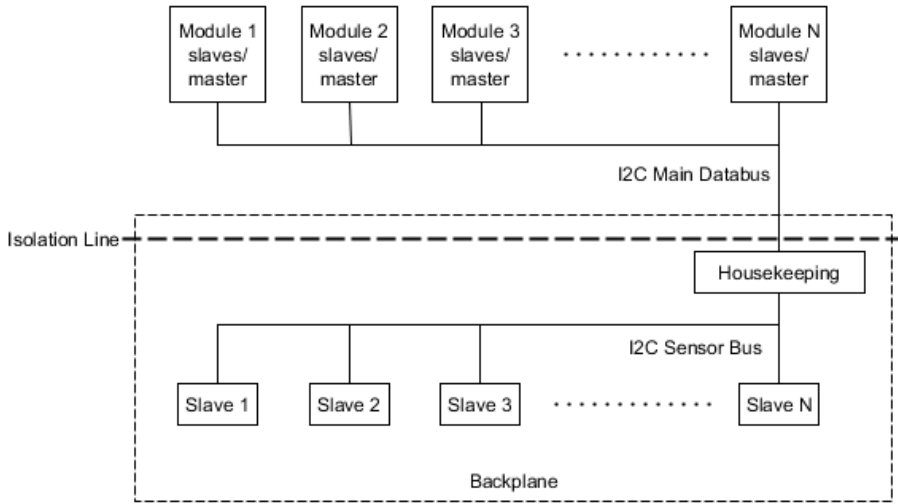


Figure 3.2: Sensor bus as implemented in the current design I<sup>2</sup>C bus.

backplane itself holds only current sensors used to measure each modules power consumption as well as the backplane power consumption. This is done using the INA219 IC which is a current/power monitor with an integrated I<sup>2</sup>C interface. The the power consumption is measured at each module slot both for the 3.3V and the 5.0V rails. This part of the current design is incomplete with no connections to a master reading the sensor bus data.

The main drawback of this design is that it does not allow for sensors to be placed on modules without connecting them to the main I<sup>2</sup>C data buses. This results in a design where sensors are spread across different buses. The housekeeping module has not been realized in this design and the exact details of how the sensor bus is connected to the main data bus is not documented. An issue also encountered in current designs was the need for more sensors of a single type than allowed by that sensor types configurable address space.

### Proposed changes

In section 3.1.1 a new CAN bus based data carrier is proposed to replace the original dual I<sup>2</sup>C bus system. However, when it comes to sensor peripherals the I<sup>2</sup>C bus is a well suited bus. The I<sup>2</sup>C bus is typically used for attaching peripheral ICs to processors. The I<sup>2</sup>C bus used only as a sensor bus also

provides a better domain separation between sensor data and the main data bus. This can allow the OBCs to gather sensor data while the main data bus is in use or down and vice versa. The new proposed I<sup>2</sup>C bus design also allow sensors on the module cards to be connected to the same bus as well. This design will also allow micro controllers placed on module cards to access the sensor bus and reading the satellite sensor data as well should the module card design require it. The new sensor bus design is outlined in figure 3.3. The figure shows a single I<sup>2</sup>C bus used as a sensor bus in the satellite. It allows both sensors located on the backplane and sensors located on modules to be connected to the same bus. Further more the figure 3.3 shows the use of I<sup>2</sup>C buffers to ensure electrical isolation between the modules and the backplane to avoid fault propagation in the event of module failures. The buffers also allow the modules to be hot-swapped while the system is powered up in a lab environment. The I<sup>2</sup>C buffer chip can also be used to disconnect a module from the I<sup>2</sup>C bus by powering down the module side of the buffer chip. This is discussed in more detail in section 3.1.7. Figure 3.3 also shows an I<sup>2</sup>C MUX used before the slave sensor ICs on the backplane. This allows for the same type of I<sup>2</sup>C sensors to be used on the module cards without worrying about them having the same address as the sensors placed on the I<sup>2</sup>C bus on the backplane. As in the current design each module slot will have INA219 sensors to monitor both the power consumption of the 3.3V and the 5.0V. It is estimated that 16 of these sensors will be needed in the backplane design as well as at least 4 on the EPS module card. As this particular sensor only have 16 different configurable addresses a MUX is needed.

### 3.1.3 Power distribution

The EPS provides power to the satellite in the form of two redundant 5.0V rails and two redundant 3.3V rails. The backplane must be able to distribute this power to the connected modules in a reliable manner. Modules must be allocated a maximum power draw over time as well as peak current limits. The backplane must manage power in a way that ensures module functionality but does not allow modules to, through a fault or otherwise, consume excessive amounts of the satellites power. The backplane must also be able to disconnect faulty modules form the power rails as well as disconnecting faulty power rails from the modules.



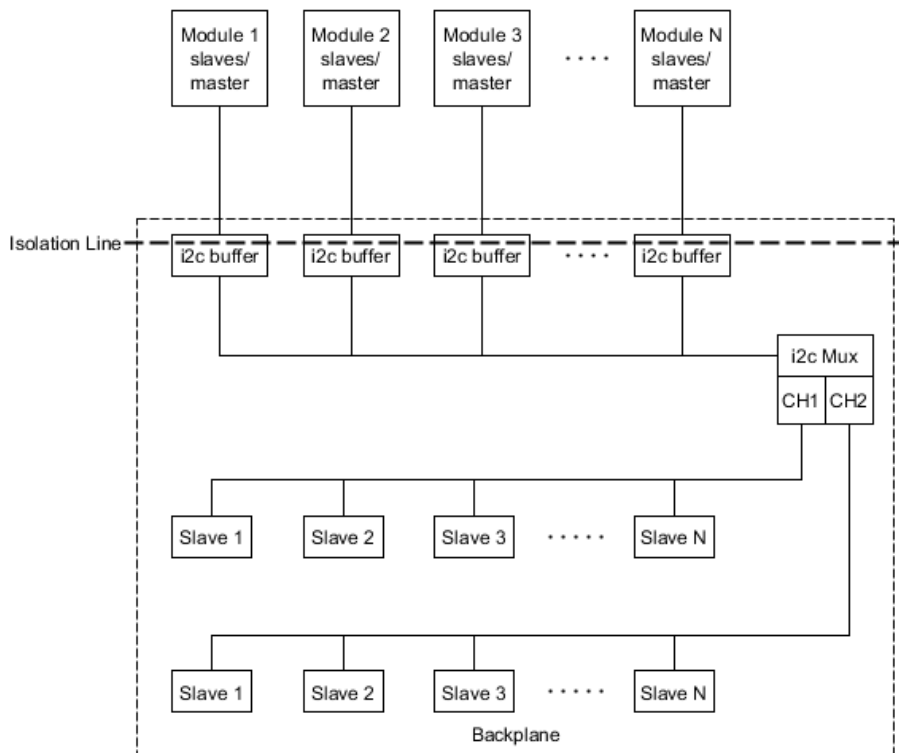


Figure 3.3: New sensor bus design using I<sup>2</sup>C and a bus multiplexer.

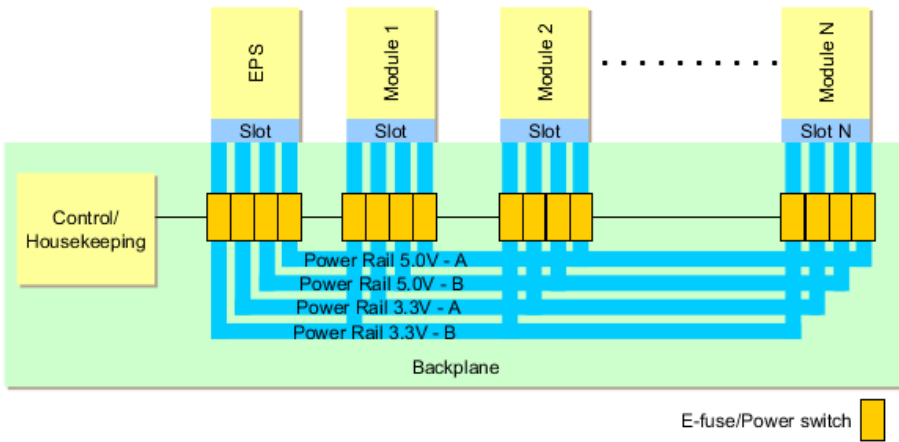


Figure 3.4: The current backplane power distribution.

### Current state

The EPS provides four power rails to the backplane as shown in figure 3.4. On the backplane the four power rails is distributed to the modules through E-fuses. The e-fuses can be set to disconnect when the current draw passes a limit set by the resistor network connected to the E-fuse. This allows us to define different current limits for the different slots based on each modules need. The e-fuse can be reset by a control signal from the backplane control/housekeeping module if tripped. This signal can also be used to manually trigger the e-fuse should it be needed. This setup allows the power rails to be controlled in two ways for each of the rails. Either disconnecting a module from a power rail or disconnecting the EPS from a power rail effectively powering down that power rail.

The main drawback of this design is that it requires the EPS module slot to be specially designed for the EPS effectively reducing the re-usability of the slot design in a PCB layout process. It also makes use of the custom designed control bus and control logic discussed in section 3.1.1.

### Proposed changes

Very little needs to be changed in the design of the power distribution system of the satellite. The current design is well thought through and robust. However, the slots for the modules should be modified with a unified design to allow any module to be connected in to any slot. This will facilitate re-

usability in the layout process of the PCB. Proposed changes to the control signal for the e-fuses is discussed in section 3.1.5.

### 3.1.4 Lab access and programming

Access to the satellite circuit boards for testing purposes are important in a lab environment. It is important that the system allows us to ensure system functionality by physically accessing test point in the satellite and doing measurements. Important elements that needs to be accessed are buses, power rails at different locations and control signals for direct control of satellite functions. Relevant test points must be accessible both when the satellite is open and circuit boards are directly accessible, but also when the satellite is fully assembled and undergoing full scale tests. The backplane must provide a programming bus to all the connected modules allowing for the modules to be reprogrammed after the satellite is fully assembled.

#### Current state

Currently the satellite is designed with a separate test socket for ever module slot. Custom PCBs are then designed to connect to these sockets and provide test data results in the form of LED lights indicating value of logic levels. After assembly access to measurement points are given through a very-high-density cable interconnect (VHDCI) header with 68-pins. A JTAG-bus is connected to every module slot on the backplane and can be accessed through a connector on the OBC. Jumpers on the OBC sets the module address allowing us to choose what module is being programmed by the JTAG programmer connected to the OBC JTAG header.

The drawback of this design is the use of extra custom PCB cards to detect and flag the system status. This requires extra PCBs to be designed. As well as the area need for extra headers on the backplane itself taking up important PCB real-estate. Also, the programming of modules connected to the JTAG bus is done via the OBC. This makes reprogramming the satellite while fully assembled difficult or impossible as access to the OBC is required.

#### Proposed changes

In the new design we remove the custom PCBs and test headers. As many as possible of the test points currently routed to these custom PCBs will be routed to the 68-pin VHDCI header. Measurement points not needed when the satellite is fully assembled will be broken out on the backplane

PCB itself and assessable using a logic analyzer. A second 68-pin VHDCI header will be added should the amount of needed measurement points exceed the available pins in a single 68-pin VHDCI header. The JTAG bus must be accessible through the VHDCI header. Setting the address of what slot is being programmed by the JTAG bus should also be done through the VHDCI header. This will allow the satellite to be fully accessible and reprogrammable after assembly or any at state before fully assembled.

### 3.1.5 Backplane Control

The backplane must provide functions that allows each module slot to be controlled and its states to be set. The system must be able to set states for power-rails connected to the slots as well as connecting or disconnecting the buses or signals going to and from the module via the module slot. This is important in order to limit fault exposure and to contain and isolate failures to a single module preventing propagation of said failure to other modules. Control of these states is exercised by the OBC modules and the backplane must provide a way for the OBC to control these states and functions.

#### Current state

In the current design the backplane control signals and state holders are referred to as housekeeping. Figure 3.5 shows an overview of how backplane functionality is currently controlled. An arbitration scheme based on a custom arbitration bus between the two OBCs decides who can write to the module slot logic. When one of the OBCs win the arbitration it is given access to bus driver chips located on the backplane. These ICs are used to drive the respective address bus and state register bus. Using the address bus driver the active OBC can place an address on the address bus. The address bus is three wires allowing for 3bit addressees used to match against 3bit address predetermined in the hardware of a module slot. When the module slot recognize its address the slot state shiftregister is enabled allowing new data to be clocked in to it in a serial manner via the system state register bus. The system state register bus consists of wires connected the state shiftregister allowing data, clock and set signals to be controlled. Given that the address on the address bus is a match, the OBC can then clock the desired states in to the shiftregister using the serial bus driver ending with a set signal moving the data in the shift register from the serial input to the parallel output effectively setting the slot states. Should states change as the result of automated responses generated in the slot is self, such as a e-fuse being triggerred. The changed state will set and store an

IRQ in the IRQ register. The IRQ register is effectively a D flip-flop used to store the interrupt until such a time that the OBC can read it.

The biggest issue with this design is the complexity, it requires a lot of hardware to work. A custom arbitration bus. Four bus drivers effectively creating six buses. Two serial buses for each OBC on the module side to control the bus driver ICs. As well as two custom buses on the backplane to control the state register. Each module slot also requires address matching hardware, a state shift register and IRQ logic. This is effectively the main bulk of all the electronics on the backplane.

### Proposed changes

As discussed in section 3.1.1 changing the data bus to a CAN bus based data bus would remove the need for arbitration between the two OBCs as the arbitration would be inherent in the CAN bus standard based on the OBCs defined CAN bus address. However this would only work for setting the states of the backplane slots if these the setting of said states were subject to this arbitration. It would not be desirable that an OBC could have access to the system state register bus even though it had lost arbitration inherent in the CAN protocol. However, using the CAN bus to access and set the logic states of the backplane would make setting the backplane states subject to the arbitration in the CAN bus. In figure 3.6 CAN IO expanders are connected directly to the CAN bus on the backplane and controllable from the OBCs via the CAN bus. The CAN IO expanders are quite simple ICs with a CAN bus port on the one side and eight I/O pins on the other side. The I/O pins can then be set high or low by sending a message via the CAN bus to the CAN IO expander. In figure 3.6 D flip-flop are used as state holder. Each slot has a single D flip-flop for each state effectively storing the state. CAN IO expander number 1 has its first output pin connected to the clock input of all D flip-flop's specific to the first slot. The second pin connected to all D flip-flop's specific to the second slot and similarly for all consecutive slots. This allows a message to be sent from an OBC over the CAN bus setting the CAN IO expander pin high effectively clocking any states that may be on the input of the D flip-flop to the outputs storing those states in the D flip-flop's. What states are stored are given by CAN IO expander number 2 which has its outputs connected in parallel to all slots. CAN IO expander number 3 is configured to receive inputs. If an IRQ is raised a message is generated and put on the CAN bus detailing what input was sensitized and allowing both OBCs to read it.

This backplane housekeeping system allows the six serial buses shown in figure 3.5 to be replaced by a single CAN bus connection. It also does away

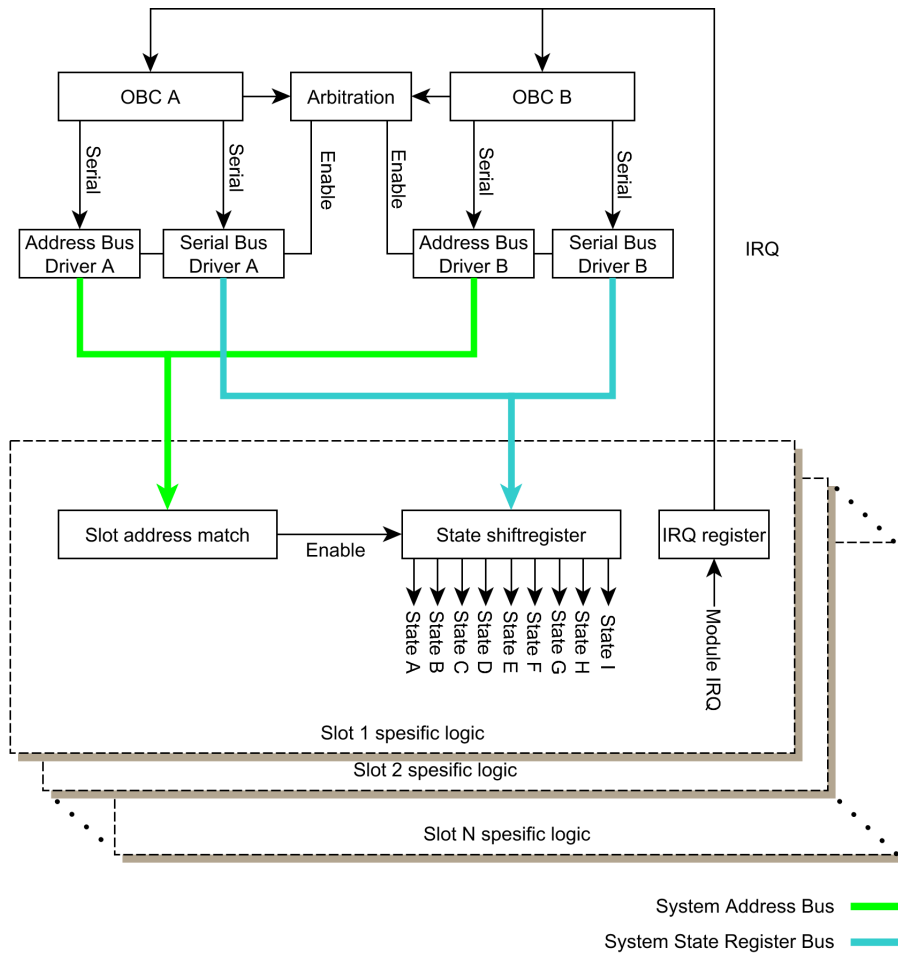


Figure 3.5: Figure showing the current version of the housekeeping electronics used to set module slot states for controlling backplane slot functionality.

the need for a custom arbitration bus and the bus driver logic. Further more it makes use of the standard CAN protocol and removes the need for implementing custom serial communication protocols in the OBC. It also allows the slot header it self to be reduced in pin count and makes it possible to remove connections between the OBC and the backplane that was only used by the OBC making a more universal connection between modules and the backplane possible as proposed in section 3.1.6. These changes would also make it possible to add a third OBC should it be desirable to make a system based on three ore more OBCs voting for fault tolerance in a later iteration of the satellite.

### 3.1.6 Module slot requirements

As briefly discussed in section 2.1.1 a module is a unit that slots in to the backplane in one of several slots. A module normally consists of a circuit board that performs a specific function or fulfills a certain role in the satellite system as a whole. A module slot on the backplane itself is basically a header that provides the module with the necessary physical connections to perform its task in the satellite system.

#### Current state

In the current system there are three distinct slot types. Slave slots, master slots and a single EPS slot. The slave slot is the most basic slot. The system has four identical slots of this type and the header in this slot provides a slave module with the necessary connections when it comes to power, data buses and the programming bus. As well as some utility connections for use in a lab environment. The second slot type is the Master slot. It provides the same connectivity as the the slave slot but has an extended header in order to facilitate the use of the arbitration, address and register buses described in section 3.1.5. The EPS slot is the only slot that needs extra pins to provide power from the module as all other modules receives power through the slot header. The extra power pins providing power to the backplane and the other modules via the backplane also comes in the form of an extended header. All three slots have different sized headers to prevent a module of the wrong type to be inserted in to a slot. An important note is that the headers are designed with diagonal split with equal pins on each side. This makes it possible to rotate a module 180° in a slot.

The main drawback of this design is the predefined headers limiting the number of choices when it comes to module placement and also requires

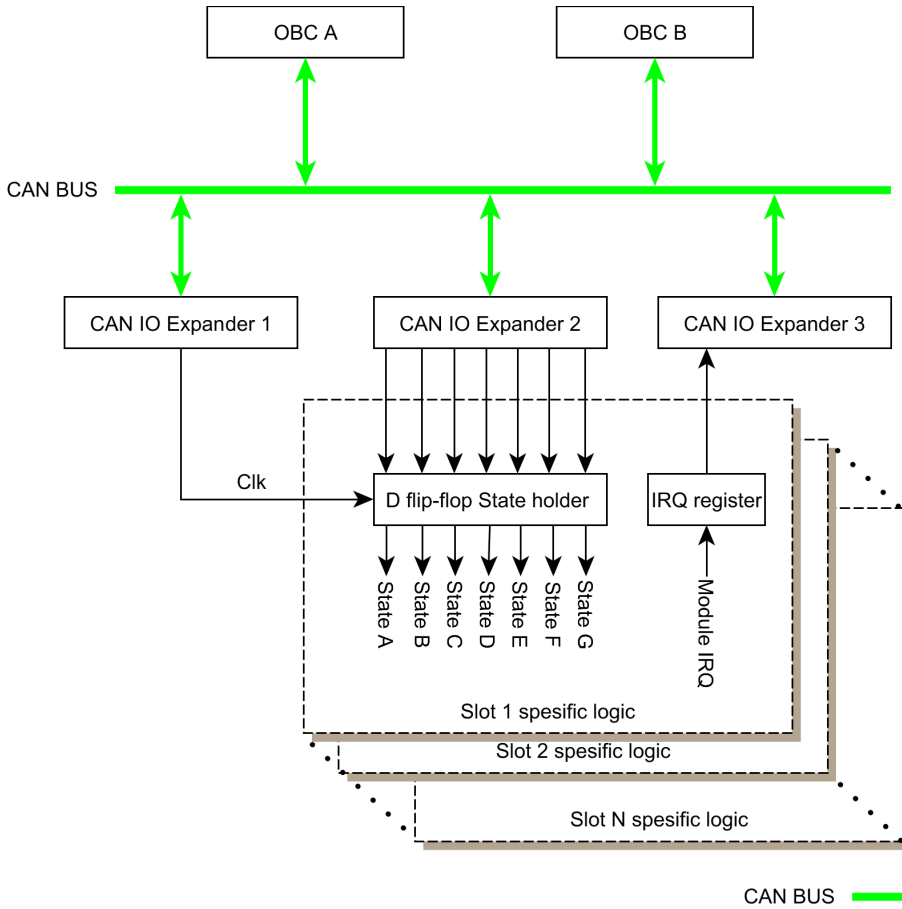


Figure 3.6: Figure showing the proposed new version of the housekeeping electronics used to set module slot states for controlling backplane slot functionality.



three different headers with with different schematics and different layout requirements.

### Proposed changes

In the new design a single uniform slot and header layout for all module types is proposed. In section 3.1.5 the proposed changes to the backplane control system involve replacing the current serial buses used for backplane control functions and arbitration with a control scheme based on messages over the CAN bus. This effectively removes the need for any additional pins in the header of the master slots making them identical to the slave slots. By including the extra pins used by the EPS to deliver power to the backplane in all the slots a uniform slot design can be achieved. This will greatly reduce the complexity in the schematic design and allow for reuse-ability in the PCB layout process simplifying the layout process.

#### 3.1.7 Fault isolation

As discussed in section 2.2 faults due to radiation in space may be unavoidable when using "of-the-shelf" electronics that are not radiation hardened in a satellite. It will ultimately lead to faults in the semiconductors in those components. This means that our options are limited when it comes to Fault-avoidance. Efforts are better spent on Fault-tolerance. This means that by design we can tolerate faults in our system and still perform the mission critical tasks of the satellite. The satellite design already provides fault-tolerance in key areas. The EPS provides redundant power rails. and using a set of two OBCs, each with its own radio, also provides redundancy. The backplane must facilitate these redundancies and provide fault isolation. Fault isolation is a type of fault-tolerance. By isolating a fault to a single module and preventing it or a potential resulting error or failure in the module from spreading to other modules the system becomes more fault-tolerant.

### Current state

As discussed in section 3.1.3 the backplane provides redundant power of both 5.0V and 3.3V to each module. As discussed in section 3.1.1 the current I<sup>2</sup>C data bus is also designed with redundancy in mind consisting of two parallel buses. It also features I<sup>2</sup>C repeaters that features electrical isolation and can be shut down in order to disconnect the I<sup>2</sup>C bus from the module should a failure be present on the module side. Figure 3.7 shows isolation towards

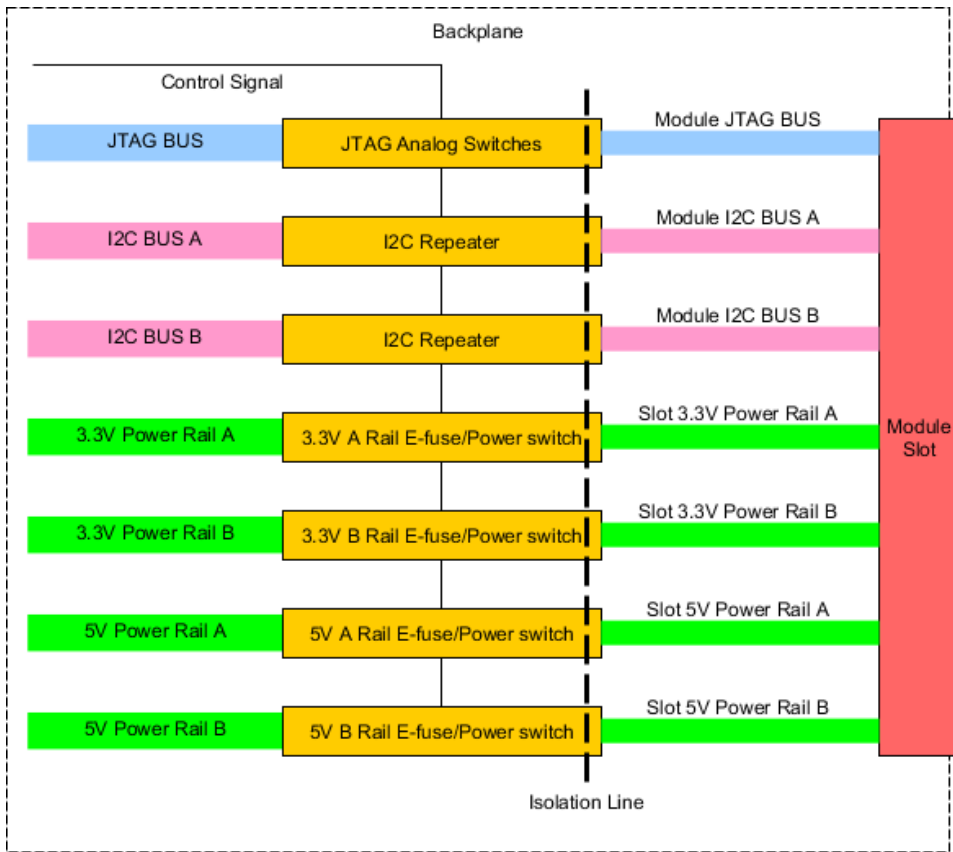


Figure 3.7: The current isolation towards module slots to prevent fault propagation.

the module slot. It also shows analog switches disconnecting the JTAG bus from the module providing further isolation.

### Proposed changes

The approach to fault isolation and module fault tolerance achieves acceptable fault confinement in modules and requires little change to be implemented in the new design. As shown in figure 3.8 the proposed change to a CAN bus as data carrier mentioned in section 3.1.1 is isolated through the use of a CAN transceiver. The can transceiver operates without any input from the backplane control signals. Should a fault present itself that inhibits a module from using the CAN bus the module is effectively dead and can be disconnected by powering down the power rails to that module.

In figure 3.8 a utility signal has been added. This is a signal that can be used for utility functions towards the module. It should be controllable from the an OBC via the CAN bus as described in section 3.1.5. Typically this signal can be used to hard reset a module or it can be used to deploy the antennas on the antenna module should they fail to deploy. The I<sup>2</sup>C bus has been reduced to a single bus dedicated to collecting sensor data and is electrically isolated from the module through an I<sup>2</sup>C repeater as in the current design.

### 3.1.8 Proposed design summary

A new design is proposed where the dual I<sup>2</sup>C bus is removed as main data bus and replaced with a single CAN bus. A separate sensor-bus using I<sup>2</sup>C provides domain separation between the data bus and sensor bus. The introduction of CAN bus enable us to remove all custom arbitration logic on the backplane as arbitration with priority is inherent in the CAN bus protocol. Each module slot will receive its own state holders controlled by simple CAN IO-expanders connected to the CAN bus. This allow us to remove the custom designed shift-register based serial data bus currently used for slot control. It also allows any module connected to the CAN bus to control the backplane. Each module slot on the backplane will be redesigned with a uniform slot header design enabling reusability in the PCB layout process and making the design more flexible while reducing complexity.

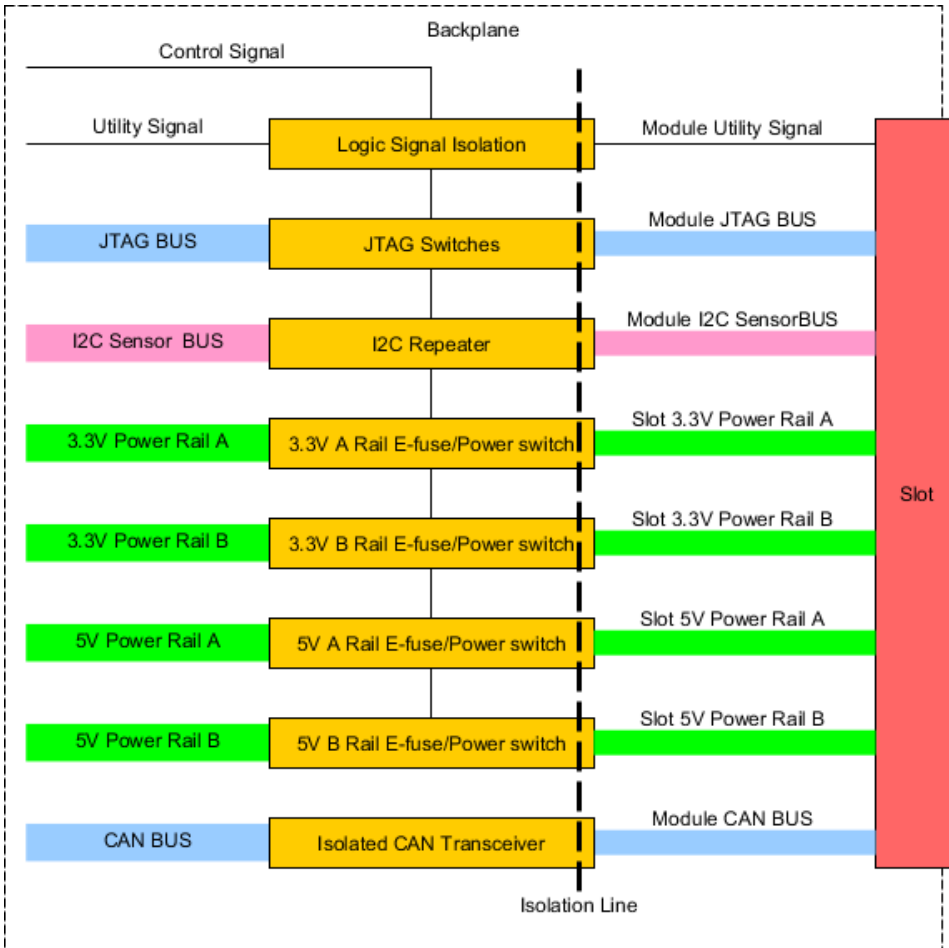


Figure 3.8: The proposed new isolation towards module slots to prevent fault propagation.

# Chapter 4

## Design implementation

This chapter will cover the actual implementation of the changes to the backplane design proposed in chapter 3. The different ways to implement the proposed changes will be evaluated and criteria such as fault tolerance will be reviewed for the new implementations. The chapter will make some references to schematics and relevant parts of the schematics will be included in figures in this chapter. The complete schematics can be found in the appendix in chapter 9. Lastly this chapter will suggest a test approach and methodology to verify the design.

### 4.1 Backplane implementation overview

An overview showing the new revision of the backplane design is presented in Figure 4.1. A total of seven module slots are marked in red with an eighth virtual module slot containing the housekeeping logic. The housekeeper can be addressed from any module slot allowing any slot to control parameters of any other slot. In practice only the OBS modules connected will be controlling slot functions.

### 4.2 Data carrier (implementation)

As proposed in the previous chapter section 3.1.1 the new data carrier is implemented as a CAN bus. Figure 4.2 shows each module having a CAN controller and a CAN transceiver. The CAN controller implements the CAN protocol and CAN functionality towards the user of the CAN bus. In the satellite modules talking to the CAN bus the Atmel AVR UC3C 32-bit Flash Microcontrollers [9] are used. These microcontrollers have the CAN

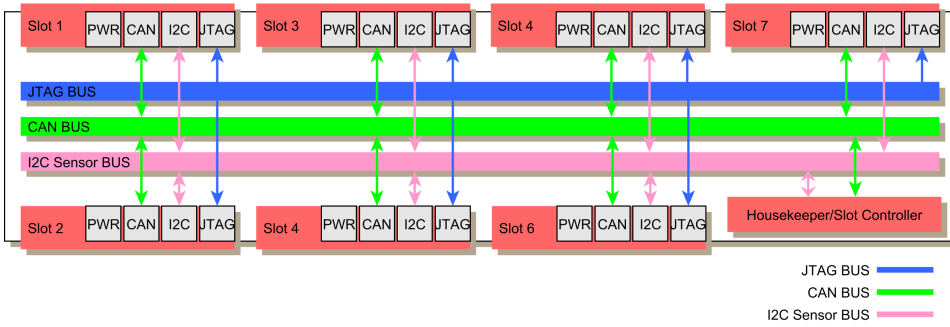


Figure 4.1: Overview of the new backplane design implementation.

controller built in to the microcontroller taking care of the CAN data link layer as per the ISO 11898-1 standard [7]. When it comes to the CAN transceivers the ISO1050 Isolated CAN Transceiver [10] based on the ISO 11898-2/3 Medium Access Unit [11] standards is used. It provides electrical isolation between the module and the backplane and performs the level translations between the CAN controller providing serial data through RXD and TXD serial data pins and the CAN bus data lines on the backplane who uses CAN H and CAN L signals. The CAN bus simply consists of two data-lines on the backplane with end termination resistors. Further more in figure 4.2 the housekeeping is shown as a virtual module. The housekeeping logic itself is designed as a module but is not physically placed on a different PCB. However, the structure and isolation is designed as a module. This will be further discussed in section 4.6.

### 4.3 Power distribution (implementation)

When it comes to power distribution as mentioned in section 3.1.3 the design needs very few changes. Most of the changes done are with respect to the control signals that will be covered in section 4.6. The implementation of the e-fuse used on the backplane is shown in figure 4.3. The circuit based on the TPS25942A eFuse IC from Texas Instruments [12]. The integrated back-to-back FETs provide bidirectional current control well suited for systems with multiple power sources. The power rail delivers power on the input pin (9) and delivers power through the output pin (4). The overpower protection is set through a resistor network connected to the EN/UVLO pin (14) and OVP (15) pin respectively. The resistors connected to EN/UVLO are used for setting programmable undervoltage lockout threshold. An undervoltage event on the input will open the internal FET disconnecting the rail from

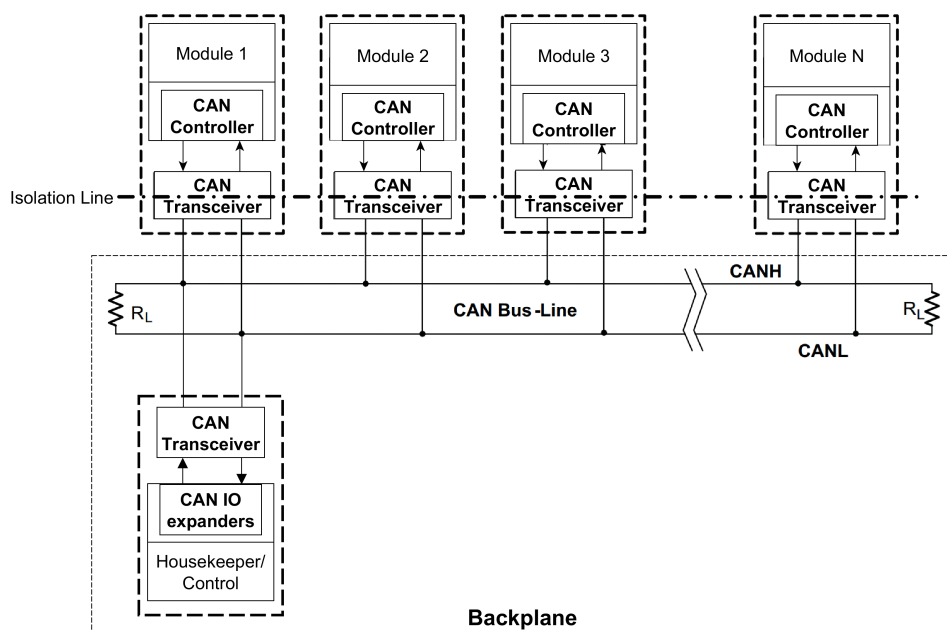


Figure 4.2: New implementation of the data carrier making use of CAN bus. The figure shows how the modules are connected to the backplane bus while placed off board. And the "virtual" module for housekeeping implemented on the backplane.

the output. Similarly the resistors connected to the OVP pin is used to set the programmable overvoltage protection threshold. An overvoltage event will also open the FET disconnecting the output from the supply rail. The ILIM pin (17) is used to set the overload and short-circuit current limit. This is done by connecting a resistor of a size determined by equation 4.1 to ground.

$$I_{(LIM)} = \frac{89000}{R_{(ILIM)}} \quad (4.1)$$

When the internal FET has been opened either by over-voltage, under-voltage or over-current the e-fuse can be reset by pulling EN/UVLO low and then back high. If the voltage on EN/UVLO is held below 0.6V the e-fuse will be held in an off state keeping the rail disconnected from the output. Driving EN/UVLO to ground will be discussed in section 4.6 as this is considered part of the houskeeping/backplane control logic. Component C10 is a capacitor used to regulate the ramp up time of the output and will be determined by each modules requirement.

The implementation shown in figure 4.3 represents the power delivery implementation for a single power line in a single module slot. Each module slot has four of these circuits. One for each of the two 3.3V power lines and one for each of the two 5.0V power lines. With a total of 7 module slots in the design the implementation outlined in figure 4.3 is repeated 28 times trough out the design with varying resistor network values. As well as four additional times for the virtual backplane houskeeping module for a total of 32 times.

The actual power lines consist of four lines traced in parallel from one side of the board to the other side of the board connecting to each of the module slots on the away. Every power line is connected both to outbound and inbound pins on the header in each slot as discussed in section 4.7. This allows the EPS module to be connected to any of the slots. It also allows more then one EPS to be installed in the backplane provided the EPS modules used can be connected in parallel.

## 4.4 Sensor network (implementation)

As described in section 4.3 four power lines are provided to each module slot. After the e-fuse the two lines of same voltage are joined and provides a single point where the current flow can easily measured on its way to the module slot header. Figure 4.4 shows the schematics where the current sensor is used. The INA219 is a quite simple component with a set of serial



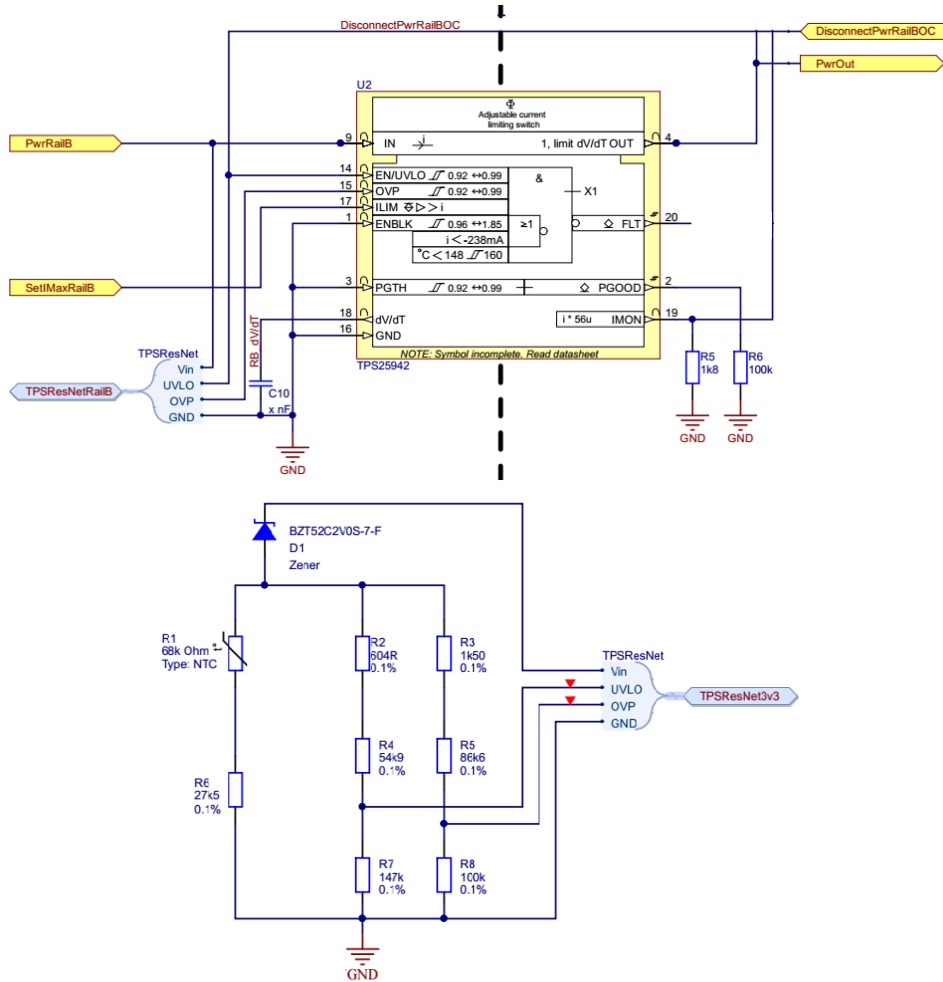


Figure 4.3: Shows the E-fuse implementation in the power distribution on the backplane. Simplified schematics showing only e-fuse related components. Complete schematic can be found in the appendix.

data pins (SCL on pin 5 and SDA on pin 6) in the form of an I<sup>2</sup>C bus connection. The I<sup>2</sup>C bus can be used by any master to request a current measurement. The INA219 gets a current readout by measuring the voltage difference between pin 1 and 2 connected over a 40mOhm shunt resistor placed in the power line. The I<sup>2</sup>C address can easily be set to one of 16 different addresses based by connecting the A0 (7) and A1 (8) pins to either SCL, SDA, VCC or GND. This is done in the top level schematics in order to easily being able to change the values if needed. One of the problems addressed in section 3.1.2 was the limited number of addresses the INA219 has at its disposal. We must be able to measure the current consumption at two different points in each of the module slots as well as two places in supply for the virtual housekeeping module. The his results in the number of required implementations of the schematics in figure 4.4 reaching 16 in total. We know that the INA219 is used in the EPS design as well putting the total number of sensors over the limit of 16 address spaces. To overcome this an I<sup>2</sup>C MUX is used as shown in figure 4.5. This moves the INA219 sensors on the backplane to a higher address space avoiding conflict with any components using a similar address space on the modules.

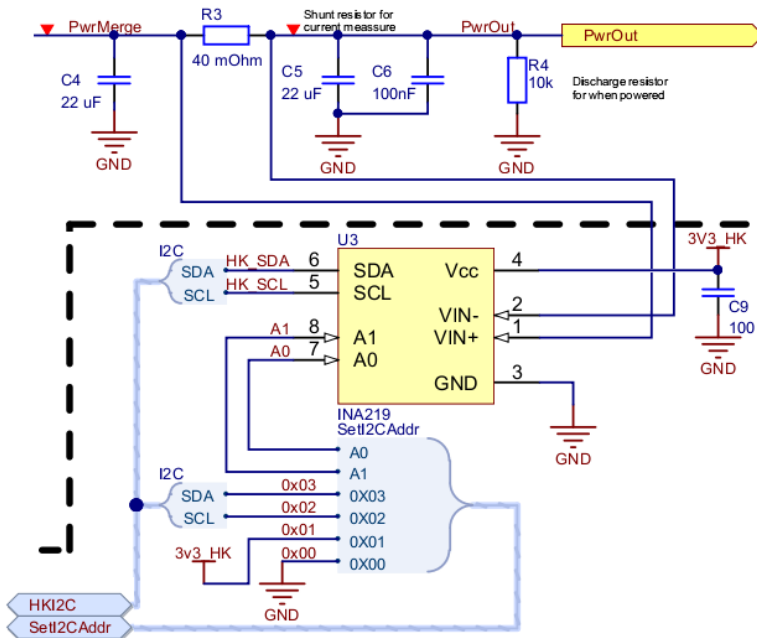


Figure 4.4: Simplified schematics showing how the INA219 sensor is used to measure power consumption. Complete schematics can be found in the appendix.

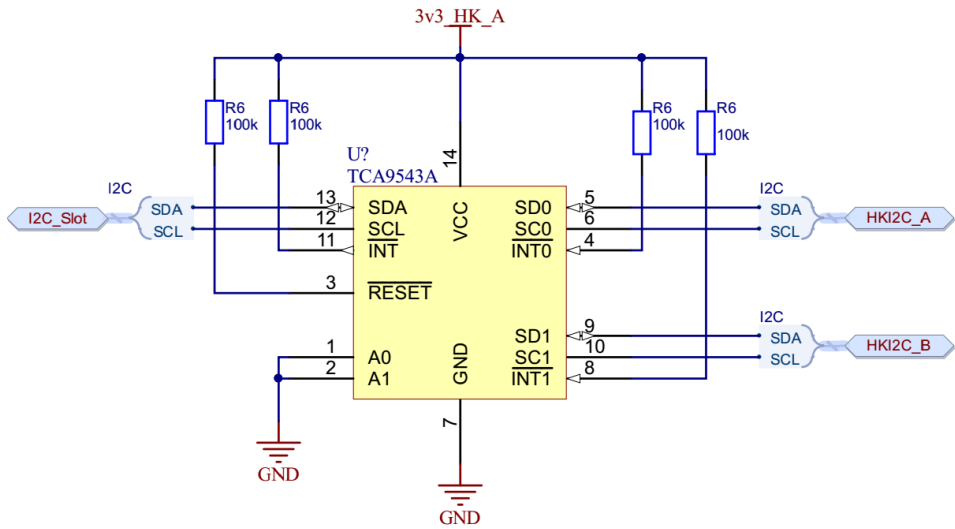


Figure 4.5: I<sup>2</sup>C multiplexer used to increase and separate the address space of the sensors on the backplane and the modules.

The current sensors on the backplane are all connected to the I<sup>2</sup>C bus. This bus is made available to all the modules through the module slot header as shown in section 4.7. This allows both masters and slaves to connect to the same bus and either provide information or request it. This structure allows any master located on any module to request any sensor data individually from any of the sensors connected to the sensor bus. In the current design the only masters in the system are the two OBCs. However, it is possible to allow the microcontrollers located on other modules such as theADCS to be connected to the I<sup>2</sup>C bus allowing functions such as tracking and orientation controls of the satellite to be accessed over the I<sup>2</sup>C bus in the event of a main data bus failure.

## 4.5 Lab access and programming (implementation)

Lab access is provided through a single VHDCI (very-high-density cable interconnect) header as shown in figure 4.7. Providing 68 pins it allows the satellite to be accessed in all states of assembly. Especially once the satellite is fully assembled the VHDCI header provides access to the sensor bus, CAN bus, power rails, battery charging points and battery control signals. It also has available pins if it is later decided that more test points need to be connected to the header making them available in a fully assembled state.

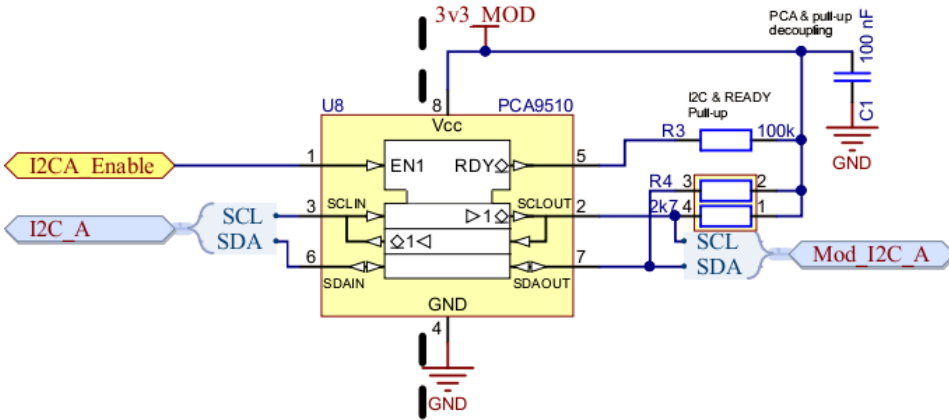


Figure 4.6: Shows the use of an I<sup>2</sup>C repeater to provide isolation and hotswap capabilities between the modules and the backplane. Complete schematics can be found in the appendix.

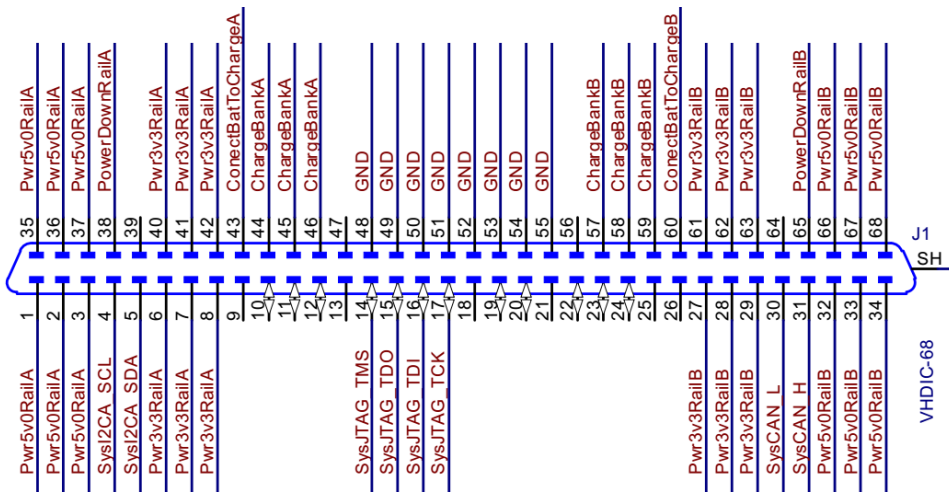


Figure 4.7: Shows the use of VHDCI header to provide lab access to the backplane functions and buses for programming, charging, power and diagnostics.

Parameter	Controlled via	Granularity of control
Module power 3v3 A	CAN	Individually for each slot
Module power 3v3 B	CAN	Individually for each slot
Module power 5v0 A	CAN	Individually for each slot
Module power 5v0 B	CAN	Individually for each slot
Reset/Utility	CAN	Individually for each slot
I <sup>2</sup> C connection ON/OFF	CAN	Individually for each slot
JTAG connection ON/OFF	CAN	Individually for each slot
Housekeeping Power 3v3 A	CAN	Global
Housekeeping Power 3v3 B	CAN	Global
Housekeeping Power 5v0 A	CAN	Global
Housekeeping Power 5v0 B	CAN	Global

Table 4.1: Shows the parameters that can be controlled on the backplane. The table also shows how the parameters are controlled and to what granularity.

## 4.6 Backplane Control (implementation)

In section 3.1.5 the backplane control logic is discussed. The implementation provides the OBC modules with a way to control all the backplane states via the CAN bus. In practice any number of modules can be allowed to control the backplane provided they are connected to the CAN bus. Table 4.1 shows the parameters that can be controlled using the backplane control logic. It also shows how they are controlled and to what granularity.

### 4.6.1 CAN IO expanders

In the implementation of the control circuitry CAN IO-expanders are used. Figure 4.8 shows an overview of the implementation. It shows two separate CAN IO-expanders each connected to and controlled by the main CAN databus. CAN IO expander nr. 1 can load a desired state on its pins 1 to 7. These values are then propagated to the input of D flip flops in all the slots. CAN IO-expander nr. 2 then toggles the pin corresponding to the module the state is to be stored in effectively making a single clock pulse moving the values on the input side of the D flip flops to the output side. The D flip flops acts as memory storing the slot state with it's output connected to the enable and ON/OFF signals for the controllable elements of the slot in question. By clocking the D flip flops on more than one slot an identical state can simultaneously be clocked to multiple module slots. As

discussed in section 2.2.1 bit flips may occur when electronics is placed in space. In actual schematics is shown in figure 4.9 we make use of an extra IO-expander. These IO-expanders together provide the "set state" pulse to each of the module slots. By using two of these IO-expanders where the pins are connected through AND gates, identical levels of the pins in the two IO-expanders are required before the state is clocked to a module. This effectively removes the possibility of bit-flips randomly being clocked to a state as it would require identical bit flips in both IO-expanders in order to propagate the new state to the slot in question.

### 4.6.2 Slot state holder

In figure 4.10 the implementation of the slot state holder circuitry is shown. This circuit is implemented for each of the slots and holds the respective slots state in the D flip flop IC's. The desired state of the slot is set on the D1 input pins on the left side of the seven first D flip flops. When the state on the C1 input of the D flip flops are toggled the value held on the D1 input is propagated to the output of the circuit. The output from the D flip flop provides the control signal for controlling the slot hardware in the form of enable/disable or ON/OFF signals. The default state of this signal can be chosen by selecting the regular output or the inverted output. In the top part of the figure an RC-circuit provides a delay when powering on the backplane holding the active low reset input low during startup ensuring the D flip flops is in a known default state on startup. The power rail control signals for a single voltage is spread across two different IC's ensuring that at least one rail of each 5v0 and 3v3 volts are still present should a D flip flop IC fail.

The last D flip flop holds the IRQ which is triggered by one of the fault indicator signals raised in a fault situation by one of the slot e-fuses as discussed in section 4.3. On a powerfault where one or more of the e-fuses are tripped the ModPwrFaultOC pin will be pulled low resetting the D flip flop and raising the SlotFaultIRQ output pin. The slot IRQ pin is connected to a CAN IO-expander that generates a predefined message on the CAN bus when the values of the pins change. This message can be read by the OBC connected to the CAN bus. Each slot has its own IRQ number and IO in the IRQ CAN IO-expander. Of the 8 IO pins available on the IRQ CAN IO expander 7 are used to receive IRQs, one for each slot. The last is used to reset all D flip flops holding IRQs. This way the D flip flop can hold the IRQ until it is cleared by the OBC. A single pin on the IRQ CAN IO-expander is used to reset all IRQs. Care must be taken that all IRQ inputs are read before a reset is initiated to ensure no IRQs are missed.

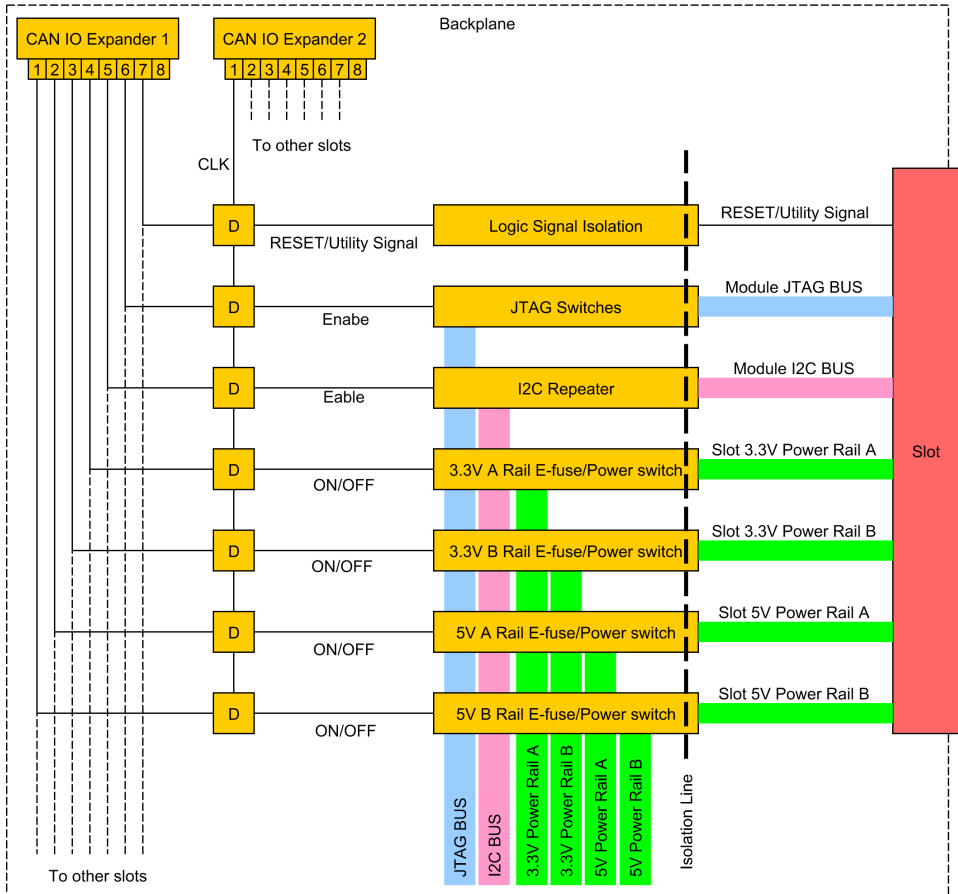


Figure 4.8: Block schematic showing the slot controller and how states are set and stored in D flip flops using CAN IO-expanders controlling the different slot functionalities.

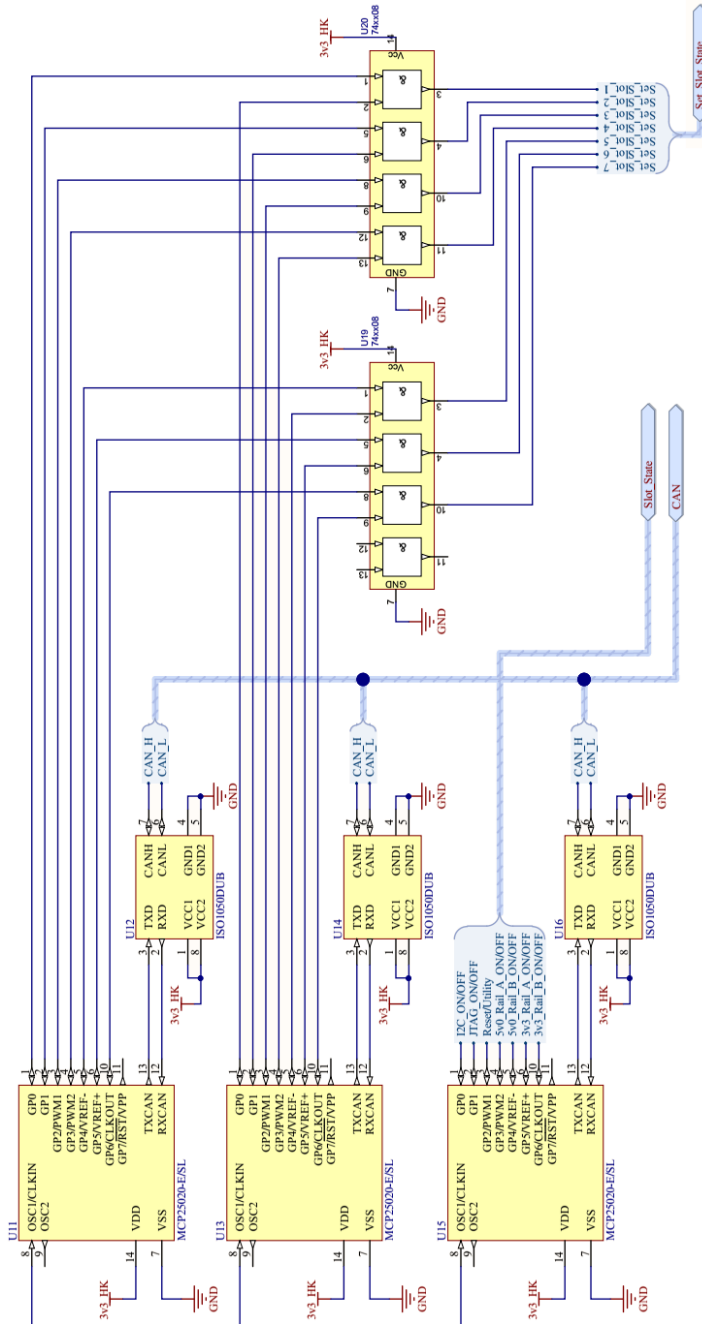


Figure 4.9: Schematic showing the CAN IO-expanders used control slot states by setting the D flip flops of the respective slot.



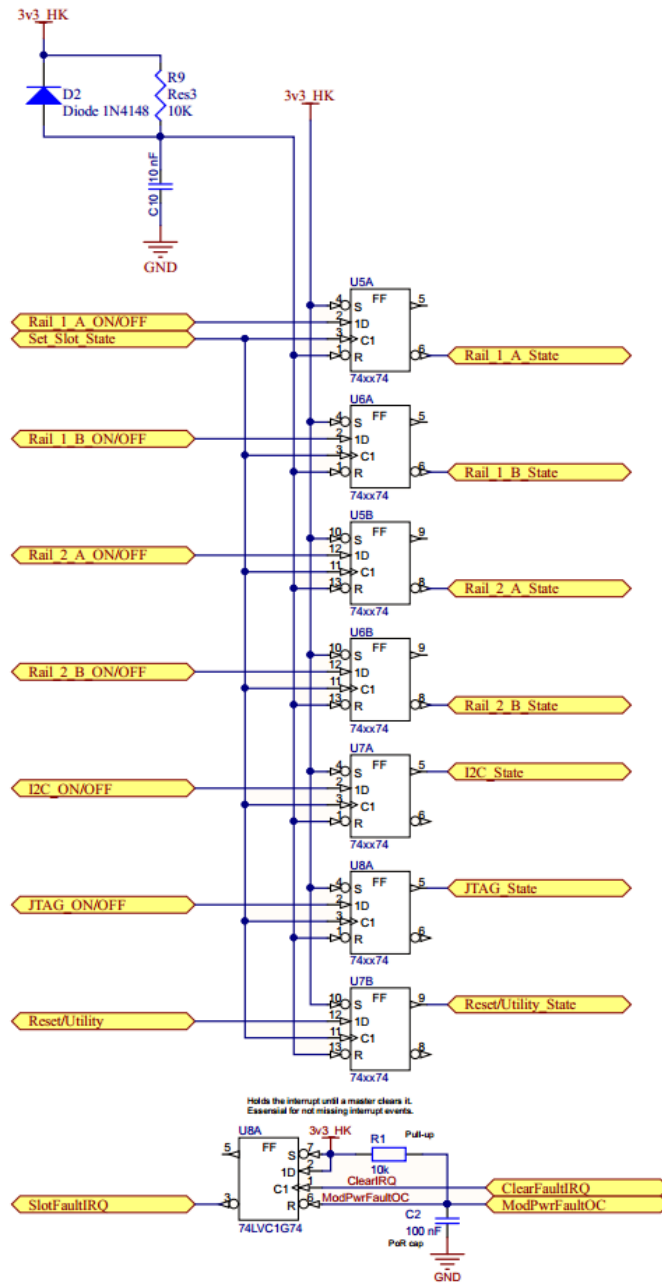


Figure 4.10: Schematic showing the D flip flops used to hold slot states in a given slot. IRQ state holder and power on reset functionality is also shown.

### 4.6.3 Control-logic power and power cycling

Section 4.3 covers how power distribution to the satellite modules is designed in a fault tolerant manner. However it is also important to power the backplane logic in a way that is equally robust as the implementation for the modules. Figure 4.11 shows an excerpt from the schematics detailing how the backplane logic is powered. Two backplane voltages (3v3\_HK\_A and 5v0\_HK\_B) are provided through the same type of e-fuse circuits covered in 4.3. This provides the same fault isolation when it comes to the backplane power as each of the modules slots have. The 3v3\_HK\_A power line is used to power most function of the backplane. Almost all IO-expanders, all sensors and the I2C sensor bus, as well as all the D flip flops holding the slot states uses the 3v3\_HK\_A voltage. The second voltage 5v0\_HK\_B is used to provide all can transceivers with power on the backplane side allowing for the isolation between modules and the backplane on the CAN bus. 5v0\_HK\_B is also used to provide the control logic shown in figure 4.12 with power. This is the control logic for the backplane power making use of a CAN IO-expander to control the e-fuses providing the backplane power. It can be used to power down or power cycle all backplane logic powered by the 3v3\_HK\_A voltage. As discussed in section 4.6.1 and section 4.6.2 a powercycle of the CAN IO-expanders or the D flip flops holding the slot states will result in the default states being produced in these circuits effectively providing a way to "reset" all the backplane logic. As the CAN transceivers are powered by the 5v0\_HK\_B voltage we retain the ability use modules and to send CAN messages between modules even when the backplane logic is powered down. Also, the e-fuses that provide power from the satellite power rails to each module are unaffected by a power down of the backplane logic. We effectively loose the ability to send control signals to these e-fuses, however the basic over/under-voltage protection and current limiting functionality remains. This allows us to power down or effectively enter a "powersave" mode when it comes to power management of the backplane without loosing any functionality in the modules them selves and the communication between the modules. The two power states are shown in table 4.2.

The e-fuses providing the 5v0\_HK\_B power also have their own control circuit similar to the one shown in 4.12. It is however important that this control logic is never used to POWER DOWN the 5v0\_HK\_B as this will result in a situation where the CAN transceiver circuits are powered down and we are unable to send messages to this control logic to power it back up. The control circuit controlling the 5v0\_HK\_B should only be used to reset the e-fuses should they be tripped. It is also important to note that the E-fuses

Power states	Via	Functionality
Fully powered	CAN	Retain full control over all slots.
Powersave (No 3v3_HK_A)	CAN	No slot control, no sensor bus, only CAN bus working, default slot states on power-up.

Table 4.2: Shows the two different power states the backplane can be set to and what level of backplane control is possible in each state.

used to provide the 5v0\_HK\_B are of the "auto-retry" variety. Meaning that if the fuse is tripped due to a over/under-voltage or over current situation the fuse will automatically try to power back up the 5v0\_HK\_B voltage.

#### 4.6.4 Module programming and inter-module programming

Most of the satellites modules employ some form of microcontroller or multiple microcontrollers. These microcontrollers are programmed using a JTAG programmer. As discussed in section 3.1.4, whilst the microcontrollers on the module cards can easily be accessed when the satellite is in a disassembled state it must also be possible to program these modules after the satellite is fully assembled. Figure 4.13 shows how this is implemented in the satellite. Both the JTAG bus and the CAN bus can be accessed through the VHDCI header from the outside of the satellite in a fully assembled state. By sending a message on the CAN bus the CAN IO-expanders can be configured to "open" the analog switches in each module slot allowing the backplane JTAG bus to be connected to the chosen module creating a single path from the external VHDCI Header to the module for programming. All other module switches should be in a "closed" or disconnected state ensuring only the desired module receives the JTAG signals. Any "reset signals required during the programming must use the "reset/utility" signal provided in the slot state holder as shown in figure 4.8.

The current implementation of the JTAG bus also enables reprogramming of modules while in flight. Effectively allowing for software updates and new functionality to be added to the satellite at a later point. Figure 4.14 shows how this can be implemented on the module side. The OBC would receive a binary file from the ground station. By making use of a MUX on the module side of the JTAG bus the JTAG bus can be connected to GPIO pins on the MCU. Then both the analog switches in the module slots of both the OBC MCU and the target MCU can be opened making a connection directly from the OBC GPIO to the target MCU JTAG pins.

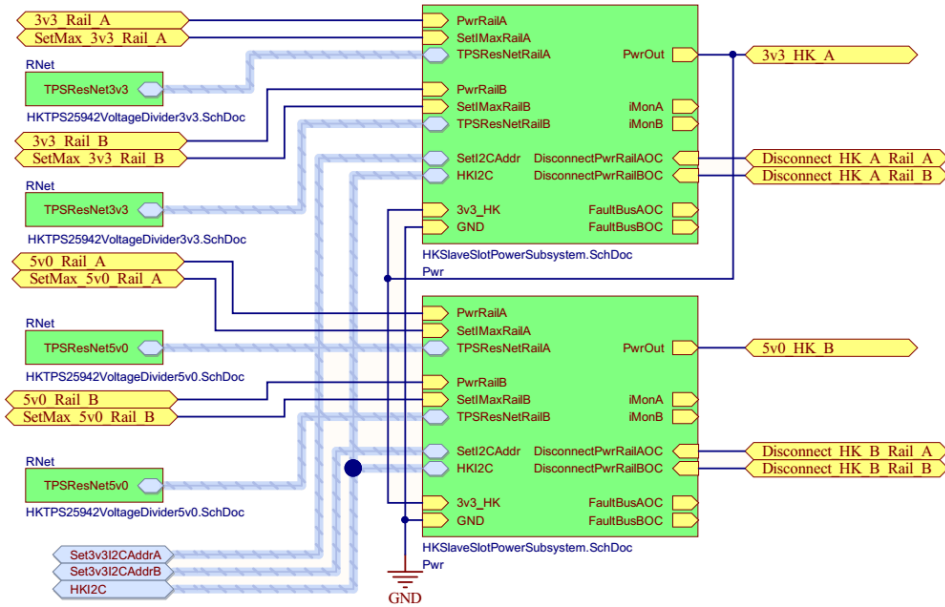


Figure 4.11: Schematic showing how the backplane functionality and house-keeper is powered.

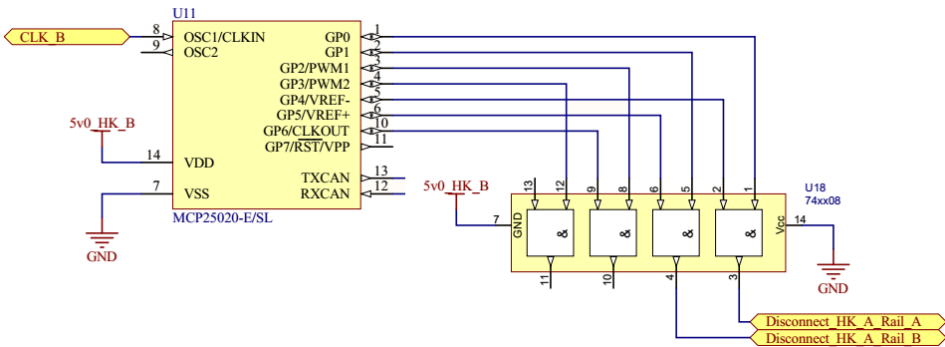


Figure 4.12: One of two mostly identical schematics showing how control signals for the housekeeping power are generated.

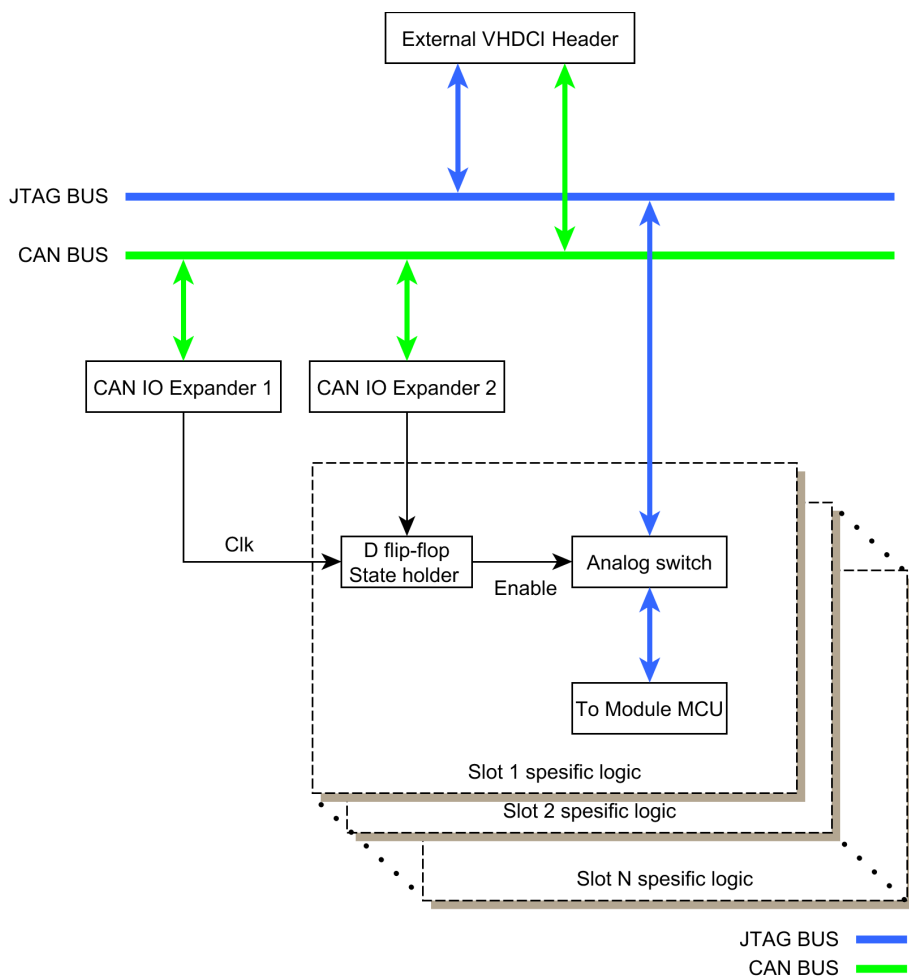


Figure 4.13: Shows how a module is programmed in a fully assembled state by making use of CAN bus control signals and the JTAG bus for programming.

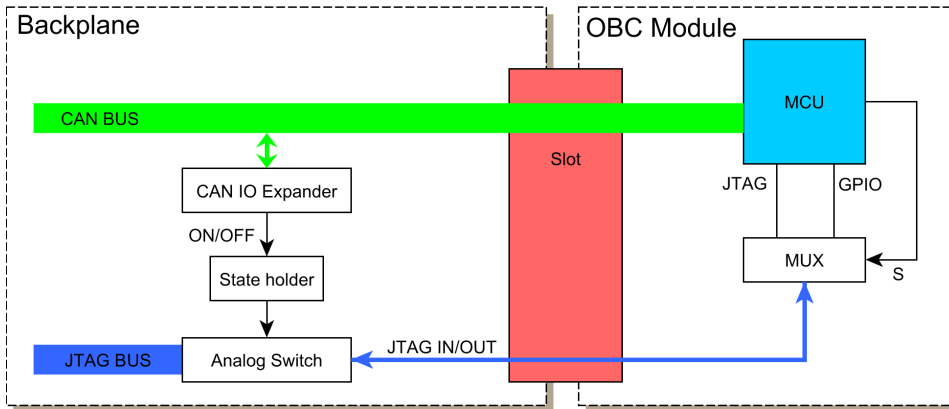


Figure 4.14: Shows inter module re-programming can be achieved by making use on a MUX on the module side of the OBC.

Using this method the OBC MCU can be used to reprogram other modules using a typical "bit banging" method essentially just feeding the binary file to the target MCU via the GPIO pins. This shows how the backplane can facilitate this type of "over the air" updates to any module while the satellite is in operation. Even though this relies heavily upon whether or not the MUX capability is implemented in the OBC the backplane has the ability to facilitate such functionality in the new design.

#### 4.6.5 Control intervals and fault avoidance

As discussed in section 2.2 space is a hostile environment when it comes to electronics. Faults are best avoided or by design tolerated and should be prevented from causing errors which may lead to failures. Steps can be taken when it comes to the frequency in which control signals are sent to the backplane that can reduce or eliminate some types of faults. Especially faults caused by SEE(Single Event Effects) can be reduced by carefully planned timing of control signals to the backplane. In the design the components most vulnerable to bit flips or similar faults caused by SEE's(Single Event Effects) are the D flip flops that hold the state of the logic in each slot. Table 4.3 shows each of the control signals in used in the seven module slot and the severity of a fault occurring on any of the D flip flops holding the state of the signal. A fault on any of the control signals to the e-fuses may disconnect the power rail from the module. These power rails have redundancy and a single power rail being disconnected from a module will not affect the operation of the module. Further more the disconnection of the power rail

Parameter	Default state	Failure consequence
Module power 3v3 A	High	Low (if single fault)
Module power 3v3 B	High	Low (if single fault)
Module power 5v0 A	High	Low (if single fault)
Module power 5v0 B	High	Low (if single fault)
Reset/Utility	Low	High
I <sup>2</sup> C connection ON/OFF	Low	Low
JTAG connection ON/OFF	Low	Low

Table 4.3: Shows the severity of a fault occurring on a given slot control signal.

will trigger an IRQ event making the OBC (On-Board Controller) aware of the event. The JTAG and the I2C control signals are considered non-critical and a fault on either of these signals does not pose any threat to the operation of the satellite. The consequence of a fault on the Reset/Utility signal is highly dependent on what the signal is used for in the module it is connected to and should always be considered critical. A fault on this signal may result in holding a module in a reset state effectively disabling a whole module. As discussed in section 4.6.4 the slot reset signal is in the case of the two OBC's connected to the OBC's MCU reset pin making it possible for one OBC to re-program the other OBC. However, this also makes the OBC vulnerable to a fault on the D flip flop holding that particular state.

As discussed in section 4.6.1 the D - flip flop are controlled by the CAN IO-expander who in turn are controlled by the OBC's (On-Board Controller) via the CAN bus. By ensuring the two OBC's hold a register in memory of the states of all slots we can prevent faults in the D flip flops from escalating. The OBC can, based on timed intervals, update the states of every module slot there by overwriting any single fault that may have occurred. If both OBC's do this individually on timed intervals single power rail disconnection faults are corrected without modules being shut down. And faults where one of the OBC's is disabled by a reset signal held low will be corrected by the other OBC. Suggested control intervals of each of the modules and its states are presented in the appendix section 9.1.

## 4.7 Module slot (implementation)

Each module slot on the backplane makes use of a header which accommodates all the power and communication connections made between a module

and the backplane as shown in figure 4.15. The central 24 pin header in figure 4.15 contains all connections used in a module whilst the two outer 10 pin headers are only used by EPS modules. All power coming from the EPS to power the backplane and other modules is located at the two outermost headers. While any power being delivered to a module is done through the central header. The header layout is designed with a diagonal symmetry which stops modules from being damaged when inserted the wrong way. However, functionality such as the CAN bus and JTAG will not work if the module is inserted the wrong way. Complete symmetry could have been achieved by choosing a bigger header, however this would hinder backwards compatibility. As the EPS module only makes use of the outer most headers and the I<sup>2</sup>C bus it is possible to rotate the EPS 180° making orientation of the battery pack customizable. This three part header layout makes it possible to use the same layout for every module in the system making the modules movable and the satellite more customizable. The separate headers for the EPS power and control signals are not implemented on modules other than the EPS. This removes any conflicts arising from modules being placed in a wrong socket. As discussed in section 3.1.6 a unified module slot and header design facilitates re-usability in the layout process allowing for a faster and less complex PCB layout.

## 4.8 Fault isolation (implementation)

As discussed in section 3.1.7 each of the module slots have been carefully implemented to provide isolation between a module and the backplane. This isolation can effectively prevent failures in modules from propagating faults to other modules. Each of the signals going to and from a module has been carefully evaluated to ensure high fault isolation and the system to be fault tolerant.

Figure 4.16 shows the I<sup>2</sup>C repeater circuit used to provide a barrier between the module side and the backplane side of the sensor bus. Using this repeater we have the ability to hot swap a module in and out of the I<sup>2</sup>C bus. This prevents faults or errors on the I<sup>2</sup>C bus on the backplane side when modules are powered down or up. It also allows modules to be added or removed while the satellite is powered up during testing in a lab environment. Using the I2CA\_Enable signal a module can be disconnected from the backplane preventing a faulty module from locking up the I<sup>2</sup>C bus.

Figure 4.17 shows the circuit providing isolation between a module the backplane on the JTAG bus making use of the 74xx4066 analog switches. By default the analog switches used are in an open non-conducting state



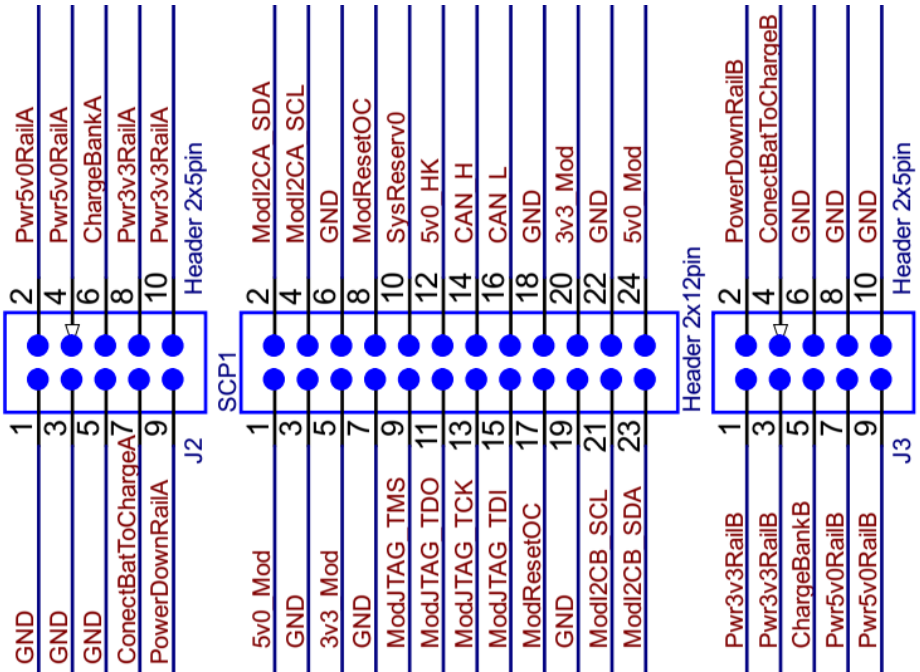


Figure 4.15: Shows the layout of the backplane slot headers where modules connect to the backplane.

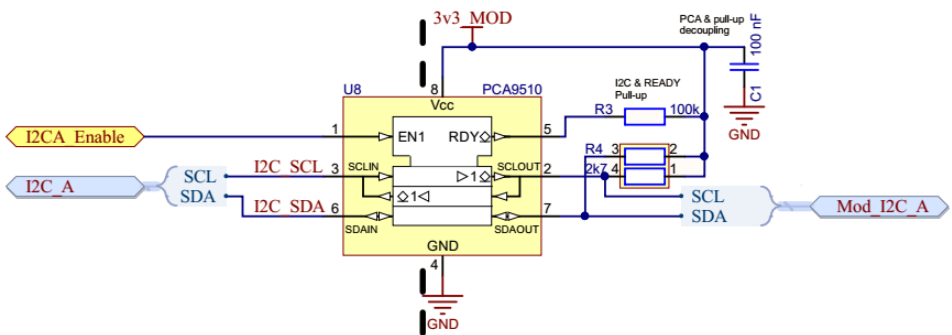


Figure 4.16: Shows how an I<sup>2</sup>C repeater is used to improve fault isolation between the backplane and the module.

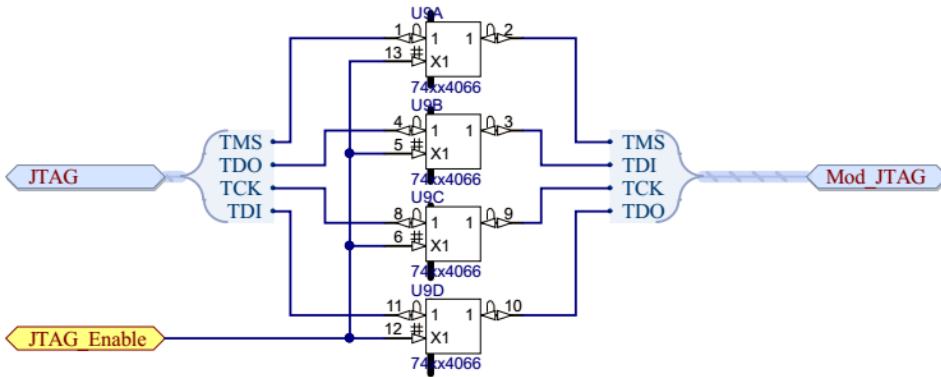


Figure 4.17: Shows how analog switches are used to connect or isolate a module from the JTAG bus.

effectively disconnecting the JTAG bus from all the modules. This circuit is described in more detail in section 4.6.4. Even though this circuit is primarily used for programming purposes, by default it also provides fault isolation preventing a module failure to affect other modules via the JTAG bus.

Figure 4.18 shows the circuit Using the ISO7420 [13] providing a simple galvanic isolation up to 2500 VRMS for the single reset/utility signal provided to the module. This prevents faults on the module side to propagate to the backplane side in the event of a module failure.

#### 4.8.1 Fault isolation control

When a failure occurs it can manifest in several different ways. Digital signals can manifest faults as so-called "stuck at" faults, where the logic level are either "stuck-at 0" or "stuck-at 1" or bit flips where a stored value flips to the opposite value in a memory element. As discussed in section 2.2 the space environment and charged particles in space can heavily expedite these types of failures. From the backplane point of view these faults can manifest in a module in the following ways.

- Unresponsive module
- Unusual power draw (Module short circuit or power failure)
- Bus being "Stuck" or "occupied" constantly by a module
- Backplane failure

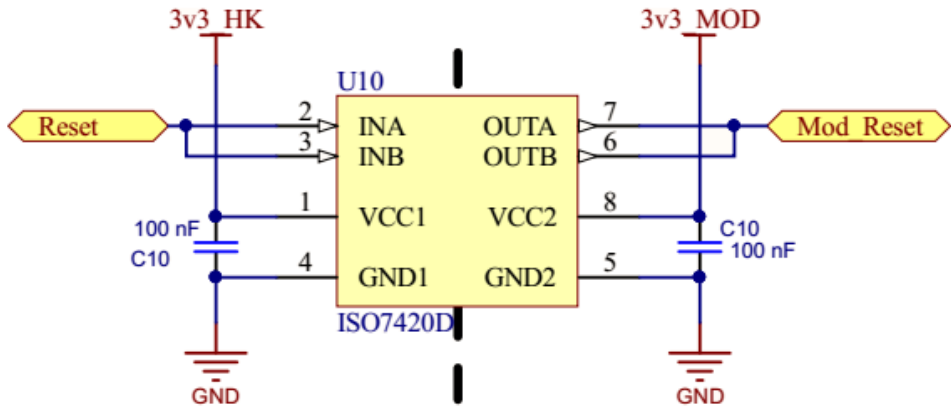


Figure 4.18: Shows how simple galvanic isolation circuit is used to provide fault confinement in the utility/reset signal towards the module.

These faults can be handled either automatically or by one of the OBC's direct control.

### Unresponsive module

An unresponsive module can be reset by an OBC using the CAN bus to control the module slot states providing the module with a reset signal. Alternatively the module can be power cycled or completely shut down. All these actions require the problem to be actively handled by the OBC.

### Unusual power draw (module short circuit or power failure)

In the case of power draw exceeding the power draw limit of the module set in the configuration of the E-fuse's the fault is handled automatically by the e-fuses shutting the module down. However, should a module exhibit unusual behaviour in the form of constant power draw higher than normal but not high enough to trip the e-fuses possibly resulting in unwanted battery depletion, the OBC's has the ability to detect and handle it appropriately. The sensor bus can provide details on each modules power draw. The OBC's can then take action in the form of resetting or temporarily or permanently disabling the affected module.

### Bus being "Stuck" or "occupied" constantly by a module

As discussed in section 2.3 CAN bus is quite robust at particularly these types of situations. In the case of the I<sup>2</sup>C bus the OBC's can disconnect a

module from the I<sup>2</sup>C bus completely effectively removing the troublesome module from the bus.

### **Backplane failure**

In the case of a backplane failure where the backplane logic is stuck or the backplane short-circuits. The OBC's can power down all the logic of the backplane leaving only the power to the modules and the CAN bus operational.

## **4.9 Proposed testing methodology**

Once the design has been fully realized it is crucial that a proper test regime is employed to verify the functionality of the backplane. The backplane PCB should be thoroughly tested before it is used with other modules or assembled in the satellite.

### **4.9.1 What to test (functionality)**

Proposed elements that should be tested are as follows:

- Voltage and current delivery to modules
- Short circuits
- CAN bus and I<sup>2</sup>C bus functionality
- Slot state holders
- PCB vacuum tests

### **4.9.2 How to test (physical access)**

Physical access to the test points needed to perform the testes mentioned in section 4.9.1 can be gained either through the module header in each slot or in the VHDCI header on the PCB.

### **4.9.3 Test bench**

A simple test bench is proposed requiring the following hardware:

- LAB power supply providing 3v3 and 5v0 volt.

- A computer with peripherals able to read and write to the CAN bus and I<sup>2</sup>C bus using a software terminal.
- High power resistors (5W-10W) for simulating module loads.
- Logic analyzer to measure logic levels on several pins at once.

### Test A

The backplane must be powered through the VHDCI header using a LAB powersupply providing 3v3 and 5v0 volt. The CAN bus and I<sup>2</sup>C bus must also be connected to a PC terminal via the VHDCI header. Appropriately sized high power resistors for simulating module loads must be placed in all module slots from the 3v3 and the 5v0 voltages to ground. The test is performed as follows:

1. Power on the powersupply providing the backplane with the different required voltages.
2. Using the PC terminal connected to the I<sup>2</sup>C bus, read data from each of the 16 current sensors on the backplane. (Data set 1)
3. Using the PC terminal connected to the CAN bus disconnect the e-fuses one by one reading the sensor data from the correlating current sensor on the I<sup>2</sup>C bus. (Data set 2)
4. Verify data set 1 and 2 against expected values.

This test verifies that the module slots receive power, that the CAN bus and I<sup>2</sup>C bus are working and that we have control of the e-fuses and powering down module slots.

### Test B

The backplane must be powered through the VHDCI header using a LAB powersupply providing 3v3 and 5v0 volt. The CAN bus and I<sup>2</sup>C bus must also be connected to a PC terminal via the VHDCI header. High power resistors are not needed.

1. Power on the powersupply providing the backplane with the different required voltages.
2. Using a logic analyzer or oscilloscope measure all module slot 3v3 and 5v0 voltages. (Data set 3)

3. Short circuit the 5v0 power of the first module and record the resulting IRQ message on the CAN bus.
4. Short circuit the 3v3 power of the first module and record the resulting IRQ message on the CAN bus.
5. Repeat for all module slots. (Data set 4)
6. Verify that both e-fuse tripped during the short circuit of a voltage by ensuring that each short circuit provided exactly 1 IRQ correlating to that slot. (Data set 5)
7. Repeat the measurement of all module slot 3v3 and 5v0 voltages by using a logic analyzer or oscilloscope . (Data set 6)
8. Verify data sets 3, 4, 5 and 6 against expected values.

### Test C

Using a vacuum chamber repeat test A.

# Chapter 5

## Results

This chapter will briefly present test results from tests performed on parts of the circuit assembled on a breadboard. As the project does not include a finished PCB design a full scale test and verification of the design has not been possible.

### 5.1 Verification of power control signals.

A concern that was raised early in the development process was the complications involved in being able to partially power down the circuitry on the back plane. Especially the functionality where e-fuses powering modules remain powered on, whilst the control logic used to provide control signals to the e-fuses is powered down. This can result in two different undesired scenarios. Firstly the non-powered electronics may essentially "float" behaving in an undefined manner and potentially cause unintended shut downs in the part of the system it controls. Alternatively, the electronics remaining powered can "reverse power" the component potentially harming it or drawing an extra power. A design as shown in 5.1 has been assembled and tested for potential problems.

#### 5.1.1 Test A

Testing the ability of the D flip flop (U2A in Figure 5.1) to drive the E-Fuse EN (U1 pin 14) low disabling the output. The following procedure was used:

1. Powering up the circuit. Measuring points **PowerIn**, **PowerOut** and **Test point A** ensuring all are 5V. (Data set 1)





2. Provide 5V on the **Value** input and a single clock pulse on the **Set** input effectively zeroing out the inverted output driving Test point A and EN LOW.
3. Taking measurements at **PowerIn**, **PowerOut** and **Test point A** (Data set 2)
4. Provide 0V on the **Value** input and a single clock pulse on the **Set** input putting 5V on the inverted output driving Test point A and EN HIGH.
5. Taking measurements at **PowerIn**, **PowerOut** and **Test point A** (Data set 3)

### Test A Results

	PowerIn	PowerOut	Test point A
Data set 1	5.04V (HIGH)	5.04V (HIGH)	5.04V (HIGH)
Data set 2	5.04V (HIGH)	0.0mV (LOW)	80mV (LOW)
Data set 3	5.04V (HIGH)	5.04V (HIGH)	5.04V (HIGH)

Table 5.1: Measurements form Test A driving E-Fuse EN LOW.

Table 5.1 Data set 2 shows that the D flip flop output is able to drive the EN pin on the E-fuse low there by powering down the PowerOut pin.

#### 5.1.2 Test B

Testing short-circuit protection of the E-Fuse by making an intentional short-circuit. Then after removing the short-circuit powering back up the output. The following procedure was used:

1. Powering up the circuit. Measuring points **PowerIn**, **PowerOut** and **Test point A** ensuring all are 5V. (Data set 1)
2. Connecting a wire from **PowerOut** to ground effectively creating a short-circuit.
3. Taking measurements at **PowerIn**, **PowerOut** and **Test point A** (Data set 2)
4. Removing the short-circuit wire and taking measurements at **PowerIn**, **PowerOut** and **Test point A** (Data set 3)

5. In order to reset the E-Fuse the EN pin must be toggled. Provide 5V on the **Value** input and a single clock pulse on the **Set** input effectively zeroing out the inverted output driving Test point A and EN LOW.
6. Provide 0V on the **Value** input and a single clock pulse on the **Set** input putting 5V on the inverted output driving Test point A and EN HIGH.
7. Taking measurements at **PowerIn**, **PowerOut** and **Test point A** (Data set 4)

### Test B Results

	PowerIn	PowerOut	Test point A
Data set 1	5.08V (HIGH)	5.08V (HIGH)	5.04V (HIGH)
Data set 2	5.08V (HIGH)	0.0mV (HIGH)	5.04V (HIGH)
Data set 3	5.08V (HIGH)	0.0mV (HIGH)	5.04V (HIGH)
Data set 4	5.08V (HIGH)	5.08V (HIGH)	5.04V (HIGH)

Table 5.2: Measurements form Test B E-Fuse output short-circuit.

Table 5.2 shows that the E-Fuse is successfully tripped disconnection Power-Out from PowerIn when its output is short-circuited. After the short-circuit has been removed the E-Fuse remains off until the control signal is toggled and the output is successfully powered back up. Current measurements shows the current being limited to 740mA for a short period before the e-fuse shuts down conforming to the current limit set by R1.

### 5.1.3 Test C

Testing consequences of power-loss in the D flip flop circuitry controlling the E-fuse. The following procedure was used:

1. Powering up the circuit. Measuring points **PowerIn**, **PowerOut** and **Test point A** ensuring all are 5V. (Data set 1)
2. Disconnecting supply power to all circuitry in Powerdomain A in Figure 5.1.
3. Taking measurements at **PowerIn**, **PowerOut** and **Test point A** (Data set 2)

4. Reconnecting supply power to all circuitry in Powerdomain A.
5. Taking measurements at **PowerIn**, **PowerOut** and **Test point A** (Data set 3)

### Test C Results

	PowerIn	PowerOut	Test point A
Data set 1	5.08V (HIGH)	5.08V (HIGH)	5.04V
Data set 2	5.08V (HIGH)	5.08V (HIGH)	2.16V
Data set 3	5.08V (HIGH)	5.08V (HIGH)	5.04V

Table 5.3: Measurements form Test C disconnecting supply power to control circuitry.

Table 5.3 shows that powering down the supply voltage to the control logic does not cause adverse effects or unexpected behaviour. The pull-up resistor network connected to EN successfully holds the voltage higher then the cut of voltage allowing the E-fuse to continue to provide power to the subsequent circuitry.



## Chapter 6

# Discussion

As stated in the problem description the existing backplane design has been evaluated and improved upon with respect to how it interconnects all other subsystems of the satellite. A critical choice in the design process was to implement a single CAN bus as a data carrier and CAN based IO expanders to control the backplane functionality. Combined with the unified slot header design in all slots this has simplified the backplane design considerably providing a solution that is a lot easier to layout in PCB design. However, it has also made the satellite highly dependent on the CAN bus and the CAN bus has become a single point of failure. Taking in to account the robustness of the CAN bus during the design process this has been considered an acceptable trade-of in order to reduce the satellite complexity and design time frame. Introducing a second redundant CAN bus could have solved this issue but would have required considerable redesign on the modules of the satellite as well as increasing the design complexity. The CAN IO-expanders used in the new design has been chosen to reduce the complexity of the design and to keep the backplane "dumb" or void of any micro controllers requiring the development of control software. It can be argued however that replacing all the CAN IO-expanders with a single and relatively simple rad-hard micro controller would make the design more versatile and also replace several components on the backplane with a single one. Even though this would introduce an other single point of failure in the system.



## Chapter 7

# Conclusion

A new revision of the NUTS backplane design has been presented where relevant theory has been applied. The new design significantly reduces complexity while increasing the robustness of the backplane. Problems with address spacing on I<sup>2</sup>C bus has been addressed and solved. Further more a uniform slot header design has been achieved making module placement more flexible, module design simpler and PCB layout easier. The power distribution system and power control signals have been partially assembled and functionality verified. A series of tests for verification of the PCB functionality after production has been presented. The end result provides a solid platform for PCB layout an further development of the backplane and NUTS.





# Chapter 8

## Bibliography

- [1] Cubesat project webpage. <http://www.cubesat.org>, [Online; accessed 16-May-2016].
- [2] Roger Birkeland. Nuts-1 mission statement, June 14, 2011.
- [3] J. Scarpulla and A. Yarbrough. What could go wrong? the effects of ionizing radiation on space electronics. *The Aerospace Corporation*, 4(2):15–19, 2003.
- [4] Malcolm Macdonald and Viorel Badescu. *The International Handbook of Space Technology*. Praxis Publishing, Chichester, UK, 2014.
- [5] NXP Semiconductors. *I<sup>2</sup>C-bus specification and user manual*, April 2014. UM10204 Rev. 6.
- [6] Steve Corrigan. *Application Report: Introduction to the Controller Area Network (CAN)*. Texas Instruments, August 2002 (revised July 2008). SLOA101A, Industrial Interface.
- [7] International Organization for Standardization. *Road vehicles – Controller area network (CAN) – Part 1: Data link layer and physical signalling*, 2015. ISO 11898-1:2015.
- [8] Ingulf Helland. *Backplane v4.0 schematics*. NTNU. Rev 4.
- [9] Atmel. *AT32UC3C 32-bit AVR® Microcontroller*, 2012. 32117D–AVR–01/12 - doc32117.
- [10] Texas Instruments. *ISO1050 Isolated CAN Transceiver*, June 2009 - revised January 2015. SLLS983I.

- [11] International Organization for Standardization. *Road vehicles – Controller area network (CAN) – Part 2: High-speed medium access unit*, 2003. ISO 11898-2:2003.
- [12] Texas Instruments. *TPS25942x/44x eFuse Power MUX with Multiple Protection Modes*, June 2014 - revised March 2015. SLVSCE9A.
- [13] Texas Instruments. *ISO742x Low-Power Dual-Channel Digital Isolators*, June 2009 - revised July 2015. SLLS984I.

# Chapter 9

## Appendix

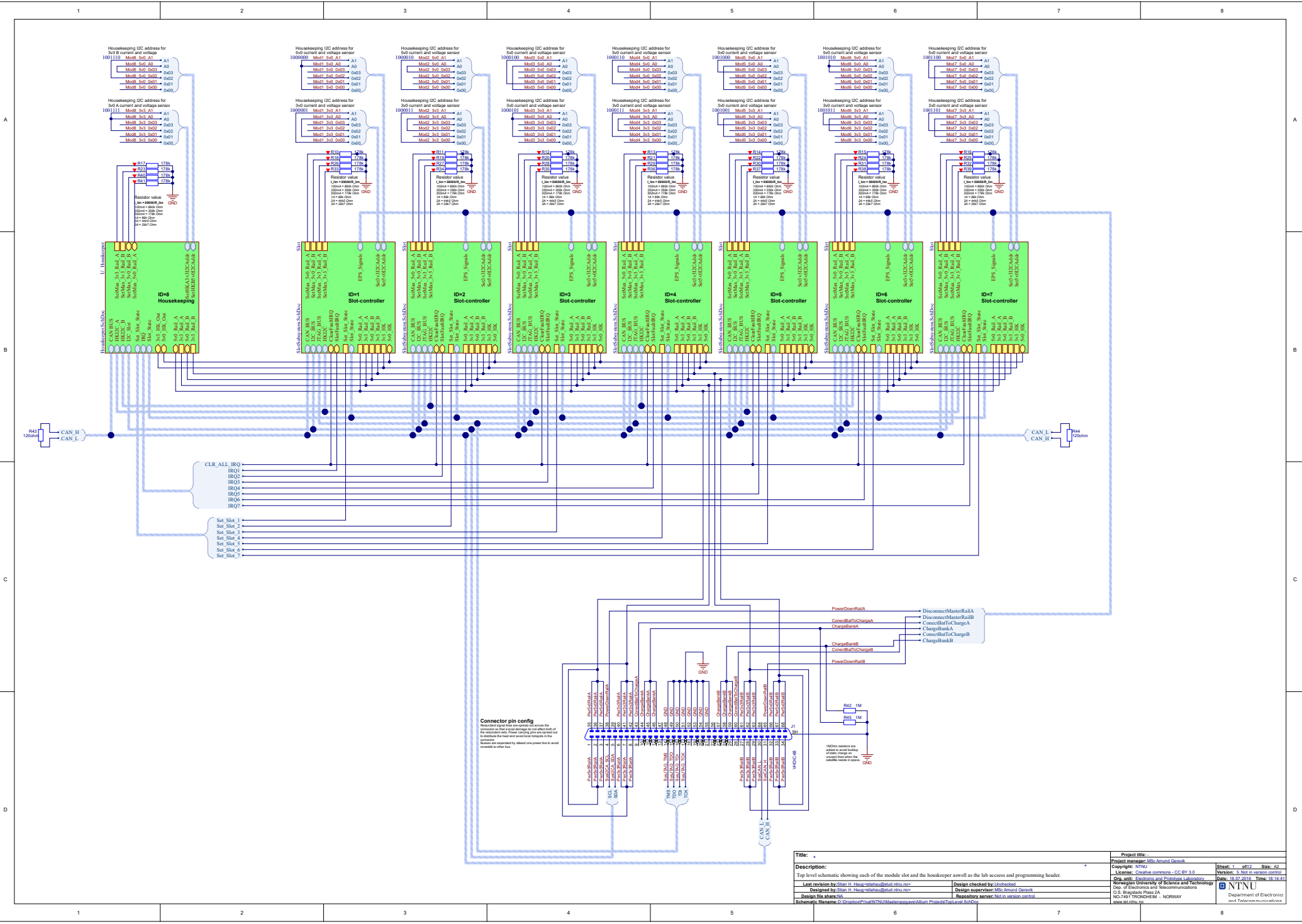
### 9.1 Slot state update interval

Module	Parameter	Controlled by	Interval
OBC A	3v3 A	OBC B	Short
OBC A	3v3 B	OBC B	Short
OBC A	5v0 A	OBC B	Short
OBC A	5v0 B	OBC B	Short
OBC A	Reset/Utility	OBC B	Short
OBC A	I <sup>2</sup> C ON/OFF	OBC B or OBC A	Short
OBC A	JTAG ON/OFF	OBC B or OBC A	Short
OBC B	3v3 A	OBC A	Short
OBC B	3v3 B	OBC A	Short
OBC B	5v0 A	OBC A	Short
OBC B	5v0 B	OBC A	Short
OBC B	Reset/Utility	OBC A	Short
OBC B	I <sup>2</sup> C ON/OFF	OBC A or OBC B	Short
OBC B	JTAG ON/OFF	OBC A or OBC B	Short
ADCS	3v3 A	OBC A or OBC B	Short
ADCS	3v3 B	OBC A or OBC B	Short
ADCS	5v0 A	OBC A or OBC B	Short
ADCS	5v0 B	OBC A or OBC B	Short
ADCS	Reset/Utility	OBC A or OBC B	Short
ADCS	I <sup>2</sup> C ON/OFF	OBC A or OBC B	Short

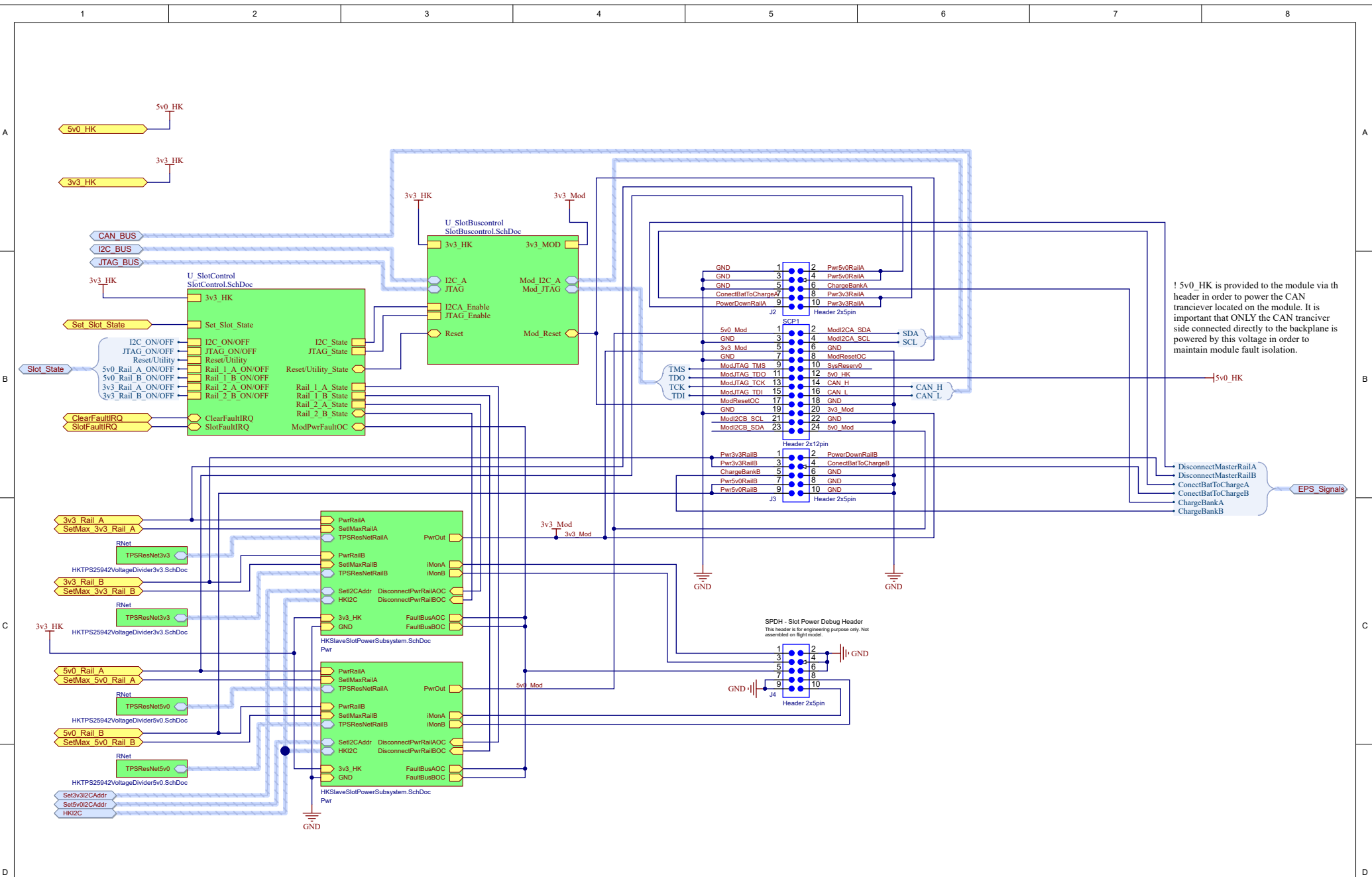
ADCS	JTAG ON/OFF	OBC A or OBC B	Short
EPS	3v3 A	NA	NA
EPS	3v3 B	NA	NA
EPS	5v0 A	NA	NA
EPS	5v0 B	NA	NA
EPS	Reset/Utility	NA	NA
EPS	I <sup>2</sup> C ON/OFF	OBC A or OBC B	Long
EPS	JTAG ON/OFF	NA	NA
Antenna	3v3 A	OBC A or OBC B	Long
Antenna	3v3 B	OBC A or OBC B	Long
Antenna	5v0 A	OBC A or OBC B	Long
Antenna	5v0 B	OBC A or OBC B	Long
Antenna	Reset/Utility	OBC A or OBC B	Long
Antenna	I <sup>2</sup> C ON/OFF	NA	NA
Antenna	JTAG ON/OFF	NA	NA
Payload	3v3 A	OBC A or OBC B	Long
Payload	3v3 B	OBC A or OBC B	Long
Payload	5v0 A	OBC A or OBC B	Long
Payload	5v0 B	OBC A or OBC B	Long
Payload	Reset/Utility	OBC A or OBC B	Long
Payload	I <sup>2</sup> C ON/OFF	OBC A or OBC B	Long
Payload	JTAG ON/OFF	OBC A or OBC B	Long
Unused Slot	3v3 A	NA	NA
Unused Slot	3v3 B	NA	NA
Unused Slot	5v0 A	NA	NA
Unused Slot	5v0 B	NA	NA
Unused Slot	Reset/Utility	NA	NA
Unused Slot	I <sup>2</sup> C ON/OFF	NA	NA
Unused Slot	JTAG ON/OFF	NA	NA
Backplane	3v3 A	OBC A or OBC B	Medium
Backplane	3v3 B	OBC A or OBC B	Medium
Backplane	5v0 A	OBC A or OBC B	Medium
Backplane	5v0 B	OBC A or OBC B	Medium

## 9.2 Design Schematics

The following pages contains the complete schematics for the backplane design revision 5. Sheet numbers 10, 11 and 12 are unaltered since revision 4 and where designed by Ingulf Helland [8].

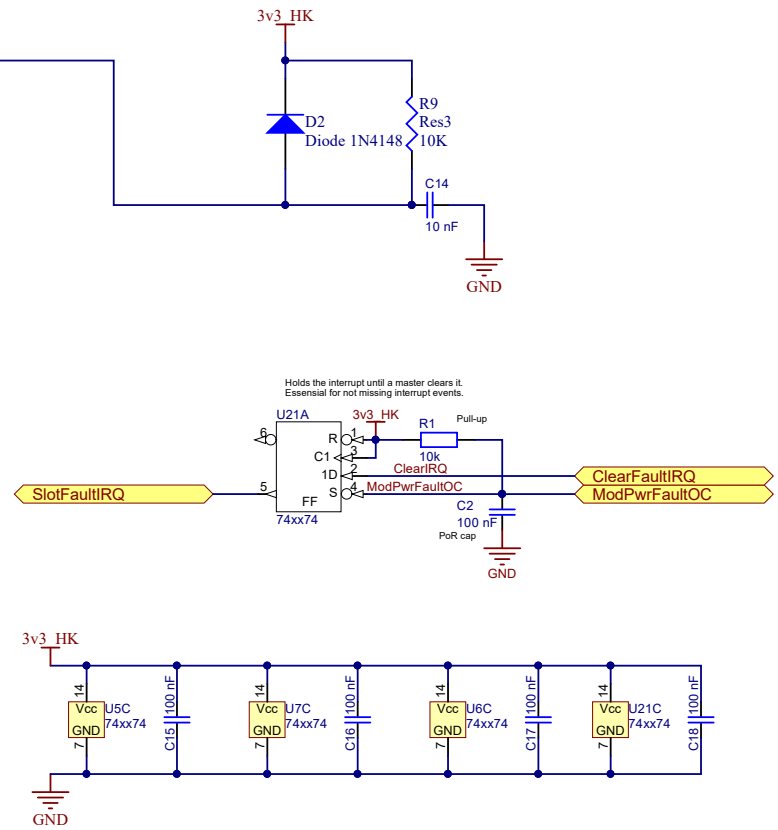
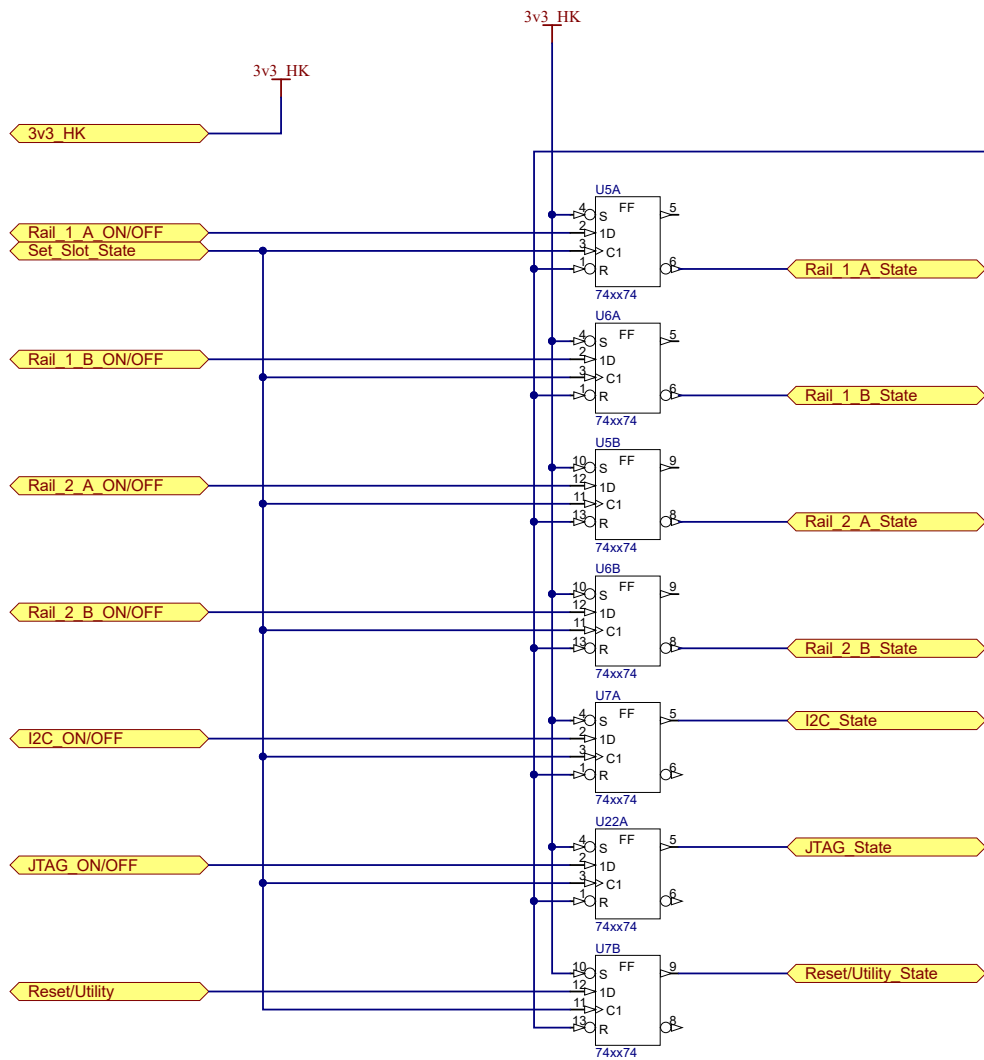


Title:		Project title:	
Description:		Project manager: MSc Armand Geysels	
Top level schematic showing each of the module slot and the housekeeping as well as the lab access and programming leader.		Copyright: NTNU	
Last revision by: Einar H. Hånes (eha@ntnu.no)		License: Creative Commons - CC BY 3.0	
Designed by: Einar H. Hånes (eha@ntnu.no)		Version: 5.10.0 (version control)	
Created by: Einar H. Hånes (eha@ntnu.no)		Date: 18.07.2018 Time: 18:14:41	
Design checked by: Ulf Hestmark		Department of Electronics and Telecommunications	
Design supervisor: MSc Armand Geysels		NTNU	
Repository server: Not in version control		Department of Electronics and Telecommunications	
Schematic, changes: C:\Project\Frog\NTNU\Masterpage\Album\Project\Top level.sch		N-747 TRONDHEIM, NORWAY	



! 5v0\_HK is provided to the module via the header in order to power the CAN transceiver located on the module. It is important that ONLY the CAN transceiver side connected directly to the backplane is powered by this voltage in order to maintain module fault isolation.

<b>Title:</b> Module slot subsystem		<b>Project title:</b> NUTS - NTNU Test Satellite	
<b>Description:</b> Module slot subsystem containing module fault isolation, state holders and power distribution.		<b>Project manager:</b> MSc Amund Gersvik	
<b>Last revision by:</b> <a href="mailto:Sitan.H.Haugstatahu@stud.ntnu.no">Sitan H. Haugstatahu@stud.ntnu.no</a>		<b>Copyright:</b> NTNU	
<b>Designed by:</b> <a href="mailto:Sitan.H.Haugstatahu@stud.ntnu.no">Sitan H. Haugstatahu@stud.ntnu.no</a>		<b>License:</b> Creative commons - CC BY 3.0	
<b>Design file share:</b> NA		<b>Org. unit:</b> Electronic and Prototyping Laboratory	
<b>Schematic filename:</b> D:\Dropbox\Privat\NTNU\Masteroppgave\Altium\Projects\SlotSubsystem.SchDoc		<b>Design supervisor:</b> MSc Amund Gersvik	
		<b>Dep. of Electronics and Telecommunications</b>	
		<b>Norwegian University of Science and Technology</b>	
		<b>NTNU</b>	
		Department of Electronic and Telecommunications	
		O.S. Bregheds Plass 2A NO-7491 TRONDHEIM - NORWAY <a href="http://www.iet.ntnu.no">www.iet.ntnu.no</a>	
		<b>Sheet 2 of 12</b> <b>Size: A3</b>	
		<b>Version:</b> 5. Not in version control	
		<b>Date:</b> 18.07.2016 <b>Time:</b> 18:14:41	



**Title:** Slot state holder and IRQ circuitry

**Description:**

**Last revision by:** Stian H. Haug <stiahau@stud.ntnu.no>

**Designed by:** Stian H. Haug <stiahau@stud.ntnu.no>

**Design file share:** //felles.ansatt.ntnu.no/time/iet/Prototypelab

**Schematic filename:** D:\Dropbox\Privat\NTNU\Masteroppgave\Altium Projects\SlotControl.SchDoc

**Design checked by:** Unchecked

**Design supervisor:** MSc Amund Gersvik

**Repository server:** NA

**Project title:** NUTS - NTNU Test Satellite

**Project manager:** MSc Amund Gersvik

**Copyright:** NTNU

**License:** Creative commons - CC BY 3.0

**Org. unit:** Electronic and Prototype Laboratory

**Norwegian University of Science and Technology**

Dep. of Electronics and Telecommunications

O.S. Bragstads Plass 2A

NO-7491 TRONDHEIM - NORWAY

www.iet.ntnu.no

**Sheet:** 3 **of:** 12 **Size:** A4

**Version:** 5. Not in version control

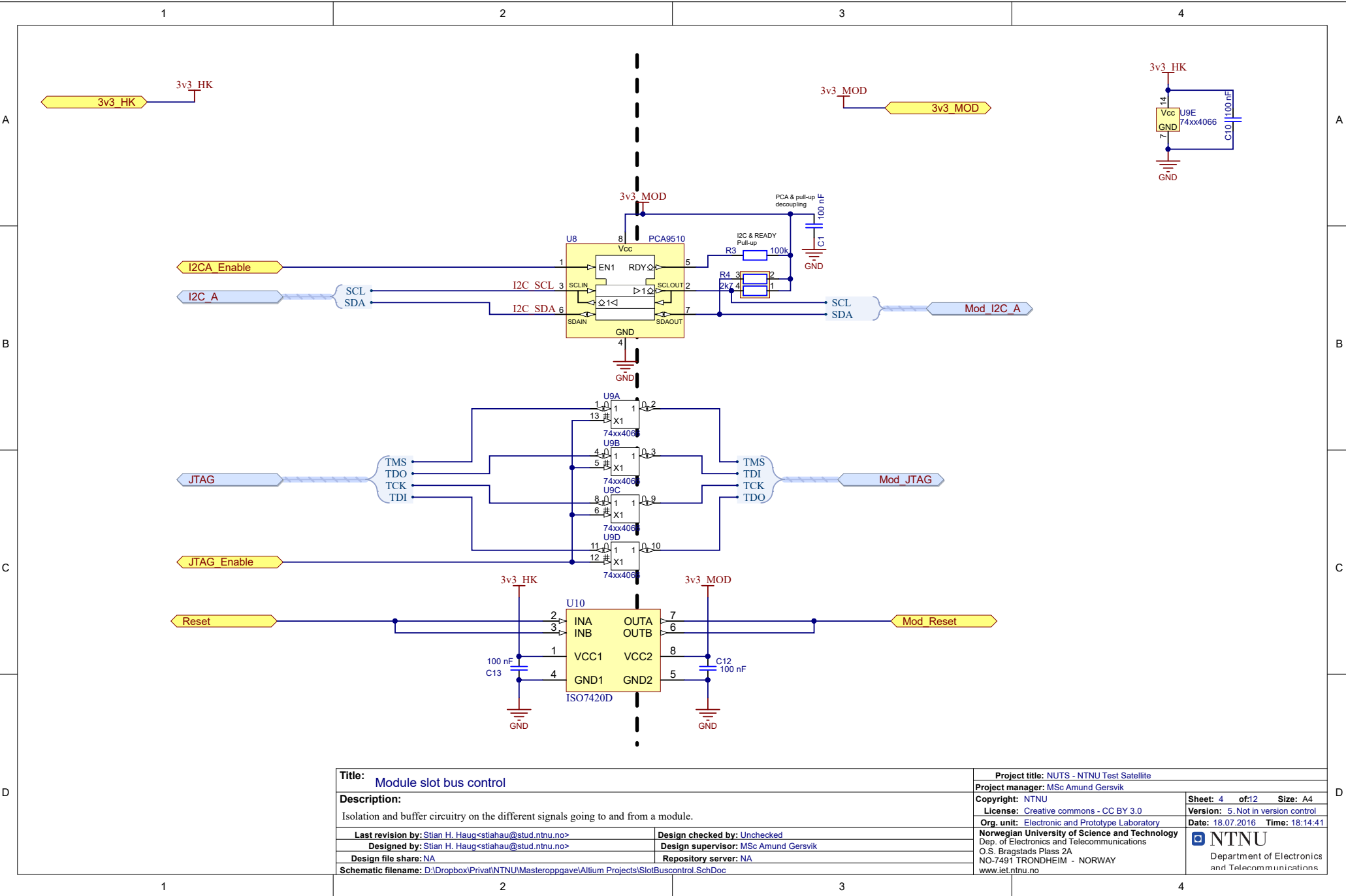
**Date:** 18.07.2016 **Time:** 18:14:41

**NTNU**

Department of Electronics

and Telecommunications

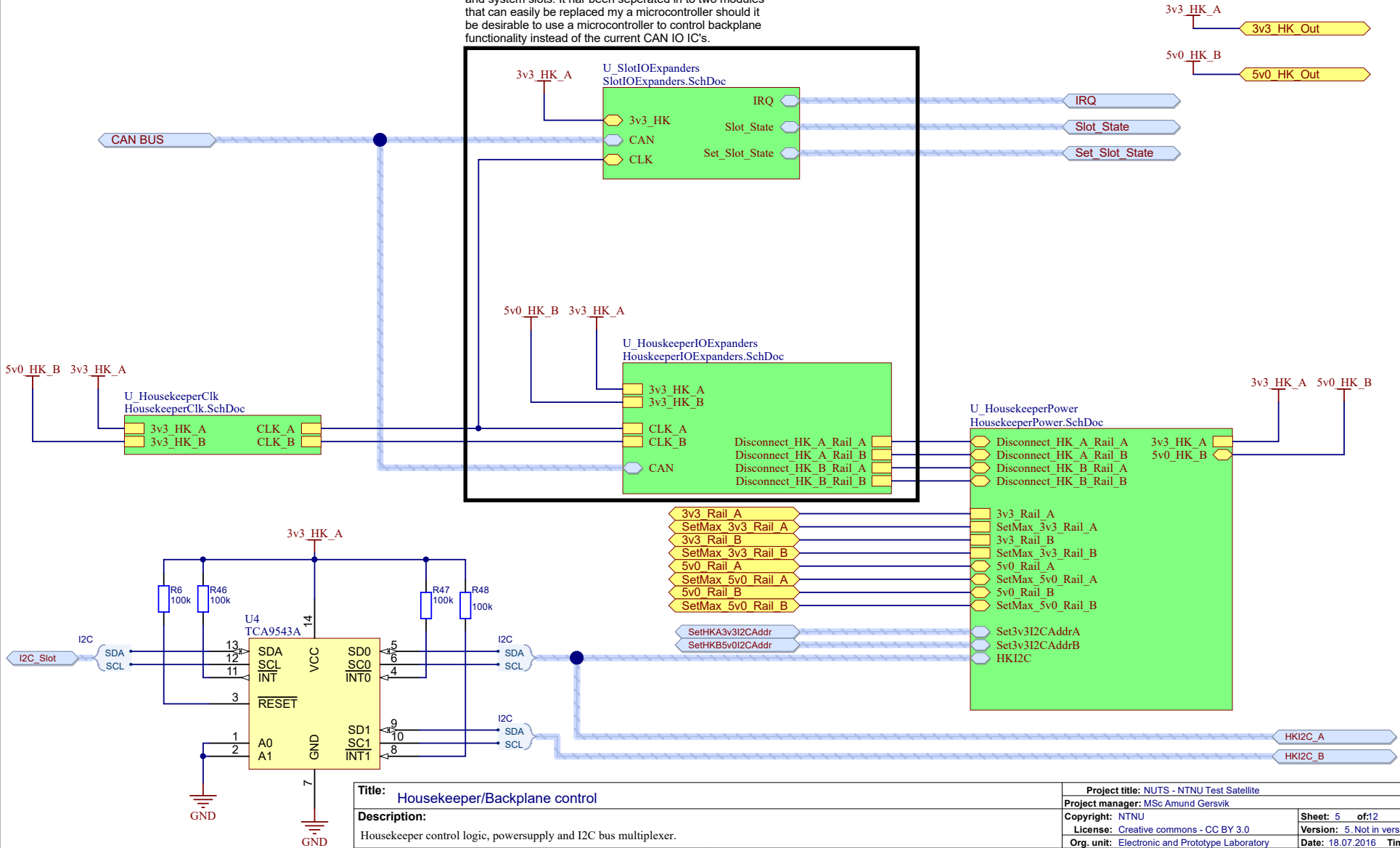




<b>Title:</b> Module slot bus control		<b>Project title:</b> NUTS - NTNU Test Satellite	
<b>Description:</b> Isolation and buffer circuitry on the different signals going to and from a module.		<b>Project manager:</b> MSc Amund Gersvik	
<b>Last revision by:</b> Stian H. Haug<stiahau@stud.ntnu.no>		<b>Copyright:</b> NTNU	
<b>Designed by:</b> Stian H. Haug<stiahau@stud.ntnu.no>		<b>License:</b> Creative commons - CC BY 3.0	
<b>Design file share:</b> NA		<b>Org. unit:</b> Electronic and Prototype Laboratory	
<b>Schematic filename:</b> D:\Dropbox\Privat\NTNUMasteroppgave\Altium Projects\SlotBuscontrol_SchDoc		<b>Version:</b> 5. Not in version control	
<b>Design checked by:</b> Unchecked		<b>Date:</b> 18.07.2016	
<b>Design supervisor:</b> MSc Amund Gersvik		<b>Time:</b> 18:14:41	
<b>Repository server:</b> NA		<b>Sheet:</b> 4 of 12	
		<b>Size:</b> A4	
		<b>NTNU</b> Department of Electronics and Telecommunications	

### Backplane housekeeping control logic

The marked area contains the control logic for backplane and system slots. It has been separated in to two modules that can easily be replaced by a microcontroller should it be desirable to use a microcontroller to control backplane functionality instead of the current CAN IO IC's.



**Title:** Housekeeper/Backplane control

**Description:** Housekeeper control logic, powersupply and I2C bus multiplexer.

Last revision by: Stian H. Haug<stiahau@stud.ntnu.no>

Designed by: Stian H. Haug<stiahau@stud.ntnu.no>

Design file share: NA

Schematic filename: D:\Dropbox\Privat\NTNU\Masteroppgave\Altium Projects\Housekeeper\_SchDoc

Design checked by: Unchecked

Design supervisor: MSc Amund Gersvik

Repository server: NA

Project title: NUTS - NTNU Test Satellite

Project manager: MSc Amund Gersvik

Copyright: NTNU

License: Creative commons - CC BY 3.0

Org. unit: Electronic and Prototype Laboratory

Norwegian University of Science and Technology

Dep. of Electronics and Telecommunications

O.S. Bragstads Plass 2A

NO-7491 TRONDHEIM - NORWAY

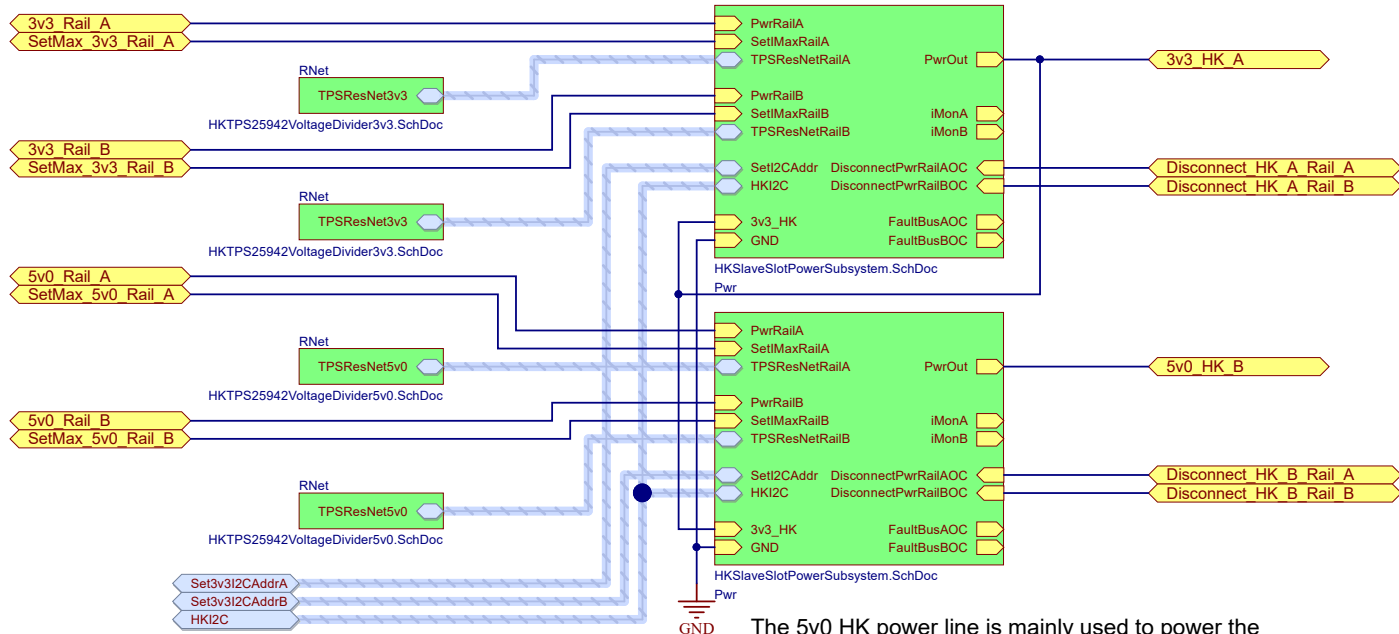
www.iet.ntnu.no

Sheet: 5 of 12 Size: A4


Version: 5. Not in version control

Date: 18.07.2016 Time: 18:14:41

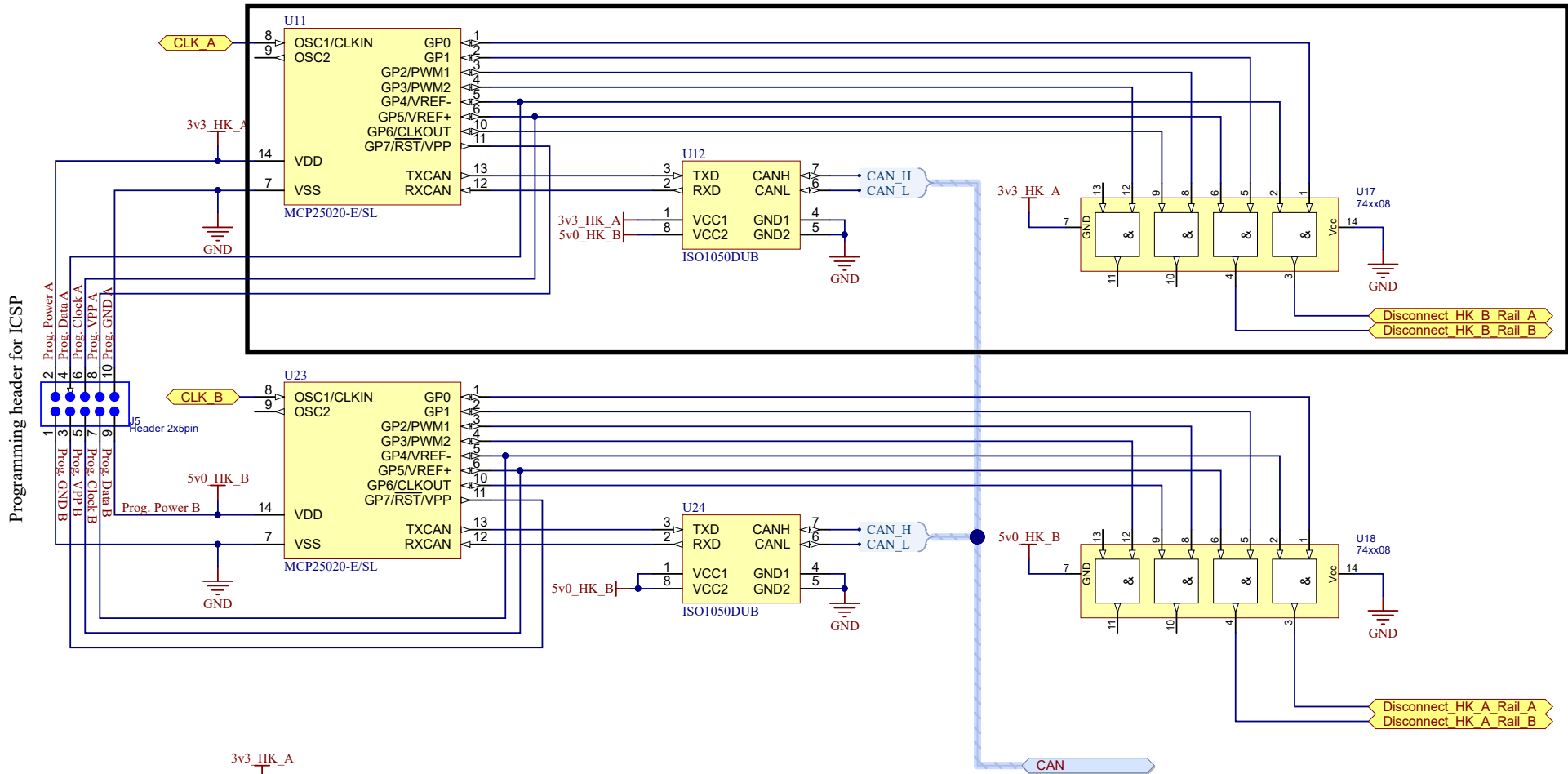




The 5v0 HK power line is mainly used to power the CAN H/L side of the CAN Trancivers used i the design. It is critical that AUTO RETRY E-Fuses are used as a powering down of the 5v0 HK voltage will leave the system unable to send and recive CAN messages on the CAN bus and thus unable to power the 5v0 voltage back up!

<b>Title:</b> Housekeeper/Backplane power		<b>Project title:</b> NUTS - NTNU Test Satellite	
<b>Description:</b> E-fuses, power and currentsensing circuitry for houskeeper and backplane logic.		<b>Project manager:</b> MSc Amund Gersvik	
<b>Last revision by:</b> Stian H. Haug<stiahau@stud.ntnu.no>		<b>Copyright:</b> NTNU	
<b>Designed by:</b> Stian H. Haug<stiahau@stud.ntnu.no>		<b>License:</b> Creative commons - CC BY 3.0	
<b>Design file share:</b> NA		<b>Sheet:</b> 6 of 12 <b>Size:</b> A4	
<b>Schematic filename:</b> D:\Dropbox\Privat\NTNU\Masteroppgave\Altium Projects\HousekeeperPower.SchDoc		<b>Version:</b> 5. Not in version control	
<b>Design checked by:</b> Unchecked		<b>Org. unit:</b> Electronic and Prototype Laboratory	
<b>Design supervisor:</b> MSc Amund Gersvik		<b>Date:</b> 18.07.2016 <b>Time:</b> 18:14:42	
<b>Repository server:</b> NA		<b>Norwegian University of Science and Technology</b>	
		Dep. of Electronics and Telecommunications	
		O.S. Bragstads Plass 2A	
		NO-7491 TRONDHEIM - NORWAY	
		www.i.et.ntnu.no	
		 Department of Electronics and Telecommunications	

The 5v0 HK is controlled by the CAN IO-expander marked by the black box. This control logic should NEVER be used to POWER DOWN the 5v0 HK power as this will result in a situation where the CAN transceiver circuits are powered down and we are unable to send messages to this control logic to power it back up. This circuit should only be used to reset the e-fuses should they be tripped.



Programming header for ICSP

**Title:** Housekeeping CAN IO-expanders

**Description:**  
CAN IO-expanders for controlling housekeeper and backplane power.

Last revision by: Stian H. Haug <stiahau@stud.ntnu.no>

Designed by: Stian H. Haug <stiahau@stud.ntnu.no>

Design file share: NA

Schematic filename: D:\Dropbox\Privat\NTNU\Masteroppgave\Altium Projects\Housekeeper\IOExpanders.SchDoc

Design checked by: Unchecked

Design supervisor: MSc Amund Gersvik

Repository server: NA

Project title: NUTS - NTNU Test Satellite

Project manager: MSc Amund Gersvik

Copyright: NTNU

License: Creative commons - CC BY 3.0

Org. unit: Electronic and Prototype Laboratory

Norwegian University of Science and Technology

Dep. of Electronics and Telecommunications

O.S. Bragstads Plass 2A

NO-7491 TRONDHEIM - NORWAY

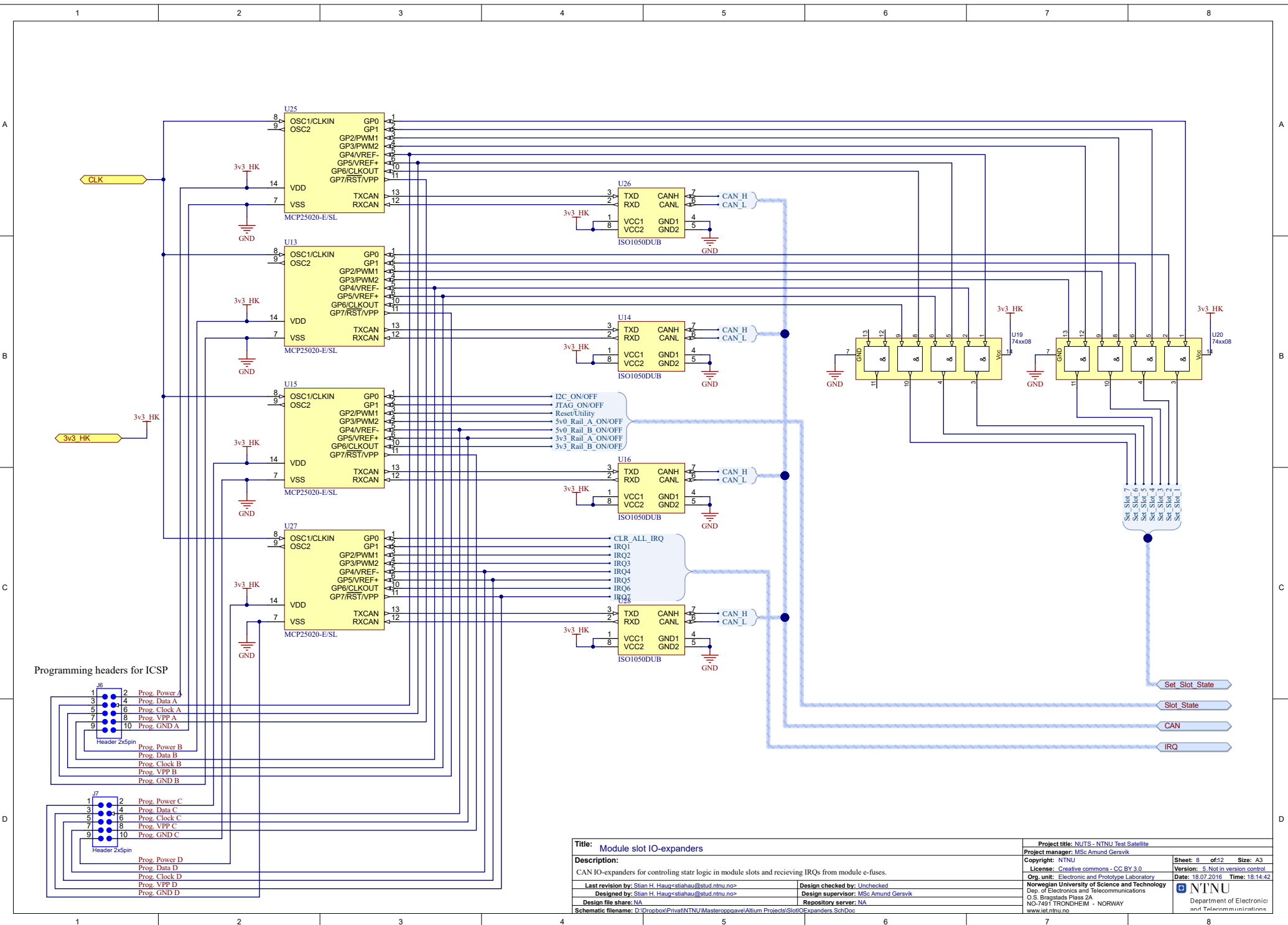
www.iet.ntnu.no

Sheet: 7 of 12 Size: A4

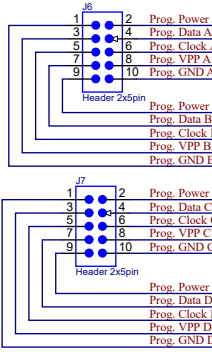
Version: 5. Not in version control

Date: 18.07.2016 Time: 18:14:42





Programming headers for ICSP



<b>Title:</b> Module slot IO-expanders		<b>Project title:</b> NUTS - NTNU Test Satellite	
<b>Description:</b> CAN IO-expanders for controlling start logic in module slots and receiving IRQs from module e-fuses.		<b>Project manager:</b> MSc Amund Gersvik	
<b>Last revision by:</b> Sitan H. Haugstahaug@stud.ntnu.no		<b>Copyright:</b> NTNU	
<b>Designed by:</b> Sitan H. Haugstahaug@stud.ntnu.no		<b>License:</b> Creative commons - CC BY 3.0	
<b>Design file share:</b> NA		<b>Org. unit:</b> Electronic and Prototype Laboratory	
<b>Schematic filename:</b> D:\Dropbox\Privat\NTNU\Masteroppgave\Altium Projects\SlotIOExpanders\SchDoc		<b>Norwegian University of Science and Technology</b> Dep. of Electronics and Telecommunications O.S. Bragestads Plass 2A NO-7491 TRONDHEIM - NORWAY www.iel.ntnu.no	
<b>Design checked by:</b> Unchecked		<b>Date:</b> 18.07.2016	
<b>Design supervisor:</b> MSc Amund Gersvik		<b>Time:</b> 18:14:42	
<b>Repository server:</b> NA		<b>Sheet:</b> 8 of 12	
		<b>Size:</b> A3	
		<b>Version:</b> 5. Not in version control	
		<b>NTNU</b> Department of Electronic and Telecommunications	

1

2

3

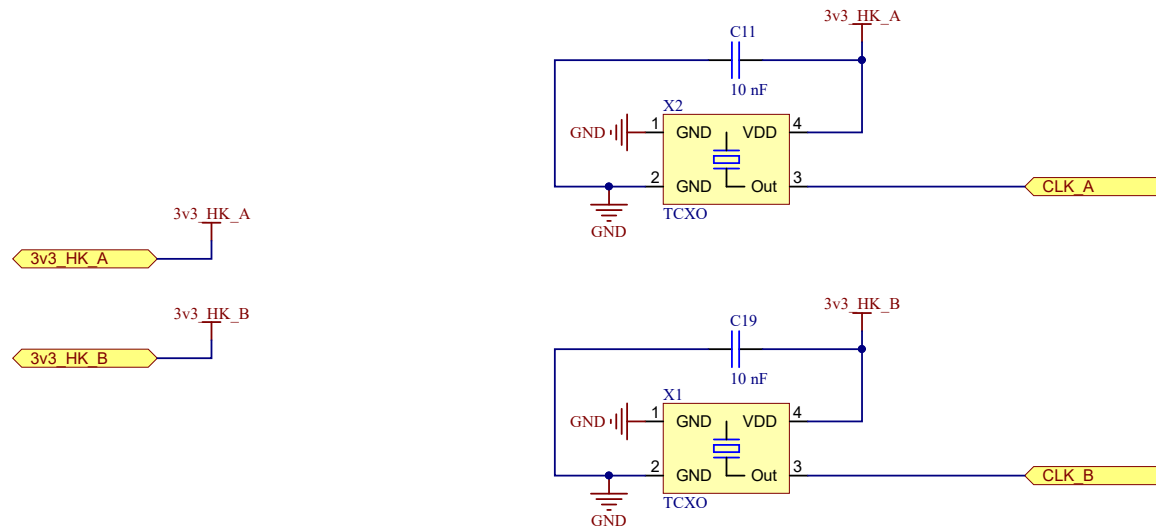
4

A

B

C

D



**Title:** Housekeeper clock sources

**Description:**

Clock sources providing clock to housekeeper logic.

Last revision by: Stian H. Haug<stiahau@stud.ntnu.no>

Designed by: Stian H. Haug<stiahau@stud.ntnu.no>

Design file share: NA

Schematic filename: D:\Dropbox\Privat\NTNU\Masteroppgave\Altium Projects\HousekeeperClk\_SchDoc

Design checked by: Unchecked

Design supervisor: MSc Amund Gersvik

Repository server: NA

Project title: NUTS - NTNU Test Satellite

Project manager: MSc Amund Gersvik

Copyright: NTNU

License: Creative commons - CC BY 3.0

Org. unit: Electronic and Prototype Laboratory

Norwegian University of Science and Technology

Dep. of Electronics and Telecommunications

O.S. Bragstads Plass 2A

NO-7491 TRONDHEIM - NORWAY

www.iet.ntnu.no

Sheet: 9 of 12 Size: A4

Version: 5. Not in version control

Date: 18.07.2016 Time: 18:14:42

NTNU

Department of Electronics

and Telecommunications

1

2

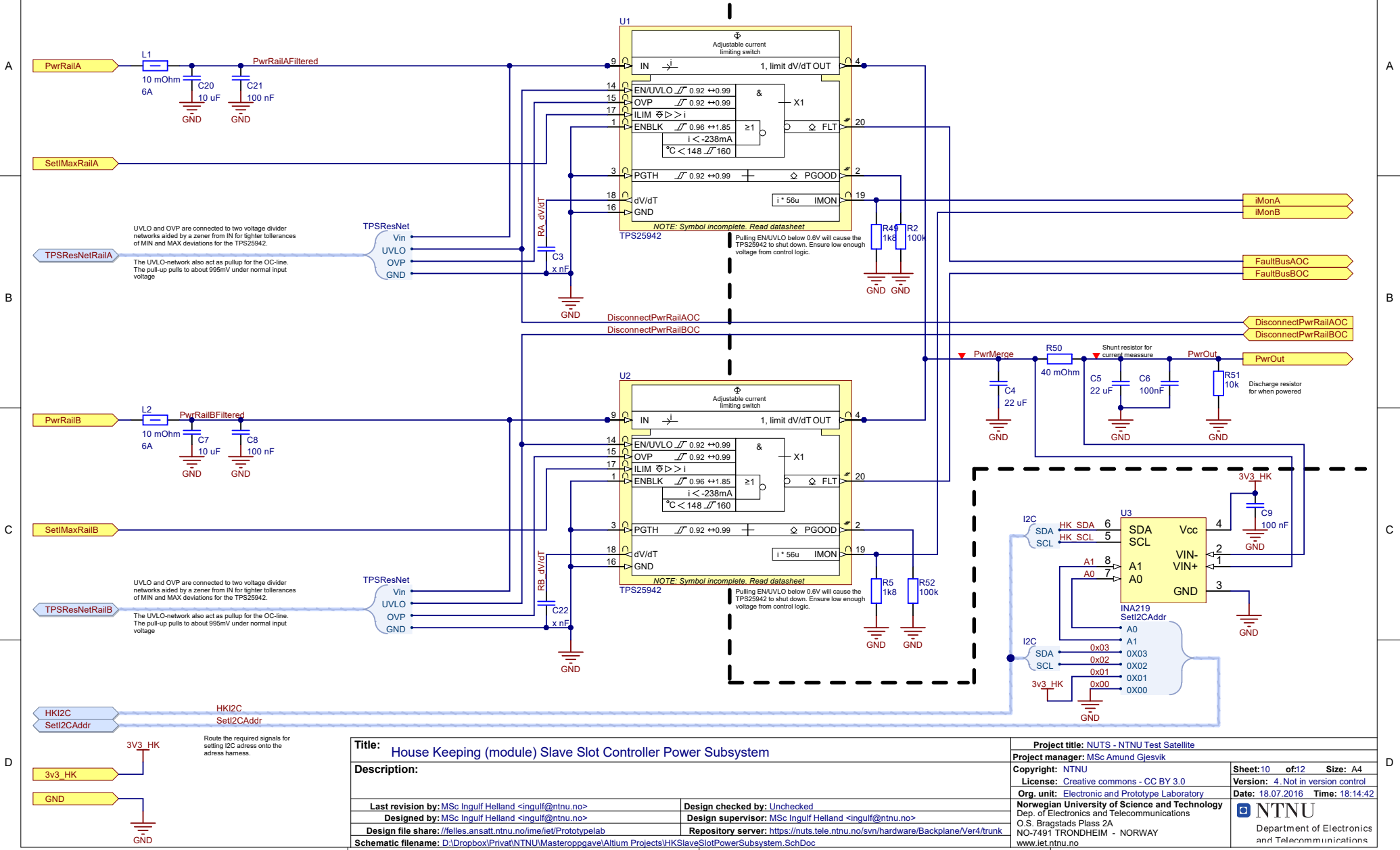
3

4

D

### Fault isolation line

Any signal crossing this line needs to be individually and thoroughly checked for proper fault isolation




**Title:** House Keeping (module) Slave Slot Controller Power Subsystem

**Description:**

Last revision by: MSc Ingulf Helland <ingulf@ntnu.no>  
 Designed by: MSc Ingulf Helland <ingulf@ntnu.no>  
 Design file share: //felles.ansatt.ntnu.no/ime/iet/Prototypelab

Design checked by: Unchecked  
 Design supervisor: MSc Ingulf Helland <ingulf@ntnu.no>  
 Repository server: https://nuts.tele.ntnu.no/svn/hardware/Backplane/Ver4/trunk  
 Schematic filename: D:\Dropbox\Private\NTNU\Masteroppgave\Altium Projects\HKSlaveSlotPowerSubsystem\_SchDoc

<b>Project title:</b> NUTS - NTNU Test Satellite	
<b>Project manager:</b> MSc Amund Gjesvik	
<b>Copyright:</b> NTNU	<b>Sheet:</b> 10 of 12 <b>Size:</b> A4
<b>License:</b> Creative commons - CC BY 3.0	<b>Version:</b> 4. Not in version control
<b>Org. unit:</b> Electronic and Prototype Laboratory	<b>Date:</b> 18.07.2016 <b>Time:</b> 18:14:42
<b>Norwegian University of Science and Technology</b>	
Dep. of Electronics and Telecommunications	
O.S. Bragstads Plass 2A	
NO-7491 TRONDHEIM - NORWAY	
www.iet.ntnu.no	
 Department of Electronics and Telecommunications	

1

2

3

4

A

A

B

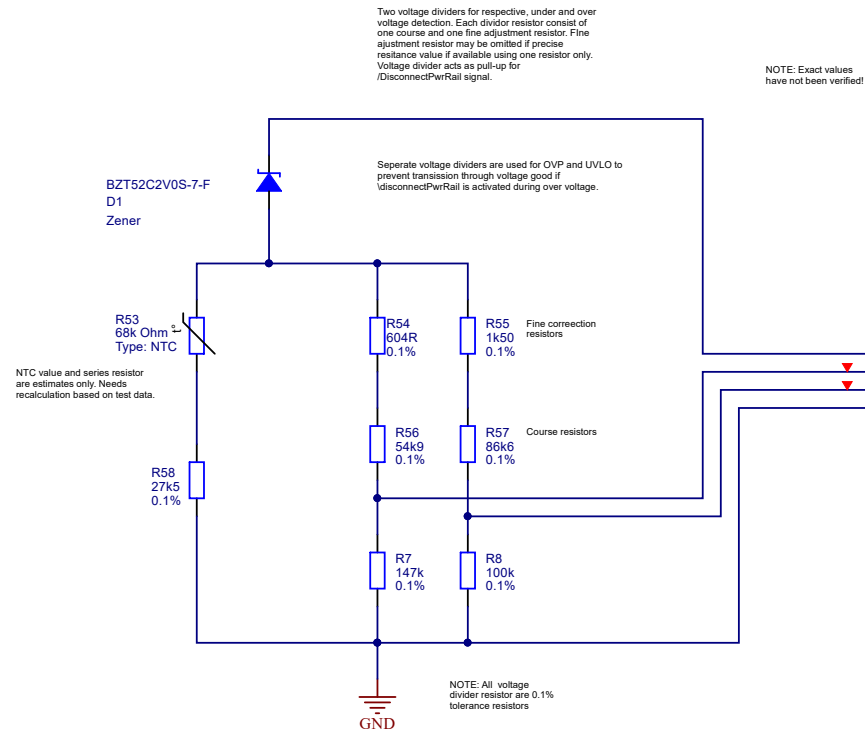
B

C

C

D

D



**Title:** TPS25942-voltage divider for 3v3

**Description:**

**Last revision by:** MSc Ingulf Helland <ingulf@ntnu.no>

**Designed by:** MSc Ingulf Helland <ingulf@ntnu.no>

**Design file share:** //felles.ansatt.ntnu.no/time/iet/Prototypelab

**Schematic filename:** D:\Dropbox\Privat\NTNU\Masteroppgave\Altium Projects\HKTPS25942VoltageDivider3v3.SchDoc

**Design checked by:** Unchecked

**Design supervisor:** MSc Ingulf Helland <ingulf@ntnu.no>

**Repository server:** https://nuts.tele.ntnu.no/svn/hardware/Backplane/Ver4/trunk

**Project title:** NUTS - NTNU Test Satellite

**Project manager:** MSc Amund Gersvik

**Copyright:** NTNU

**License:** Creative commons - CC BY 3.0

**Org. unit:** Electronic and Prototype Laboratory

**Norwegian University of Science and Technology**

Dep. of Electronics and Telecommunications

O.S. Bragstads Plass 2A

NO-7491 TRONDHEIM - NORWAY

www.iet.ntnu.no

**Sheet:** 11 **of:** 12 **Size:** A4

**Version:** 4. Not in version control

**Date:** 18.07.2016 **Time:** 18:14:42

**NTNU**  
Department of Electronics  
and Telecommunications

1

2

3

4



1

2

3

4

A

A

B

B

C

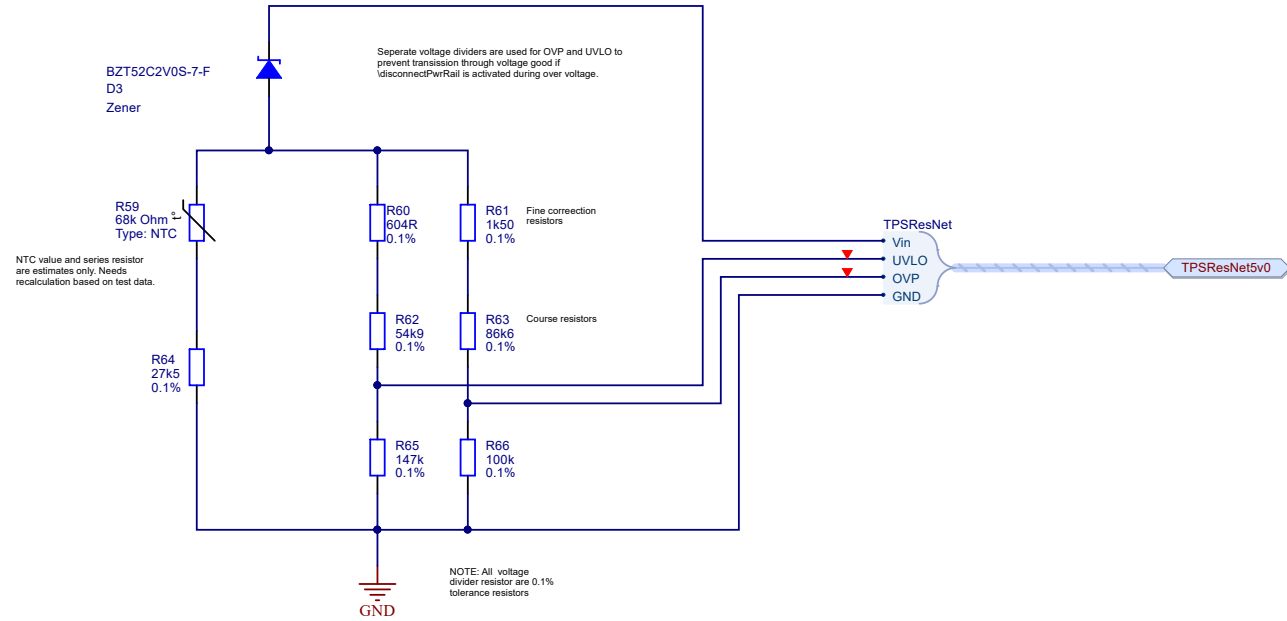
C

D

D

Two voltage dividers for respective, under and over voltage detection. Each divider resistor consist of one course and one fine adjustment resistor. Fine adjustment resistor may be omitted if precise resistance value is available using one resistor only. Voltage divider acts as pull-up for /DisconnectPwrRail signal.

NOTE: Exact values have not been verified!



<b>Title:</b> TPS25942-voltage divider for 3v3		<b>Project title:</b> NUTS - NTNU Test Satellite	
<b>Description:</b>		<b>Project manager:</b> MSc Amund Gersvik	
<b>Last revision by:</b> MSc Ingulf Helland <ingulf@ntnu.no>		<b>Copyright:</b> NTNU	<b>Sheet:</b> 12 <b>of:</b> 12 <b>Size:</b> A4
<b>Designed by:</b> MSc Ingulf Helland <ingulf@ntnu.no>		<b>License:</b> Creative commons - CC BY 3.0	<b>Version:</b> 4. Not in version control
<b>Design file share:</b> //felles.ansatt.ntnu.no/time/iet/Prototypelab		<b>Org. unit:</b> Electronic and Prototype Laboratory	<b>Date:</b> 18.07.2016 <b>Time:</b> 18:14:42
<b>Schematic filename:</b> D:\Dropbox\Privat\NTNU\Masteroppgave\Altium Projects\HKTPS25942VoltageDivider5v0.SchDoc		<b>Norwegian University of Science and Technology</b> Dep. of Electronics and Telecommunications O.S. Bragstads Plass 2A NO-7491 TRONDHEIM - NORWAY www.iet.ntnu.no	<b>NTNU</b> Department of Electronics and Telecommunications

1

2

3

4