Hardware- und Software-Kompatibilitätstests für den Bordrechner eines Kleinsatelliten

Von der Fakultät Luft- und Raumfahrttechnik und Geodäsie
der Universität Stuttgart zur Erlangung der Würde
eines Doktor-Ingenieurs (Dr.-Ing.) genehmigte Abhandlung

Vorgelegt von
Michael Wolfgang Fritz
aus Stuttgart

Hauptberichter:	Prof. Dr. rer. nat. Hans-Peter Röser
Mitberichter:	Prof. Dr.-Ing. Rudolf Benz
	Prof. Dr.-Ing. Roger Förstner
Tag der mündlichen Prüfung:	29. April 2013

Institut für Raumfahrtsysteme der Universität Stuttgart
2013

Quand tu veux construire un bateau, ne
commence pas par rassembler du bois,
couper des planches et distribuer du
travail, mais reveille au sein des hommes
le desir de la mer grande et large.

Antoine de Saint Exupéry (1900-1944)

Inhaltsverzeichnis

Abbildungsverzeichnis

Abkürzungsverzeichnis

APID	Application Process Identifier
BCH	Bose, Chaudhuri und Hocquenghem
CAD	Computer Aided Design
CADU	Channel Access Data Unit
CCSDS	Consultative Committee for Space Data Systems
CLTU	Command Link Transmission Unit
CPDU	Command Pulse Distribution Unit
CPU	Central Processing Unit
DLR	Deutsches Zentrum für Luft- und Raumfahrt
ESA	European Space Agency
ESATAN	European Space Agency Thermal Analysis Network
FPGA	Field Programmable Gate Array
GmbH	Gesellschaft mit beschränkter Haftung
GNU	GNU's Not Unix
GPS	Global Positioning System
HPC	High Priority Command
I/O	Input/Output
I²C	Inter-Integrated Circuit
IBIS	Integrated Bus for Intelligent Sensors
IEEE	Institute of Electrical and Electronics Engineers
IRS	Institut für Raumfahrtsysteme
ITAR	International Traffic in Arms Regulations
JTAG	Joint Test Action Group
Map ID	Multiplexer Access Point Identifier

MDVE	Model-based Development and Verification Environment
MOIS	Manufacturing and Operations Information System
NCTRS	Network Control and Telemetry Routing System
PCDU	Power Control and Distribution Unit
POSIX	Portable Operating System Interface
Proba	Project for On-Board Autonomy
PUS	Packet Utilization Standard
QEMU	Quick Emulator
RIU	Remote Interface Unit
RMAP	Remote Memory Access Protocol
RS	Recommended Standard
RT	Real-Time
RTAI	Real-Time Application Interface
RTEMS	Real-Time Operating System for Multiprocessor Systems
SCOS	Spacecraft Control and Operation System
SIF	Service Interface
SPARC	Scalable Processor Architecture
TC	Telecommand
TCP/IP	Transmission Control Protocol/Internet Protocol
TM	Telemetry
TMS	Thermal Modelling Suite
UART	Universal Asynchronous Receiver Transmitter
UDP	User Datagram Protocol
UML	Unified Modeling Language
USB	Universal Serial Bus
VME	Versa Module Europa

Zusammenfassung

Die Universität Stuttgart hat in Kooperation mit industriellen Partnern im Jahr 2010 das Konzept eines neuartigen Bordrechners für Satelliten vorgestellt. Dieser Bordrechner nutzt modernste Prozessor- und Datenbustechnik. Er wird beim sich am Institut für Raumfahrtsysteme der Universität Stuttgart in Entwicklung befindenden Kleinsatelliten Flying Laptop erstmals eingesetzt.

Ein solcher Bordrechner muss im Betrieb fehlerfrei funktionieren. Auch die auf dem Bordrechner laufende Bordsoftware darf keine systematischen Fehler haben. Fehler in Hardware oder Software können zu einem Komplettausfall des Satelliten führen. Um dieses Risiko zu minimieren, sind während der Entwicklung des Satelliten eine Vielzahl an Bordrechnertests nötig. Für den Bordrechner des Flying Laptops wurde im Rahmen dieser Arbeit ein Prüfstand entwickelt, mit dem ein Großteil der geforderten Tests möglich ist. Durch diesen Prüfstand wurde die Entwicklung der Bordsoftware beschleunigt und deren funktionale Verifikation ermöglicht. Außerdem konnte verifiziert werden, dass die von verschiedenen Herstellern entwickelten Baukomponenten des Bordrechners zueinander kompatibel sind. Darüber hinaus wurde gezeigt, dass die Schnittstellen zwischen Bordrechner und Bordkomponenten sowie zwischen Bordrechner und Missionskontrollsystem ordnungsgemäß funktionieren. Schließlich wurde die gesamte Kommandierungskette vom Missionskontrollsystem über den Bordrechner hin zu exemplarisch gewählten Bordkomponenten erfolgreich getestet.

Die vorliegende Dissertation beschreibt außerdem, welche Planungs- und Entwicklungsarbeiten notwendig waren, um einen solchen Prüfstand aufzubauen. Die dabei aufgetretenen Probleme werden erläutert und dazu erarbeitete Lösungen dargestellt. Für die durchgeführten Bordrechnertests wurden unterschiedliche Konfigurationen des Prüfstands entwickelt und genutzt. Die Konfigurationen und deren jeweiliger Zweck werden beschrieben. Für jede Konfiguration wird anhand eines Anwendungsbeispiels veranschaulicht, wie sich die damit durchgeführten Tests gestalteten. Die erhaltenen Ergebnisse werden dargestellt und bewertet. Geplante Konfigurationen für die nächsten Schritte der Entwicklung werden erläutert. Abschließend wird die weitere Verwendbarkeit des Prüfstands bei der Entwicklung des Kleinsatelliten Flying Laptop diskutiert.

English Abstract

Testing of hardware and software compatibility for a small satellite on-board computer

In cooperation with industrial partners, the University of Stuttgart presented a concept for a modern satellite on-board computer in 2010. This on-board computer is based on a state-of-the-art processor and applies latest interface standards. The Institute of Space Systems is currently developing the small satellite Flying Laptop, which is the first platform for the deployment of this on-board computer.

Spacecraft on-board computers have to work correctly in operation. The applied on-board software must not have any kind of systematic errors. Hardware or software errors can cause system failure disabling the complete satellite. In order to reduce these risks, a lot of on-board computer tests are necessary before launching the satellite. In the frame of this dissertation, a test bench has been developed enabling many of the required tests for the Flying Laptop on-board computer. The application of this test bench has accelerated the development of on-board software and allowed its functional verification. The parts of the on-board computer have been developed by various companies. The application of this test bench has proved that these parts are compatible with each other. It has been demonstrated that the interfaces between on-board computer and connected components work properly. In addition, the interface between on-board computer and mission control system has been successfully tested. Finally, the complete command chain from the mission control system via the on-board computer through to exemplary on-board equipment has been verified.

The present dissertation further describes the design and development for the implementation of such a test bench. Emerging problems are explained and developed solutions depicted. Configurations which have been made possible applying this setup are described. The particular purpose of each configuration is illustrated with application examples. Performed tests and obtained results are shown and evaluated. Configurations for the next steps of development have been planned and are described. Finally, the future usage of the setup for the development of the small satellite Flying Laptop is discussed.

1 Einleitung

Sowohl die Leistungssteigerung von Rechnern [Sweeting 2009], wie auch das größer werdende Leistungsvermögen von Raumflugkörpern [Wertz 2011] führten dazu, dass Bordrechnern von Satelliten, Raumsonden, Raumfahrzeugen und Raumstationen immer mehr Funktionen zugewiesen wurden [Eickhoff 2011]. Die Universität Stuttgart hat in [Eickhoff et al. 2010b] und [Eickhoff et al. 2011] ein Konzept für den Bordrechner eines Kleinsatelliten vorgestellt, der gängige Raumfahrtstandards erfüllt und dem Stand der Technik entspricht. Damit kann dieser Bordrechner künftig auch in industriellen Projekten verwendet werden. Den dabei mitwirkenden industriellen Partnern wird außerdem eine sogenannte In-Orbit Qualification[1] ihrer Baukomponenten ermöglicht, wodurch ein weiterer Mehrwert entsteht.

Bei der Realisierung dieses Konzepts sollte zunächst eine Umgebung geschaffen werden, mit der der Bordrechner getestet werden kann. Diese Tests sollten die spätere bodenseitige Kommandierung ebenso mit einschließen wie die Interaktion zwischen Bordrechner und Bordkomponenten des Satelliten. Auch die Bordsoftware sollte damit funktional verifiziert werden können. Der Bordrechner muss dazu mechanisch und elektrisch in eine entsprechende Testumgebung integriert werden. Um die Entwicklung der Bordsoftware so effizient wie möglich zu gestalten, sollte für entsprechende Software-Tests der Bordrechner mit einem simulierten Raumfahrzeug verbunden werden [Falke 2009]. Bei der Verifikation der Bordsoftware mit dieser Methodik ist darauf zu achten, dass die dafür verwendeten Modelle von Satellitenkomponenten im Simulator korrekt sind und auch möglichst viele Fehlerfälle der realen Hardware simuliert werden können.

Der zu verifizierende Bordrechner basiert auf einem LEON3-FT kompatiblen Prozessor [Eickhoff et al. 2010b, Eickhoff et al. 2011]. Er kommt auf dem in [Grillmayer et al. 2004, Falke 2009, Grillmayer 2010, Kuwahara 2010 und Zeile 2012] vorgestellten Kleinsatelliten Flying Laptop zum Einsatz. Die Kommandierung des Satelliten wird mit Telekommandos durchgeführt, die den Standards des Consultative Commit-

[1] Da es sich hier um einen festen englischen Terminus handelt, wird auch die englische Schreibweise verwendet.

tee for Space Data Systems[2] (CCSDS) entsprechen [CCSDS 2003a, CCSDS 2003b, CCSDS 2010] und den Packet Utilization Standard[3] (PUS) verwenden [ECSS 2003]. Die Kommandierung der Bordkomponenten vom Bordrechner wird durch Ansteuerung einer dedizierten Schnittstellenkomponente über den ebenfalls standardisierten Feldbus SpaceWire durchgeführt [ECSS 2008]. Da die Bordsoftware nicht nur all diese Pakete prozessiert, sondern darüber hinaus die gesamte Satellitensteuerung übernimmt, muss die zur Verifizierung der Software genutzte Simulationsumgebung die dafür notwendigen Anforderungen erfüllen.

In den nachfolgenden Kapiteln wird beschrieben, wie diesen Anforderungen an eine Testumgebung für Bordrechner und Bordsoftware Rechnung getragen werden kann. Dafür wird zunächst die Anwendungsplattform Flying Laptop und der Bordrechner selbst betrachtet. Anschließend wird die dafür genutzte und für die Hardware- und Software-Tests speziell adaptierte Verifikationsumgebung beschrieben, in die der Bordrechner integriert wurde. Die damit aufgebauten Konfigurationen und durchgeführte Tests werden exemplarisch erläutert. Abschließend wird dargestellt, welche Ergebnisse sich aus den unterschiedlichen Tests ergaben und wie diese zu bewerten sind. Die Weiterverwendung der Umgebung für noch ausstehende Projektaufgaben außerhalb des Rahmens dieser Arbeit wird diskutiert.

[2] zu Deutsch: beratender Ausschuss für Datensysteme im Weltraum
[3] zu Deutsch: Paketverwendungsstandard

2 Der Kleinsatellit Flying Laptop

Die University of Surrey hat bereits im Jahr 2002 ein Konzept zur Verwendung standardisierter Kommunikationsprotokolle zwischen Bodenstation und Raumfahrzeug, wie sie in der Raumfahrtindustrie zum Einsatz kommen, sowie moderne Bordrechnerarchitekturen in universitären Kleinsatelliten vorgestellt [Zheng et al. 2002]. Trotzdem wird davon bislang kaum Gebrauch gemacht. Insbesondere in Deutschland entwickelte universitäre Kleinsatelliten werden nicht mit solchen Protokollen kommandiert, stattdessen sind Amateurfunkprotokolle im Einsatz [Schilling 2009]. Auch moderne Bordrechner sind aufgrund begrenzter finanzieller Möglichkeiten vieler Universitäten kaum eine Option.

Vom Institut für Raumfahrtsysteme (IRS) wurde im Jahr 2004 ein Konzept für den Kleinsatelliten Flying Laptop vorgestellt [Grillmayer et al. 2004]. Bei der Fertigstellung des Designs wurde ein Schwerpunkt darauf gelegt, standardisierte Kommunikationsprotokolle, die dem CCSDS Standard entsprechen, zu verwenden [Eickhoff et al. 2010b]. Außerdem kommt ein modernes Bordrechnersystem zum Einsatz, das im nachfolgenden Kapitel näher erläutert wird. Auch die dafür ausgewählten SpaceWire Schnittstellen[1] entsprechen dem Stand der Technik. Abbildung 2.1 zeigt links ein CAD[2] Modell sowie rechts ein nicht funktionales Modell des Kleinsatelliten Flying Laptop. In Abbildung 2.2 ist die für den Flug vorgesehene Struktur des Flying Laptop zu betrachten. Die quaderförmige Konfiguration mit eingeklappten Solarpaneelen hat Abmessungen von 60x70x84 cm^3. Die Gesamtmasse des Satelliten beträgt zwischen 120 und 130 kg. Die Struktur wird teilweise in Sandwichbauweise mit kohlenstofffaserverstärktem Kunststoff als Deckschicht und einem Stützkern in Wabenform aus Aluminium realisiert, um sowohl hohe Steifigkeit wie auch einen geringen thermischen Ausdehnungskoeffizienten zu ermöglichen [Lengowski et al. 2007]. Ansonsten werden reine Aluminiumteile für die Struktur verwendet, die die durch Bordkomponenten dissipierte Wärme besser leiten. Temperatursensoren, elektrische Heizer und Multilayer Insulation tragen zur Regelung der Temperatur bei.

[1] Sämtliche im Text genannten Hardwareschnittstellen werden in Anhang B und C ab Seite 115 detailliert beschrieben.

[2] Computer Aided Design; zu Deutsch: rechnergestützter Entwurf

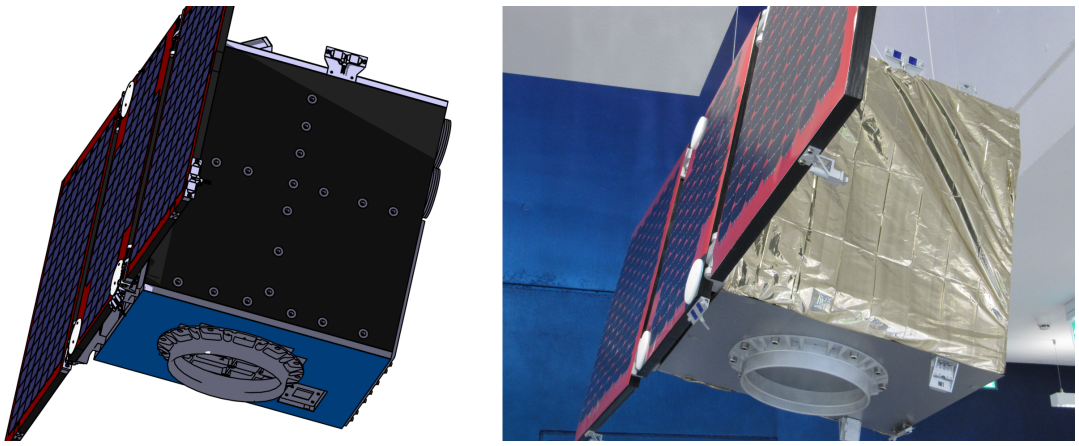


Abbildung 2.1: Kleinsatellit Flying Laptop

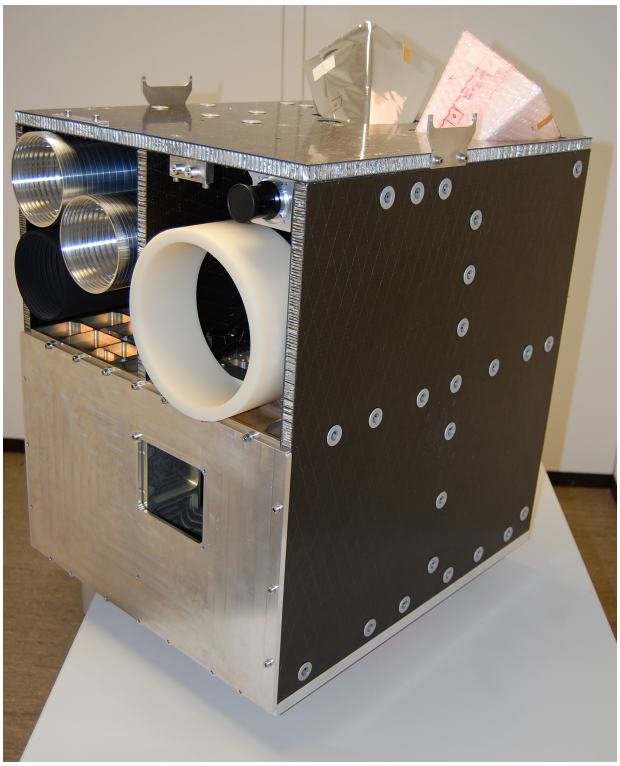


Abbildung 2.2: Struktur des Flying Laptop

Als Nutzlasten sind für den Flying Laptop verschiedene Kameras und ein optischer Kommunikationslink vorgesehen. Die Lageregelung erfolgt mit einem Lageregelungssystem zur Ausrichtung auf beliebige Punkte, das aus Sonnensensoren, Sternsensoren, Faseroptischen Kreiseln, GPS-Empfängern, Magnetometern, Reaktionsrädern und Magnettorquern besteht [Grillmayer 2010]. Abbildung 2.3 zeigt die Lagerregelungskomponenten. Die Kommunikation vom Boden erfolgt mit einer eigenen Bodenstation im S-Band Frequenzbereich.

Der Satellit wird als Huckepack-Nutzlast gestartet. Dabei wird freie Kapazität auf der beim Start eines oder mehrerer großer Satelliten verwendeten Trägerrakete für

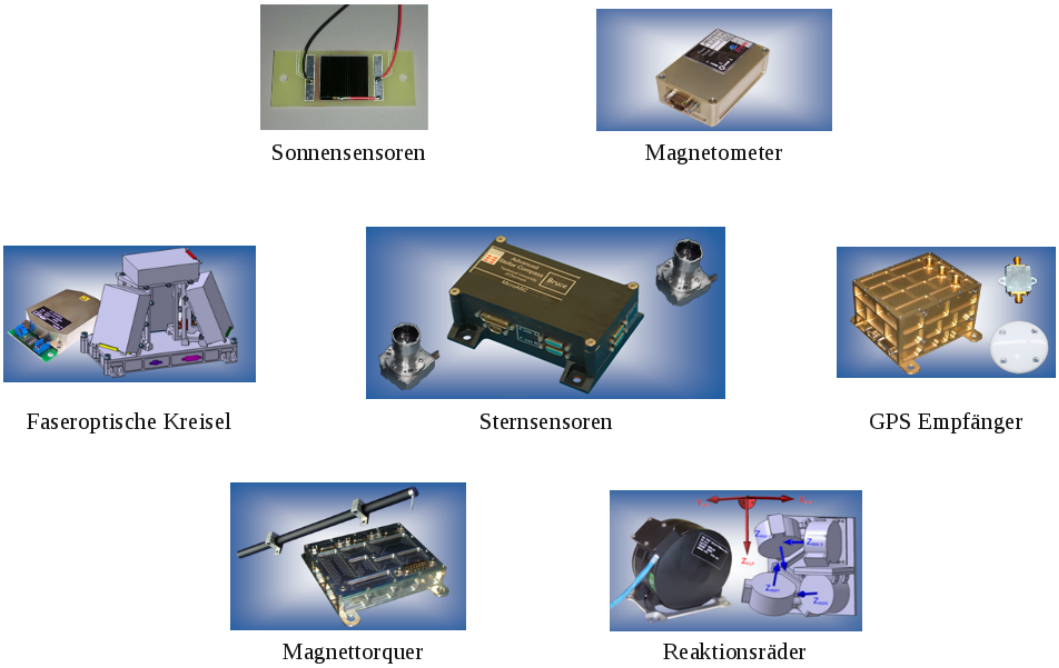


Abbildung 2.3: Lageregelungskomponenten des Flying Laptop [Grillmayer 2010]

kleinere Satelliten genutzt. Erst nach der erfolgreichen Trennung dieser großen Satelliten, der sogenannten Hauptnutzlast, von der Oberstufe der Trägerrakete werden nach und nach auch die Kleinsatelliten davon abgetrennt. Der Orbit hängt damit aufgrund des nicht vorhandenen Bahnregelungssystems von der Hauptnutzlast der Trägerrakete ab. Vorgesehen ist der Start in einen sonnensynchronen Orbit. Ein sonnensynchroner Orbit ist annähernd polar, der Satellit fliegt also über die Nord- und Südpolgebiete der Erde hinweg. In allen Bahnen, die eine Inklination ungleich 90° aufweisen, wird aufgrund der Erdabplattung ein Drehmoment auf die Bahn erzeugt. Dadurch verschiebt sich die Rektaszension des aufsteigenden Knotens. Bei einem sonnensynchronen Orbit verschiebt sie sich genau so schnell, dass die Bahn in einem Jahr genau einmal in Richtung der Erdrotation um die Erde geführt wird. Damit hat die Bahn immer den gleichen Winkel zur Verbindungslinie zwischen Erde und Sonne und damit immer den gleichen Anteil der Zeit im Erdschatten gemessen an der Gesamtumlaufzeit. Für einen sonnensynchronen Orbit steht durch die Höhe automatisch auch die Inklination fest [Wertz 2011]. Durch Störungen, die die Umlaufbahn des Satelliten beeinflussen und durch das nicht vorhandene Bahnregelungssystem kann die Sonnensynchronizität jedoch während des Betriebs des Flying Laptop nicht dauerhaft sichergestellt werden.

Abbildung 2.4 zeigt das elektrische Blockschaltbild des Flying Laptop. Deutlich sichtbar sind die vielen Hardwareschnittstellen und Verbindungen zur Übertragung

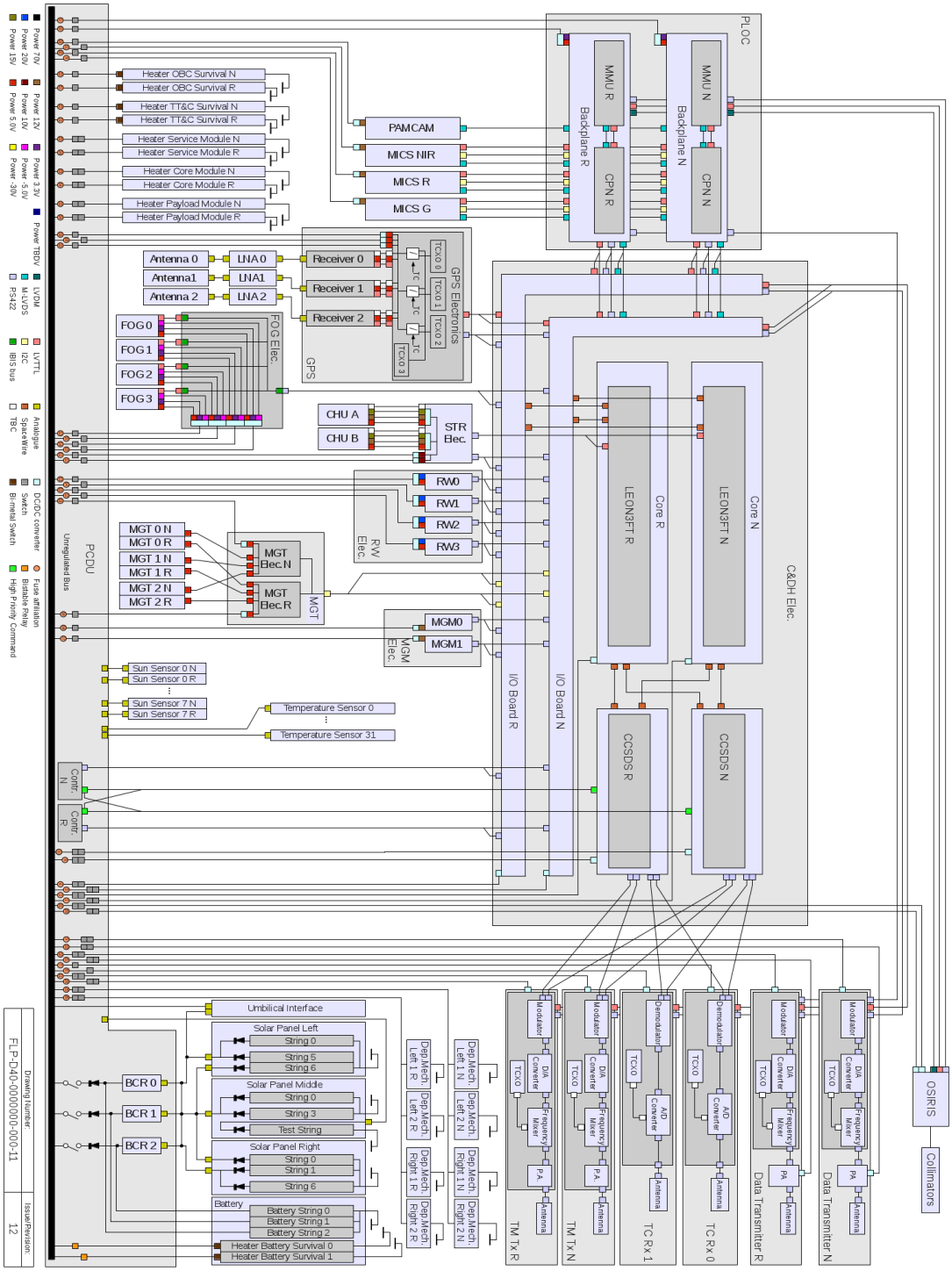


Abbildung 2.4: Elektrisches Blockschaltbild des Flying Laptop

von Daten und Strom an Bord des Satelliten. Die meisten dieser Verbindungen sollen wiederum aus mehreren Leitungen bestehen. Aus dem Blockschaltbild geht außerdem hervor, dass viele Komponenten mit dem in der Mitte abgebildeten Bordrechnersystem verbunden sind.

Mit den oben beschriebenen Schwerpunkten, insbesondere mit dem Bordrechnersystem, wird demonstriert, dass auch universitäre Kleinsatelliten mit modernen und standardisierten Konzepten realisierbar sind. Ein solcher Satellitenbus kann später

erneut gebaut und mit anderer Nutzlast versehen werden. Außerdem lernen beteiligte Studenten und Absolventen früh übliche Standards und moderne Hardware kennen. Im nachfolgenden Kapitel wird das Bordrechnersystem und die darauf laufende Bordsoftware genauer beschrieben. Mit Kenntnissen hinsichtlich des Aufbaus der Hardware und der Architektur der Bordsoftware werden dann die Umgebung für die Kompatibilitätstests entwickelt und die Tests selbst durchgeführt.

3 Der Bordrechner des Flying Laptop

Das Bordrechnersystem von Satelliten besteht oft aus zwei getrennt zu betrachtenden Einheiten: Einem Nutzlastrechner für die Ansteuerung der Nutzlast, der Prozessierung, Speicherung und Übertragung von Nutzlastdaten sowie einem Plattformrechner zur Ansteuerung und Regelung der Plattform, dem Empfang der Telekommandos vom Boden, der Erzeugung von Informationen hinsichtlich des Satellitenstatus sowie zum Fehlermanagement [Eickhoff 2011]. Der Ausfall von Nutzlastrechnern führt zum eingeschränkten Betrieb der Nutzlast, wohingegen der Ausfall von Plattformrechnern den Verlust der gesamten Mission bedeuten kann.

Da ein solches Bordrechnersystem im Weltall nicht durch direkten Eingriff verändert werden kann, muss es in hohem Maße zuverlässig und fehlertolerant sein. Darüber hinaus muss es die Randbedingungen an Bord wie Volumen, Masse und Leistung erfüllen und gleichzeitig robust gegenüber Umweltbedingungen wie Vakuum, erhöhter Strahlung sowie Vibrationen und Schocks beim Start sein. Auch die maximalen und minimalen operationellen Temperaturen sind in der Regel weiter auseinander liegend als bei Rechnern auf der Erde. Diese Anforderungen resultieren in niedrigen Prozessortaktraten und geringem Arbeitsspeicher. Außerdem sollte ein Bordrechnersystem die Möglichkeit zur Software-Aktualisierung im Flug bieten, um auch noch nach dem Start erkannte Software-Fehler beheben zu können [Ley 2009].

Auch das Bordrechnersystem des Kleinsatelliten Flying Laptop besteht aus einem Plattformrechner und einem Nutzlastrechner [Eickhoff et al. 2010b]. Die vorliegende Arbeit behandelt ausschließlich den Plattformrechner, der im weiteren Text zur Vereinfachung Bordrechner genannt wird. Auf diesem läuft die für den Satellitenbetrieb kritische Software, die mit Hilfe der später beschriebenen Kommandierungskette verifiziert werden kann.

3.1 Hardware

Der Bordrechner des Kleinsatelliten Flying Laptop besteht aus vier jeweils einfach redundanten Platinen, die alle in ein eigenes Gehäuse eingebaut sind. Neben der Prozessorplatine, dem sogenannten Bordrechnerkern, handelt es sich dabei um zwei baugleiche Schnittstellenkarten und eine Stromversorgungsplatine. Die Schnittstellenkarten sollen einerseits die Verbindung zum Kommunikationssystem, andererseits die Verbindung zu Sensoren und Aktuatoren herstellen. Abbildung 3.1 stellt schematisch den mechanischen Aufbau des Bordrechners dar. Das Gesamtkonzept wurde in [Eickhoff et al. 2010b] der Öffentlichkeit vorgestellt.

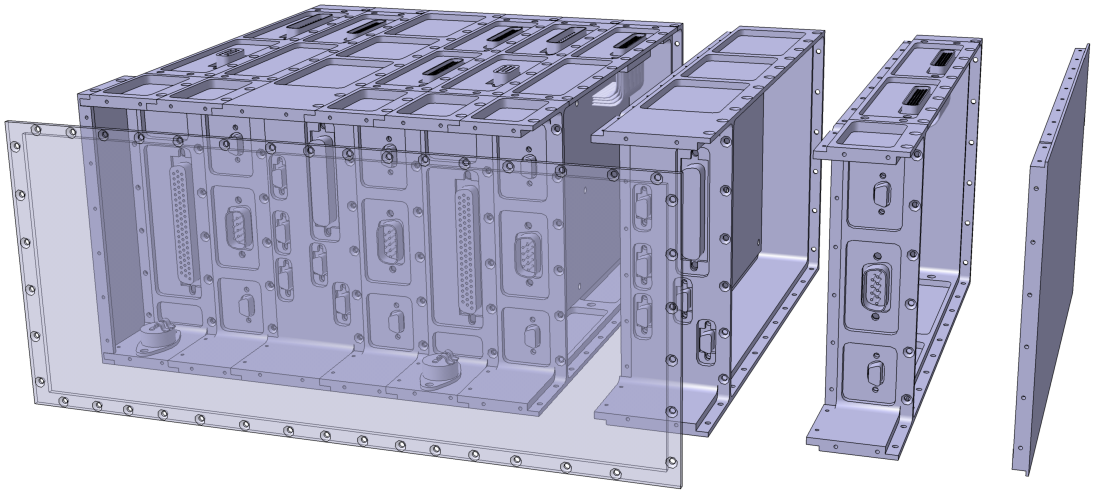


Abbildung 3.1: Mechanischer Aufbau des Bordrechners [Eickhoff 2011]

Bordrechnerkern

Der Bordrechnerkern ist mit einem LEON3-FT-kompatiblen Prozessor bestückt. Der LEON basiert auf der SPARC V8 Mikroprozessorarchitektur [Gaisler 2002] und implementiert eine sogenannte System-On-A-Chip Architektur. Dabei werden sowohl der Prozessorkern wie auch diverse Bausteine für Schnittstellen auf einen einzigen Chip integriert. Zusätzliche integrierte Schaltkreise sind damit nicht mehr notwendig, was zur Miniaturisierung beiträgt. Der LEON3-FT wird beim Flying Laptop in der Ausführung UT699 der Firma Aeroflex verwendet. Abbildung 3.2 stellt den schematischen Aufbau der UT699 Prozessorplatine dar und zeigt, welche zusätzlichen Elemente außerhalb der System-On-A-Chip Architektur verfügbar sind. Diese Version bietet neben vier SpaceWire Schnittstellen, über die der Bordrechner mit anderen Komponenten kommunizieren kann, auch eine RS-422 Schnittstelle für die Ausga-

be von Meldungen, eine Ethernet Schnittstelle sowie flüchtigen und nichtflüchtigen[1] Speicher. Auf dem UT699 steht außerdem eine JTAG[2] Schnittstelle zur Konfiguration des Rechners und zur Fehlersuche auf demselben zur Verfügung. Die vier SpaceWire Schnittstellen werden zur Ansteuerung der nominellen und redundanten Peripherieplatinen des Bordrechners eingesetzt.

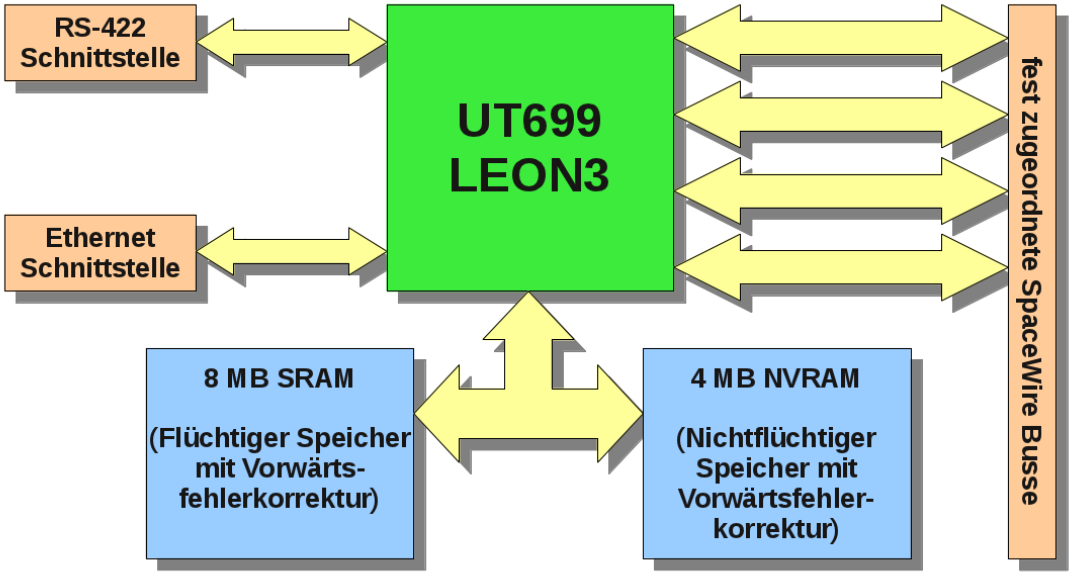


Abbildung 3.2: Blockdiagramm der Bordrechnerkernplatine

I/O Board

Eine der beiden Peripherieplatinen dient dazu, die Komponenten des Satelliten, z.B. Sensoren und Aktuatoren, anzusteuern. Diese Platine wird im weiteren Text in Anlehnung an [Eickhoff et al. 2010b] und [Eickhoff et al. 2011] I/O Board genannt. In der Mitte von Abbildung 2.4 sind das nominelle I/O Board und dessen redundante Ausführung als L-förmig dargestellte Elemente zu sehen. Das I/O Board bietet einen Teil der Funktionalität einer sogenannten Remote Interface Unit (RIU), wie sie häufig im kommerziellen Satellitenbau eingesetzt wird. Ein Ingenieurmodell[3] der Platine ist in Abbildung 3.3 dargestellt. Das I/O Board hat eine Vielzahl an RS-422, I²C und IBIS Schnittstellen für die Kommunikation mit den Komponenten. Um eine

[1] Auf flüchtigen Speicher geschriebene Informationen gehen nach einer Trennung von der Spannungsversorgung verloren. Auf nichtflüchtigem Speicher bleiben sie in einem solchen Fall erhalten.

[2] Die in IEEE Spezifikation 1149.1 beschriebene Joint Test Action Group (JTAG) hat sich seit 1990 zu einem Standard durchgesetzt und wird von vielen Elektronikherstellern berücksichtigt [Stollon 2011].

[3] Ingenieurmodell ist ein Begriff aus der Modellphilosophie. Weitere Details dazu sind in Anhang D ab Seite 120 zu finden.

bestimmte Komponente vom Bordrechnerkern aus zu kommandieren, wird von diesem aus ein nach dem Remote Memory Access Protocol (RMAP) formatiertes Paket über die SpaceWire Schnittstelle an das I/O Board gesendet. RMAP beinhaltet unter anderem eine Adresse, an die das Paket weitergeleitet wird. Je nach Adresse wird der Inhalt des RMAP Pakets auf einem dedizierten Speicherbereich des I/O Boards abgelegt, der einer bestimmten Komponente zugeordnet ist. Von dort aus leitet das I/O Board den Inhalt des RMAP Pakets an die Komponente weiter.

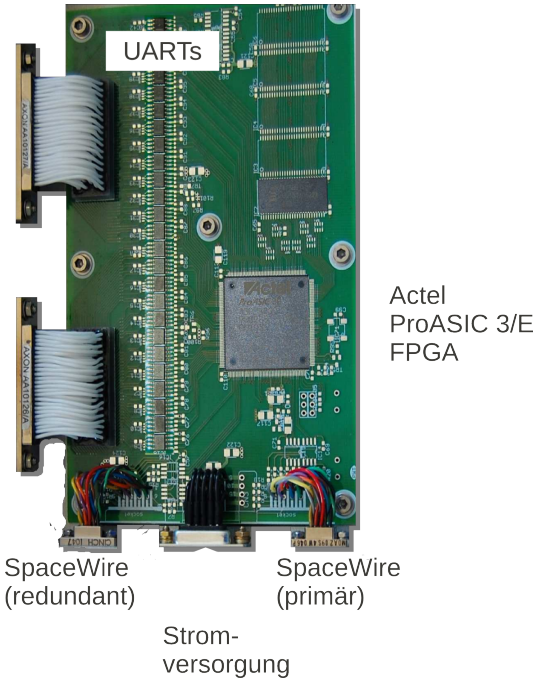


Abbildung 3.3: Ingenieurmodell des I/O Boards [Bucher 2011]

CCSDS Board

Um den Zweck und die Funktionsweise der zweiten Peripherieplatine besser zu verstehen, sind zunächst weitere Details hinsichtlich der Kommunikation im späteren Betrieb zu erläutern. Bei der Kommunikation mit Raumfahrzeugen wird wie in Abbildung 3.4 dargestellt zwischen der Kommandierungs- und Informationsrichtung mit den Begriffen Telekommando und Telemetrie unterschieden.

Bei beiden Übertragungsrichtungen werden die Daten dabei zunächst in sogenannte Protokolle verpackt, die je nach Typ diverse Merkmale wie beispielweise Prüfsummen beinhalten können. Anschließend werden diese Datenprotokolle für die Übertragung auf der Funkstrecke moduliert. Nach dem Senden und Empfangen durch Antennen kann das Signal wieder demoduliert und Daten aus den Protokollen entpackt werden.

Abbildung 3.4: Kommunikation mit einem Raumfahrzeug

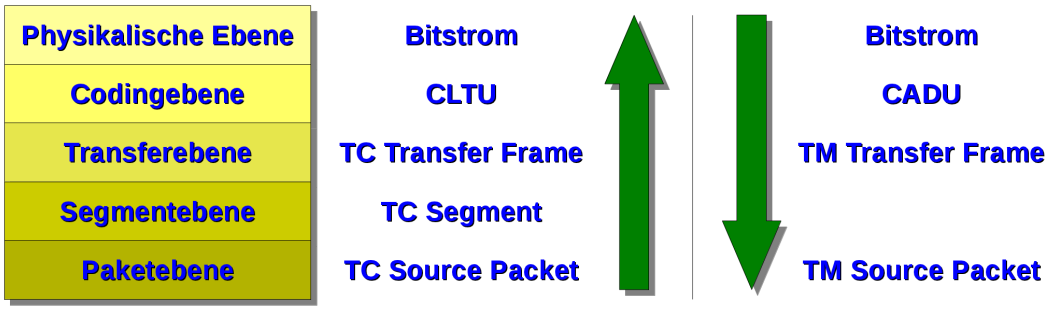


Abbildung 3.5: Ebenen nach CCSDS Standards

Zur Kommunikation mit dem Kleinsatelliten Flying Laptop wurde die Protokollierung der Telekommando- und Telemetriepakete nach CCSDS gewählt. Dieser Standard unterteilt Telekommandos und Telemetrie in mehrere Ebenen, und zwar wie in Abbildung 3.5 dargestellt in Paketebene, Segmentebene, Transferebene, Codingebene und physikalische Ebene. Diese Ebenen finden sich auf den ersten drei Ebenen des in Anhang A beschriebenen OSI-Schichtenmodells wieder. Auf Paketebene werden

am Boden wie in Abbildung 3.6 dargestellt sogenannte *Telecommand (TC) Source Packets* definiert, die einen bestimmten Befehl für den Satelliten enthalten. Diese Befehle werden auf Segmentebene zu einem *TC Segment* zusammengefasst. An Bord hingegen existiert wie in Abbildung 3.7 zu sehen für *Telemetry (TM) Source Packets* nur die Paketebene. Der nächste große Rahmen, der am Boden ein *TC Segment*, an Bord ein *TM Source Packet* enthält, wird *TC Transfer Frame* bzw. *TM Transfer Frame* genannt und findet auf Transferebene statt. Hier werden weitere Informationen gespeichert, beispielsweise bezüglich der Identifikation des Raumfahrzeugs. Bevor die Transfer Frames an die Funkstrecke weitergeleitet werden, werden sie in einen weiteren Rahmen auf der Codingebene verpackt, der sich für Telekommandos *Command Link Transmission Unit (CLTU)* und für Telemetrie *Channel Access Data Unit (CADU)* nennt. Mit dieser weiteren Einrahmung kann auf der physikalischen Ebene erkannt werden, wann CLTU bzw. CADU beginnen. CADUs haben eine spezifizierte Größe, bei CLTUs wird aufgrund der variablen Länge auf diese Weise auch das Ende des Pakets markiert. Außerdem wird der Inhalt einer CLTU in 8 Byte großen Einheiten übertragen. Dabei handelt es sich um 7 Datenbytes und ein Kontrollbyte. Dadurch können Fehler sowohl erkannt wie auch automatisch korrigiert werden[4]. CADUs werden am Ende des Pakets mit 160 Kontrollbytes versehen, die den gleichen Zweck haben. Allerdings wird hier eine andere Art der Vorwärtsfehlerkorrektur verwendet. Abbildung 3.5 zeigt anschaulich, auf welchen Ebenen welche Protokolle definiert sind. Weitere Details zu den beschriebenen Standards sind in [CCSDS 2003a, CCSDS 2003b, CCSDS 2006b, CCSDS 2006a und CCSDS 2010] zu finden.

Die zweite Peripherieplatine dient dazu, Telekommandos und Telemetrie zwischen verschiedenen Ebenen des CCSDS Standards [CCSDS 2006a] zu konvertieren. Diese Platine wird im weiteren Text, ebenfalls in Anlehnung an [Eickhoff et al. 2010b] und [Eickhoff et al. 2011] CCSDS Board genannt und ist in Abbildung 2.4 in der oberen Mitte der Darstellung rechts zu sehen. Beim Ingenieurmodell und Flugmodell handelt es sich um die gleiche Platine wie die des I/O Board. Als Entwicklungsmodell wird die in Abbildung 3.8 dargestellte Platine verwendet. Von den vielen Hardwareschnittstellen, die das spätere Flugmodell der Platine bietet, werden zwei RS-422 Schnittstellen zur Kommunikation mit den beiden Transmittern und zwei weitere zur Kommunikation mit den Receivern genutzt. Außerdem werden zwei RS-422 Schnittstellen mit der Einheit zur Leistungsregelung und -verteilung (PCDU)

[4]Diese Art der Vorwärtsfehlerkorrektur wird nach dessen Erfindern Bose, Chaudhuri und Hocquenghem auch BCH-Code genannt [CCSDS 2006a].

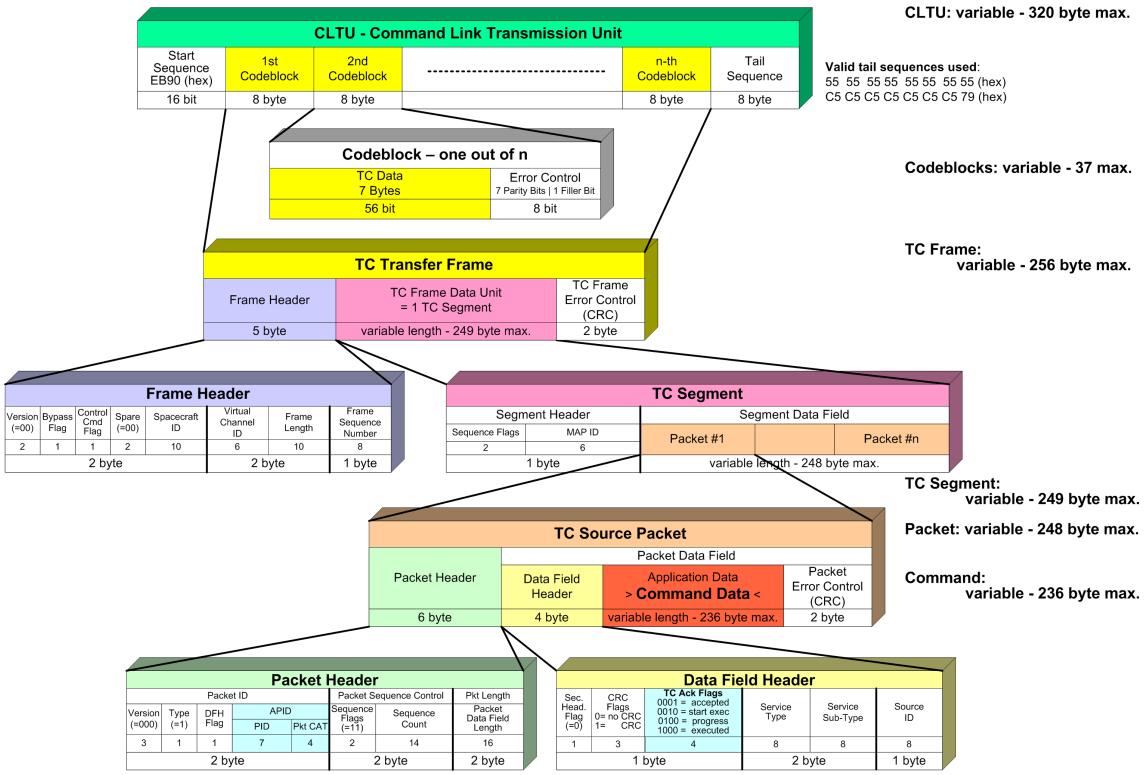


Abbildung 3.6: Aufbau von Telekommandodaten nach CCSDS [Eickhoff 2011]

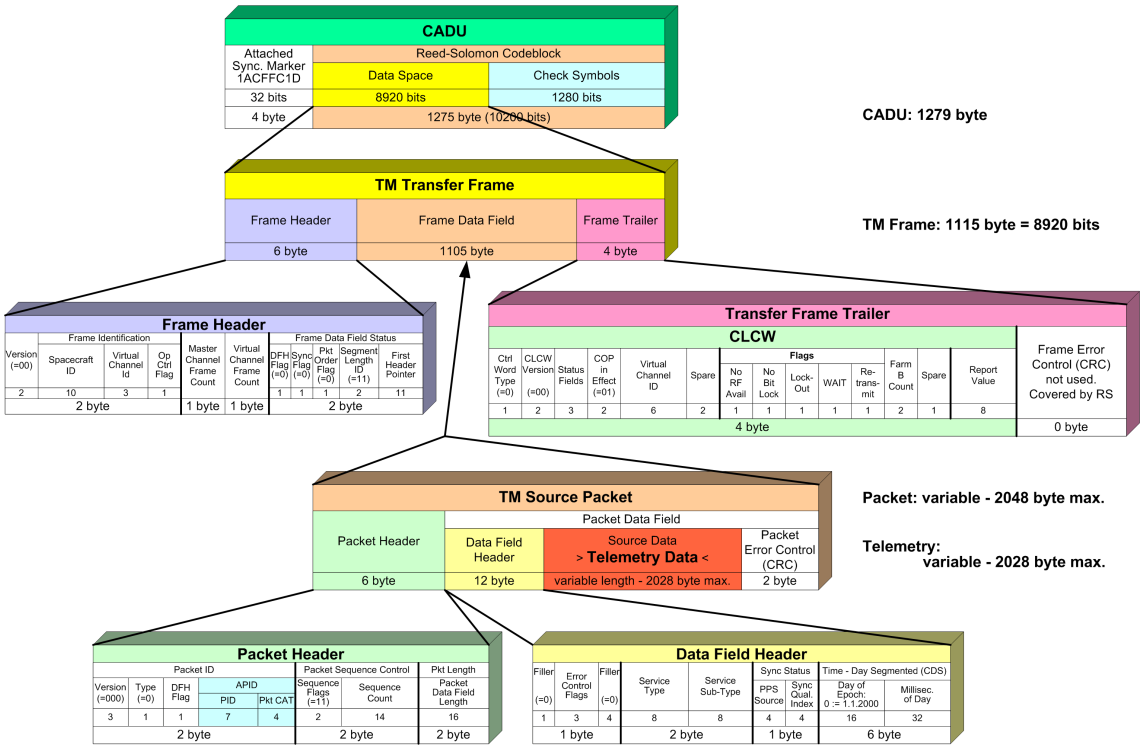


Abbildung 3.7: Aufbau von Telemetriedaten nach CCSDS [Eickhoff 2011]

verbunden. Die anderen Schnittstellen bleiben ungenutzt.

Die beiden CCSDS Boards können vom Boden gezielt angesprochen werden. Dies geschieht durch Verwendung hierarchisch angeordneter Kanäle, welche in Abbildung 3.9 schematisch dargestellt sind. Der physikalische Kanal entspricht dabei den von

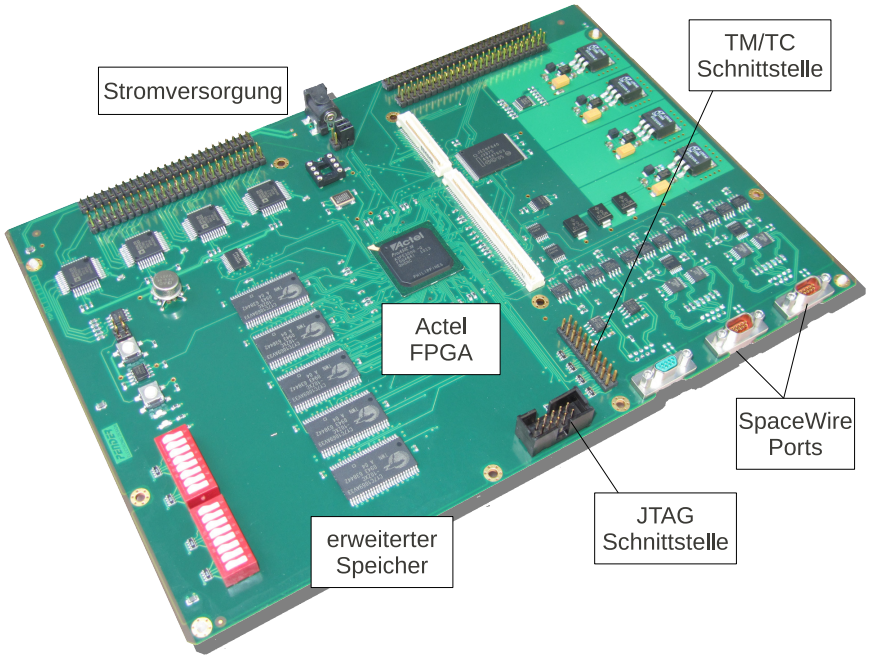


Abbildung 3.8: Entwicklungsmodell des CCSDS Boards [Bucher 2011]

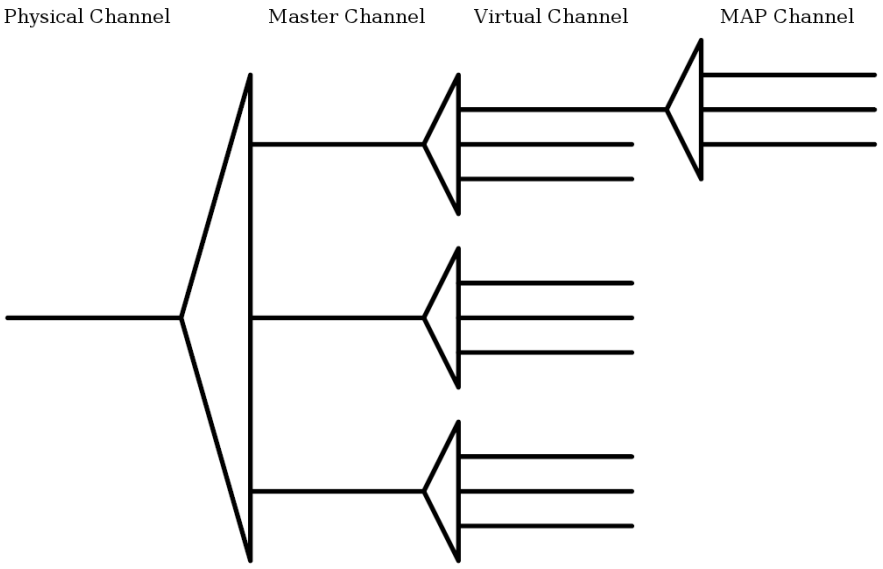


Abbildung 3.9: Kanäle nach CCSDS [CCSDS 2010]

einem CCSDS Board empfangenen validen Daten. Danach folgt der Hauptkanal, der sich nach der zu einem TC Transfer Frame gehörenden Information *Spacecraft Identification* richtet. Diese Information kann bereits zum dedizierten Weiterleiten des Telekommandos genutzt werden. Meist ist pro Raumfahrzeug nur eine solche Identifikationsnummer vorhanden. Schließlich folgen die ebenfalls im TC Transfer Frame enthaltenen virtuellen Kanäle, an die sich wiederum ein Multiplexer mit je 64

Kanälen anschließt. Für Telekommandos existiert beim Flying Laptop nur ein physikalischer Kanal mit einem Hauptkanal. Eine Unterscheidung der Pakete am CCSDS Board ist also erst mit den virtuellen Kanälen möglich. Hier sind für Telekommandos insgesamt vier Kanäle spezifiziert, pro Platine zwei. Durch eine Kreuzkopplung zwischen Demodulatoren und CCSDS Boards empfangen im Normalbetrieb beide Platinen dasselbe Telekommando, allerdings nimmt aufgrund des im Telekommando enthaltenen virtuellen Kanals nur eins von beiden die Daten an. Je nach Identifikationsnummer des virtuellen Kanals wird das Paket dann über die RS-422 Schnittstelle an die PCDU weitergeleitet oder auf einem internen Speicher des CCSDS Boards abgelegt. Die Verwendung von Multiplexern ist im Flying Laptop nicht vorgesehen.

Wurden die empfangenen Daten auf dem internen Speicher des CCSDS Boards abgelegt, kann sich der Bordrechnerkern von dort die entsprechenden Telekommandos abholen. Telemetrie kann vom Bordrechner auf dem Speicher des CCSDS Boards abgelegt und der Sendevorgang eingeleitet werden. Dabei wird die Telemetrie vom CCSDS Board von der Paketebene in die Codingebene konvertiert.

Das CCSDS Board wurde im Rahmen dieser Arbeit mit einem Überbrückungskabel an das bodenseitige Missionskontrollsystem angeschlossen. So wurde die Kompatibilität zur Bodenstation überprüft. Da Sender und Empfänger noch nicht verfügbar waren, kamen sie nicht zum Einsatz. Die Funkstrecke kann aber jederzeit integriert und getestet werden. Für den Test der Funkstrecke würde sich beispielweise ein Relaissatellit[5] eignen.

Mit dem CCSDS Board können außerdem dedizierte Daten direkt an die PCDU weitergeleitet werden. Mit einer solchen Weiterleitung kann die auf dem Bordrechnerkern laufende Bordsoftware umgangen werden. Auf diese Weise kann bei einem Fehler in der Bordsoftware, beispielsweise einer Endlosschleife, der Bordrechner durch Trennung und Wiederherstellung der Stromversorgung neu gebootet werden. Die dabei genutzten Telekommandos werden in Kapitel 4.5.3 genauer erläutert.

Stromversorgungsplatine

Der vierte und letzte Platinentyp im Bordrechner sind die Stromversorgungsplatinen. Diese dienen hauptsächlich dazu, die Elemente des Bordrechners mit der PCDU zu verbinden und die Stromversorgung sicherzustellen. Außerdem werden einige Signale des Bordrechners in dieser Platine elektrisch konvertiert, um kompatibel mit entsprechenden Bordkomponenten zu sein.

[5] Ein Relaissatellit kann ein empfangenes Signal weiterleiten, auch zurück an den Sender. Von einer Bodenstation aus und an diese zurück finden damit zwei Atmosphärendurchgänge statt, was das Signal beeinflusst und so reale Bedingungen herstellt.

Das Gesamtkonzept mit den oben beschriebenen Elementen ist in Abbildung 3.10 schmetatisch skizziert. Die im Normalbetrieb des Satelliten mit Leistung versorgten Teile sind rot hinterlegt. Mit diesem Konzept genügt es also, nur das CCSDS Board heiß redundant zu fahren.

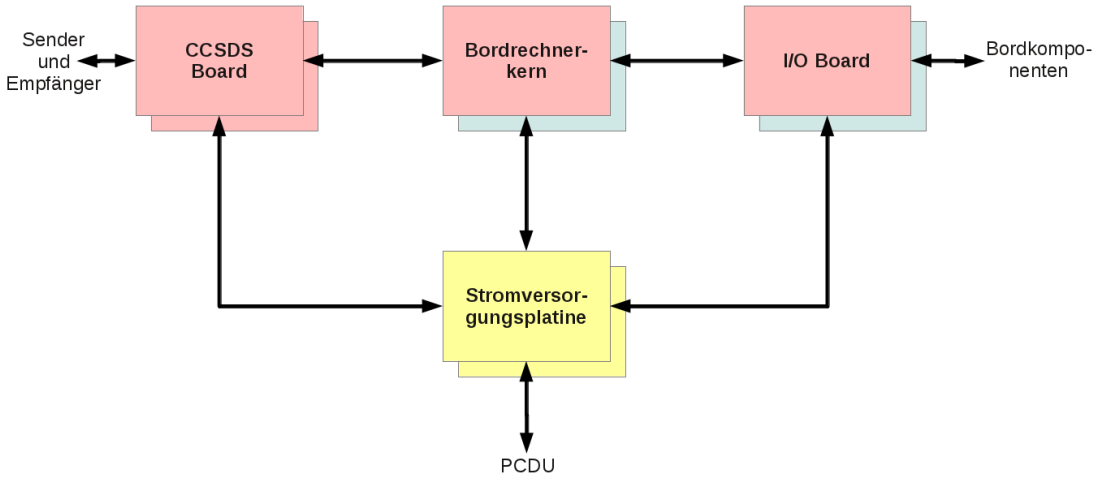


Abbildung 3.10: Gesamtkonzept des Bordrechners

3.2 Software

Die auf dem Bordrechner laufende Software besteht aus einem Betriebssystem und dafür programmierten Prozessen [Eickhoff et al. 2010b]. Zu den Aufgaben des Betriebssystems gehört [Baumgarten 2007]

- die Steuerung des zeitlichen Ablaufs der Prozessbearbeitung, im Englischen *Scheduling* genannt. Dafür kann der Nutzer dem Betriebssystem Vorgaben machen, indem er zum Beispiel Prioritäten für Prozesse setzt.
- Daran schließt sich die Aufteilung der Prozessorkapazität auf verschiedene Prozesse an, die ebenfalls vom Nutzer beeinflusst werden kann.
- Die sogenannte Interprozesskommunikation, die Kommunikation und Synchronisation von Prozessen untereinander wird auch vom Betriebssystem gehandhabt.
- Weitere wichtige Aufgaben sind das Laden von Prozessen in den Arbeitsspeicher sowie das Starten und Beenden von Prozessen.

- Die Koordination und Synchronisierung des Zugriffs auf gemeinsame Betriebsmittel wie Speicherbausteine und weitere Hardwarekomponenten werden vom Betriebssystem durchgeführt.

- Die Verwaltung von Daten inklusive der Handhabung von Zugriffen auf Daten durch die Vergabe von Rechten sind ebenfalls dem Betriebssystem zugewiesen.

Ein Betriebssystem wird daher auch als Software angesehen, die die Nutzung eines Rechners überhaupt ermöglicht [Baumgarten 2007]. Im Bordrechnerkern des Kleinsatelliten Flying Laptop kommt als Betriebssystem RTEMS[6] zum Einsatz, das ursprünglich nur militärischen Zwecken vorbehalten war. Mittlerweile wurde der Quelltext des Betriebssystems offen gelegt. RTEMS ist unter einer modifizierten Variante der GNU General Public License[7] verfügbar, und damit kostenfrei erhältlich. Des weiteren unterstützt es die Prozessorarchitektur des LEON3 sowie offene Standards für Programmierschnittstellen wie zum Beispiel POSIX[8] [Freund 2011]. Dies sind neben der Offenlegung des Quelltextes die Hauptgründe, warum es für die Verwendung auf dem Bordrechner des Flying Laptop ausgewählt wurde. Dieses Betriebssystem wurde als Echtzeitbetriebssystem entworfen. Im Gegensatz zu herkömmlichen Betriebssystemen können bei Echtzeitbetriebssystemen zeitliche Randbedingungen für Berechnungen sowohl festgelegt und durch Anwendung bestimmter Methoden auch eingehalten werden [Koolwal 2009]. Die Korrektheit der zeitlichen Ausführungen ist für das Auslesen von Sensoren, das Ausführen von Regelschleifen sowie das Ansteuern von Aktuatoren an Bord eines Satelliten essentiell. Würde beispielsweise ein Aktuator zu spät angesteuert, könnte die Funktionsfähigkeit des Lageregelungssystem eingeschränkt werden. Da die Entwicklung von RTEMS nicht am IRS durchgeführt wird und schon auf mehreren Raumfahrzeugen, z.B. auf dem Satelliten Proba-2[9] [Gantois et al. 2006] oder den interplanetaren Raumsonden Dawn und Venus Express [Kalibera et al. 2010] zum Einsatz kam, wird davon ausgegangen, dass die funktionale Korrektheit von RTEMS gewährleistet ist. Da RTEMS jedoch vorwiegend auf anderen Bordrechnern eingesetzt wurde, müssen verwendete Funktionalitäten auf dem Bordrechner des Flying Laptop verifiziert werden, um die korrekte Interaktion zwischen Hardware und Software sicherzustellen. Der Schwerpunkt der weiter unten beschriebenen Bordsoftwareverifikation liegt jedoch auf den

[6] Abkürzung für Real-Time Operating System for Multiprocessor Systems, zu Deutsch: Echtzeitbetriebssystem für Multiprozessorsysteme
[7] zu Deutsch: Allgemeine Öffentliche GNU-Lizenz
[8] Abkürzung für Portable Operating System Interface, zu Deutsch: Übertragbare Betriebssystemschnittstelle
[9] Abkürzung für Project for On-Board Autonomy, zu Deutsch: Projekt für Bordautonomie

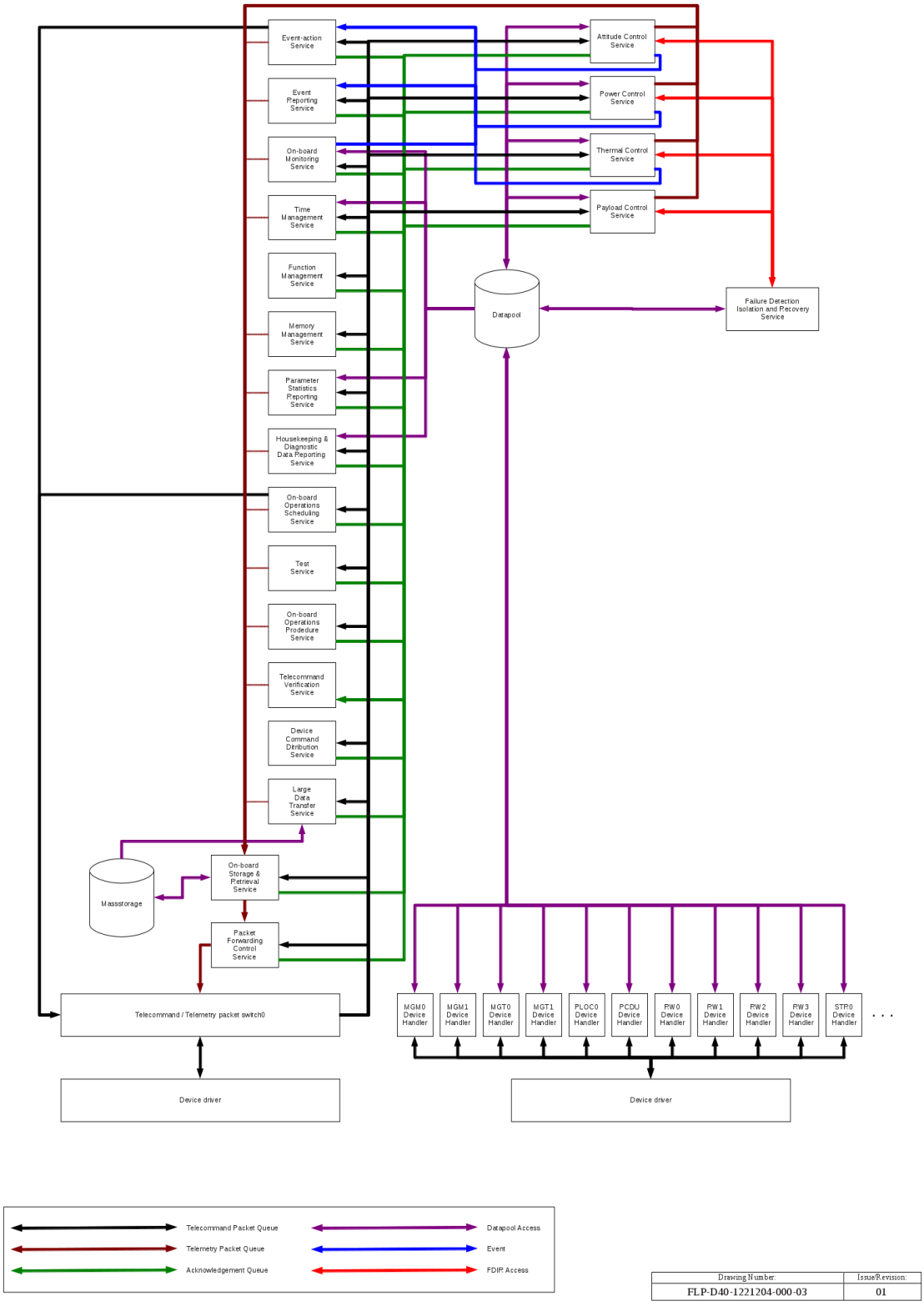


Abbildung 3.11: Architektur der Bordsoftware für den Kleinsatelliten Flying Laptop

am IRS entwickelten Prozessen.

Diese Prozesse beinhalten [Eickhoff 2011]

- Treiber zur Kommunikation über die Hardwareschnittstellen des Bordrechners,

- Steuerungssoftware zur Handhabung der Protokolle für die Kommunikation mit den Bordkomponenten des Satelliten,
- eine Struktur zur Handhabung des Datenbestands[10] des Satelliten,
- Kontrollalgorithmen für die Lageregelung, das Thermalkontrollsystem, die Nutzlast und das Energiekontrollsystem,
- die Enkodierung von Telemetrie sowie die Dekodierung von Telekommandos,
- Routinen zur Weiterverarbeitung der Telekommandopakete, sogenannte Service Handler,
- Algorithmen für die Fehlererkennung, deren Isolierung und die anschließende Wiederherstellung der Funktionalität,
- eine Überwachungsroutine für die Parameter im Datenbestand,
- Routinen zur Handhabung von Ereignissen, Kontrollprozeduren und Bordspeicher,
- Software zur Ansteuerung einer Wartungsschnittstelle des Bordrechners sowie
- einer Routine zur geordneten Ausführung aller genannten Aufgaben.

Abbildung 3.11 zeigt viele dieser Prozesse für den Kleinsatelliten Flying Laptop, die im weiteren Text Bordsoftware genannt werden. Da die Fehlfunktion eines dieser Elemente an Bord zum Komplettausfall des Satelliten führen kann, müssen alle Elemente der Bordsoftware und deren Zusammenspiel funktional verifiziert werden. Die dafür verwendete Umgebung, deren Aufbau und Funktionsweise werden im nachfolgenden Kapitel dargestellt.

[10] Im Fachjargon heißt dieser Bestand Datenpool.

4 Aufbau des Prüfstands

Zu den Kompatibilitätstests der Hardware und Software des Bordrechners gehören die

- Auswahl und Anpassung einer geeigneten Testumgebung zur Durchführung der Tests,
- die Integration aller zu testenden Komponenten und Geräte sowie notwendiger Software in diese Umgebung,
- die Planung der Tests,
- deren Durchführung mit einem möglichst hohen Automatisierungsgrad,
- die Auswertung und Bewertung der Ergebnisse
- sowie deren Dokumentation.

Die Wiederverwendbarkeit sowohl des entwickelten Prüfstands wie auch der ausgearbeiteten Methodik sollte für die Entwicklung zukünftiger weiterer Projekte möglichst hoch sein. Für die Realisierung der genannten Punkte war ein ingenieurwissenschaftlicher Arbeitsprozess, im nachfolgenden Text Engineering Process genannt, nötig, um diesen Anforderungen gerecht zu werden.

4.1 Begriffsdefinition

Zunächst werden einige Begriffe erläutert, die mehrfach im Text verwendet werden: Verifikation[1] ist der Nachweis, dass ein System so arbeitet, wie es spezifiziert wurde [Zave 1984]. Insbesondere im Bereich der Softwareentwicklung versteht man unter Verifikation den Nachweis der Spezifikation einer entsprechenden Entwicklungsphase [NASA 1996]. Funktionale Verifikation ist ein Begriff, der mit der Verbreitung des Rechners einhergeht. So befasst sich die funktionale Verifikation weder mit dem

[1] nach dem lateinischen Wort veritas, zu Deutsch: Wahrheit

Entwurf der Hardware noch mit deren physikalischen Eigenschaften, sondern ausschließlich damit, ob ein Rechner oder ein System die spezifierten Funktionen korrekt ausführt. Das ist genau dann der Fall, wenn zum einen die Software, die diese Funktionen beinhaltet, korrekt arbeitet und zum anderen die Software mit der Hardware kompatibel ist. Außerdem muss auch die Hardware fehlerfrei sein. Das bedeutet, dass die Software auf der echten Hardware zu verifizieren ist.

4.2 Problemstellungen

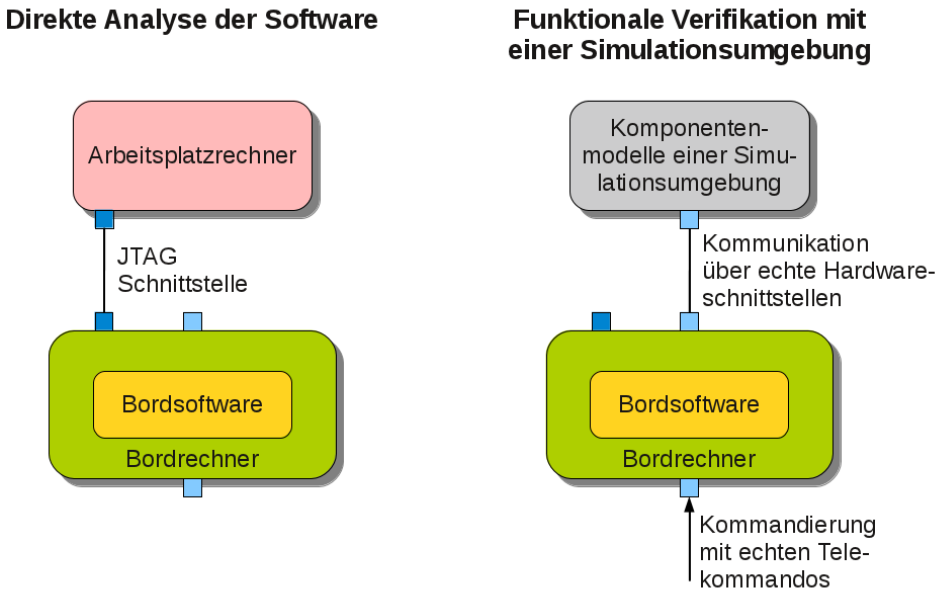


Abbildung 4.1: Verifikation der Bordsoftware auf dem Bordrechner

Wie in Abbildung 4.1 vereinfacht dargestellt, gibt es zwei Möglichkeiten um die Software auf der echten Hardware verifizieren. Einerseits kann mit Fehlersuchprogrammen über entsprechende Schnittstellen wie der JTAG Schnittstelle der Ablauf der Bordsoftware überwacht werden. Damit kann zwar die interne funktionale Korrektheit überprüft werden, nicht aber das Zusammenspiel mit den angeschlossenen Komponenten. Hierfür sind dann andererseits alle Hardwareschnittstellen des Rechners entsprechend zu stimulieren und auch auszulesen. Da aber beispielsweise die Aktuatoren an Bord die Lage des Satelliten verändern können und dadurch die Eingangswerte an den Sensoren des Satelliten entsprechend beeinflusst werden, müssen diese Querbeeinflussungen berücksichtigt werden. Da komplexe Aufbauten mit der echten Hardware die Kapazitäten und finanziellen Möglichkeiten eines Universitätsinstituts übersteigen würden, sind hierfür Simulationsmodelle nötig. Diese Modelle

müssen in Echtzeit das Zusammenspiel der Komponenten im Hinblick auf Lage, Temperatur, Leistung und Umgebung berücksichtigen. Außerdem müssen sie die Datenprotokolle nutzen, die auch später die echten Komponenten haben, sowie ein reales Antwortverhalten bieten. Des weiteren muss der Bordrechner mit echten Telekommandos kommandiert werden, die denen im Missionskontrollsystem entsprechen. Ebenso muss die Telemetrie des Bordrechners entsprechend vom Missionskontrollsystem empfangen und angezeigt werden können.

4.3 Methodik

Es ist also eine Simulation des Satellitensystems nötig, wie sie in [Eickhoff 2009] beschrieben wird. Damit sollte die auf dem Bordrechner laufende Bordsoftware in einer sogenannten Hardware in the Loop[2] Umgebung verifiziert werden können. Verifikationsläufe müssen exakt reproduzierbar sein, um Änderungen in der Bordsoftware bewerten zu können. Ein möglichst hoher Automatisierungsgrad trägt zur Erleichterung der Reproduzierbarkeit, aber auch zum vereinfachten Starten und Herunterfahren des Prüfstands bei. Da die Komplexität der Verifikationsumgebung groß ist, unterstützt ein standardisierter Aufbau die Handhabung. Nach der Durchführung eines Verifikationslaufs sollte ein Verifikationbericht automatisch erstellt und archiviert werden können. So können anschaulich dargestellte Ergebnisse jederzeit eingesehen werden.

4.4 Verifikationsumgebung

4.4.1 Architektur

Ein wichtiges Element der Verifikationsumgebung ist die Simulationsinfrastruktur. Da die Bordsoftware in Echtzeit läuft, muss auch der Simulator in Echtzeit simulierte Informationen an den Bordrechnerschnittstellen auslesen und liefern. [Falke 2009] beschreibt die dafür ausgewählte Simulationsumgebung und deren grundsätzlichen Aufbau. Die Simulationsumgebung beruht auf einer Technologie, die von der Astrium GmbH in Friedrichshafen entwickelt wurde [Eickhoff et al. 2006]. In vorangegangenen Veröffentlichungen wird diese Technologie auch immer wieder “Model-based Development and Verification Environment[3]” oder kurz MDVE genannt. Für Ker-

[2] Dieser englische Terminus wird auch im Deutschen verwendet, da es keine adäquate Übersetzung gibt. Gemeint ist die Integration realer Komponenten in eine Simulationsumgebung [Roddeck 2006].

[3] zu Deutsch etwa: Modellbasierte Entwicklungs- und Verifikationsumgebung

neltechnologie[4] und Systemmodelle wurde von Astrium der Systemsimulator ohne Komponentenmodelle als kompilierter Binärcode zur Verfügung gestellt. Am IRS wurden dafür Modelle der Komponenten des Kleinsatelliten Flying Laptop entwickelt. Abbildung 4.2 zeigt die prinzipielle Interaktion zwischen System- und Komponentenmodellen im Simulator. Diese Simulationsinfrastruktur wurde bereits vor dem Beginn dieser Arbeit im Rahmen einer anderen Dissertation [Falke 2009] am IRS aufgebaut. Da der Bordrechner bei der Fertigstellung jener Arbeit noch nicht konzipiert war, konnten die zu dessen Anbindung nötigen Komponentenmodelle im Simulator und Methoden noch nicht entwickelt werden. Im nachfolgenden Text werden nun die gegenüber dieser Erstinstallation notwendigen Schritte beschrieben.

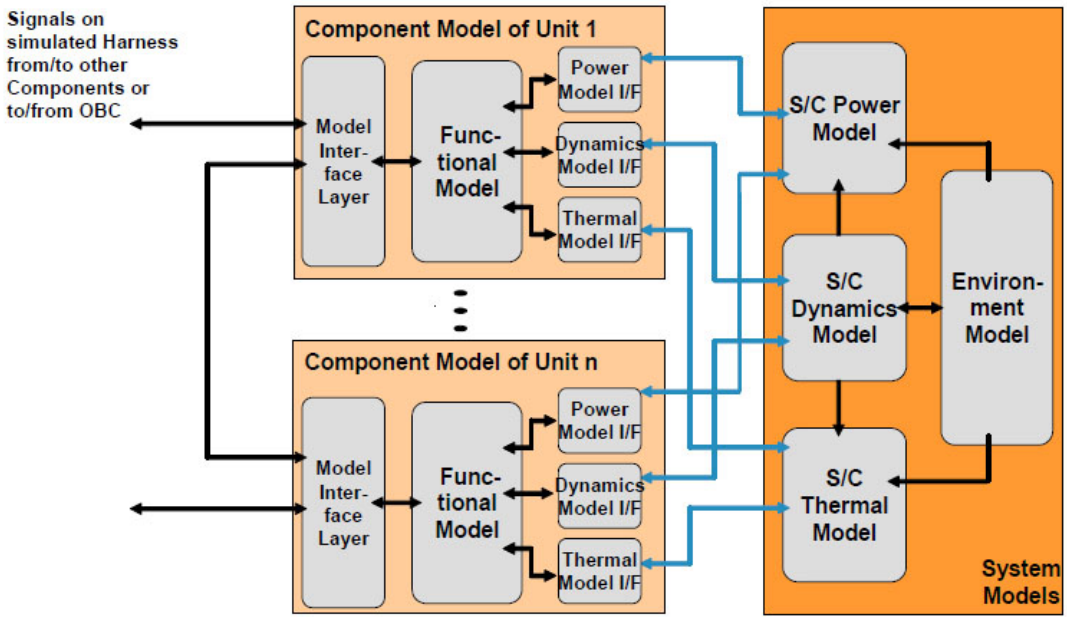


Abbildung 4.2: System- und Komponentenmodelle [Eickhoff 2009]

Eine weitere Simulationsumgebung, die sich dafür eignet Systemtopologien abzubilden, ist die in [Eickhoff et al. 2010a] erstmals vorgestellte Simulationssoftware OpenSimKit. Hier besteht aufgrund der Verwendung einer GNU General Public License voller Zugriff auf jeglichen Simulatorquelltext. Mit OpenSimKit lassen sich außerdem durch Anbindung des Prozessoremulators QEMU (Quick Emulator) und Verwendung der Modellierungssprache SystemC ganze Rechner simulieren. Auf den simulierten Rechnern kann dann Bordsoftware ebenfalls an eine Simulationsumgebung angeschlossen werden [Ziemke et al. 2011]. Zur Erstellung einer echtzeitfähigen Hardware in the Loop Umgebung ist OpenSimKit jedoch aufgrund der Verwendung

[4]Ein Kernel ist der elementare Bestandteil eines Betriebssystems oder einer Anwendung

der Programmiersprache Java und der dazugehörigen virtuellen Maschine[5] eher ungeeignet. Die Virtualisierung schränkt den direkten Zugriff auf Hardware und damit die Echtzeitfähigkeit des Simulators ein.

Abbildung 4.3: Verifikationsumgebung aus [Falke 2009]

Abbildung 4.3 zeigt den Stand dieser Verifikationsumgebung zu Beginn dieser Dissertation. Der Simulator konnte bereits vom Missionskontrollsystem SCOS-2000 aus angesteuert werden, auf welches später nochmal eingegangen wird. Die kleinen grauen Kästchen im Simulator stellen symbolisch die oben erwähnten System- und Komponentenmodelle dar. Nicht abgebildet ist ein an den Simulator angeschlossenes FPGA[6]-Board, da es nicht weiter verwendet wurde.

4.4.2 Erweiterung der Architektur

Modellierung des I/O Boards

Der in [Falke 2009] beschriebene und im Rahmen dieser Arbeit für die Tests angepasste Simulator ruft System- und Komponentenmodelle zyklisch auf, wobei die Simulationszeitschrittweite für jedes Modell vom Anwender festgelegt werden kann. Sie sollte aber nicht zu hoch gewählt sein, damit sich im Simulator keine Verzögerungen einstellen. Die höchste dieser Zeitschrittweiten beträgt 100 Hz. Im Umkehrschluss bedeutet dies aber, dass zwischen den Aufrufen eines Modells durch den Simulatorkernel mindestens 10 ms vergehen. Damit ist ein solches Modell zur Kommunikation mit verbundener Hardware ungeeignet, da das reale Antwortverhalten nicht gewährleistet wird. Verzögerungen von bis zu 10 ms entsprechen nicht den in der Bordsoftware zur Verfügung stehenden Zeitfenstern, womit Antworten des Simulators am Bordrechnerkern zu spät ankommen. Außerdem müssen dem Bordrechnerkern die Daten als SpaceWire-Signal zur Verfügung gestellt werden. Für die Modellierung des

[5] Eine virtuelle Maschine ist ein virtueller Computer, also das Modell eines Computers in Software.
[6] Field Programmable Gate Array. Dieser englische Terminus wird auch in der deutschen Sprache so verwendet. Übersetzt heißt er etwa feldprogrammierbare Gatterlogik. Gemeint ist eine programmierbare Hardwareschaltung.

I/O Boards war also eine spezielle Architektur zu entwickeln, die einerseits die Ausführung als Modell und andererseits ein reales Antwortverhalten ermöglicht. Dafür wurde in [Pflüger 2010] ein Kommunikationprinzip entwickelt, welches in Abbildung 4.4 abgebildet ist.

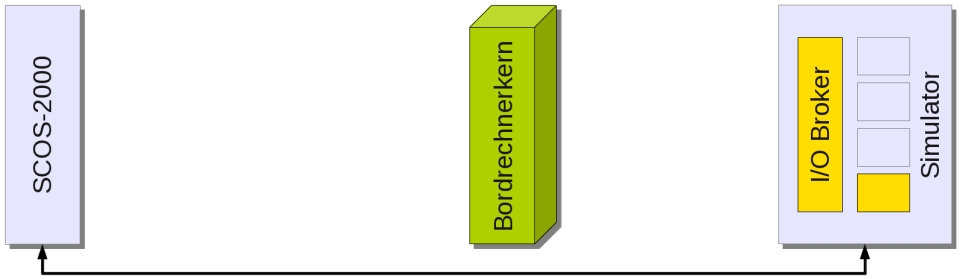


Abbildung 4.4: Verifikationsumgebung mit Bordrechnerkern

Auf der rechten Seite der Abbildung ist der Simulationsrechner abgebildet. Im Simulator wurde ein Komponentenmodell namens I/O Board Frontend implementiert, das durch den simulierten Harness mit anderen Komponenten verbunden ist und damit die Verbindung des I/O Boards zu Bordkomponenten modelliert. Über den simulierten Kabelbaum kann es mit den angeschlossenen Komponenten Daten austauschen. Im Bild ist dies das kleinere der beiden gelben Kästchen. Parallel dazu wurde ein Mittlerprogramm, der sogenannte I/O Broker entwickelt. Durch Kommunikation mit dem I/O Board Frontend über einen gemeinsam genutzten Speicher stehen dem I/O Broker ständig die neuesten Werte der Komponenten zur Verfügung. Der I/O Broker versteht das zur Kommunikation zwischen Bordrechnerkern und I/O Board genutzte Remote Memory Access Protocol und kann entsprechend Daten aus diesem Protokoll entpacken und Daten in das Protokoll verpacken. Dabei werden die Adressen der I/O Board Hardware simuliert. Da es ein vom Simulator unabhängig laufendes Programm ist, kann es Anfragen auf einer Netzwerkschnittstelle sofort beantworten. Dabei kommen TCP/IP[7] Protokolle zum Einsatz. Das bedeutet, dass RMAP zusätzlich in ein solches Protokoll gepackt wird. I/O Broker und I/O Board Frontend simulieren zusammen also das Verhalten des echten I/O Boards, mit der Ausnahme, dass sie keine SpaceWire Schnittstelle ansprechen. Stattdessen wurde die Netzwerkkarte des Simulationsrechners mit dem in Abbildung 4.5 zu sehenden Ethernet SpaceWire Router verbunden. Dort wird das TCP/IP Protokoll entpackt und das darin enthaltene RMAP Paket auf eine SpaceWire Schnittstelle weitergeleitet, mit der wiederum der Bordrechner verbunden wurde. Dies ist in Abbildung 4.6 dargestellt.

[7]Transmission Control Protocol/Internet Protocol, eine Familie von Netzwerkschnittstellen

Abbildung 4.5: Ethernet SpaceWire Router

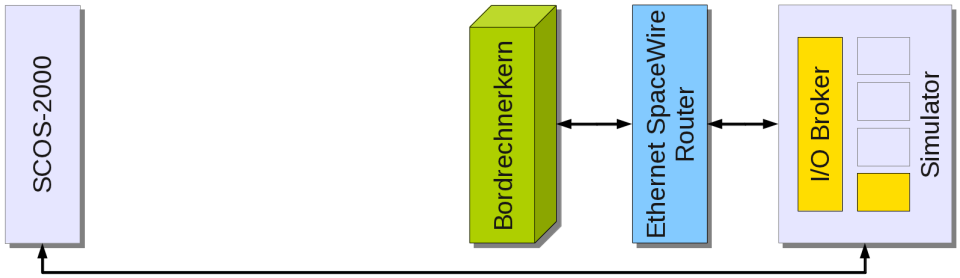


Abbildung 4.6: Anbindung des Simulators an den Bordrechnerkern

Echtzeitfähigkeit des Simulators

Der Simulatorrechner hat Antworten auf Anfragen des Bordrechners immer in einem vorgegebenen Zeitfenster zu liefern, das aufgrund der Architektur der Bordsoftware deutlich kleiner als 1 ms Sekunde ist. Aus diesem Grund kann der Simulator nicht auf herkömmlichen Betriebssystemen wie Windows XP, Windows 7 oder Linux laufen. All diese Betriebssysteme sind nicht deterministisch. Das bedeutet, dass deren Verhalten im Hinblick auf zeitliche Vorgaben nicht voraussagbar ist. Durch sogenannte System Interrupts können laufende Prozesse des Betriebssystems ausgebremst werden. Außerdem können Prozesse mit höchster Priorität nicht sofort in vollem Umfang den Prozessor auslasten. Daher kann nicht sichergestellt werden, dass der Simulator Antworten im gewünschten Zeitfenster liefert. Zur Umgehung dieses Problems wurde ein Echtzeitbetriebssystem benötigt. Bei der industriellen Verwendung des Simulators kommen proprietäre Echtzeitbetriebssysteme wie VxWorks, aber auch Linux Derivate wie RedHawk zum Einsatz. Beide haben gemeinsam, dass sie nicht frei verfügbar sind. In der industriellen Anwendung werden sie jedoch benötigt, da in einer industriellen Simulationsumgebung Bordrechnerelemente wie das I/O Board des Flying Laptop selten simuliert werden. Die simulierten Komponenten müssen über eine Vielzahl von Hardwareschnittstellen an den Bordrechner angeschlossen werden können, damit sie Stück für Stück durch echte Hardware ersetzt werden können. Am Simulationsrechner kommen dabei typischerweise VME-Karten[8] zum

[8] Versa Module Europa (VME) ist ein Bussystem, das rechenintensive Aufgaben des Prozessors übernehmen kann. Es wurde erstmals 1981 eingesetzt.

Einsatz, die die Verwendung der genannten Echtzeitbetriebssysteme nötig machen. Ein mit VME-Karten bestücktes Chassis ist in Abbildung 4.7 dargestellt.

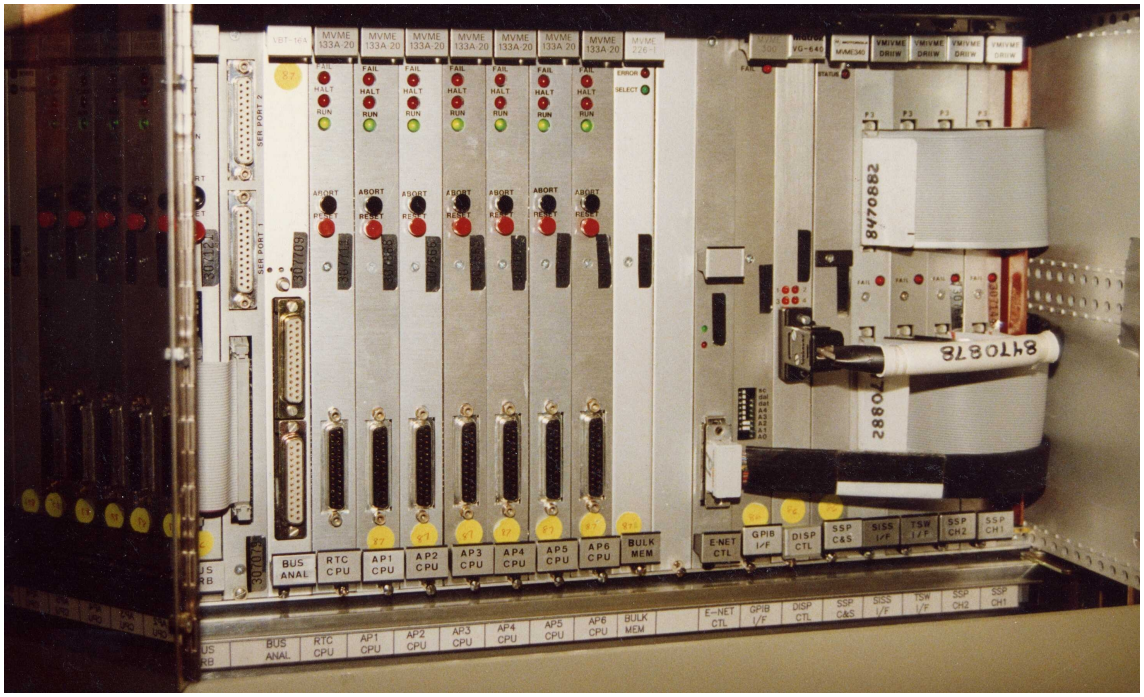


Abbildung 4.7: VME-Karten in einem Chassis ©JEKLsoft

Wie oben beschrieben wird in der Simulationsumgebung des Flying Laptop ein I/O Board Modell eingesetzt, da bei der Entwicklung des Satelliten wie in Abbildung 4.8 dargestellt auf den sukzessiven Austausch von simulierten Modellen und Hardware verzichtet wird. Daher wurden solche Karten nicht benötigt, was die Flexibilität bei der Verifikation einschränkt, aber die Kosten niedrig hielt und zugleich die Auswahl an Echtzeitbetriebssystemen vergrößerte. Wie in [Freund 2011] beschrieben, wurde für die Simulationsumgebung des Flying Laptop ein Patch[9] für den Linux Betriebssystemkern gewählt. Das hat den Vorteil, dass das bislang verwendete Linux Betriebssystem auf den Simulatorrechnern weiterverwendet werden konnte. Durch den Patch wurde es echtzeitfähig gemacht. Damit ist darauf laufende Software, z.B. der Simulator, nicht nur resistent gegen Interrupts. Prozesse höherer Priorität können bei Aktivierung auch umgehend Prozessorauslastungsanteile von Prozessen niedriger Priorität übernehmen. Prinzipiell eigneten sich die Patches Xenomai, RTAI[10] und RT-Preempt[11] für diesen Zweck. Xenomai hat einen eigenen Kernel, der den Linux Betriebssystemkern echtzeitfähig macht. Außerdem ermöglicht es durch Bereitstellung einer Vielzahl an Programmierschnittstellen die Portierbarkeit

[9] Ein Patch korrigiert oder erweitert eine bestehende Software. Patch ist das englische Wort für flicken oder ausbessern.
[10] Real-Time Application Interface, zu Deutsch: Schnittstelle für Echtzeitanwendungen
[11] von Real-Time Preemption, zu Deutsch etwa: Zuvorkommen durch Echtzeit

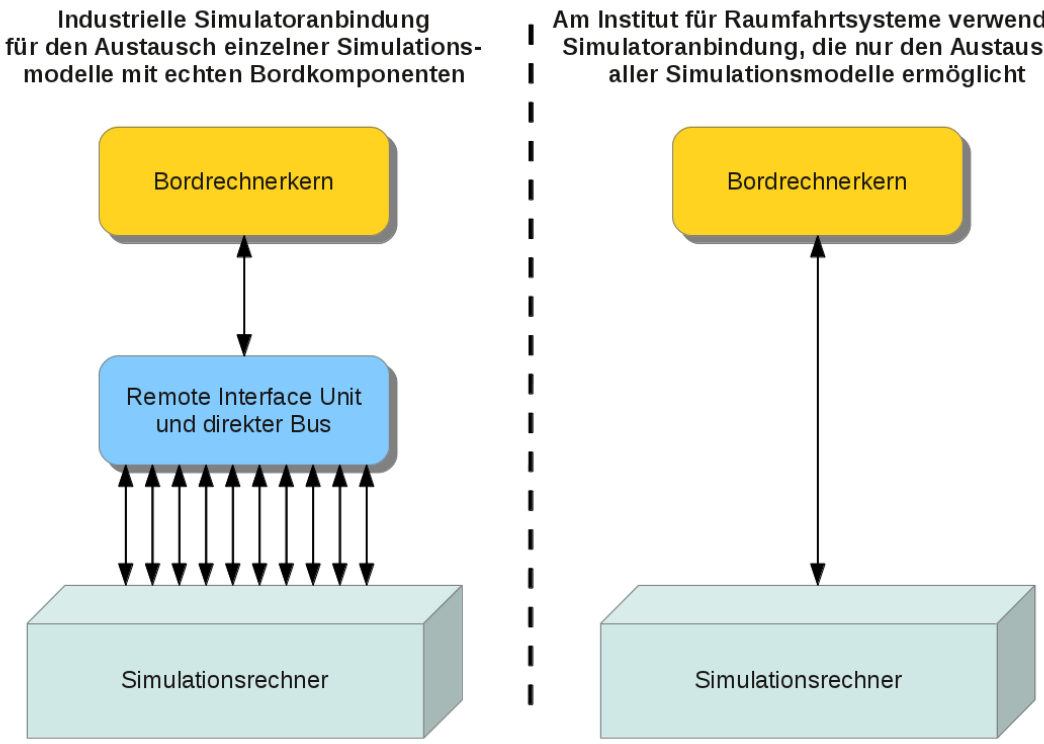


Abbildung 4.8: Unterschiedlich ausgeführte Simulationsumgebungen

von Anwendungen, die für andere Betriebssysteme, z.B. VxWorks entworfen wurden. Bei RTAI wird ebenfalls ein eigener Kernel zwischen Linux Betriebssystemkern und Hardware implementiert, der Interrupts des Linux Betriebssystemkern abfangen kann. Das verhindert aber den Zugriff auf serielle Schnittstellen, Netzwerkkarten und andere Geräte vom Linux Betriebssystem aus. Dafür müssen Treiber neu geschrieben werden. Um diese nutzen zu können, muss entsprechend der Quelltext von Anwendungen angepasst werden. RT-Preempt hat keinen eigenen Kernel, sondern erweitert und verändert den Linux Betriebssystemkern. Dadurch können Programme direkt echtzeitfähig gemacht werden, ihr Quelltext muss nicht länger angepasst werden. Das verkürzt wiederum die Entwicklungszeit dedizierter Anwendungen. Es können jedoch nicht alle System-Interrupts mit RT-Preempt abgefangen werden [Freund 2011]. Der wichtigste Unterschied zwischen den drei beschriebenen Architekturen ist in Abbildung 4.9 zu sehen.

Da der Quelltext des Simulatorkernels und der entsprechenden Systemmodelle nicht vorliegt, sondern nur die entsprechende Binärdatei, können diese Elemente des Simulators weder eingesehen noch verändert werden. Dadurch scheideten Xenomai und RTAI aus. Um RT-Preempt sinnvoll verwenden zu können, wurde der Linux Betriebssystemkern, wie in [Freund 2011] beschrieben, zunächst durch einen Patch verändert, anschließend von Hand zur Optimierung modifiziert und schließlich neu

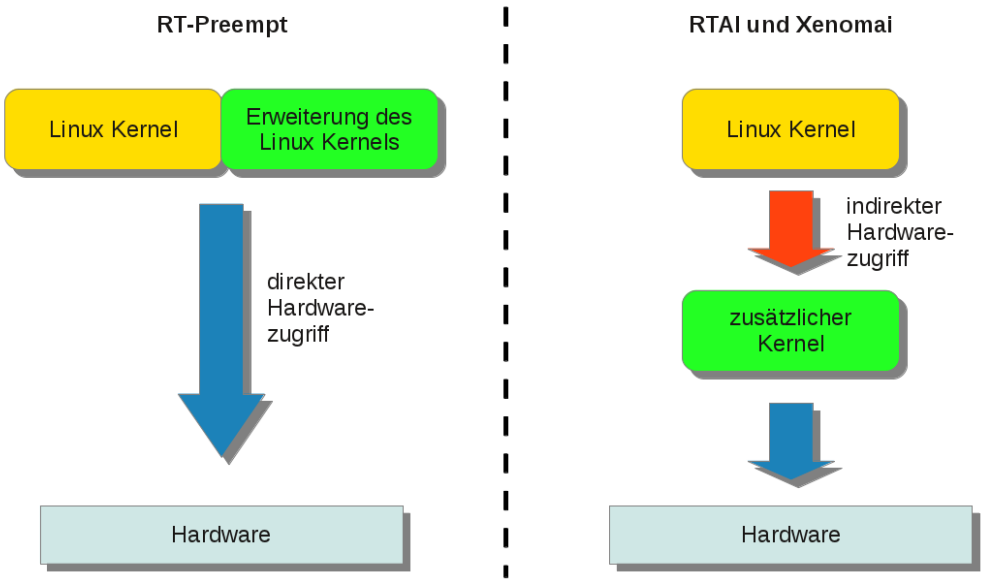


Abbildung 4.9: Prinzipieller Unterschied verschiedener Echtzeit-Linux Architekturen

kompiliert. Um den Simulator vor System-Interrupts zu schützen, wurde ein Rechner mit vier Prozessorkernen gewählt. Sowohl dem Simulator als auch dem unabhängig laufenden I/O Broker wurde je ein Prozessorkern zugewiesen. Alle weiteren Prozesse inklusive die des Betriebssystems sind auf die beiden anderen Prozessoren verteilt. Die dem Simulator und dem I/O Broker zugewiesenen Prozessoren wurden außerdem gegen Interrupts geschirmt. Abbildung 4.10 veranschaulicht das gewählte Prinzip. Damit konnten die durch das Betriebssystem verursachten maximalen Latenzen[12] auf unter 10 µs verringert werden.

Die vom Betriebssystem verursachten Latenzen waren um die vom Ethernet SpaceWire Router verursachten Latenzen zu ergänzen, um die Gesamtlatenz der Simulationsinfrastruktur zu berechnen. Dazu wurde der Router mit dem Simulationsrechner verbunden. Die Gesamtlatenz wurde mit einem einfachen Prinzip gemessen. An den Router wurde eine Schleifenschaltung angeschlossen, welche ausgehende mit eingehenden Pins verbindet. Mit diesem Aufbau wurde ein Paket vom Simulationsrechner an den Router geschickt. Dieses Paket war sehr kurz, um die Übertragunsdauer an sich so gering wie möglich zu halten. Das Paket wurde vom Router empfangen und wieder an den Simulationsrechner gesendet. Damit wurde die gesamte Strecke zweimal durchlaufen. Die Zeitdauer zwischen dem Absenden des Pakets und dessen

[12] Als Latenz wird die Zeit zwischen dem Befehl zum Start eines Prozesses und dessen tatsächlichen Starts verstanden.

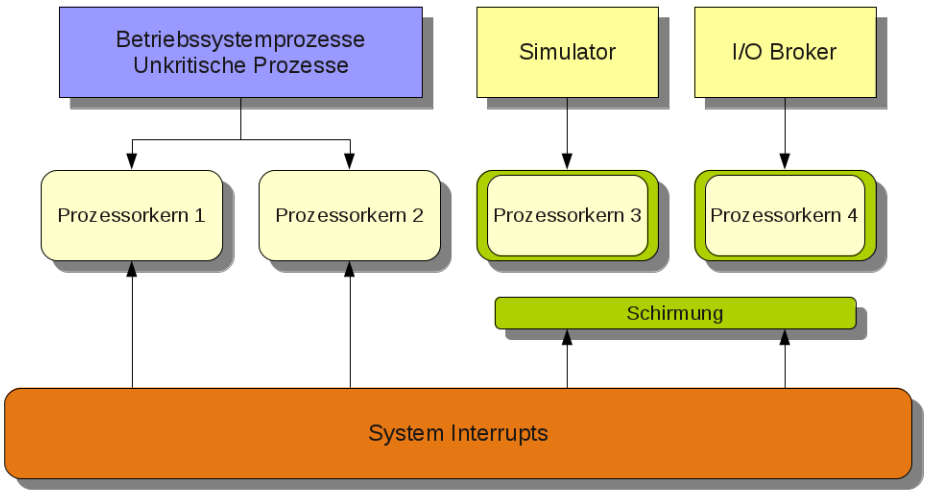


Abbildung 4.10: Simulationsrechnerarchitektur zur Reduzierung von Latenzen

Empfang wurde gemessen. Die Latenz der Strecke entsprach etwa der Hälfte dieser gesamten Zeitdauer. Über die gesamte Strecke wurden maximale Latenzen von 70 µs gemessen, was den Anforderungen des Flying Laptop genügt. Durch diesen Aufbau konnte der Bordrechner des Flying Laptop mit der Simulationsumgebung verbunden und getestet werden. Mit dem reinen Simulator fand jedoch weder ein komfortabler Simulatorbetrieb statt noch konnte der Bordrechner mit realen Bodenkommandos kommandiert werden. Daher wurde um diese Simulationsumgebung herum weitere Infrastruktur aufgebaut.

Kommandierung des Bordrechners

Um die reale Kommandierung des Bordrechners zu ermöglichen, wurde eine Infrastruktur ausgewählt und angepasst, wie sie später auch in der Bodenstation weiterverwendet werden kann. Dies schließt nicht das Antennensystem und dessen Ansteuerung ein, dafür aber die Software für das Missionskontrollsystem und ein Telemetrie und Telekommando Frontend zur Konvertierung der Telekommandos von der Paketebene in die Codingebene und entsprechend der Telemetrie von der Codingebene in die Paketebene. Zur Automatisierung des Missionskontrollsystems wurde eine Prozeduransteuerung ausgewählt, integriert und angebunden.

Als Software für das Missionskontrollsystem wurde in [Brandt 2007] die von der europäischen Weltraumorganisation ESA mitentwickelte und genutzte Softwareinfrastruktur SCOS-2000[13] ausgewählt und am IRS aufgesetzt. Dieser Schritt ist nicht

[13] Abkürzung für Spacecraft Control and Operation System, zu Deutsch: System zur Steuerung und für den Betrieb von Raumfahrzeugen

als Teil der vorliegenden Arbeit zu betrachten. SCOS-2000 bietet eine Standardschnittstelle zur Kommunikation zwischen verschiedenen Bodenstationen an [Brandt 2007] und wurde bereits im Rahmen vieler Missionen verwendet und entsprechend verbessert [Noguero et al. 2005]. Darüber hinaus bietet die Nutzung von SCOS-2000 den Vorteil, auf Missionskontrollinfrastrukturen der ESA und des deutschen Zentrums für Luft- und Raumfahrt (DLR) zurückgreifen zu können, falls dies aufgrund eigener Engpässe erforderlich sein sollte. SCOS-2000 bietet auch volle Unterstützung für den CCSDS Standard, der für die Kommunikation zwischen Bodenstation und Flying Laptop ausgewählt wurde. Ebenfalls unterstützt wird der Packet Utilization Standard PUS nach [ECSS 2003], der die Handhabung der Pakete in der Bordsoftware beschreibt. Abbildung 4.11 zeigt Bildschirmelemente von SCOS-2000.

Abbildung 4.11: Missionkontrollsystem SCOS-2000

Da die gewählte Software für das Missionskontrollsystem nicht nur zur Kommandierung des Bordrechners, sondern auch zur Steuerung und Überwachung der Simulationsinfrastruktur genutzt werden sollte, wurde in [Brandt 2007] eine spezielle Software entwickelt, die die von SCOS-2000 abgeschickten Pakete analysiert und diese entweder an den Simulationsrechner weiterleitet oder für ein Telemetrie und Telekommando Frontend vorhält. Es handelt sich also um ein Vermittlerprogramm und wird daher Proxy[14] genannt. Der Proxy läuft auf demselben Rechner wie SCOS-2000.

SCOS-2000 kann in der gewählten Version ausschließlich Telekommandos auf Paketebene abschicken. Daher wurde ein Telemetrie und Telekommando Frontend beschafft und, wie in [Bucher 2011] beschrieben, mit SCOS-2000 verbunden. Der Proxy unterscheidet zwischen Kommandos an den Simulator, Kommandos an den

[14]Proxy ist das englische Wort für Vertreter. Als Proxy wird in der Regel ein von zwei Kommunikationspartnern unabhängiges Gerät bzw. eine unabhängige Software bezeichnet, die die Kommunikation regelt und in dieser Hinsicht die Kommunikationspartner “vertritt”.

Bordrechner und Kommandos zur Steuerung des Frontends. Das Frontend nimmt die Konvertierung der Telekommandos von der Paketebene in die Codingebene vor. Dafür sind sowohl komplexe Algorithmen wie auch eine Vielzahl an Zusatzinformationen nötig. Da es Firmen gibt, die damit Erfahrung haben und bereits mehrere Missionen mit einem solchen Frontend ausgestattet haben, wurde entschieden, das Frontend einzukaufen. Dieses ist auch in der Lage, Telemetrie von der Codingebene in die Paketebene zu konvertieren. Das Frontend besteht aus einem normalen Arbeitsplatzrechner und einer Elektronikeinheit. Der Rechner kommuniziert durch seine Netzwerkschnittstelle mit dem Proxy und kann über ein gekreuztes Netzwerkkabel die Elektronikeinheit ansprechen. Diese wiederum kann mit dem Bordrechner kommunizieren und wird später zur Ansteuerung des Modulators am Boden genutzt. Die beiden Elemente sind in Abbildung 4.12 schematisch dargestellt. Die zum Frontend gehörende Software ist in Abbildung 4.13 zu sehen.

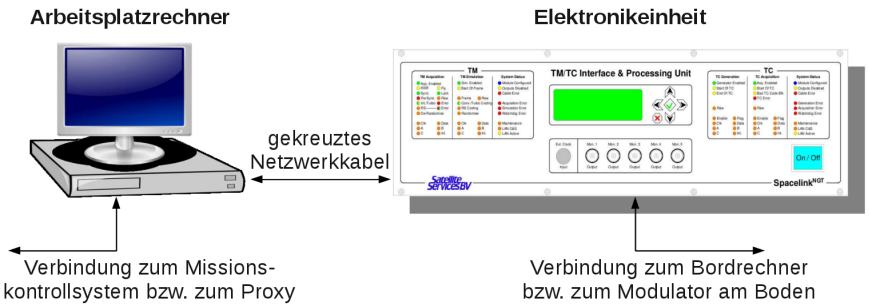


Abbildung 4.12: Telemetrie und Telekommando Frontend [Bucher 2011]

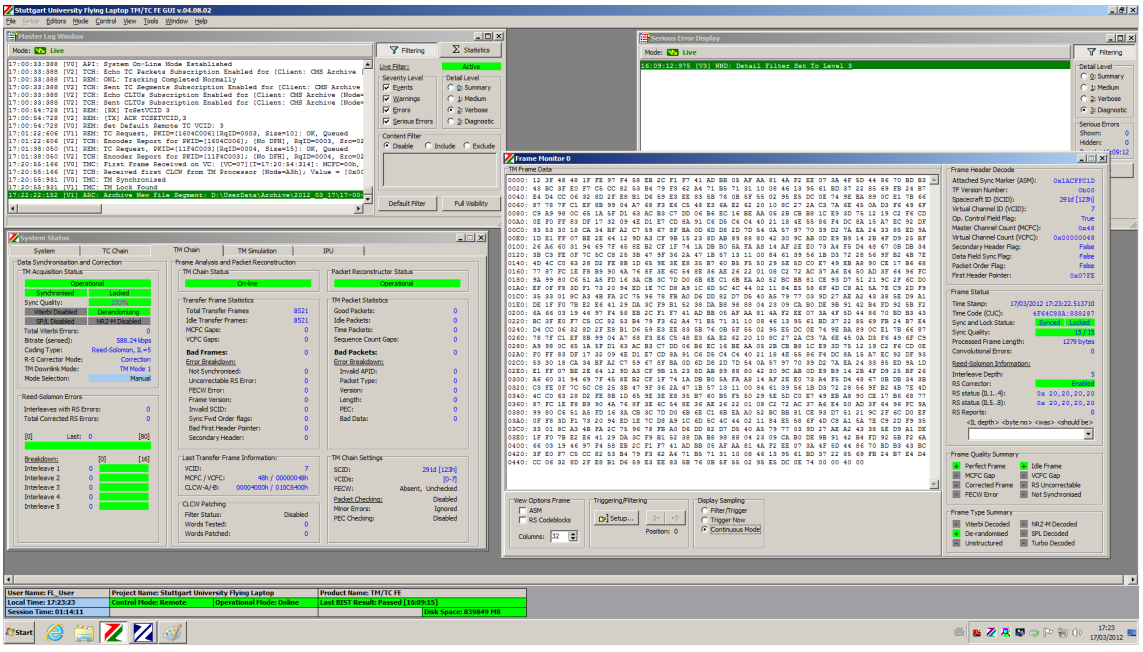


Abbildung 4.13: Auf dem Arbeitsplatzrechner laufende Software

Damit kann, wie in Abbildung 4.14 zu sehen, das CCSDS Board des Bordrechners nun auch mit dem Missionskontrollsystem verbunden werden.

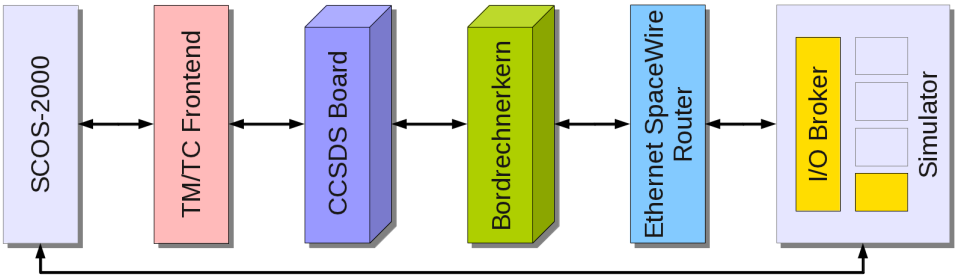


Abbildung 4.14: Reale Kommandierung des Bordrechners

Zuletzt wurde der Betrieb im Missionskontrollsystem durch den Anschluss einer Prozedursteuerung an SCOS-2000 automatisiert [Fritz et al. 2010a]. Eine solche Prozedursteuerung kann auf Telemetrie reagieren und erweitert damit die Funktionalität von SCOS-2000. Durch geeignete Prozeduren kann der Operator in der Bodenstation zeitweise ersetzt werden, was beispielsweise einen Einschichtbetrieb ermöglicht. Durch Kooperation mit der Firma Rhea wurde dem Institut das Softwarepaket MOIS[15] zur grafischen Bearbeitung und Ausführung solcher Prozeduren zur Verfügung gestellt. Darüber hinaus ermöglicht MOIS den Import der Telemetrie und Telekommando Datenbank aus SCOS-2000 und garantiert damit Datenkonsistenz. Außerdem können mit MOIS Prozeduren in verbreitete Skriptsprachen exportiert und als solche auch ausgeführt werden, was eine hohe Kompatibilität garantiert. Wie auch bei anderen Elementen der Verifikationsinfrastruktur wurde dieses Softwarepaket von Experten entwickelt und wurde laut Aussage des Herstellers bereits für 75 verschiedene Raumfahrzeuge eingesetzt. Auch MOIS kann im späteren Bodenstationsbetrieb verwendet werden. Die Wiederverwendbarkeit all dieser Elemente des Missionskontrollsystems in der späteren Bodenstation reduziert nicht nur den gesamten Entwicklungsaufwand, sondern stellt auch die Konsistenz zu den in dieser Arbeit dargestellten Tests sicher. Abbildung 4.15 zeigt die Ausführung einer mit MOIS erstellten Prozedur.

Damit wurde eine Verifikationsinfrastruktur aufgebaut, die den Anschluss des realen Bordrechners als Hardware in the Loop sowie weiterer Komponenten ermöglicht. Alle einzelnen Elemente der Verifikationsinfrastruktur sind in Abbildung 4.16 dargestellt. Bevor aber reale Komponenten eingebunden werden konnten, war die funktionale Korrektheit der Simulationsinfrastruktur zu verifizieren.

[15] Abkürzung für Manufacturing and Operations Information System, zu Deutsch: Informationssystem für Herstellung und Betrieb

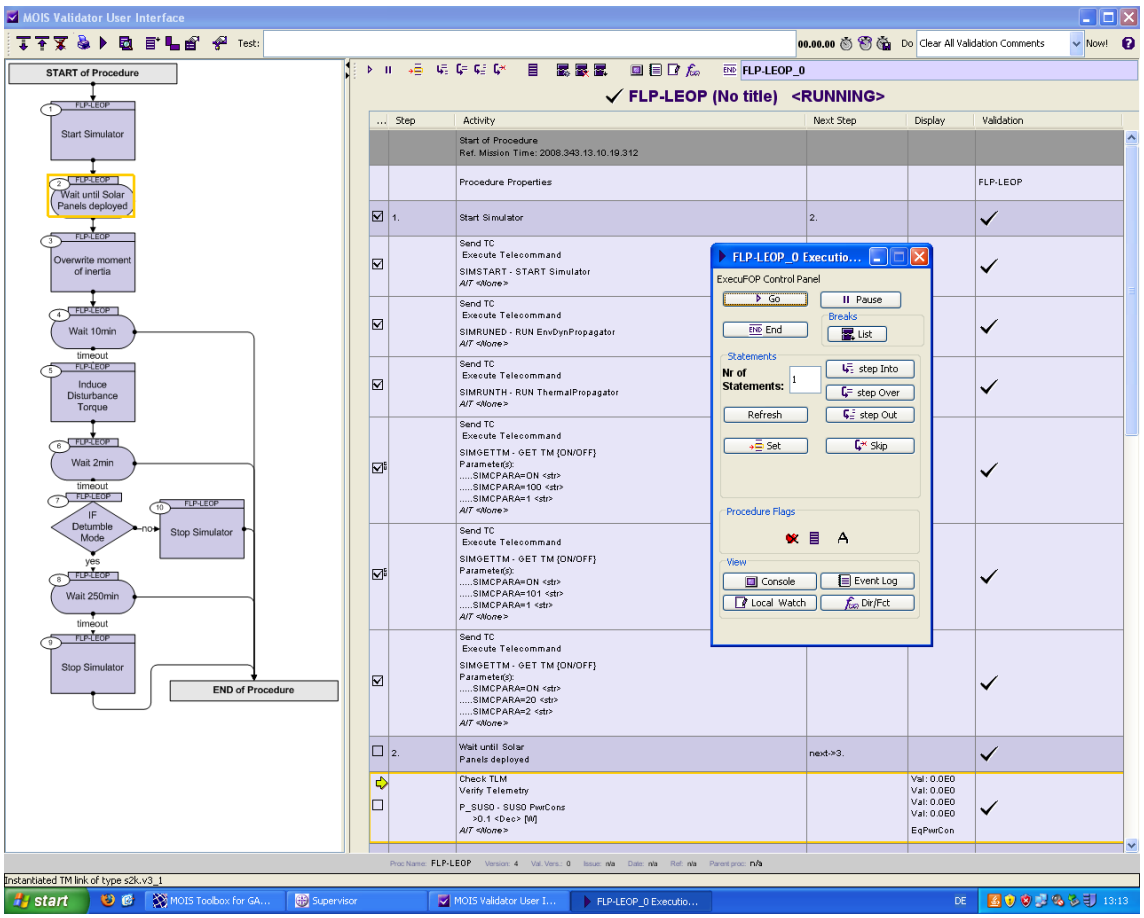


Abbildung 4.15: Prozedursteuerung MOIS [Fritz et al. 2010b]

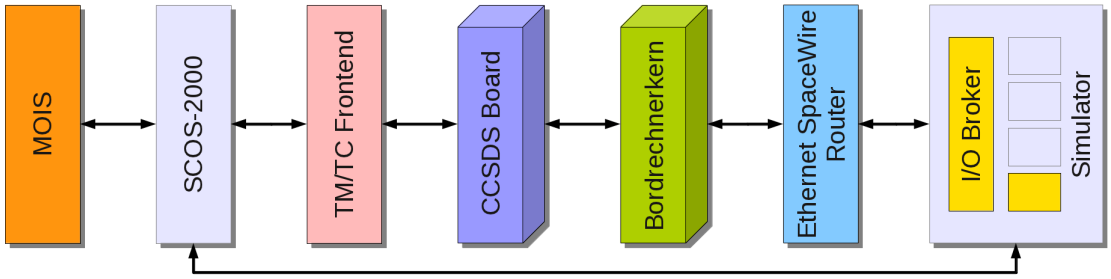


Abbildung 4.16: Vollständige Verifikationsumgebung

4.4.3 Funktionale Korrektheit des Simulators

Bevor die Simulationsumgebung für die Verifikation der Bordsoftware eingesetzt werden konnte, musste zusätzlich demonstriert werden, dass das simulierte Verhalten der Realität so nahe wie möglich kommt. Für die Überprüfung der funktionalen Korrektheit der Simulationsumgebung wurden mehrere Aspekte berücksichtigt. Zum einen wurden bereits in [Falke 2009] die Simulationsmodelle auf ihre innere Logik hin mit speziellen Tests untersucht. Nicht immer ist der Quelltext solcher Simulationsmodelle fehlerfrei. So werden semantische Fehler wie Vorzeichenfehler, falsche

mathematische Operatoren aber auch Fehler in der Reihenfolge einer Algorithmik beim sogenannten Kompilieren, dem Übersetzen von Quelltext in vom Rechner abhängigen Binärcode nicht als Fehler erkannt, da hier nur Syntaxfehler berücksichtigt werden. Da sie aber das Verhalten von Simulationsmodellen maßgeblich beeinflussen können, wurden erwartete Ergebnisse mit den vom Modell berechneten unter der Angabe von Toleranzen verglichen. Abweichungen wurden dabei dem Programmierer gemeldet.

Damit konnte aber nur die Logik und Korrektheit der Modelle an sich überprüft werden. Es ließen sich weder Probleme im Zusammenspiel der Komponenten noch nicht abgebildete Funktionen der echten Komponenten ermitteln. Zur Überprüfung solcher Aspekte wurden im Rahmen der vorliegenden Dissertation Lageregelungsalgorithmen an die Simulationsumgebung angebunden. Damit wurde in einer vom Verfasser betreuten Studienarbeit eine sogenannte Controller in the Loop Konfiguration aufgebaut, mit der nicht nur das korrekte Zusammenspiel zwischen Kontrollalgorithmus und Simulationsumgebung getestet, sondern auch die Robustheit dieser Algorithmen gezeigt werden konnte [Winter 2011]. Einige dazu gehörende Ergebnisse sind in Kapitel 6.1 zu finden.

Darüber hinaus wurde überprüft, ob die Simulationsmodelle der oben beschriebenen Umgebung mit denen der subsystemspezifischen Simulationsumgebungen konsistent sind. Da diese unabhängig voneinander aufgebaut wurden, waren identische Fehler eher selten und daher konnte auf diese Weise ein Großteil der Fehler ermittelt und korrigiert werden. Dabei wurden Simulationen mit der Lageregelungsmodellierung in Matlab/Simulink[16] sowie mit der in Abbildung 4.17 zu sehenden Thermalmodellierungssoftware ESATAN-TMS[17] in Betracht gezogen. Die simulierten Leistungen wurden punktuell mit dem Leistungverbrauch der realen Komponenten verglichen. Zuletzt wurde das Antwortverhalten des simulierten I/O Boards mit den Anforderungen für das echte I/O Board verglichen und bewertet.

Die oben erwähnte Controller in the Loop Konfiguration ist in Abbildung 4.18 dargestellt. Links oben sind dabei die Matlab/Simulink Controller zu sehen, links unten die im vorigen Kapitel beschriebene Hardware in the Loop Architektur mit Ethernet SpaceWire Router. Wie in [Winter 2011] beschrieben, wurden die Matlab/Simulink Lageregelungsalgorithmen dafür mit der im letzten Kapitel beschriebenen Simulationsumgebung verbunden, wofür eine spezielle Anwendung entwickelt wurde, die die-

[16] Matlab ist eine Software zur Lösung mathematischer Probleme. Simulink ist eine Ergänzung dazu, die die Erstellung und Nutzung grafischer Blöcke zur hierarchischen Modellierung bietet.

[17] Bei ESATAN-TMS handelt es sich um eine von der ESA genutzte und empfohlene Anwendung, die die thermische Modellierung ganzer Satelliten durch deren Diskretisierung in Knoten ermöglicht.

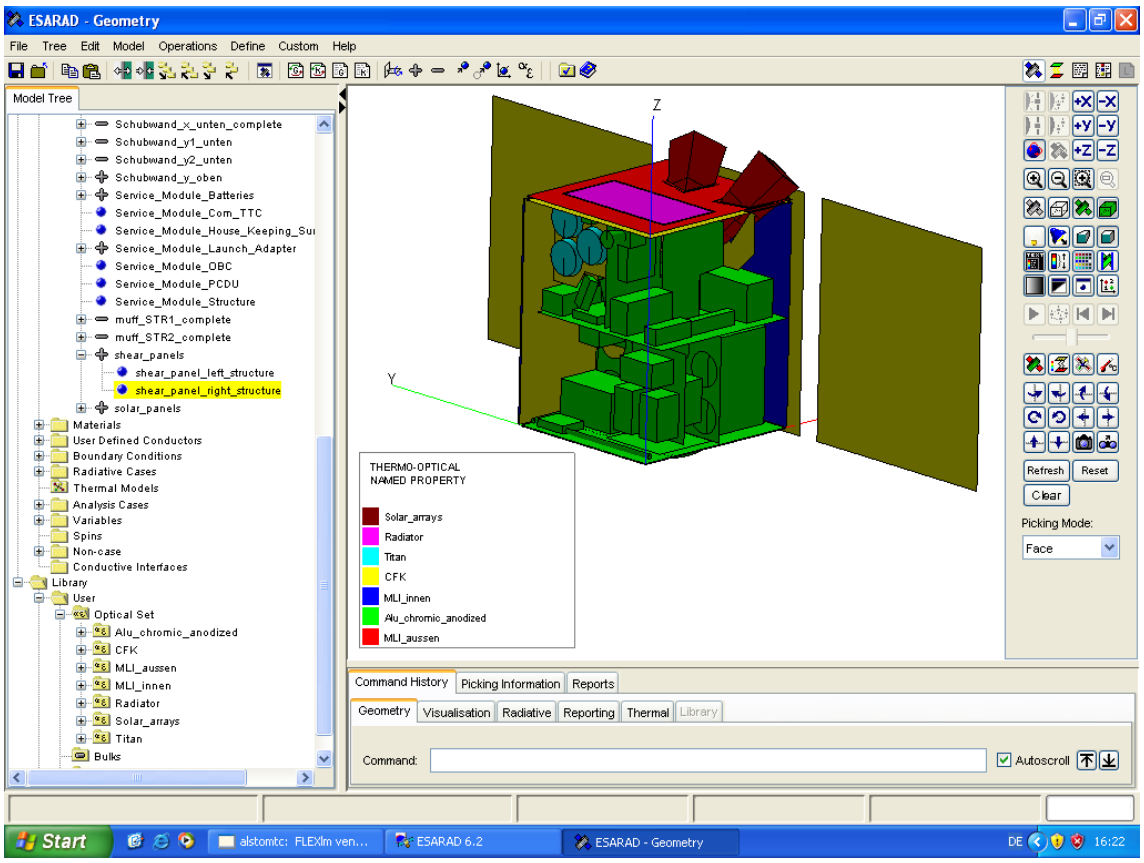


Abbildung 4.17: Thermalmodellierungssoftware ESATAN-TMS [Fritz 2009]

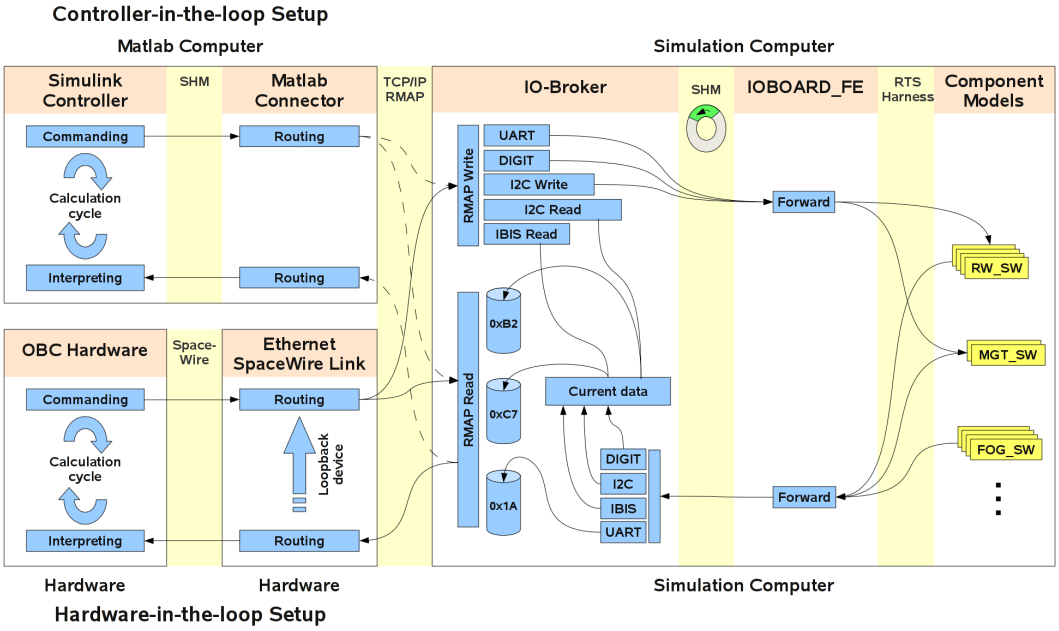


Abbildung 4.18: Architektur für ein reales Antwortverhalten des simulierten I/O Boards

se Verbindung ermöglicht. Damit konnten die Lageregelungsalgorithmen sowohl mit den in Matlab/Simulink wie auch den in der beschriebenen Simulationsumgebung implementierten Modellen des Lageregelungssystems verbunden und die Ergebnisse

verglichen werden. Dazu waren Randbedingungen wie beispielsweise Orbit, Startposition und -lage sowie simuliertes Startdatum zu beachten und entsprechend in beiden Umgebungen gleich zu wählen. Auf diese Weise konnten Fehler in beiden Simulationsumgebungen gefunden und behoben werden. Zuletzt waren die Ergebnisse wie in 6.1 beschrieben annähernd identisch.

4.5 Integration von Bordkomponenten in den Prüfstand

4.5.1 Vorgehensweise

Die Integration von Komponenten wie Bordrechnerkern, Peripherieplatinen, Lageregelungskomponenten oder PCDU in die oben beschriebene Verifikationsumgebung machte aufgrund der zu erwartenden Komplexität des gesamten Aufbaus eine standardisierte Vorgehensweise nötig. Eine solche vereinfacht auch die Reproduktion der unterschiedlichen, in Kapitel 5.1 beschriebenen Verifikationskonfigurationen. Diese Vorgehensweise ist in Abbildung 4.19 schematisch dargestellt.

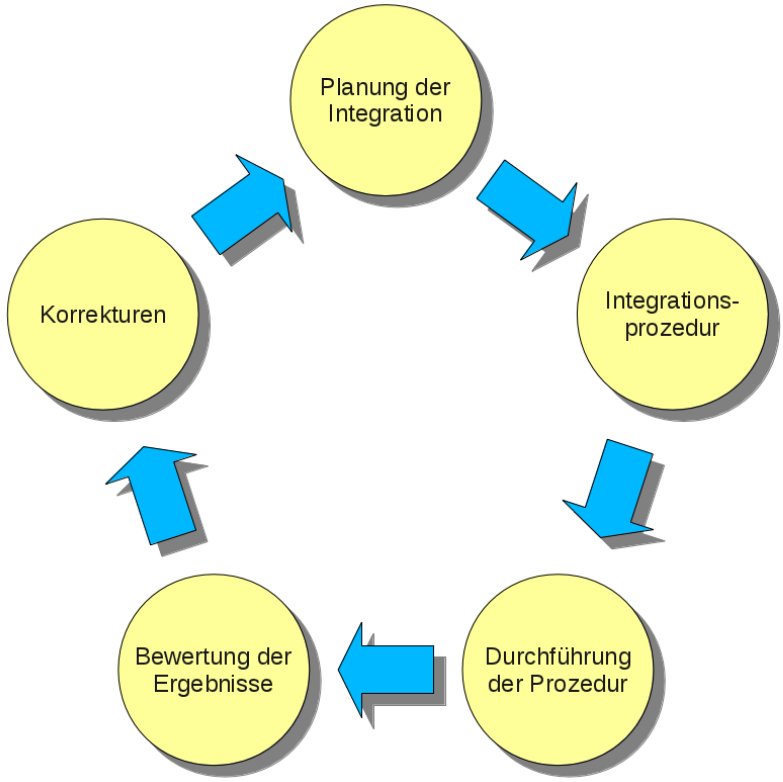


Abbildung 4.19: Vorgehensweise bei der Integration von Komponenten

Zu Beginn wurde die Integration zunächst geplant. Dabei waren unterschiedliche Faktoren zu beachten und zu koordinieren:

- Dazu gehörte die Verfügbarkeit der zu integrierenden Komponenten. Da die Komponenten von unterschiedlichen Herstellern beschafft wurden, war die Anordnung einer logischen Reihenfolge hinsichtlich der Verfügbarkeit nicht immer gegeben und musste entsprechend berücksichtigt werden.

- Die Komponenten wurden darüber hinaus zum Teil auch für andere Zwecke benötigt. Zu nennen sind die funktionale Verifikation auf Komponentenebene, Umwelttests auf Komponentenebene, die Aktualisierung der Komponentensoftware und die Überprüfung der mechanischen Kompatibilität.

- Die wichtigsten Komponenten benötigen für die korrekte Funktionsweise eine Software bzw. eine entsprechende Hardwarebeschaltung, je nachdem ob sie mit einem herkömmlichen Prozessor oder einem FPGA bestückt sind. Für die Planung war von Bedeutung, wann welche Softwareversionen zur Verfügung stehen. Damit konnten Prioritäten bei der Integration gesetzt werden.

- Mögliche Interaktionen zwischen Komponenten wurden im Rahmen dieser Planung koordiniert. Es wurde darauf geachtet, dass interagierende Komponenten möglichst zeitgleich bereit zum Test waren, um deren Kommunikation frühzeitig testen zu können.

- Aufgrund von gesetzlichen Regelungen[18] darf der Bordrechnerkern nicht manuell zugänglich sein. Nur wenige Personen haben erlaubten Zugriff. Trotzdem sollte elektronischer Zugriff auf die Platine einem erweiterten Personenkreis möglich sein, z.B. zur Bordrechnerkonfiguration und zur Ausführung der Bordsoftware. Auch das beeinflusste die Planung.

Die Koordination dieser Faktoren führte schließlich zur Ausarbeitung verschiedener, logisch geordneter Integrationsprozeduren, deren Ziele ab Kapitel 4.5.2 näher beschrieben sind. Diese waren nötig, um im Nachhinein exakt feststellen zu können unter welchen Randbedingungen die Integration und die damit verbundenen Tests stattgefunden haben. Darin wurden der Aufbau und die durchzuführenden Arbeitsschritte genau beschrieben. Auch der Test der Peripherieausrüstung wie Kabel, Netzgeräte und Erdung wurde dokumentiert. Die Prozeduren wurden jeweils von sachverständigen Ingenieuren auf mögliche Fehler geprüft, um das Risiko von Sachschäden zu minimieren. Anschließend konnten die Prozeduren durchgeführt werden.

[18] Es handelt sich um die in den Vereinigten Staaten von Amerika geltenden International Traffic in Arms Regulations, kurz ITAR. Zu Deutsch etwa: Regelungen für den internationalen Waffenhandel. Sie kommen beim Bordrechnerkern zur Anwendung, da dieser in den Vereinigten Staaten produziert wurde.

Jeder Schritt wurde dabei dokumentarisch festgehalten, um später feststellen zu können, wer den Test durchgeführt hat. Auch die Ergebnisse wurden in die Prozedur eingetragen. Nach deren Analyse konnte ermittelt werden, ob alles wie erwartet funktioniert hat. Bei Problemen waren oft Korrekturen nötig, die wiederum die Planung beeinflussen konnten und zur Überarbeitung und erneuten Durchführung von Prozeduren geführt haben. Die in Abbildung 4.19 gezeigte Vorgehensweise hat sich damit sowohl als realisierbar wie auch als effektiv erwiesen.

4.5.2 Der Bordrechner

Das Ingenieurmodell des Bordrechnerkerns des Flying Laptops war die erste Komponente, welche in die in Kapitel 4.4 beschriebene Verifikationsumgebung integriert wurde. Dabei wurde die im vorigen Kapitel beschriebene Vorgehensweise angewandt. Direkt nach der Lieferung war eine sogenannte Incoming Inspection[19] notwendig, wie sie auch für die Entwicklungsmodelle des I/O Boards und des CCSDS Boards durchgeführt wurde. Dabei wurde geprüft, ob die Lieferung vollständig war und ob keine sichtbaren Schäden vorhanden waren. Die Ergebnisse der Incoming Inspection sowie äußere Randbedingungen wurden dokumentiert und von den verantwortlichen Ingenieuren unterzeichnet. Vor der in Kapitel 5.2 zusammengefassten Inbetriebnahme des Bordrechnerkerns musste dessen Ingenieurmodell zunächst in die Verifikationsumgebung integriert werden. Das Ingenieurmodell des Bordrechnerkerns wurde, wie in Abbildung 4.20 zu sehen, mit einer Tochterplatine ausgeliefert, die beispielsweise einige Standardstecker für Schnittstellen bereit stellt. Dadurch kann bei der Entwicklung der Bordsoftware der Bordrechnerkern überwacht und konfiguriert werden.

Zunächst wurde ein Kabel zur Stromversorgung angefertigt, mit einem Multimeter dessen Korrektheit vermessen und mit einem Netzgerät verbunden, das vordefinierte Spannungen zur Verfügung stellt. Nach der Überprüfung der Spannung gegen den vorgegebenen Wert wurde das andere Kabelende mit der Tochterplatine des Bordrechnerkerns verbunden und dieser konnte schließlich mit Strom versorgt werden. Um den Bordrechnerkern später konfigurieren zu können, wurde dieser über einen USB[20]-Adapter der Firma Xilinx mit einem herkömmlichen Arbeitsplatzrechner verbunden. Ein weiteres Kabel dient zur Übertragung eines RS-422 Signals zwischen Bordrechnerkern und Arbeitsplatzrechner. Abbildung 4.21 skizziert den vollständigen Aufbau nach der Integration.

[19] zu Deutsch etwa: Eingangskontrolle

[20] USB steht für Universal Serial Bus. Es handelt sich um den für Rechner üblichen Standard zum Anschluss externer Geräte.

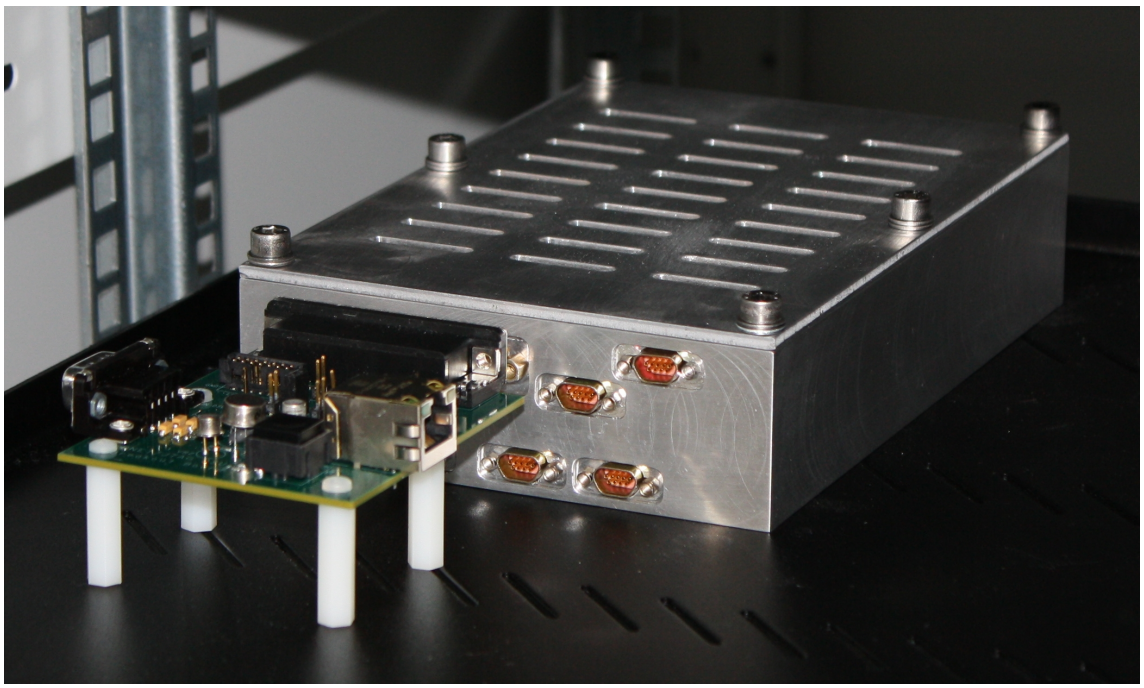


Abbildung 4.20: Ingenieurmodell des Bordrechnerkerns

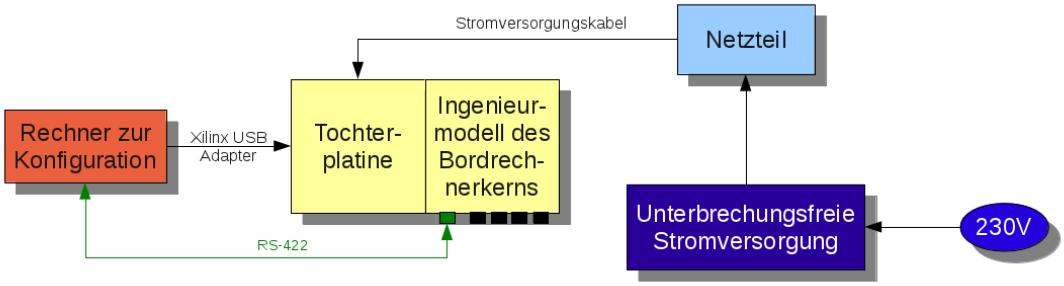


Abbildung 4.21: Schematische Skizze des integrierten Bordrechners

Nun wurden die Entwicklungsmodelle des CCSDS Boards und des I/O Boards in die Umgebung integriert. Auch hierfür wurden Stromversorgungskabel gefertigt und auf deren Pinbelegung überprüft. Nach dem Einstellen der geforderten Spannung und deren Aktivierung von einem Netzteil aus, wurde die korrekte Stromversorgung durch Überprüfung bestimmter Leuchtdioden auf den Platinen festgestellt. Beide Platinen können außerdem über eine JTAG Schnittstelle und einen entsprechenden Adapter, dem sogenannten Actel FlashPro, mit einem dedizierten Rechner verbunden werden, um beispielweise Speicherbereiche auszulesen, aber auch um die Hardwareschaltung der FPGAs neu zu schreiben und damit den funktionellen Umfang der Geräte zu verändern. Das CCSDS Board verfügt außerdem ebenfalls über Tochterplatinen. Im Gegensatz zum Bordrechnerkern dienen diese aber nicht zur Stromversorgung und Fehlererkennung, sondern zur Bereitstellung der später auch auf dem Flugmodell vorgesehenen Schnittstellen. Zuletzt wurden diese Platinen mit

SpaceWire Kabeln an den Bordrechnerkern angeschlossen. Der integrierte Zustand ist schematisch in Abbildung 4.22 dargestellt.

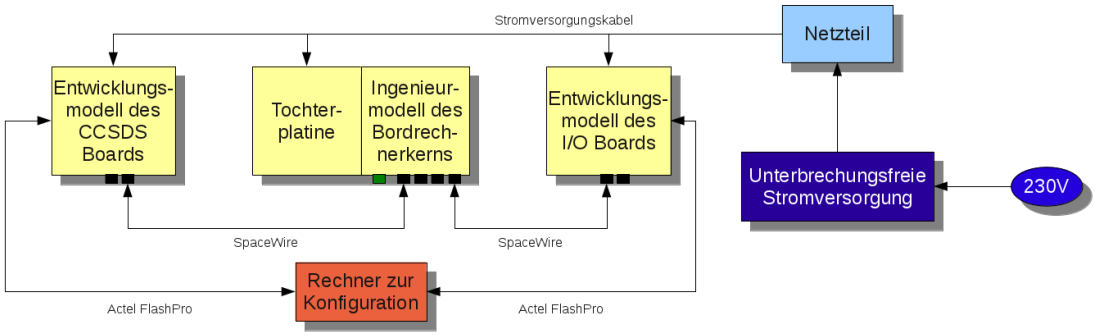


Abbildung 4.22: Schematische Skizze der integrierten Peripherieplatinen

Bei der gesamten Integration wurde Wert darauf gelegt, dass die Stecker der Schnittstellen gut und sicher zugänglich sind. Dafür wurde eine entsprechende mechanische Anordnung gewählt sowie Blenden gefertigt und eingebaut. Die integrierten Bordrechnerkomponenten sind in Abbildung 4.23 zu sehen.

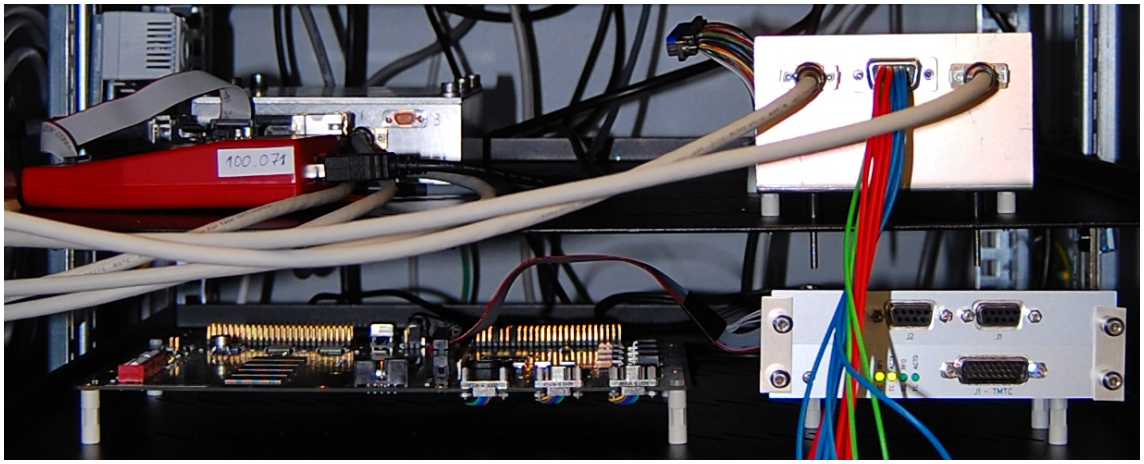


Abbildung 4.23: Integrierte Bordrechnerkomponenten

4.5.3 Weitere Bordkomponenten

Um die vollständige Kommandierungskette in Betrieb nehmen zu können, mussten an das I/O Board Bordkomponenten angeschlossen werden. Hierbei handelte es sich um eine sogenannte Open-Loop Konfiguration. Das bedeutet, dass die angeschlossenen Komponenten nicht mit dem Simulator interagierten. Dafür wäre zusätzliche Stimulations- und Analyseinfrastruktur nötig gewesen. Mit dieser Kommandierungskette konnte verifiziert werden, dass

- Telekommandos und Telemetrie auf dem Weg vom Missionskontrollsystem zum Bordrechner richtig konvertiert werden,
- der Bordrechner Telekommandos und Telemetrie richtig prozessiert
- und angeschlossene Komponenten richtig angesteuert werden können.

Da unterschiedliche Komponenten angeschlossen werden sollten, war ein Konzept zu entwickeln, das den schnellen Auf- und Abbau der Komponenten erlaubt, den Zugang zu diesen zugleich schützt und ermöglicht sowie ein dem späteren Satelliten entsprechendes Erdungskonzept bereit stellt. Das Erdungskonzept des Flying Laptop sieht vor, alle Gehäuse gegen einen Punkt zu erden. Daher müssen die an den Bordrechner angeschlossenen Komponenten den gleichen Erdungspunkt wie der Bordrechner selbst haben. Gewählt wurde eine Konstruktion aus Stahl und Aluminium, die diesen Anforderungen entspricht und in Abbildung 4.24 dargestellt ist.

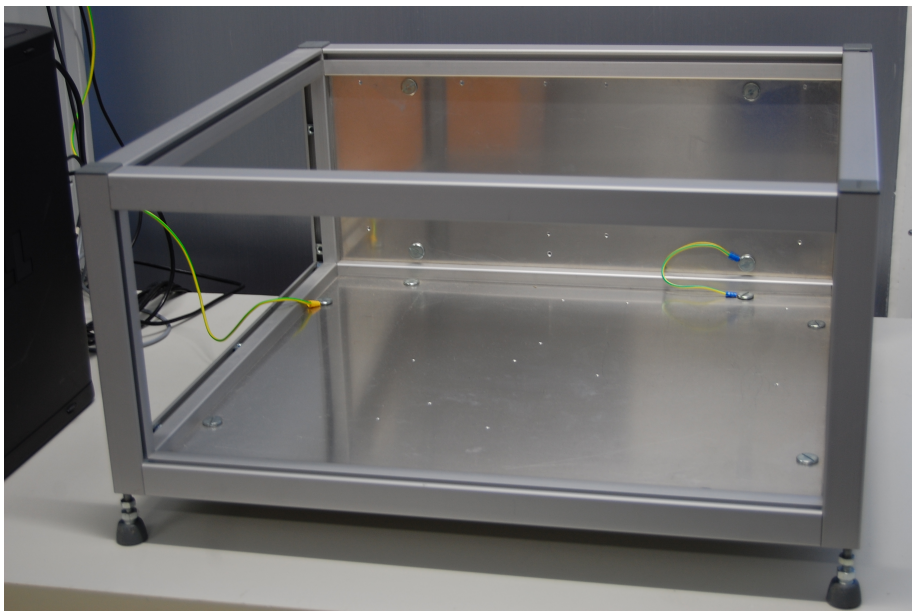


Abbildung 4.24: Konstruktion für den Test von Bordkomponenten

Darauf wurde zunächst das in Abbildung 4.25 zu sehende Ingenieurmodell der PCDU des Flying Laptop aufgebaut. Je nach Komponente werden unterschiedliche weitere Geräte benötigt, um diese anzuschließen. Für die PCDU waren das ein Netzgerät, das die PCDU mit der späteren Batteriespannung versorgt und Kabelklemmen, an denen mit einem Multimeter ausgelesen werden konnte, ob die PCDU kommandierte Ausgänge auch schaltete. Dafür waren auch spezielle Kabel vom entsprechenden PCDU Ausgang hin zu den Klemmen nötig. Auch für den Anschluss der PCDU an den Bordrechner wurden Kabel gefertigt.

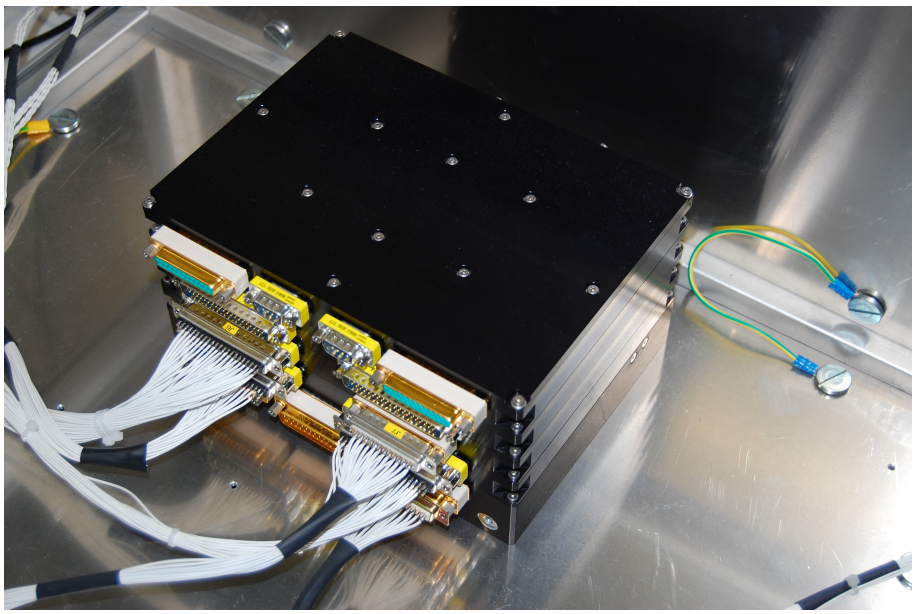


Abbildung 4.25: Ingenieurmodell der PCDU

Die PCDU kann als einzige Komponente vom Boden aus am Bordrechner vorbei kommandiert werden, um im Falle eines Ausfalls des Bordrechnerkerns oder der Bordsoftware die Stromversorgung des Rechners aus- und wieder anschalten zu können. Dazu werden so genannte High-Priority Commands[21] (HPC) verwendet. Für die korrekte Identifikation von HPCs werden die in Kapitel 3.1 beschriebenen virtuellen Kanäle verwendet. Normale Telekommandos werden vom CCSDS Board von der Codingebene in die Transferebene konvertiert und auf dem Speicher der Platine abgelegt, bis sie vom Bordrechnerkern abgeholt werden. HPCs hingegen werden bis in die Paketebene konvertiert und über eine RS-422 Schnittstelle als UART 8N1[22] direkt an die PCDU weitergeleitet. Bei vielen Satellitenprojekten gibt es zur Weiterverarbeitung eine Command Pulse Distribution Unit[23] (CPDU) [Del Re 2008]. Diese Funktion ist beim Flying Laptop jedoch in die PCDU integriert. Da die Pakete nicht vom Bordrechner prozessiert werden, konnten sie direkt in das Missionskontrollsystem integriert und verifziert werden. Darum war die PCDU auch die erste an den Bordrechner angeschlossene Komponente.

4.5.4 Vorkehrungen zur Sicherheit

Für den gesamten Prüfstand wurde das in Abbildung 4.26 dargestellte Erdungskonzept erarbeitet. Durch diesen Schaltplan wird klar ersichtlich, dass der gesamte Aufbau gegen einen Punkt geerdet ist, wodurch unterschiedliche Potentiale vermieden

[21] zu Deutsch: Kommandos hoher Priorität
[22] Siehe Anhang B
[23] zu Deutsch etwa: Einheit zur Verteilung von Befehlsimpulsen

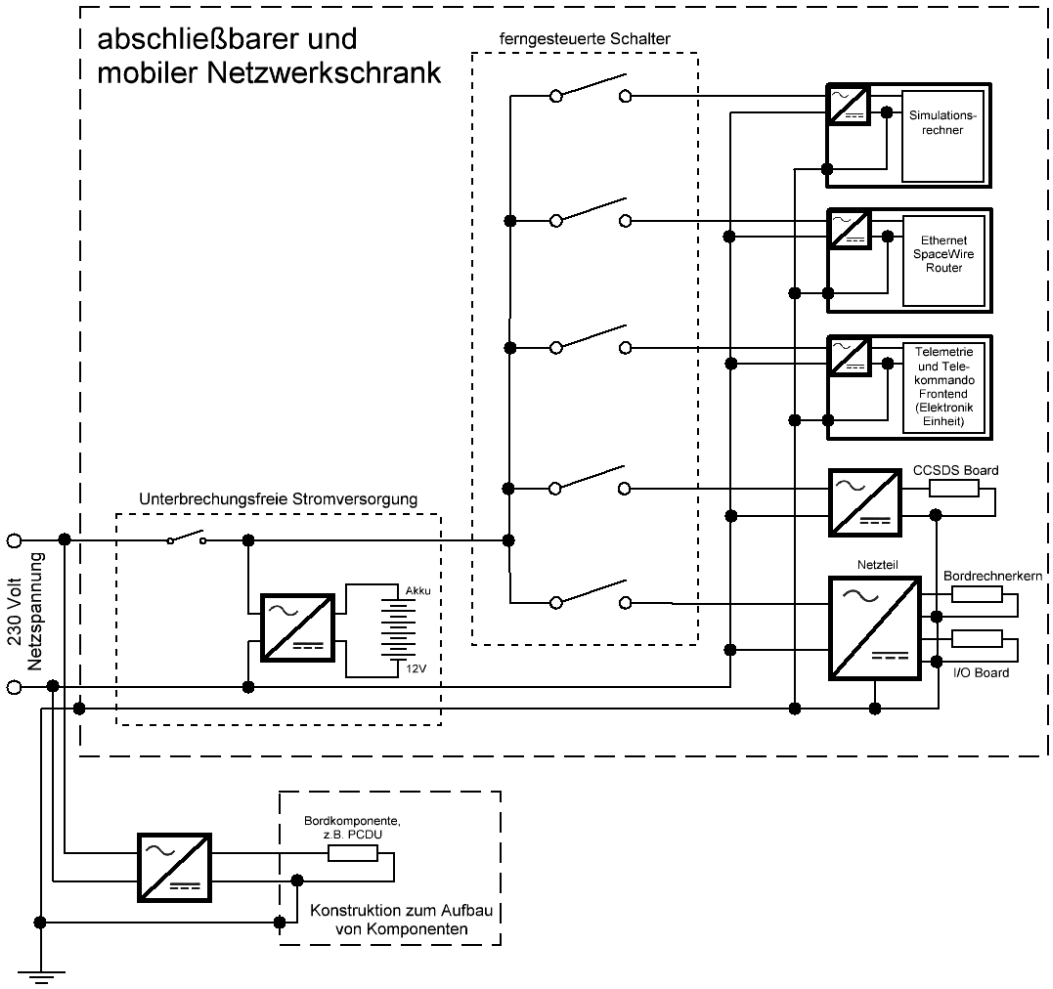


Abbildung 4.26: Erdungskonzept für den integrierten Bordrechnerkern

werden. Auch die galvanisch isolierten Spannungsausgänge der Netzgeräte wurden durch Kabel mit dem Sternpunkt verbunden. Die Gefahr einer ungewollten Spannungsentladung an der Elektronik des gesamten Bordrechners wurde damit stark verringert. Um den Bordrechnerkern vor unerlaubtem Zugriff zu schützen, wurde ein abschließbarer Netzwerkschrank beschafft. Hierin befinden sich auch Netzgeräte und die in Kapitel 4.4.1 beschriebene Simulationsinfrastruktur sowie die Elektronikeinheit des Telemetrie und Telekommando Frontends. Durch Einbau von Schaltern, die mit einem herkömmlichen Webbrowser oder via Skripte ferngesteuert betätigt werden können, ist für die Nutzung der im nachfolgenden Kapitel beschriebenen Verifikationskonfigurationen kein manueller Zugriff auf den Schrank mehr nötig. Um die Elektronik gegen Schwankungen des Stromnetzes wie Spannungsspitzen oder Ausfälle zu schützen, wurde eine unterbrechungsfreie Stromversorgung in den Schrank integriert. Mit diesem Gesamtkonzept für die Integration muss der Netzwerkschrank nur an eine übliche 230 V Steckdose und an das jeweilige Intranet angeschlossen wer-

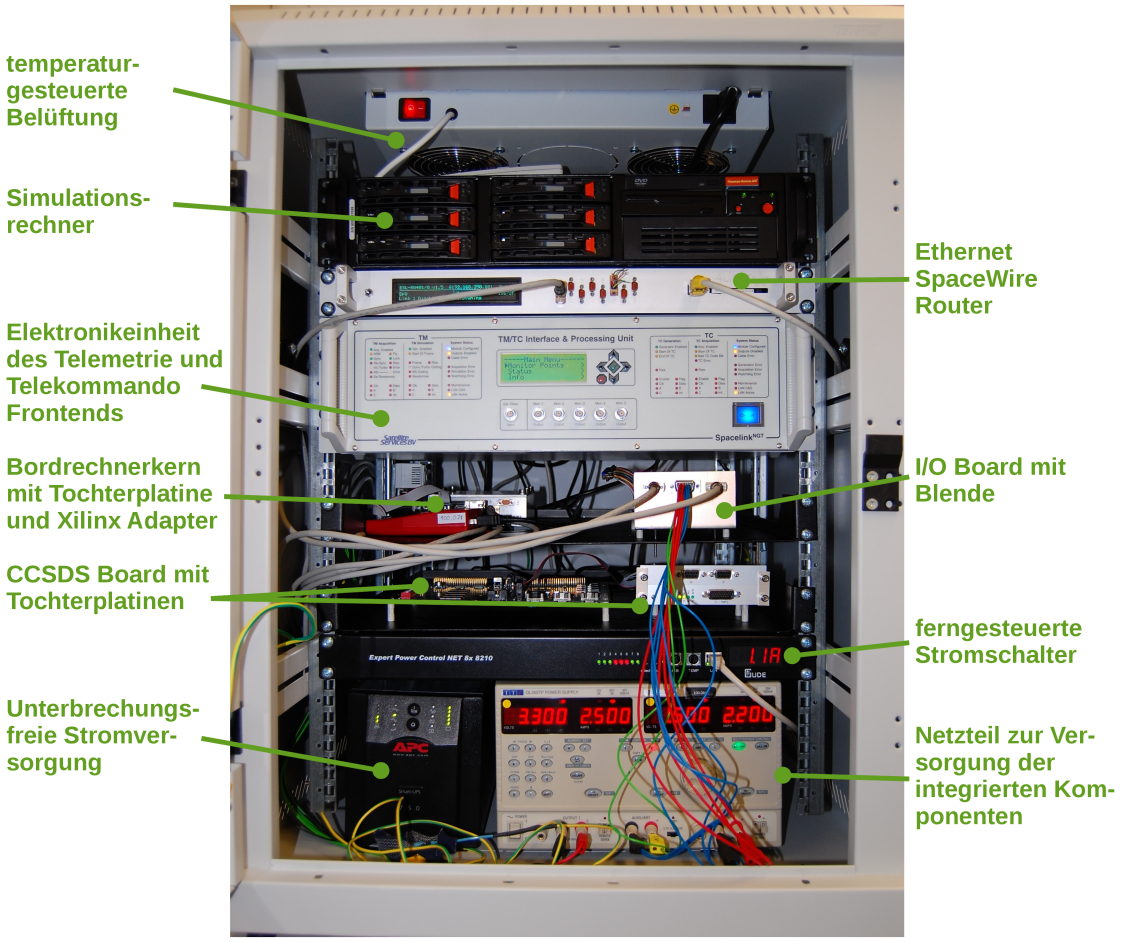


Abbildung 4.27: Netzwerkschrank mit integrierten Komponenten und Geräten

den, was den gesamten Aufbau sehr mobil macht. Die spätere Nutzung an anderen Orten wie dem Reinraum stellt also kein Problem dar. Abbildung 4.27 zeigt den ebenfalls in Abbildung 5.2 zu sehenden Netzwerkschrank. In Abbildung 4.27 wird er aber im geöffneten Zustand und im Detail mit den integrierten Komponenten dargestellt.

5 Durchführung der Tests

5.1 Verifikationskonfigurationen

Im Entwicklungsprozess eines Satelliten gibt es verschiedene Konfigurationen zur funktionalen Verifikation. Dabei werden sowohl die Funktionalitäten der Software wie auch der Hardware verifiziert. Außerdem können solche Konfigurationen auch während des Betriebs, z.B. für den Test von aktualisierter Bordsoftware oder für die Ausbildung von Operatoren genutzt werden. Auch bei der Entwicklung des Flying Laptop wurden mehrere solcher Konfigurationen entwickelt und genutzt. Abbildung 5.1 zeigt die bereits vorgestellten Hauptelemente dieser Konfigurationen in einem schematischen Überblick. In diesem sind die Elemente logisch angeordnet, sodass das Prinzip schnell zu erfassen ist. Oben zu sehen sind die Elemente der Bodenstation, die bereits in Kapitel 4.4.1 beschrieben wurden. Die Automatisierungssoftware ersetzt den Operator und steuert das Missionskontrollsystem SCOS-2000. SCOS-2000 ist wiederum mit dem Telemetrie und Telekommando Frontend verbunden, das die Daten von und an SCOS-2000 für die Übertragung aufbereitet. Später werden diese Daten durch Modulation auf eine Funkstrecke zwischen Bodenstation und Bordrechner übertragen. Da die dafür benötigten Geräte noch nicht zur Verfügung standen, wurde die Funkstrecke durch ein Kabel überbrückt. In der Mitte sind Bordrechner und PCDU abgebildet, die später Teil des Satelliten sind. Auf dem CCSDS Board werden analog zum Frontend am Boden Daten aufbereitet. Telekommandos können an den Bordrechnerkern oder die PCDU weitergeleitet werden. Der Bordrechnerkern kann wiederum über das I/O Board mit den Bordkomponenten kommunizieren. Dies können sowohl reale Komponenten wie auch die in der untersten Reihe abgebildete Simulation von I/O Board, Komponenten und Weltraumumgebung sein. Abbildung 5.2 ist eine Fotografie des Prüfstands.

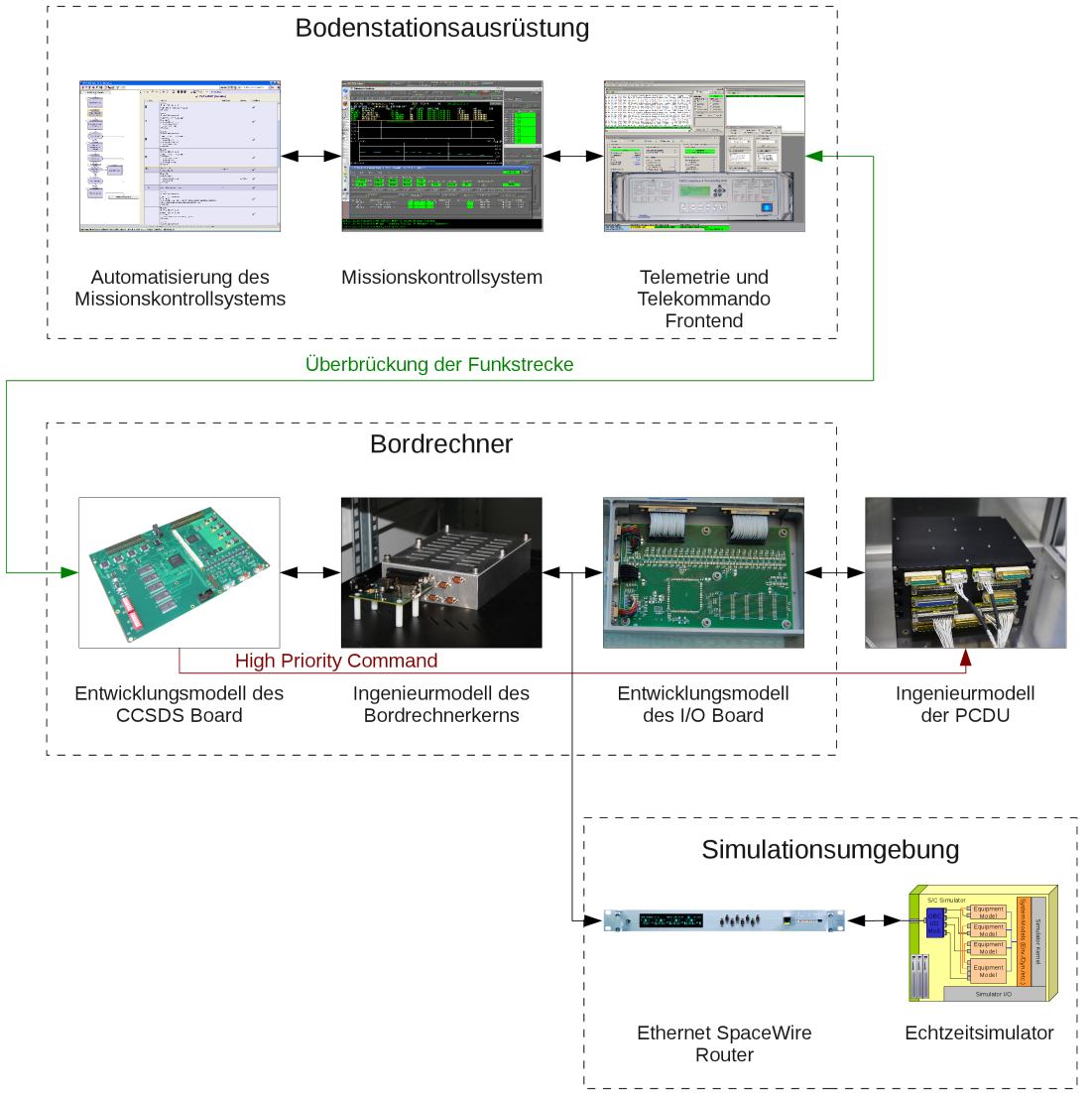


Abbildung 5.1: Logische Anordnung der Hauptelemente für die Konfigurationen

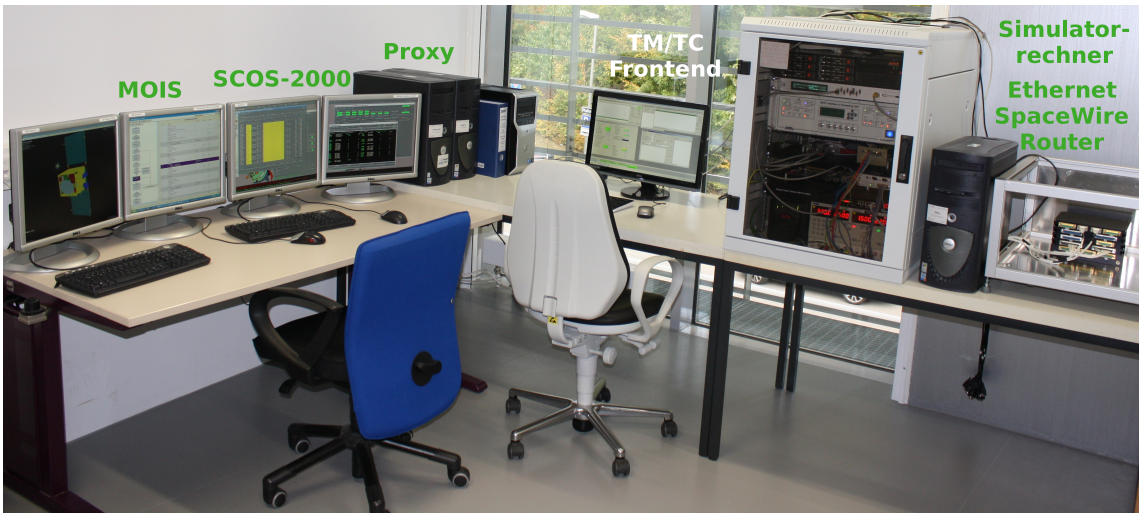


Abbildung 5.2: Fotografie der Verifikationsinfrastruktur

Abbildung 5.3 zeigt den Aufbau im Detail. Hier sind auch in Abbildung 5.1 nicht zu sehende Geräte dargestellt und die Verbindungen im Einzelnen abgebildet. Bei den zusätzlich dargestellten Geräten handelt es sich um für die Tests notwendige Peripherie, beispielsweise zur Stromversorgung. In den weiter unten vorgestellten Konfigurationen werden die aus Abbildung 5.3 benötigten Elemente verwendet. Dabei wird aus Platzgründen die Anordnung im Einzelfall etwas verändert. Die konfigurationsspezifische Kommunikation wird jeweils als UML[1] Sequenzdiagramm dargestellt.

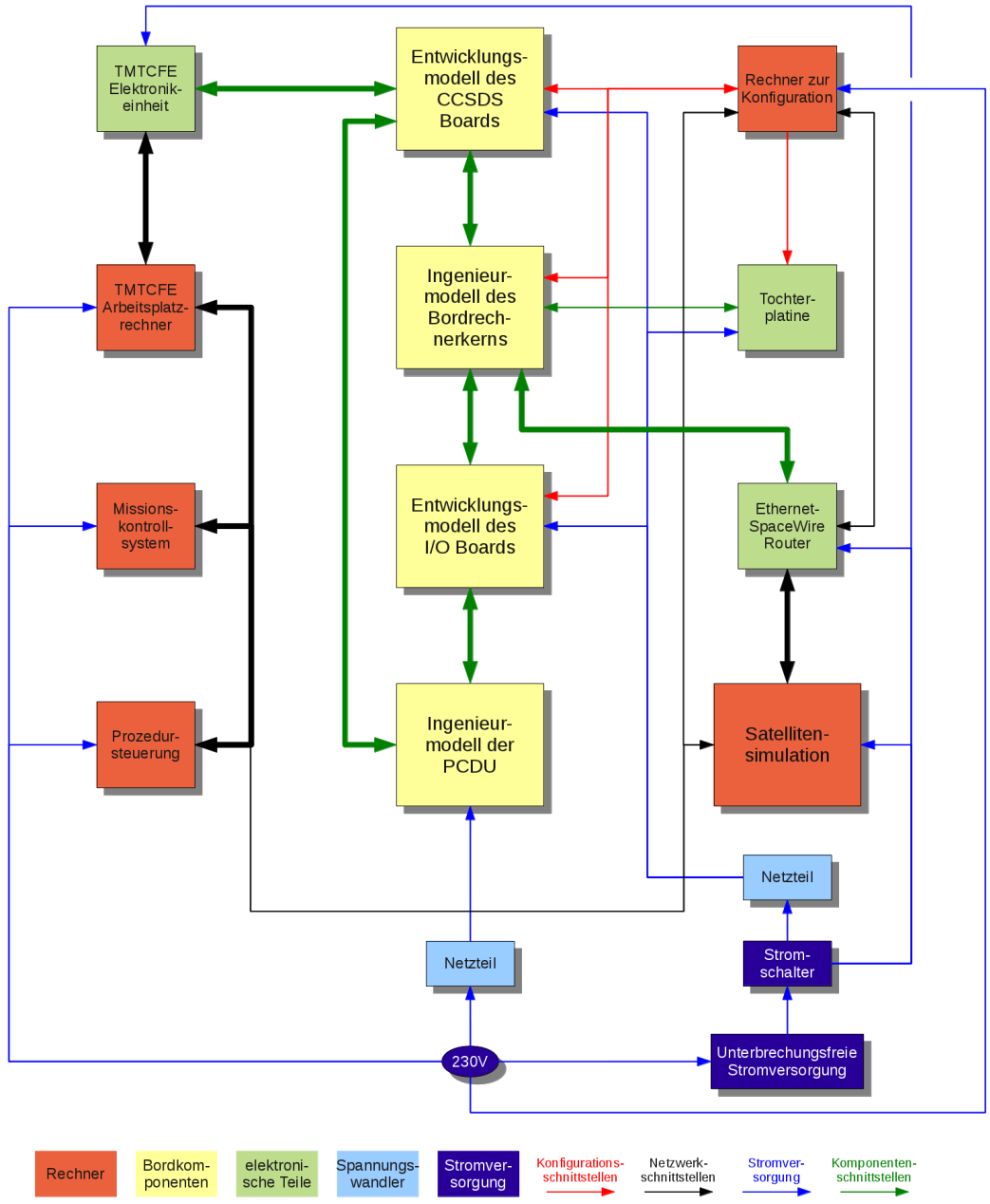


Abbildung 5.3: Alle Elemente der Verifikationsumgebung

[1] Abkürzung für Unified Modeling Language, zu Deutsch: Vereinheitlichte Modellierungssprache. Siehe Anhang E.

Die folgenden Konfigurationen wurden im Rahmen dieser Arbeit aufgebaut:

- Konfiguration 1 zur Inbetriebnahme des Bordrechnerkerns.
- Konfiguration 2 für den Test der realen Kommandierung des Bordrechners.
- Konfiguration 3, in der der Bordrechner mit der vorgestellen Simulationsumgebung getestet werden kann.
- Konfiguration 4 für Kommunikationstests zwischen realen Komponenten und dem Bordrechner.
- Konfiguration 5 zum Test der für die PCDU vorgesehenen hochprioritären Kommandos.

Diese Konfigurationen und damit durchgeführte Tests werden nun dargestellt. Außerdem werden geplante Konfigurationen dargestellt, um aufzuzeigen, für welche Tests über diese Dissertation hinaus der Prüfstand verwendet werden kann.

5.2 Inbetriebnahme des Bordrechnerkerns

Nach der mechanischen und elektrischen Integration des Bordrechnerkerns standen viele der in Abbildung 5.3 dargestellten Geräte noch nicht zur Verfügung, unter anderem das Telemetrie und Telekommado Frontend, aber auch die in Kapitel 3.1 beschriebenen Peripherieplatinen des Bordrechners sowie das Ingenieurmodell der PCDU. Für die Inbetriebnahme des Bordrechnerkerns war eine Konfiguration zu entwickeln, die schnell zu realisieren war und gleichzeitig alle zur Verifikation nötigen Funktionalitäten bot. Nur so konnte die Inbetriebnahme den im Projektplan gesetzten Zielen gerecht werden. Zur Inbetriebnahme des Bordrechnerkerns gehört zunächst das erfolgreiche Schreiben auf und Lesen von dessen Speicherelementen. Anschließend sollte eine simple Software auf den Speicher geschrieben und diese gestartet werden. Mit einer solchen sollte die Kommunikation über alle zur Verfügung stehenden Schnittstellen getestet werden. Der letzte Schritt der Inbetriebnahme ist die korrekte Handhabung der zur Kommunikation über die SpaceWire Schnittstellen genutzten Daten im RMAP Format mit einem dafür am IRS entwickelten Treiber. Abbildung 5.4 zeigt die verwendeten Elemente für den als Konfiguration 1 bezeichneten Aufbau. Neben dem Bordrechnerkern selbst handelt es sich bei den Elementen um den Ethernet SpaceWire Router und einen Arbeitsplatzrechner mit dedizierter Software für die Konfiguration des Bordrechnerkerns. Damit konnte der Bordrechnerkern über die in rot dargestellte Verbindung mit dem in Kapitel 4.5.2 erläuterten

und in Abbildung 5.5 dargestellten Xilinx Adapter von diesem Arbeitsplatzrechner aus konfiguriert und erstmals mit Software beschrieben werden. Unter Zuhilfenahme des Routers konnten außerdem die SpaceWire Schnittstellen über die in grün dargestellte Verbindung ausgelesen werden.

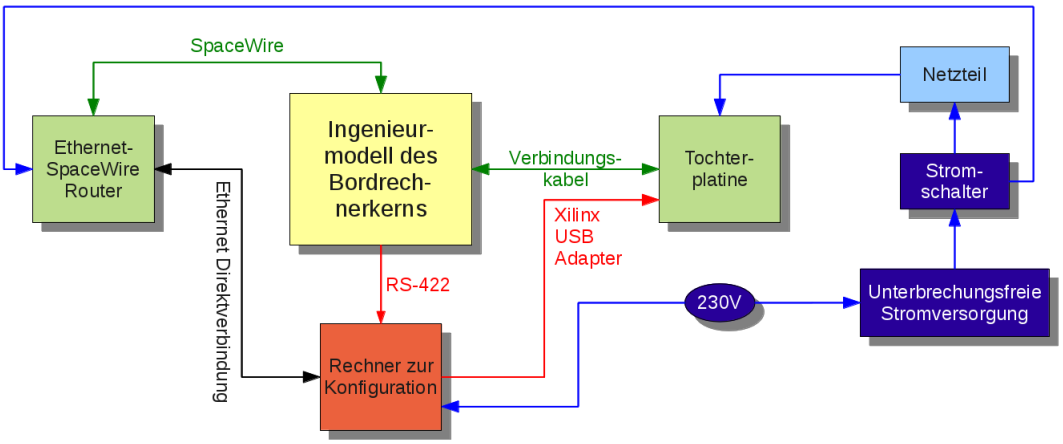


Abbildung 5.4: Konfiguration 1: Inbetriebnahme des Bordrechnerkerns

Abbildung 5.5: Xilinx Adapter zur Umsetzung von USB auf JTAG

Die Konfiguration unter Verwendung des Xilinx Adapters ist in Abbildung 5.6 näher dargestellt. Dieser setzt zunächst das vom Arbeitsplatzrechner kommende USB-Signal auf JTAG um. Mit JTAG kann direkt auf den Speicher des Bordrechnerkerns zugegriffen werden. So konnte der Speicher beschrieben und gelesen werden. Dieser Schritt war notwendig, um die auf der Platine verfügbare Hardware des Bordrechnerkerns überwachen zu können, da bestimmte Speicherbereiche dieser Hardware

zugewiesen sind [Mohr 2011]. Auf dem Arbeitsplatzrechner wurde dafür eine Software namens GRMON installiert, die für die Überwachung und Konfiguration des LEON3 Prozessors mit einem solchen Xilinx Adapter von Aeroflex Gaisler entwickelt wurde. Damit kann außerdem für den LEON kompilierte Software auf den Speicher geladen werden. Anschließend kann von GRMON aus der Prozessor angewiesen werden, die Software auszuführen. Auf diese Weise wurde auch in den nachfolgenden Konfigurationen Software auf den Bordrechnerkern geladen und gestartet.

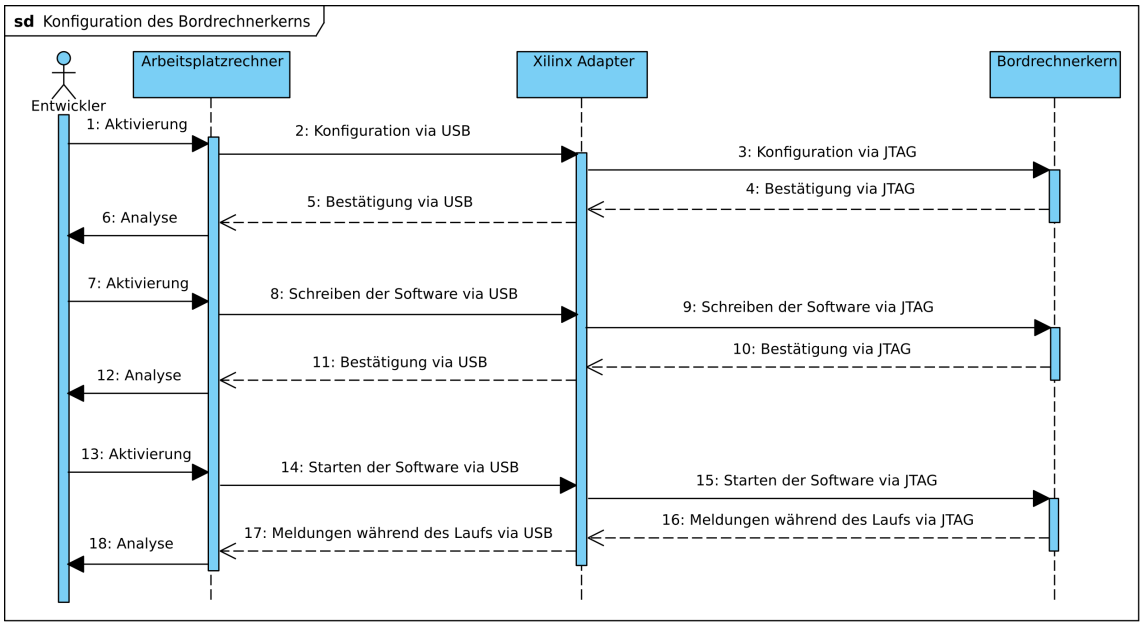


Abbildung 5.6: UML Sequenzdiagramm für das Konfigurieren des Bordrechners

In Betrieb genommen wurde auch eine Schnittstelle des Bordrechnerkerns, über die der Bordrechnerkern während der Ausführung von Bordsoftware Meldungen ausgeben kann. Solche Schnittstellen werden in der Fachsprache Service Interface (SIF) genannt [Eickhoff 2011]. Bei der UT699 Platine handelt es sich um eine RS-422 Schnittstelle. Damit konnte erstmals eine im Rahmen von [Mohr 2011] entwickelte Software-Version auf den Bordrechner geladen und gestartet werden, die

- auf die der Hardware zugewiesenen Speicherbereiche schreibt und damit die SpaceWire Schnittstellen anspricht und
- gleichzeitig entsprechende Meldungen über die RS-422 Schnittstelle ausgibt.

Dieses Kommunikationsprinzip ist in Abbildung 5.7 dargestellt. Durch die Verwendung des Ethernet SpaceWire Routers konnten die SpaceWire Schnittstellen vom Arbeitsplatzrechner ausgelesen werden. Auf diese Weise wurden beide Schnittstellen erstmals getestet. In beiden UML Sequenzdiagrammen wird die handelnde Person

als Entwickler bezeichnet, da diese Konfiguration direkt von den Entwicklern zur schnellen Fehlersuche benutzt wurde.

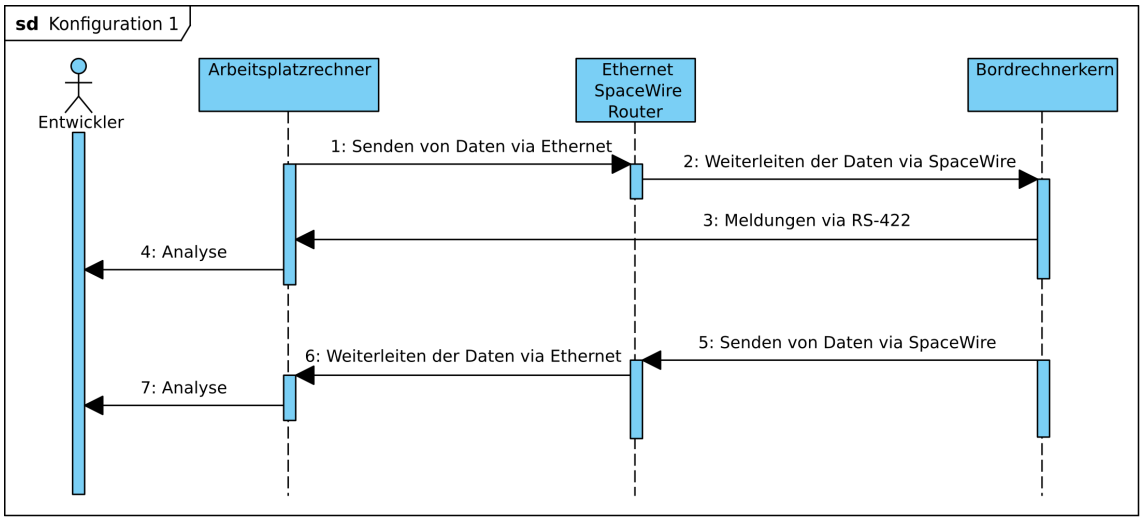


Abbildung 5.7: UML Sequenzdiagramm für Konfiguration 1

Konfiguration 1 wurde außerdem dafür verwendet, den in [Mohr 2011] beschriebenen, entwickelten und als Software kompilierten Treiber zur Handhabung des Protokolls RMAP auf dem Bordrechnerkern zu testen. Hierfür wurden vom Arbeitsplatzrechner entsprechende RMAP Pakete an den Bordrechnerkern geschickt und die entsprechenden Antwortpakete des Bordrechners analysiert. Umgekehrt konnte evaluiert werden, ob der Bordrechnerkern beim Absenden von RMAP Paketen korrekt arbeitet.

Mit Konfiguration 1 konnte also verifiziert werden, dass der Bordrechner konfiguriert werden kann und seine zur Verfügung stehenden Hardwareelemente ansprechbar sind. Dieses Ergebnis war notwendig, um bei der Verbindung des Bordrechnerkerns mit anderen Geräten Fehler auf dem Kern selbst ausschließen zu können.

5.3 Kommandierung des Bordrechners

Die nächste Konfiguration diente dazu, die Kommunikation zwischen Missionskontrollsystem und Bordrechnerkern zu testen. Nachdem ein Entwicklungsmodell des CCSDS Boards sowie das Telemetrie und Telekommando Frontend zur Verfügung standen, war diese Konfiguration 2 einsatzbereit. Wie in Abbildung 5.8 zu sehen, konnten damit Kommandos vom zur Bodenausrüstung gehörenden Missionskontrollsystem an den Bordrechner mit allen späteren Konvertierungen der Pakete vorgenommen werden. Einzig die Funkstrecke wurde hier noch überbrückt. Die Kommandierung wird in den Konfigurationen 3 bis 5 verwendet.

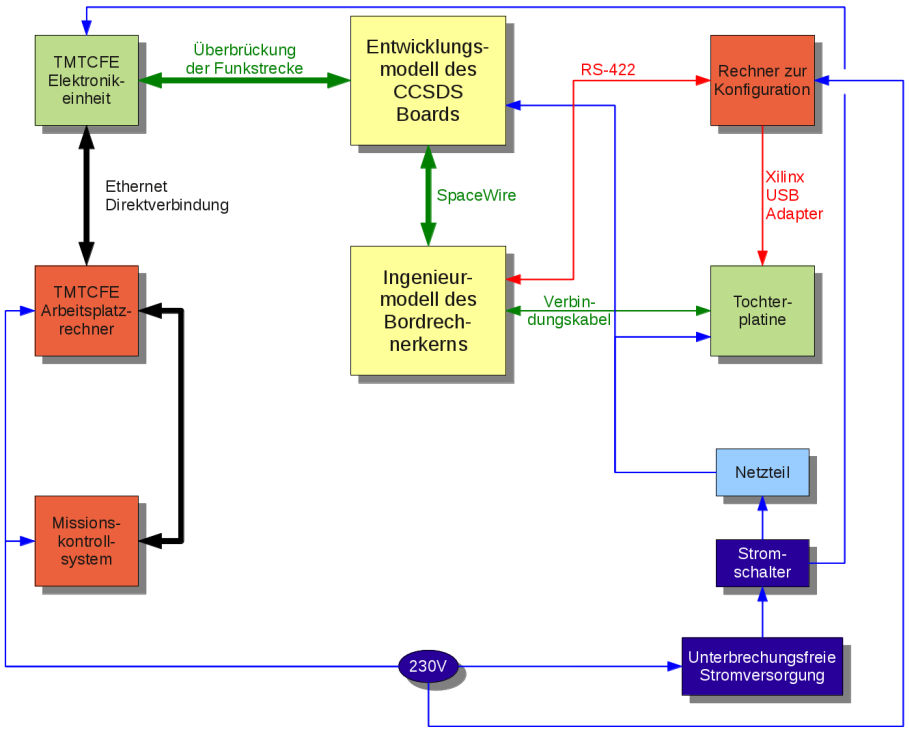


Abbildung 5.8: Konfiguration 2: Kommandierung des Bordrechners

Abbildung 5.9 zeigt die Kommunikation zwischen SCOS-2000 und dem CCSDS Board. Dabei gibt es zwei generelle Möglichkeiten: Der Operator kann Telekommandos an den Satelliten senden und von diesem ebenso Telemetrie empfangen. Um ein Telekommado zu senden, ist dieses in SCOS-2000 auszuwählen und abzusenden. SCOS-2000 wählt für das Telekommando das in der zugehörigen Datenbank vordefinierte Format. Beim Flying Laptop handelt es sich um die in Kapitel 3.1 beschriebenen Source Packets nach CCSDS. Weitere Konvertierungen in Richtung Segmentebene, Transferebene, Codingebene und physikalische Ebene übernimmt anschließend das Telemetrie und Telekommando Frontend. Die Daten werden auf dem CCSDS Board in die Transferebene rückkonvertiert und auf einem Speicher der Platine abgelegt. Sowohl SCOS-2000 wie auch das Frontend ermöglichen dem Operator durch Meldungen, den Weg, aber auch Parameter und Inhalt des Telekommandos zu verfolgen und geben Aufschluss über Fehler. Die Telemetrieseite funktioniert ähnlich: Telemetrie auf Paketebene wird von einem Speicher des CCSDS Boards gelesen und in die physikalische Ebene konvertiert. Das Frontend konvertiert die empfangene Telemetrie zurück in die Paketebene und leitet diese an SCOS-2000 weiter, wo sich der Operator die Telemetriedaten alphanumerisch und grafisch anzeigen lassen kann. Auch hier geben beide Systeme Informationen zur Telemetrie. So kann SCOS-2000 Telemetrie beispielsweise nach bestimmten, zum Telemetriepaket gehörenden

Feldern ordnen und archivieren. In beiden Fällen dient der Speicher des CCSDS Boards als Schnittstelle zum Bordrechnernkern.

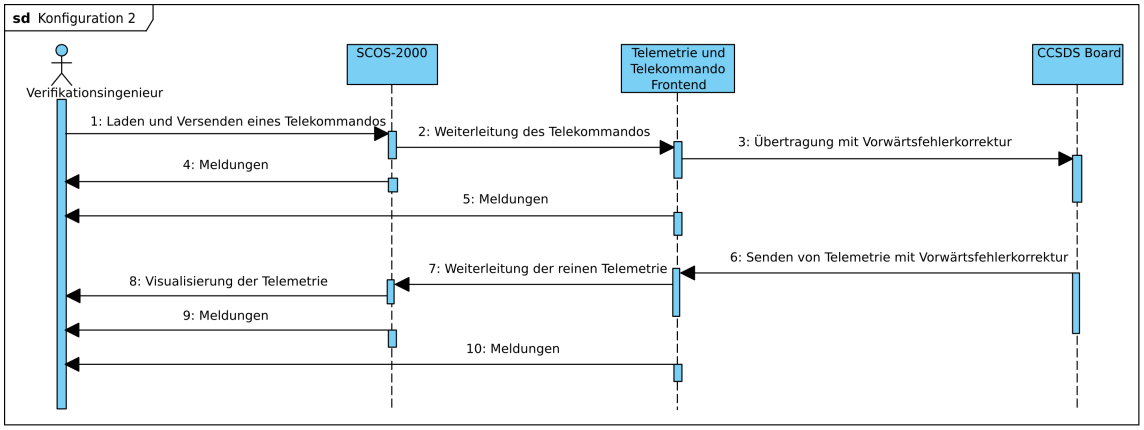


Abbildung 5.9: UML Sequenzdiagramm für Konfiguration 2

Nun war zu überprüfen, ob Telekommandos und Telemetrie auf ihrem Weg vom Missionskontrollsystem zum Bordrechnerkern und umgekehrt in allen Schritten korrekt konvertiert werden. Auch die in Kapitel 3.1 beschriebene Vorwärtsfehlerkorrektur war dabei zu testen. Nachfolgend wird die Konvertierung in beiden Richtungen je anhand eines Beispiels erläutert.

Telekommandos an den Bordrechnerkern des Flying Laptops basieren auf dem CCSDS Standard. Nach Abbildung 3.6 beinhaltet ein in SCOS-2000 definiertes Telecommand Source Packet Informationen hinsichtlich der Art, der Nummerierung und der Länge des Pakets. Darüber hinaus sind im Data Field Header Informationen zu finden, die den Paketverwendungsstandard PUS betreffen. Eine Prüfsumme bildet das Ende eines solchen Pakets. SCOS-2000 ermöglicht die flexible Gestaltung der Pakete: So können verschiedene Informationseinheiten definiert und beliebig kombiniert werden. Dadurch wird die Definition von Paketen mit unterschiedlichen PUS-Informationen oder mit unterschiedlichem Dateninhalt einfach und flexibel. Abbildung 5.10 skizziert dieses Prinzip. Es ist zu sehen, dass der CCSDS Paketkopf beispielsweise für unterschiedliche Pakettypen verwendet wird und das Baukastenprinzip die Definition vollständiger Pakete vereinfacht.

Zur Demonstration wurde ein Telecommand Source Packet gewählt, dass die aufeinanderfolgenden hexadezimalen Daten 0xCA, 0xFE, 0xCA und 0xFE enthielt. Das Muster CAFECAFE ließ sich auf dem gesamten Weg leicht verfolgen [Bucher 2011]. Abbildung 5.11 ist die hexadezimale Darstellung des Pakets in SCOS-2000. Die Paketdaten “CAFECAFE” sind auf Anhieb gut zu erkennen. Unten auf der Abbildung ist die Zuordnung zu den Paketfeldern dargestellt. Alle Felder entsprechen dabei den in [CCSDS 2003b] enthaltenen Definitionen. Die Version ist nach dem

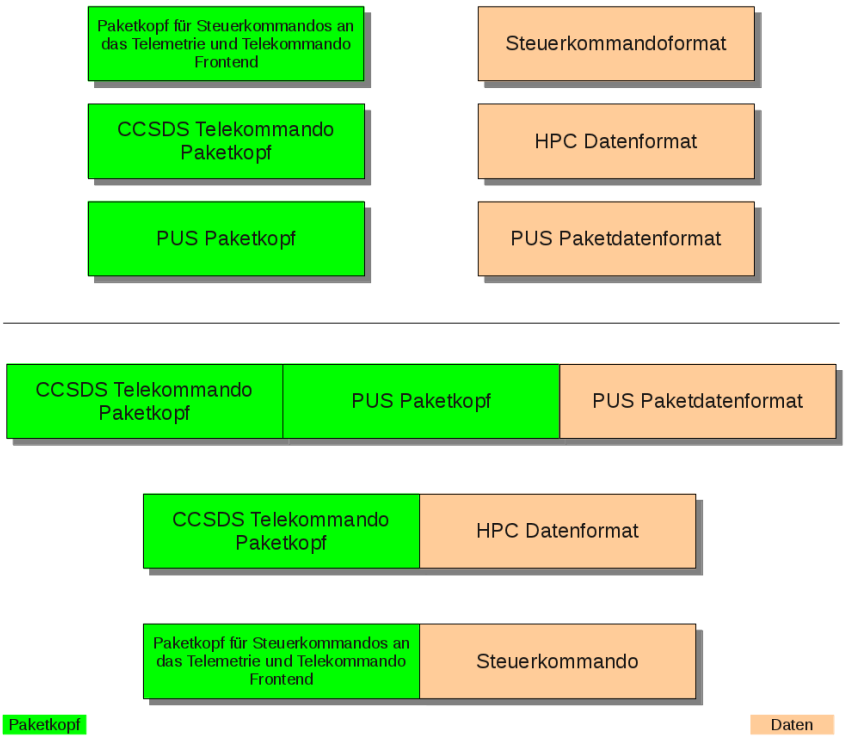


Abbildung 5.10: Flexible Gestaltung von Paketen in SCOS-2000

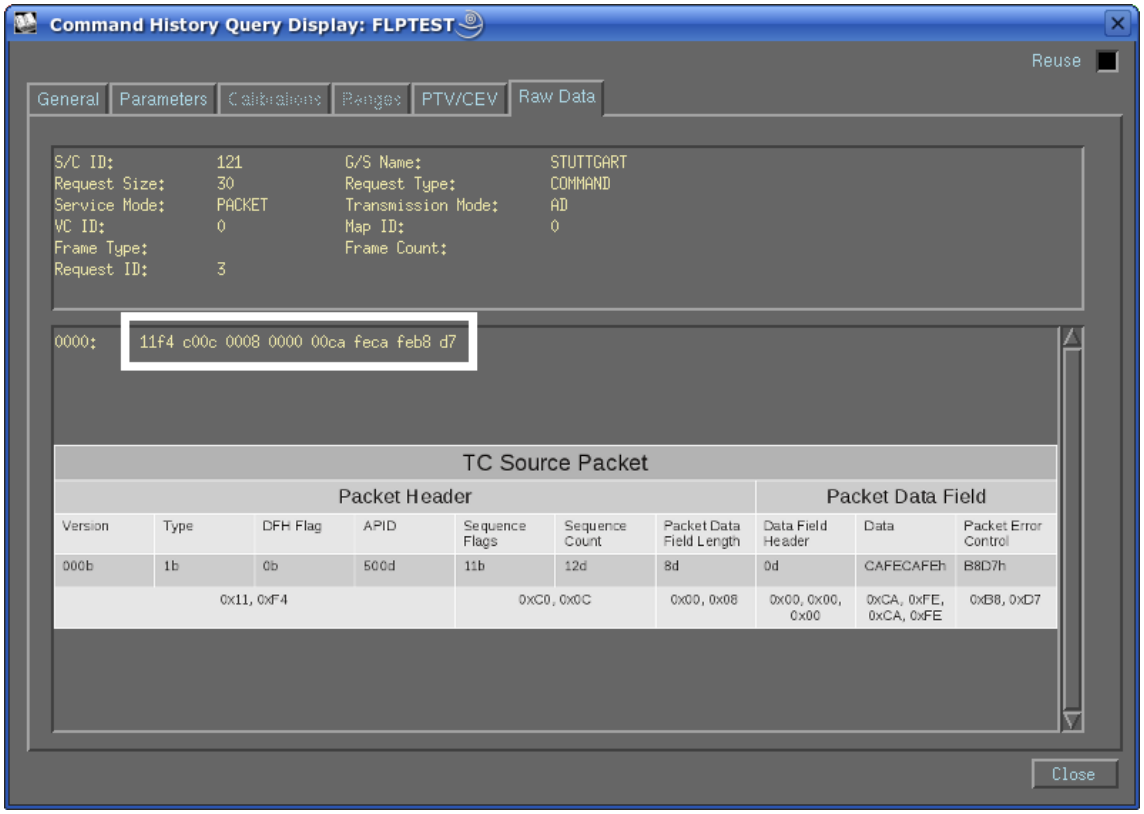


Abbildung 5.11: Telecommand Source Packet in SCOS-2000

genannten Standard stets mit “000” anzugeben. Die “1” im Feld Typ gibt an, dass es sich um ein Telekommando handelt, “0” bedeutet Telemetrie. Das Data Field

Header Flag beinhaltet die Information, ob der Data Field Header verwendbare Daten beinhaltet. Als Application Process Identification (APID) wurde 500 gewählt. Für diesen Test war wichtig, dass es sich nicht um die dem Simulator oder dem Telemetrie und Telekommando Frontend zugewiesenen APIDs für Steuerkommandos handelt; beide sind größer als 1500. Das Feld Sequence Flag gibt an, ob die Paketdaten aufgrund einer entsprechenden Größe auf mehrere Pakete verteilt sind. Falls das, wie hier im Beispiel, nicht der Fall ist, soll der Wert “11” betragen. Nach dem Zähler für Pakete gleicher APID folgt die Länge des Datenfelds. Hier ist nach CCSDS eine Besonderheit definiert: Die Länge soll um eins geringer angegeben werden, als sie tatsächlich ist. Das bedeutet, die tatsächliche Länge des Datenfelds ist hier neun. Der Data Field Header ist unter Verwendung des erläuterten Baukastenprinzips mit einer Länge von 3 Bytes definiert und mit Nullen gefüllt, da nur geprüft werden soll, ob das Paket am Bordrechnerkern ankommt. Es muss dort nicht weiter prozessiert werden. Die Daten entsprechen dem CAFECAFE-Muster und zuletzt wird eine Prüfsumme angehängt.

SCOS-2000 wurde dazu entwickelt, nicht direkt mit Bodenstationen, sondern mit einem Bodenstationsfrontend zu kommunizieren. Dieser Ansatz bot den Vorteil, dass die Entwicklung des Missionskontrollsystems unabhängig von der Kommunikation mit dem Bodenstationsnetzwerk wurde. Das erwähnte Frontend nennt sich Network Control and Telemetry Routing System[2] (NCTRS) [Jones et al. 1998]. Damit SCOS-2000 dem Bodenstationsfrontend NCTRS zusätzliche Informationen zukommen lassen kann, werden, wie in Abbildung 5.12 dargestellt, vor dem Absenden 15 Bytes zu Beginn jedes Telecommand Source Packets ergänzt. Diese Bytes werden für den Kleinsatelliten Flying Laptop aber nicht benötigt, da aus Gründen der Einfachheit des Gesamtsystems auf NCTRS verzichtet wird. Bei Verwendung von NCTRS würde das Bodenstationsfrontend vor der Weiterleitung des Pakets diese 15 Bytes prozessieren und anschließend wieder entfernen. Das Entfernen dieser Bytes wurde für die Verwendung am IRS in den in Kapitel 4.4.2 beschriebenen Proxy integriert.

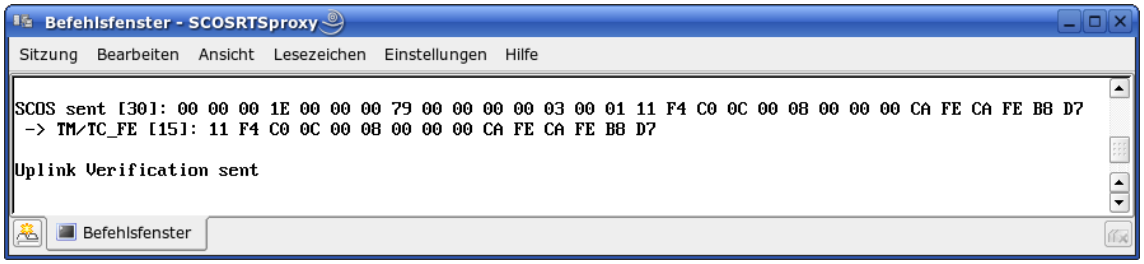


Abbildung 5.12: Paketkonvertierung nach dem Absenden aus SCOS-2000

[2]zu Deutsch etwa: System zur Netzwerksteuerung und Telemetrieweiterleitung

Nun wurde das ursprünglich definierte Paket via Netzwerk an das Telemetrie und Telekommando Frontend weitergeleitet. Hier findet die Konvertierung der TC Source Packets in Command Link Transmission Units statt. Dafür werden einige Informationen wie Zähler und Identifikation des Raumfahrzeugs automatisch hinzugefügt. Andere Informationen werden von Einstellungen am Frontend übernommen. Diese Einstellungen können wiederum von SCOS-2000 aus gesetzt werden, siehe Kapitel 4.4.2. Zunächst wird ein Segment mit der im Frontend aktuell gesetzten Map ID[3] erzeugt. Dieses Segment wird um die in Abbildung 3.6 als Frame Header gekennzeichneten Informationen sowie eine Prüfsumme zu einem TC Transfer Frame ergänzt. Hierfür werden die Definitionen aus [CCSDS 2010] verwendet. Abbildung 5.13 stellt diese vom Telemetrie und Telekommando Frontend hinzugefügten Informationen dar. Als Versionsnummer wird im genannten Standard “00” empfohlen. Das zweite Feld gibt an, ob auf einen Mechanismus, der die Annahme des Frames prüft, verzichtet werden soll. Da dieser Mechanismus zum Zeitpunkt des Tests noch nicht in die Bordsoftware integriert war, wird in dieser Konfiguration darauf verzichtet und das Bit beträgt “1”. Das Control Command Flag wird mit “0” angegeben, da es sich nicht um ein Control Command handelt. Nachdem ein nicht belegtes Feld mit “00” gefüllt wird, folgt das Feld Spacecraft Identification. Hier ist eine hexadezimale “123” eingetragen, da das verwendete Entwicklungsmodell des CCSDS Boards vor Beantragung der tatsächlichen Spacecraft ID gefertigt und dieser Wert willkürlich bestimmt wurde. Mittlerweile wurde beim Consultative Committee for Space Data Systems eine Spacecraft ID beantragt, sie lautet “25D”. Nach der Lieferung des Ingenieurmodells des CCSDS Boards mit der korrekten Spacecraft ID muss die Spacecraft ID im Telemetrie und Telekommando Frontend noch entsprechend angepasst werden. Als virtueller Kanal wird hier der nullte gewählt, damit das Paket an den Bordrechnerkern und nicht an die PCDU weitergeleitet wird, vgl. Kapitel 3.1. Die Länge des Frames wird mit “22” angegeben. Wie beim Paket ist aber die tatsächliche Länge um eins größer und beträgt 23. Die Frame Sequence Number wird auf “0” gesetzt, da auf den Mechanismus zur Überprüfung der Annahme des Frames verzichtet wird. Die Sequence Flag entspricht der Information, die unter gleichem Namen im Paket zu finden ist und beträgt daher “11”. Da innerhalb eines jeden Kanals durch Multiplexing weitere Unterkanäle angewählt werden können, existiert die Map ID. Da auch hier noch keine Informationen verfügbar sind, wird dieses Feld mit “0” belegt. Nach dem bereits oben beschriebenen TC Source Packet wird am Ende des Frames eine Prüfsumme eingefügt.

[3] Multiplexer Access Point Identifier

TC Transfer Frame											
Frame Header								TC Segment			Error Control
Version	Bypass Flag	Control Cmd Flag	Spare	Spacecraft ID	Virtual Channel ID	Frame Length	Frame Se-quence №	Sequence Flags	Map ID	TC Source Packet	CRC
00b	1b	0b	00b	123h	0d	22d	0d	11b	0d	-	5F66h

Abbildung 5.13: Telecommand Transfer Frame des versendeten Telekommandos

Zuletzt wird der TC Transfer Frame in die Command Link Transmission Unit konvertiert. Dazu wird gemäß des verwendeten BCH-Codes zunächst nach jedem siebten Byte ein Byte zur Vorwärtfehlerkorrektur eingefügt. Sollten am Ende des Frames weniger als sieben Bytes übrig sein, werden diese mit 0x55 wertigen Bytes ergänzt, sodass auch hier im Anschluss ein Byte zur Vorwärtsfehlerkorrektur eingesetzt werden kann. Der so entstandene Frame wird nun zu Beginn um einen 2 Bytes und am Ende um einen 8 Bytes großen Marker ergänzt, damit dem CCSDS Board eine Information zur Verfügung gestellt wird, wo die CLTU beginnt und wo sie endet. Die gesamte CLTU ist in Abbildung 5.14 dagestellt. Die BCH-Codes sind dabei blau eingerahmt, der Startmarker grün und der Endmarker rot. Gut zu erkennen sind auch die aufgefüllten Bytes vor dem letzten BCH-Code.

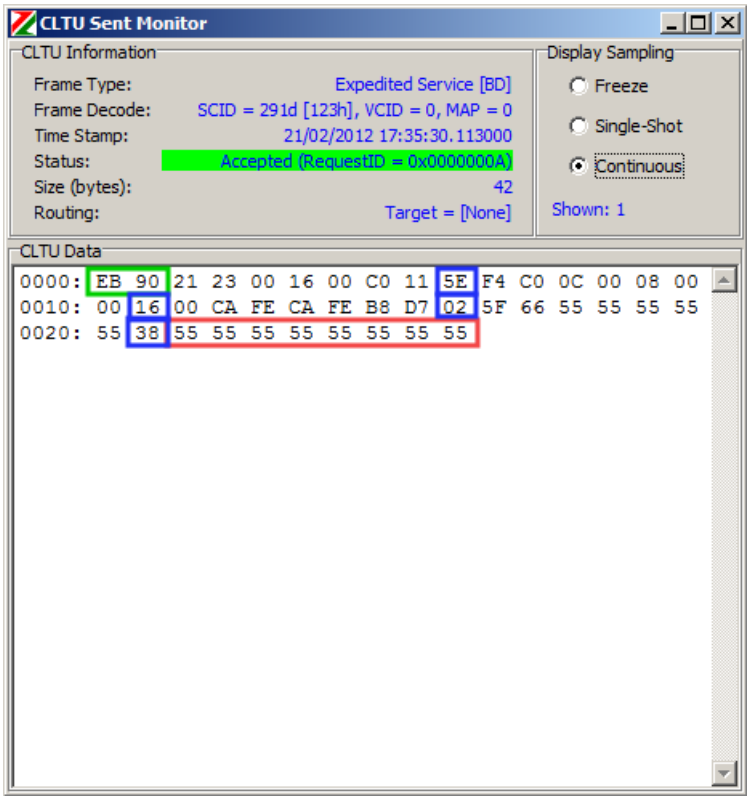


Abbildung 5.14: Command Link Transmission Unit des versendeten Telekommandos

Nun wird die CLTU vom Telemetrie und Telekommando Frontend als UART 8N1 an das CCSDS Board gesendet. Auf dem CCSDS Board wird die CLTU zunächst

in den TC Transfer Frame zurück konvertiert. Dabei werden die BCH-Codes dazu genutzt bei der Übertragung entstandene Fehler zu korrigieren. Der TC Transfer Frame wird schließlich auf einem Speicher des CCSDS Boards abgelegt und kann dort vom Bordrechnerkern aus über die SpaceWire Schnittstelle unter Verwendung von RMAP abgeholt werden. Abbildung 5.15 stellt den Zugriff des Bordrechnerkerns auf diesen Speicher dar. Dort wird jedes Byte des TC Transfer Frames mit einem weiteren Byte maskiert, damit beim Lesen vom Bordrechnerkern aus Anfang und Ende einer vom CCSDS Board empfangenen Informationseinheit erkannt werden können. Diese beiden Maskierungbytes sind in der Abbildung grün hinterlegt. Die am Ende im Rahmen der CLTU Konvertierung angehängten Bytes des Wertes 0x55 können in einem nächsten Schritt durch die Auswertung der Längenangabe des TC Transfer Frames ignoriert werden, womit der Frame wieder vollständig hergestellt ist.

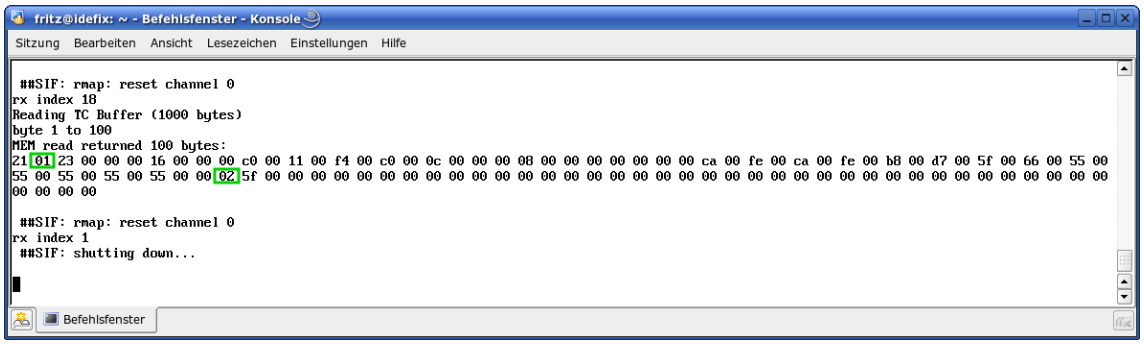


Abbildung 5.15: Auf dem CCSDS Board gespeichertes Telekommando

Auf dem Bordrechnerkern findet dann die systematische Rückkonvertierung in Pakete und deren Prozessierung statt. Abbildung 5.16 zeigt alle auf der Strecke von SCOS-2000 bis zum Bordrechnerkern durchgeführten Konvertierungen.

Wie in Kapitel 5.1 beschrieben und in Abbildung 5.9 dargestellt, sendet das Telemetrie und Telekommando Frontend nach dem Empfang eines Telekommandos von SCOS-2000 Statustelemetrie an SCOS-2000 zurück, die dort korrekt empfangen wird. Außerdem waren zum Zeitpunkt der Überprüfung noch keine Telemetriepakete definiert. Daher war es bei der Überprüfung der Telemetrie ausreichend, leere, korrekt eingepackte TM Transfer Frames, sogenannte Idle Frames, vom CCSDS Board an das Telemetrie und Telekommando Frontend zu senden. Damit konnte auf die Definition von Testtelemetrie in der Bordsoftware und in SCOS-2000 verzichtet werden. Außerdem ermöglichten Registereinträge auf dem CCSDS Board das zyklische Senden solcher Idle Frames vom CCSDS Board. Dabei wurden die Informationen des TM Transfer Frames bereits automatisch korrekt gesetzt. Auf diese Weise konnte

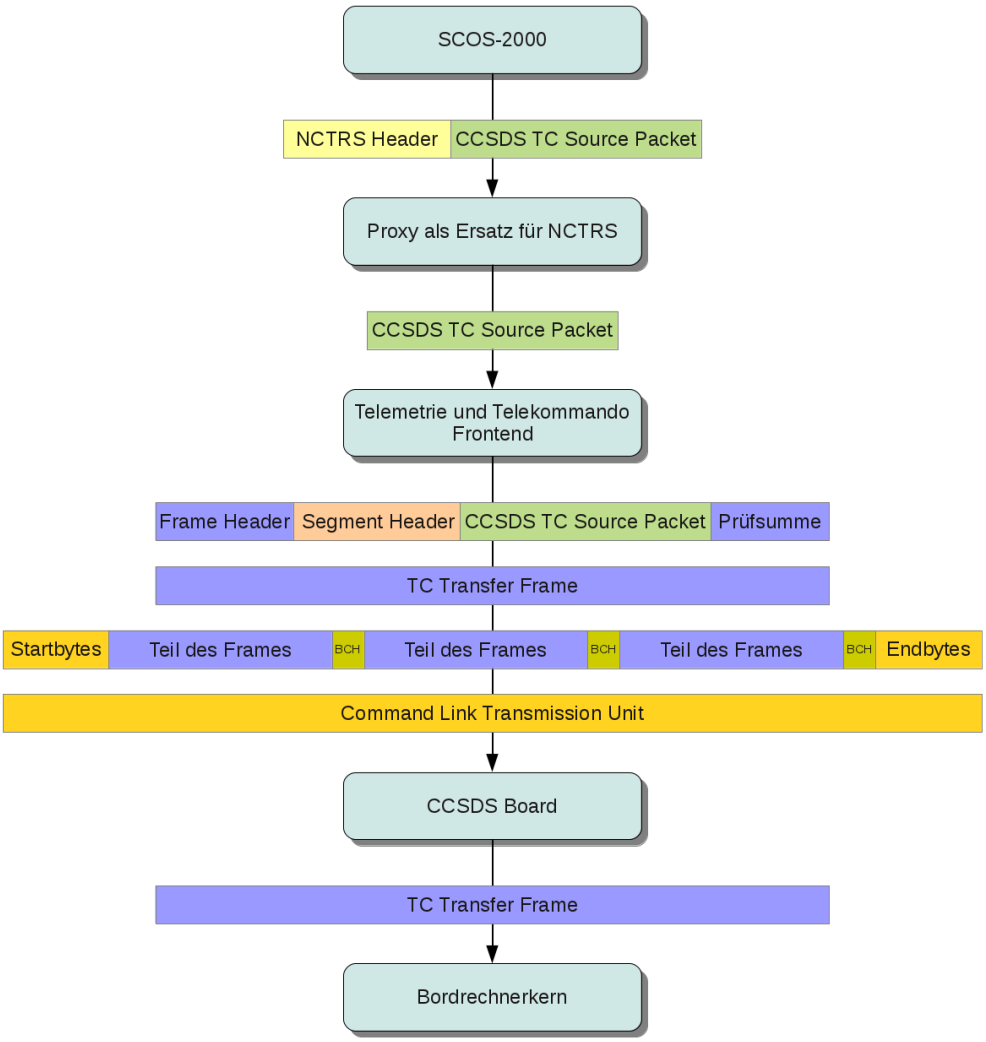


Abbildung 5.16: Konvertierungen bei der Übertragung von Telekommandos

nicht nur die korrekte Konvertierung der Telemetrie von der Bord- zur Bodenseite hin auf effiziente Weise getestet, sondern auch qualitative Aussagen durch das zyklische Senden getroffen werden. So konnten vom Telemetrie und Telekommando Frontend innerhalb weniger Minuten über 10.000 Idle Frames empfangen werden. Dabei wurden sämtliche Frames korrekt empfangen. Abbildung 5.17 zeigt ein entsprechendes Fenster des Frontends, das weitere Informationen wie beispielsweise die verwendete Vorwärtfehlerkorrekturmethode oder die aktuelle Übertragungsgeschwindigkeit enthält.

Abbildung 5.18 stellt den Empfang eines einzelnen Idle Frames dar. Dabei werden sämtliche Informationen des Frame Headers dekodiert dargestellt. Beipielsweise ist zu sehen, dass der virtuelle Kanal 7 gewählt war, der zur besseren Trennung im Missionskontrollsystem nur diesen Idle Frames vorenthalten ist. Auch über den Status und die Qualität des Frames wird Auskunft gegeben. Verwendete Dekodierungen

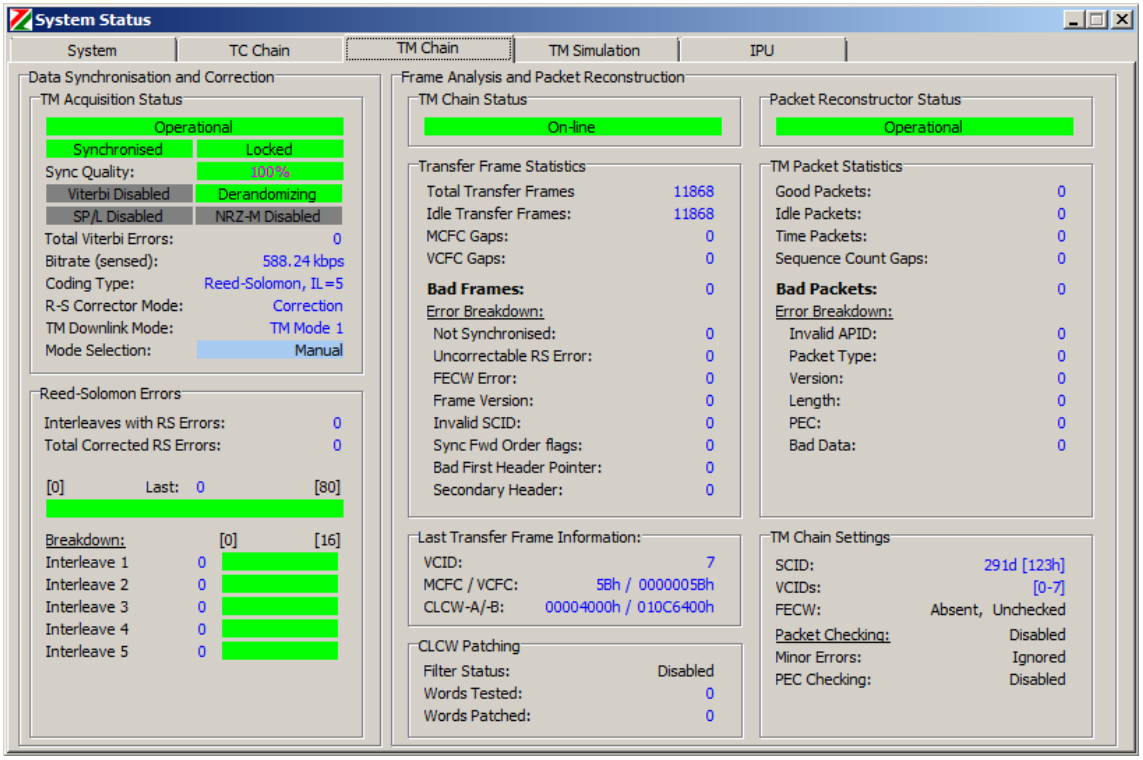


Abbildung 5.17: Empfang von Idle Frames am Telemetrie und Telekommando Front-end

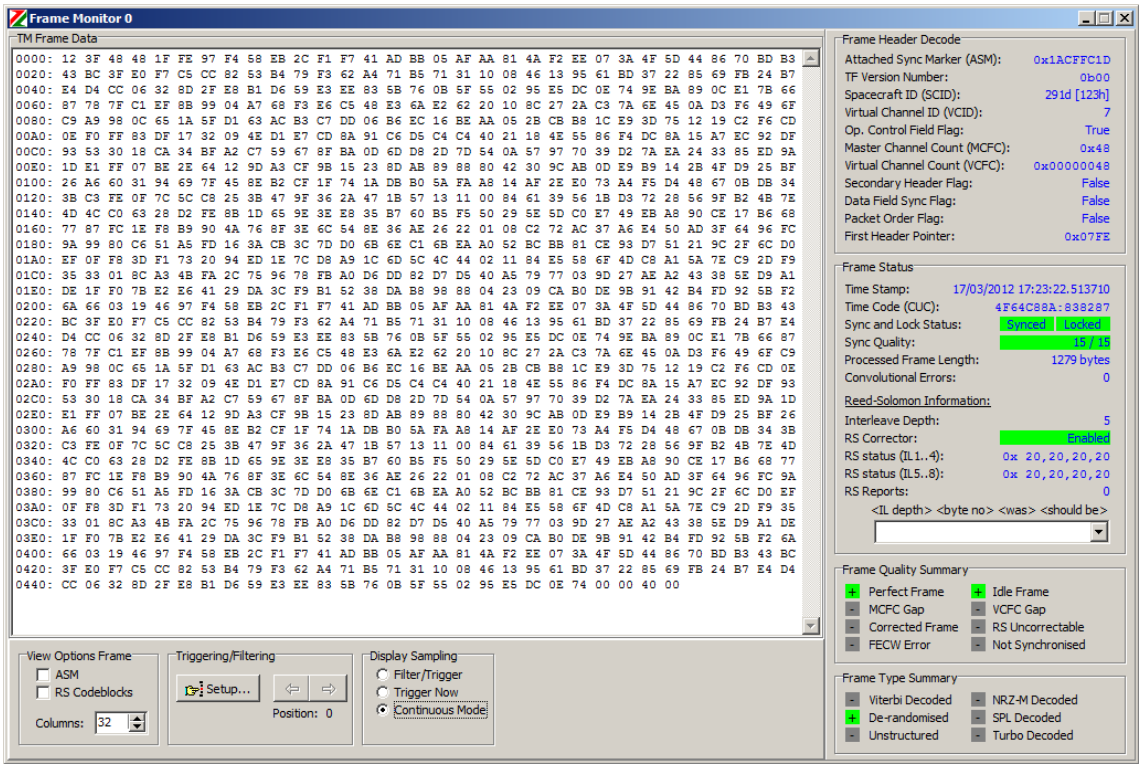


Abbildung 5.18: Empfang eines einzelnen Idle Frames

werden dargestellt.

Mit Konfiguration 2 wurde also die Ansteuerung des Bordrechners auf eine Weise ermöglicht, wie sie für dessen Betrieb übernommen werden kann. Durch die beschrie-

benen Tests wurde gezeigt, dass das Missionskontrollsystem, das Telemetrie und Telekommando Frontend, das CCSDS Board und der Bordrechnerkern kompatibel sind. Das schließt die Verwendung von Telekommandos und Telemetrie nach CCSDS ebenso ein, wie die korrekte Anwendung der Vorwärtsfehlerkorrektur und der verbundenen Schnittstellen. Durch die Anbindung des CCSDS Boards an den Bordrechnerkern konnte außerdem die Kompatibilität zwischen den Teilen des Bordrechners über SpaceWire verifiziert werden. Die korrekte Konvertierung von Telekommandos und Telemetrie zwischen Missionskontrollsystem und Bordrechnerkern konnte nach der Konfiguration aller dabei zur Anwendung kommenden Geräte ebenfalls gezeigt werden. Durch die zyklische Übertragung von über 10.000 Idle Frames wurde außerdem die Verlässlichkeit bei der Übertragung zwischen CCSDS Board und Telemetrie und Telekommando Frontend erfolgreich verifiziert. Sowohl die Tests der Telekommandostrecke wie auch die der Telemetriestrecke wurden erfolgreich abgeschlossen. Das Missionskontrollsystem ist damit einsatzbereit und zum Bordrechner kompatibel.

5.4 Funktionale Verifikation von Teilen der Bordsoftware

In Konfiguration 3 konnte damit begonnen werden erste zur Verfügung stehende Teile der Bordsoftware funktional zu verifizieren. Der Bordrechnerkern wurde anstatt mit dem echten I/O Board über den bereits in Konfiguration 1 zu anderen Zwecken verwendeten Ethernet SpaceWire Router mit dem in Kapitel 4.4.2 beschriebenen simulierten I/O Board und den daran hängenden simulierten Bordkomponenten verbunden, die ihrerseits wieder an die simulierte Weltraumumgebung gekoppelt sind. Abbildung 5.19 zeigt den Aufbau von Konfiguration 3.

Abbildung 5.20 zeigt den dafür genutzten und in Kapitel 4.4.2 beschriebenen I/O Broker nach dem Verbindungsaufbau zum Simulatormodell “I/O Board Frontend” und der Initialisierung der SpaceWire Verbindung. Durch Nutzung dieser Konfiguration konnte damit begonnen werden die in Kapitel 3.2 aufgezählten Elemente der am IRS entwickelten Software zu verifizieren, da sowohl die Verbindung zum Boden wie auch die zum Rest des Satelliten mit realen Daten versorgt werden. Für die Algorithmen zur Fehlererkennung müssen in die Simulation Fehler injiziert werden, was mit Hilfe einer dafür im Rahmen dieser Arbeit entwickelten Methode ohne tiefe Simulatorkenntnisse möglich ist.

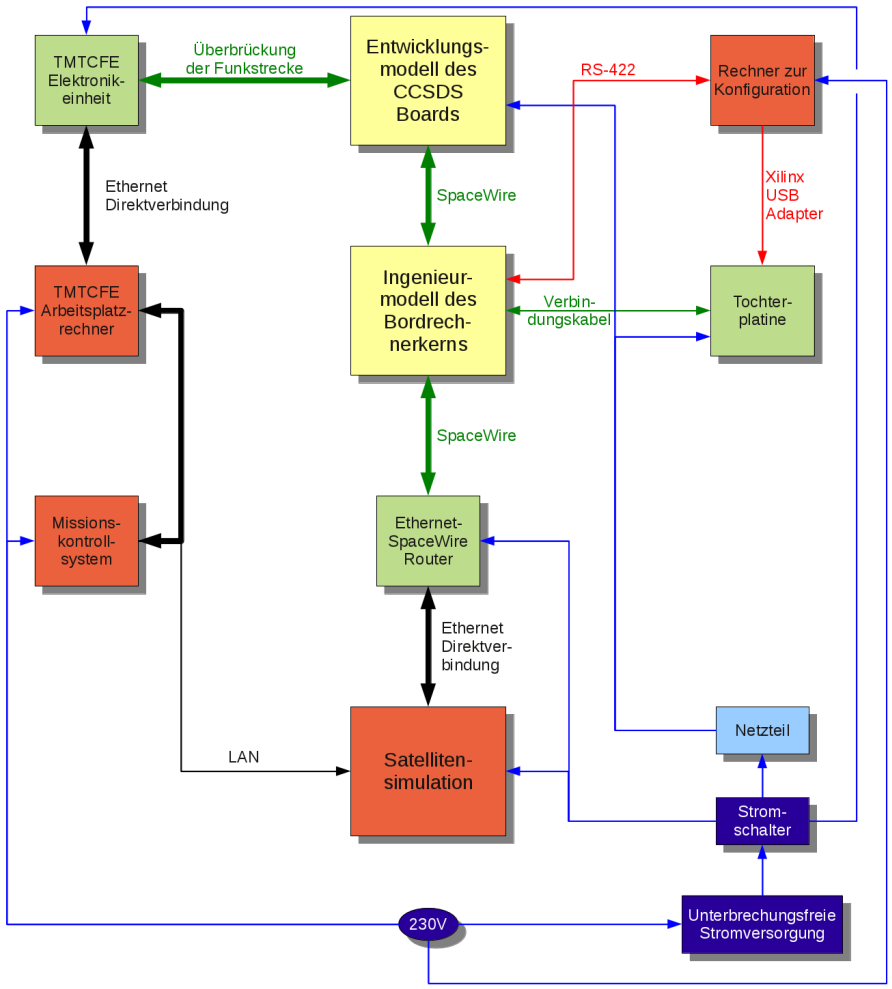


Abbildung 5.19: Konfiguration 3: Verifikationsumgebung für die Bordsoftware

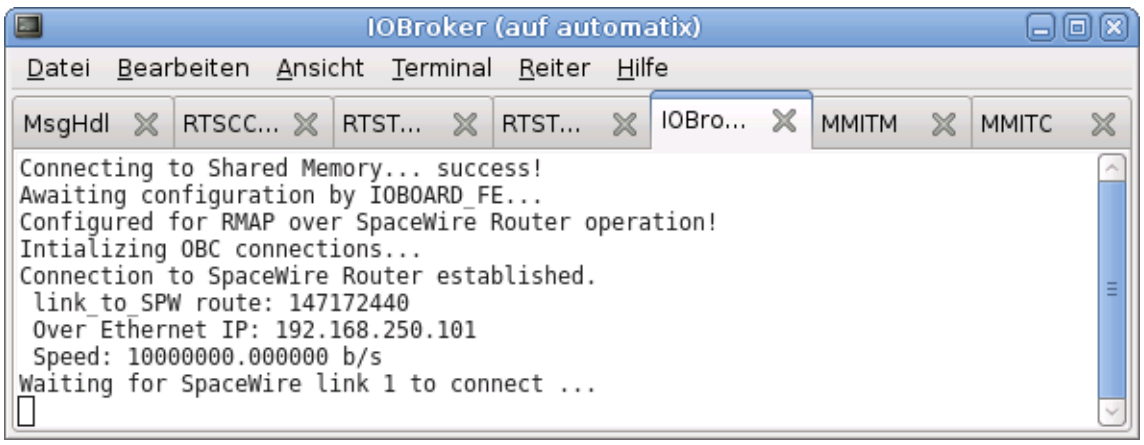


Abbildung 5.20: Verbindungsaufbau im I/O Broker

Zur Überwachung der Schaltzustände der simulierten PCDU wurde in [Winter 2011] die in Abbildung 5.21 dargestellte grafische Oberfläche entwickelt, mit der durch die farbliche Darstellung eine schnelle Erfassung möglich ist. Auch eine Umschaltung kann damit vorgenommen werden. Links in Abbildung 5.21 sind die Si-

cherungen und Schalter der simulierten PCDU zu sehen. Leitet eine Sicherung oder ein Schalter Strom, so ist sie oder er grün hinterlegt, andernfalls rot. Durch einen einfachen Klick kann der Schaltzustand geändert werden. Auf der rechten Seite sind weitere Details zum ausgewählten Element zu finden wie beispielsweise die anliegende Spannung. Mit dieser Oberfläche konnte zu Beginn der Bordsoftwareentwicklung auf die Kommunikation mit der simulierten PCDU verzichtet und benötigte Komponenten von Hand aktiviert oder deaktiviert werden.

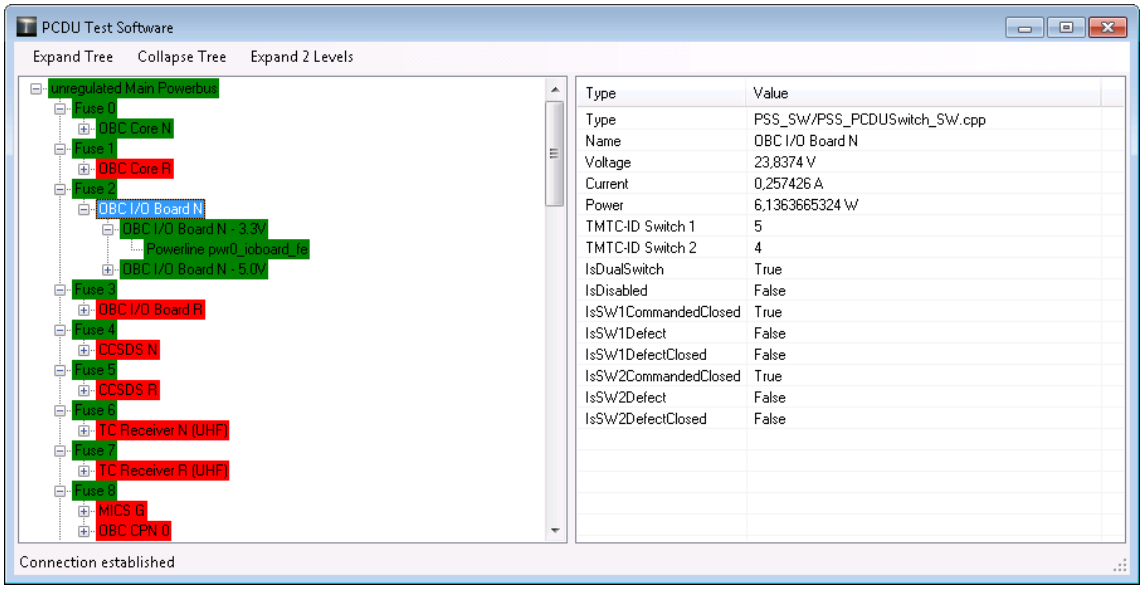


Abbildung 5.21: Überwachung der simulierten PCDU [Winter 2011]

Diese Konfiguration bot sogar Vorteile gegenüber einem Flatsat[4]. Werden beispielsweise Kontrollalgorithmen für die Lageregelung auf einem Flatsat getestet, müssen alle Sensoren koordiniert stimuliert werden. Auch die Bewegungsänderung durch die Aktuatoren hat darauf einen Einfluss. Durch Verwendung des Simulators konnte auf numerische Modelle zurückgegriffen werden, die die koordinierte Stimulation überflüssig machten. Trotzdem müssen im Lauf der Integration reale Sensoren des Flying Laptop einzeln stimuliert werden, um deren generelle Funktionalität zu testen. Zuletzt hat diese Konfiguration gegenüber einer Verifikation am Flatsat den Vorteil, dass Randbedingungen per Knopfdruck wiederhergestellt werden können. Abbildung 5.22 zeigt den aufgebauten Flatsat des ESA-Satelliten Cryosat-1.

Nach dem im vorangegangenen Kapitel beschriebenen Empfang eines Telekommandos auf dem CCSDS Board und dessen Speicherung auf demselben muss der Bordrechnerkern dieses abrufen, was in Abbildung 5.23 zu sehen ist. Jedes Byte des Telekommandos wird auf dem Speicher des CCSDS Boards mit einem zusätzlichen

[4] Als Flatsat wird der flache, elektrisch funktionale Aufbau eines Satelliten bezeichnet. Ein Flatsat wird für Kommunikationstests zwischen Bordkomponenten und zur funktionalen Verifikation eines Satelliten genutzt.

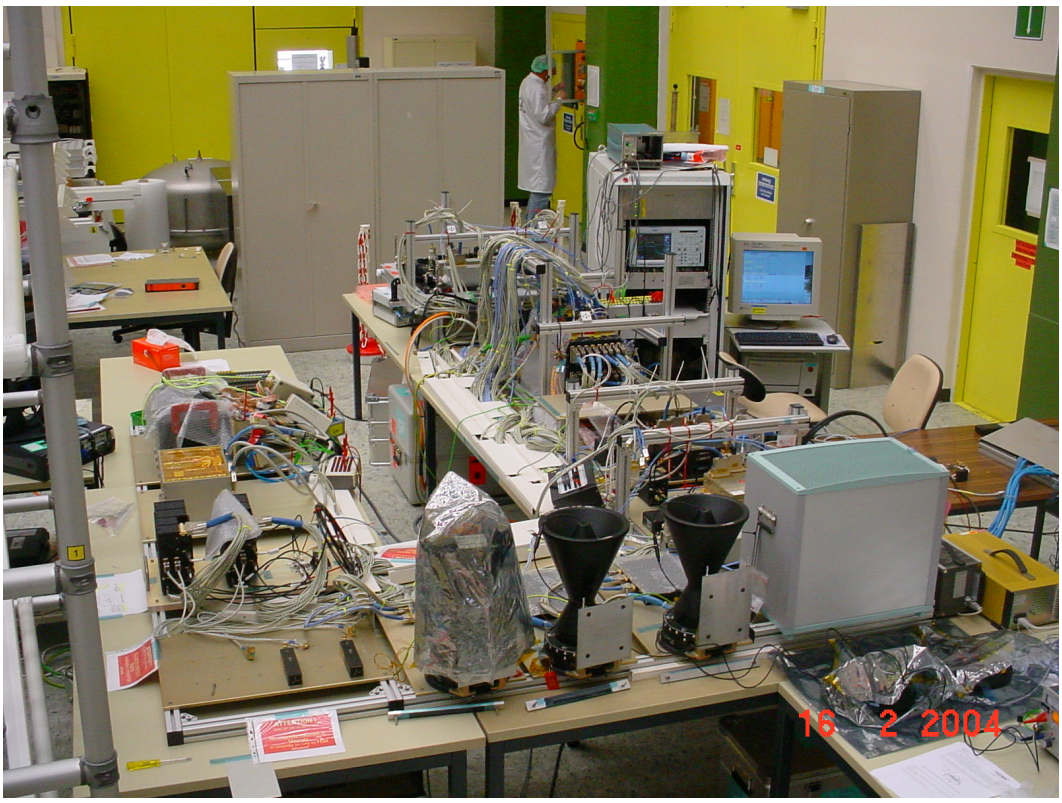


Abbildung 5.22: Flatsat des Satelliten Cryosat-1 [Eickhoff 2009]

Informationsbyte versehen. Das erste Byte eines Telekommandos erhält das Informationsbyte 1, das letzte erhält 2 und alle Bytes dazwischen werden mit 0 versehen. Aus dem hypothetischen Telekommando 55 55 55 55 wird also auf dem Speicher des CCSDS Boards 55 01 55 00 55 00 55 02. Damit der Bordrechnerkern weiß, wo er mit dem Lesen beginnen muss, gibt es einen dedizierten Zeiger[5], der diese Information enthält und der nach der Leseoperation entsprechend angepasst werden muss [Gaisler 2011]. Ein Telekommando kann die direkte Kommandierung einer Bordkomponente enthalten, normalerweise kommt diese Möglichkeit aber nur in Ausnahmefällen zum Einsatz. Stattdessen werden beispielsweise Satellitenmodi kommandiert, die die Regelalgorithmen des Bordrechners dazu veranlassen, die Bordkomponenten anzusteuern. Das Sequenzdiagramm bildet diese Logik mit Schritt 2 entsprechend ab. Schritte 3 bis 7 geschehen in aller Regel mehrfach. Es werden ja verschiedene Bordkomponenten wie Aktuatoren periodisch angesteuert. Das Gleiche gilt für die Schritte 8 bis 11, da beispielsweise diverse Sensorwerte ebenfalls periodisch abgerufen werden. Diese Werte werden wieder im Bordrechnerkern prozessiert. Damit können Telemetriedaten erzeugt werden, die entweder auf einem Speicher des Bordrechnerkerns abgelegt und bei einem Kontakt zu einer Bodenstation gesendet werden oder

[5]Zeiger ist ein fester Begriff der Informatik. Ein Zeiger enthält keine Daten, sondern eine Speicheradresse.

bei einem entsprechenden Überflug direkt an die Bodenstation übertragen werden können. Zur Übertragung ist in ein entsprechendes Register des CCSDS Boards zu schreiben, von denen es insgesamt vier gibt [Gaisler 2011].

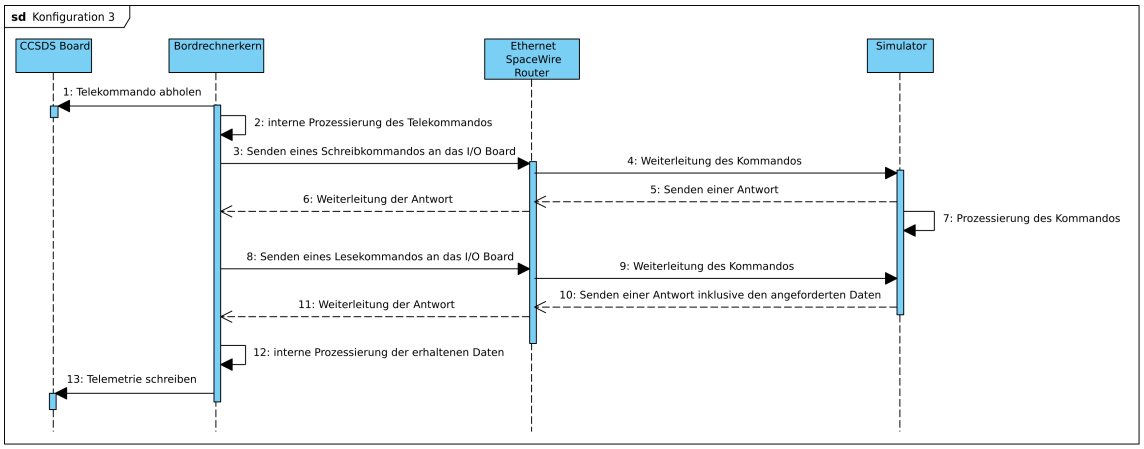


Abbildung 5.23: UML Sequenzdiagramm für Konfiguration 3

Konfiguration 3 kann von Entwicklern der Bordsoftware zur Analyse von neuem und verändertem Quelltext genutzt werden. Diese Analyse geschieht in zwei Schritten: Zunächst kann der Quelltext auf dem Arbeitsplatzrechner, der zur Entwicklung genutzt wird, kompiliert und via lokalem Netzwerk mit dem Simulator verbunden werden [Mohr 2011]. Hier dient also der Arbeitsplatzrechner als Ersatz für den realen Bordrechnerkern. Dabei wird bereits die korrekte Formatierung der Befehle zwischen Bordsoftware und I/O Board verwendet, nur die elektrische Schnittstelle ist noch nicht korrekt. Hier kommt eine Ethernet Verbindung zum Einsatz, was die Nutzung des lokalen Netzwerks ermöglicht. Für die Kommunikation mit Ethernet stehen verschiedene Standards zur Verfügung. Bei manchen wird vom Empfänger eine Antwort geliefert, auf die der Sender vor der weiteren Abarbeitung des Quelltexts wartet. Um Verzögerungen im Simulator zu verhindern, wird hier das Kommunikationsprotokoll UDP[6] genutzt, bei dem nicht auf eine Antwort gewartet wird. Der gesamte Schritt ist, wie in Abbildung 5.24 bezeichnet, als Präverifikation zu betrachten. Um sicherzustellen, dass der Quelltext sich auch für den LEON3-FT Prozessor kompilieren lässt und mit der SpaceWire Schnittstelle funktioniert, wird in einem zweiten Schritt der in Konfiguration 3 eingebettete reale Bordrechnerkern als rechnende Einheit genutzt. Diese Methodik bietet den Vorteil, dass mehrere Entwickler gleichzeitig ihre Software durch Anschluss an den Simulator analysieren können, da mehrere Entwicklungs- und Simulationsrechner, aber nur ein Ingenieurmodell des Bordrechnerkerns und ein Ethernet SpaceWire Router zur Verfügung stehen.

[6] Abkürzung für User Datagram Protocol. Demgegenüber steht das häufiger genutzte Transmission Control Protocol.

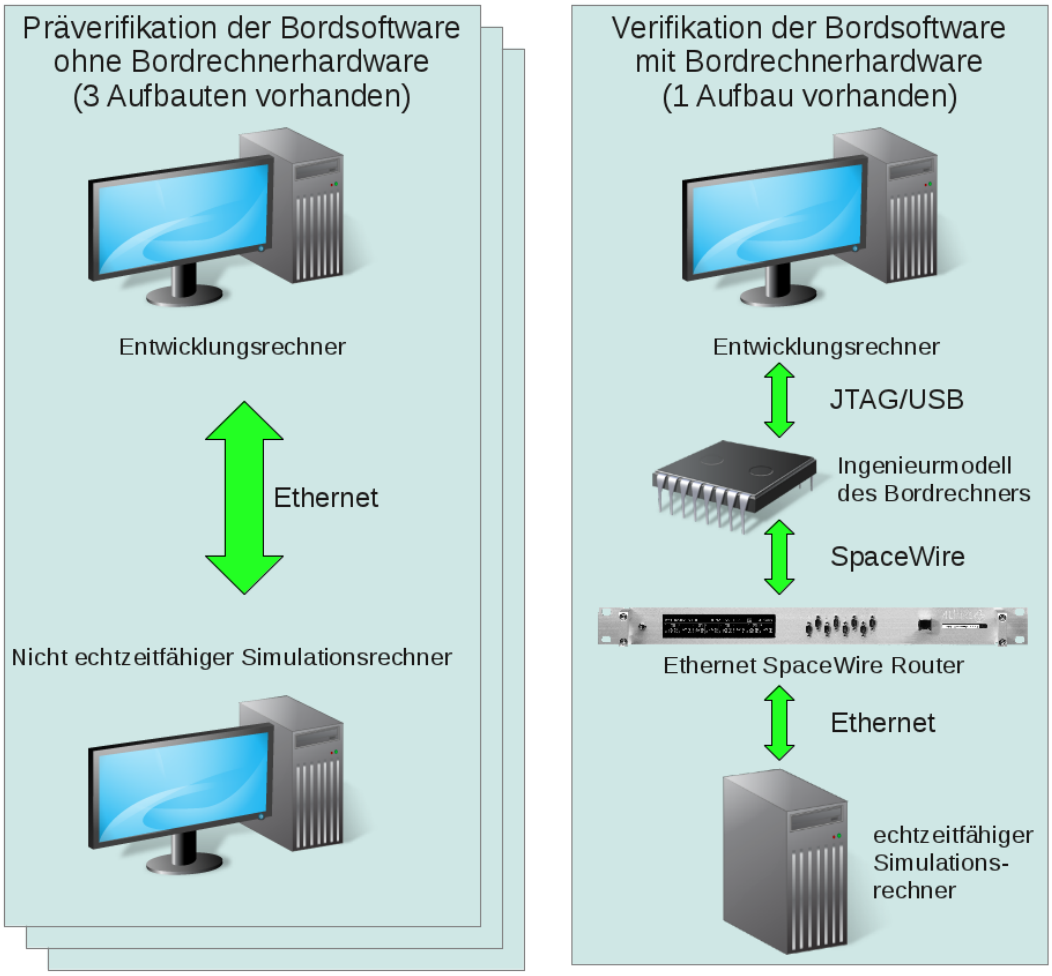


Abbildung 5.24: Verifikationsaufbauten für die Bordsoftware

Sind die Entwickler der Bordsoftware schließlich der Meinung, dass eine Funktion der Bordsoftware fertig gestellt ist, wird der kompilierte Quelltext im bei der Entwicklung genutzten Versionsverwaltungssystem mit einer Kennzeichnung versehen. Damit kann er später jederzeit per Mausklick wiederhergestellt werden. Diese Version des Quelltextes wird nun kompiliert und den Verifikationsingenieuren zur Verfügung gestellt. Außerdem kann mit der Kennzeichnung gegen Ende der Entwicklung der Bordsoftware festgestellt werden, ob sich in den für die verifizierte Funktion genutzten Dateien Änderungen ergeben haben. Falls das zutrifft, muss die entsprechende Funktion erneut verifiziert werden. Im Anschluss ist ein Vergleich der Ergebnisse durchzuführen.

Bei der Durchführung der funktionalen Verifikation muss der Verifikationsingenieur prüfen, ob alle Anforderungen an die Funktion erfüllt werden. Abweichungen, Unstimmigkeiten und Fehlfunktionen sind festzuhalten und in den weiter unten erläuterten Bericht einzufügen. War der Verifikationslauf fehlerhaft, so sind die Mängel

von den Entwicklern der Bordsoftware zu beheben und eine korrigierte Version der Bordsoftware ist zur Verfügung zu stellen. War der Verifikationslauf erfolgreich, so ist der entsprechende Quelltext als funktionsfähig zu betrachten und darf nicht mehr verändert werden.

Essentiell für die funktionale Verifikation der Bordsoftware ist die Archivierung der Ergebnisse solcher Läufe. Dazu gehört neben den Informationen hinsichtlich Bordsoftware und Simulator auch der Verlauf der Verifikation selbst. Dafür wird eine am IRS entwickelte [Fritz 2008] Software genutzt, die alle Ergebnisse und Randbedingungen archiviert. Die Software wird über eine im Rahmen dieser Arbeit entwickelte Website angesteuert, die sich von jedem Webbrowser aus öffnen lässt. Über diese Eingabemaske wird außerdem automatisch ein Bericht der Verifikation im gängigen pdf-Dateiformat erzeugt, welcher anschließend in einem für alle Beteiligten zugänglichen Ordner gespeichert wird.

Abbildung 5.25 skizziert schematisch den beschriebenen Verlauf der Bordsoftwareverifikation aus Sicht des Verifikationsingenieurs.

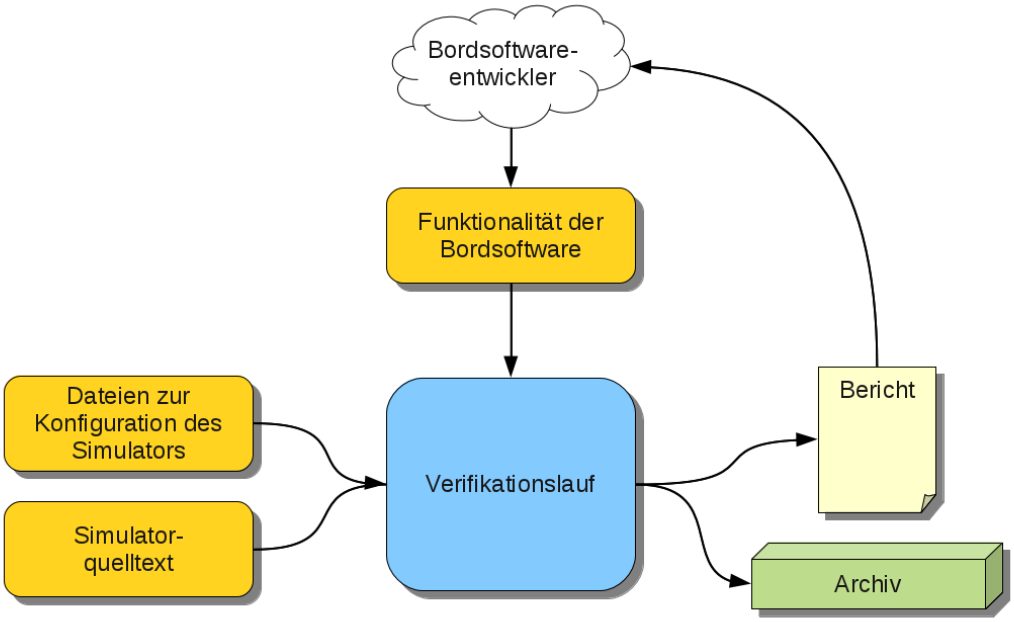


Abbildung 5.25: Prinzip der Bordsoftwareverifikation

Zur Veranschaulichung dieser Methodik soll ein Beispiel dienen. In der Bordsoftware gibt es Prozesse, die dafür verantwortlich sind, dedizierte komponentenspezifische Daten aus dem Datenbestand der Bordsoftware in ein Format zu konvertieren, das die jeweilige Bordkomponente versteht und umgekehrt. Somit wird die Kommunikation zwischen Bordsoftware und Komponente erst möglich. Folglich muss ein solcher Prozess der Bordsoftware das entsprechende Protokoll und alle Komman-

dos zu und Antworten von der ihr zugewiesenen Komponente prozessieren können. Abbildung 5.26 zeigt die Kommunikation des Bordrechnerkerns mit den simulierten Magnetometern und Magnettorquern sowie die entsprechende Darstellung des I/O Brokers. Links sind die vom Bordrechnerkern gesendeten Kommandos an die Magnettorquer sowie die Gültigkeitsprüfung der Magnetometerwerte zu sehen. Im I/O Broker rechts kann der Empfang der Kommandos an die Magnettorquer sowie eine dazugehörige Empfangsbestätigung an den Bordrechnerkern betrachtet werden. Durch den Test aller zu den Magnetometern und Magnettorquern gehörenden Kommandos und entsprechend aller zurück gesendeten Antworten und der Telemetrie konnte die Kommunikation mit den exemplarisch gewählten Bordkomponenten erfolgreich getestet werden.

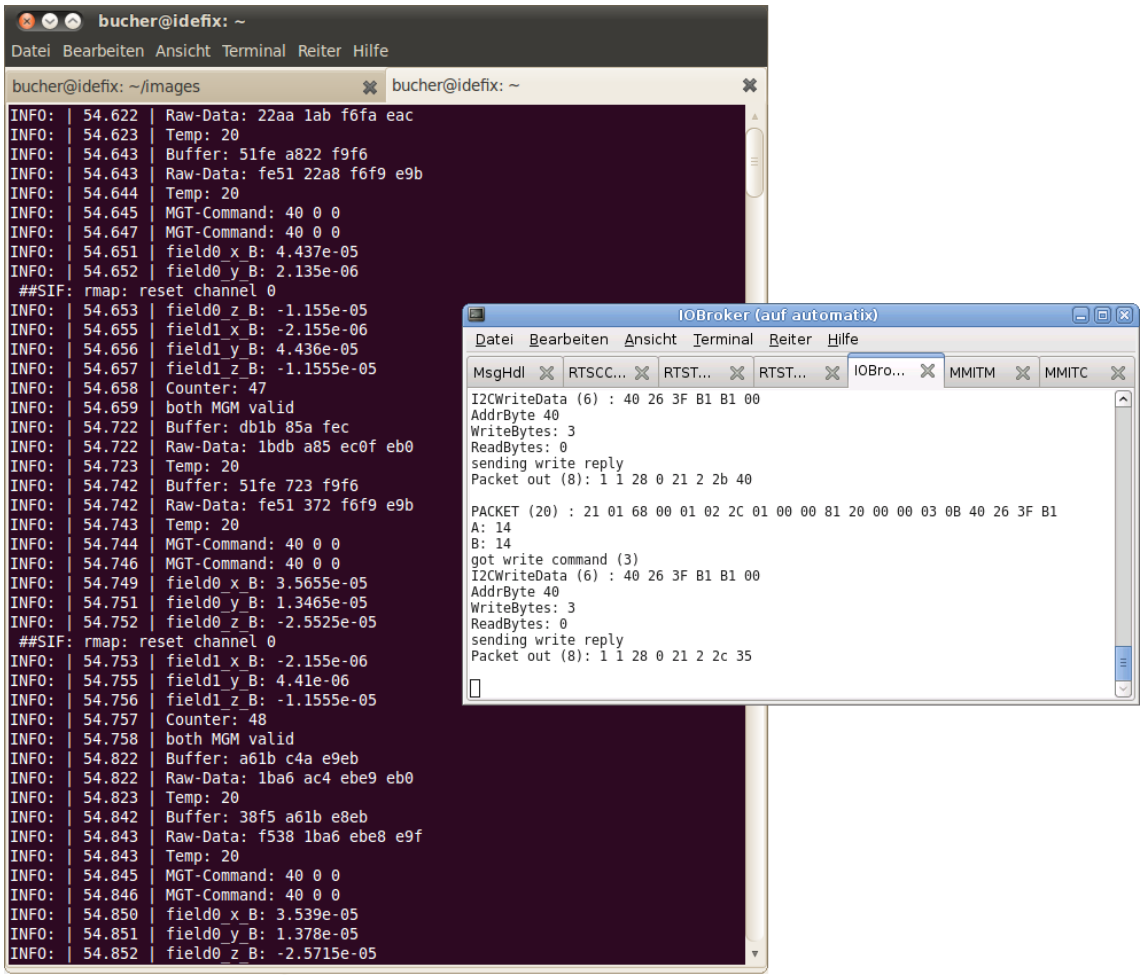
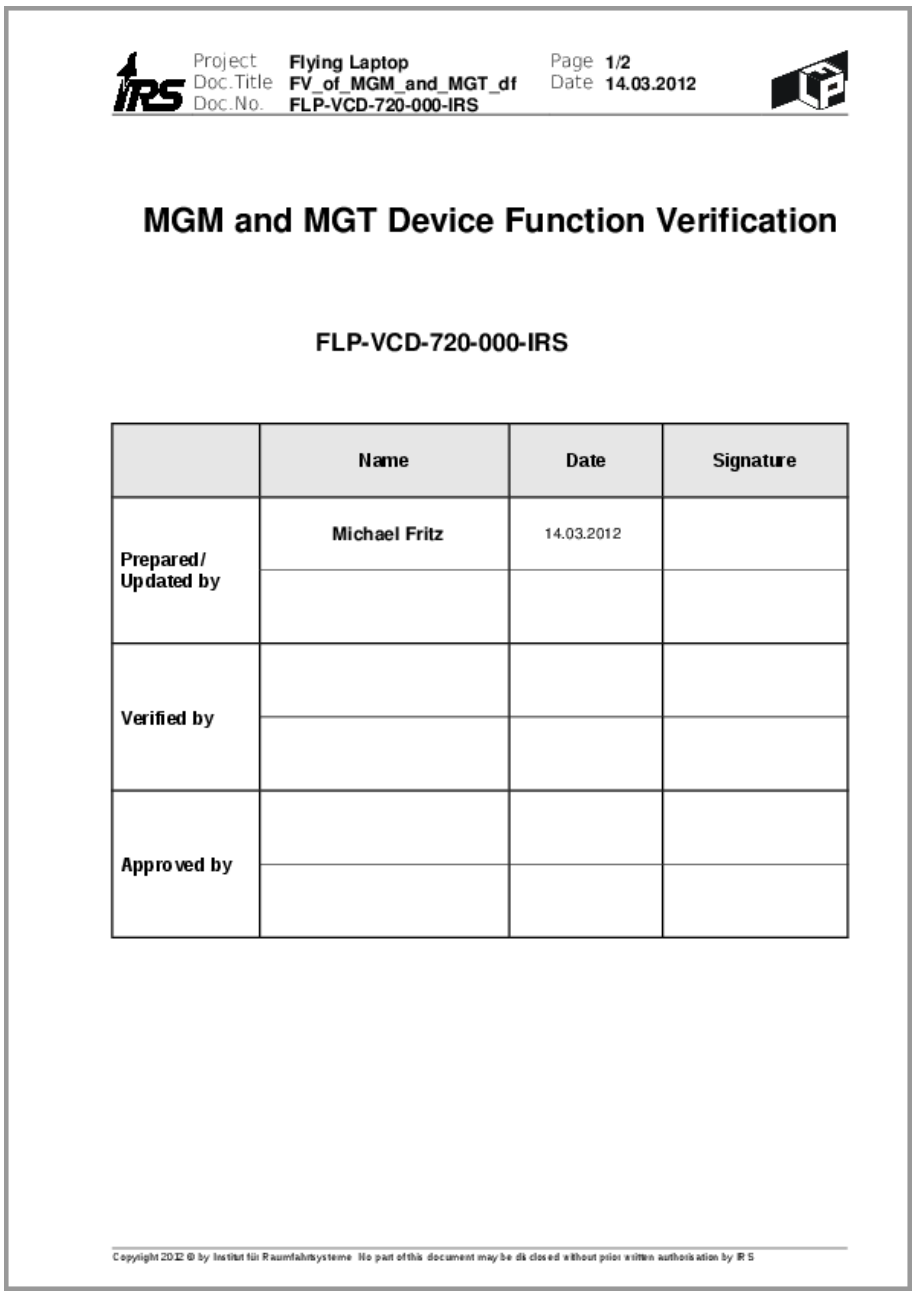


Abbildung 5.26: Kommunikation zwischen Bordrechnerkern und Simulator

Im Anschluss konnten die Ergebnisse unter Verwendung der oben erläuterten Website archiviert und dokumentiert werden. Abbildung 5.27 zeigt das Titelblatt eines solchen automatisch generierten Dokuments. Das dazu ausgearbeitete Konzept wurde in Teilen realisiert, aber noch nicht vollständig umgesetzt. Es kann für alle Zwecke angepasst werden, für die eine häufige automatische Erzeugung von Dokumenten



Project Flying Laptop
Doc.Title FV_of_MGM_and_MGT_df
Doc.No. FLP-VCD-720-000-IRS
Page 1/2
Date 14.03.2012

MGM and MGT Device Function Verification

FLP-VCD-720-000-IRS

	Name	Date	Signature
Prepared/ Updated by	Michael Fritz	14.03.2012	
Verified by			
Approved by			

Abbildung 5.27: Deckblatt eines automatisch erstelltes Verifikationsberichts

sinnvoll ist, wie zum Beispiel Tests auf Komponentenebene.

Diese Konfiguration kann über den oben beschriebenen Einsatzzweck hinaus nach dem Start des Satelliten für die Verifikation von Aktualisierungen der Bordsoftware genutzt werden, da sie ohne Flugmodelle funktionsfähig ist und entsprechende Randbedingungen im Orbit durch Simulation wiedergegeben werden können. Die Verifikation einer aktualisierten Bordsoftware ist notwendig, um das Risiko eines Satellitenverlusts zu minimieren. So kann beispielsweise Quelltext aus Gründen der Entwicklungsgeschwindigkeit auskommentiert, also deaktiviert und die erneute Einkommentierung vergessen worden sein. Darüber hinaus eignet sich diese Konfiguration für das Training von Operatoren, da Bedienungsfehler nicht den realen Satelliten

beeinflussen [Eickhoff 2009]. Es kann auch evaluiert werden, welche Benutzerrechte im Missionskontrollsystem für welchen Operator notwendig sind und welche Rechte den Satelliten beeinträchtigen können. Die Benutzerrechte für Studenten müssen beispielsweise so gewählt werden, dass eine Beeinträchtigung des Satelliten nicht stattfinden kann.

In Konfiguration 3 wurde also der Bordrechnerkern mit der ausgewählten und angepassten Simulationsumgebung verbunden. Wie in diesem Kapitel beschrieben, war auch diese Anbindung erfolgreich. Damit wurde getestet, dass der Bordrechnerkern mit den Komponentenmodellen im Simulator kommunizieren kann. Die Simulationsumgebung konnte und kann somit zur Verifikation der Bordsoftware verwendet werden.

5.5 Kompatibilitätstests mit Bordkomponenten

Die vierte Konfiguration hat einen Aufbau analog zu Konfiguration 3. Allerdings wurde hier das echte I/O Board und echte Bordkomponenten angesteuert. Da das echte I/O Board zuletzt zur Verfügung stand, konnte diese Konfiguration erst spät realisiert werden. Damit konnte begonnen werden zu verifizieren, ob

- Bordsoftware und Simulator (vgl. Konfiguration 3) möglicherweise gleiche Fehler beinhalten, was zum Beispiel durch eine fehlerhafte Spezifikation verursacht werden kann. So können bei einem Reaktionsrad einfache Vorzeichenfehler in der Spezifikation zur falschen Drehrichtung und damit einem Destabilisieren des Satelliten führen. Solche Fehler können in Konfiguration 3 nicht entdeckt werden.

- Außerdem kann verifiziert werden, ob die echten Bordkomponenten Abweichungen zur Spezifikation haben. Als Beispiel sei genannt, dass Kommandos an die Komponente falsch oder gar nicht interpretiert werden.

- Zuletzt kann so die Kompatibilität aller Schnittstellen vom Missionskontrollsystem über den Bordrechner hin zu Komponenten gezeigt werden. Auf diese Weise können Differenzen zwischen simulierten und realen Schnittstellen aufgedeckt werden.

Der Aufbau von Konfiguration 4 ist in Abbildung 5.28 dargestellt. Die PCDU diente als Beispiel für die angeschlossene Bordkomponente. Theoretisch können auch mehrere Bordkomponenten gleichzeitig angeschlossen werden. Da jedoch Platz und Netzgeräte begrenzt und die Interaktion von Bordkomponenten in dieser Umgebung nicht

vorgesehen war, wurde darauf verzichtet. Abbildung 5.28 stellt auch dar, dass sich nicht nur die Protokolle auf der Strecke vom Missionskontrollsystem zur Bordkomponente ändern, sondern auch die elektrischen Schnittstellen. Diese Konfiguration kann später so als Grundlage für den Flatsat verwendet werden, was in Kapitel 7.3 nochmal aufgegriffen wird.

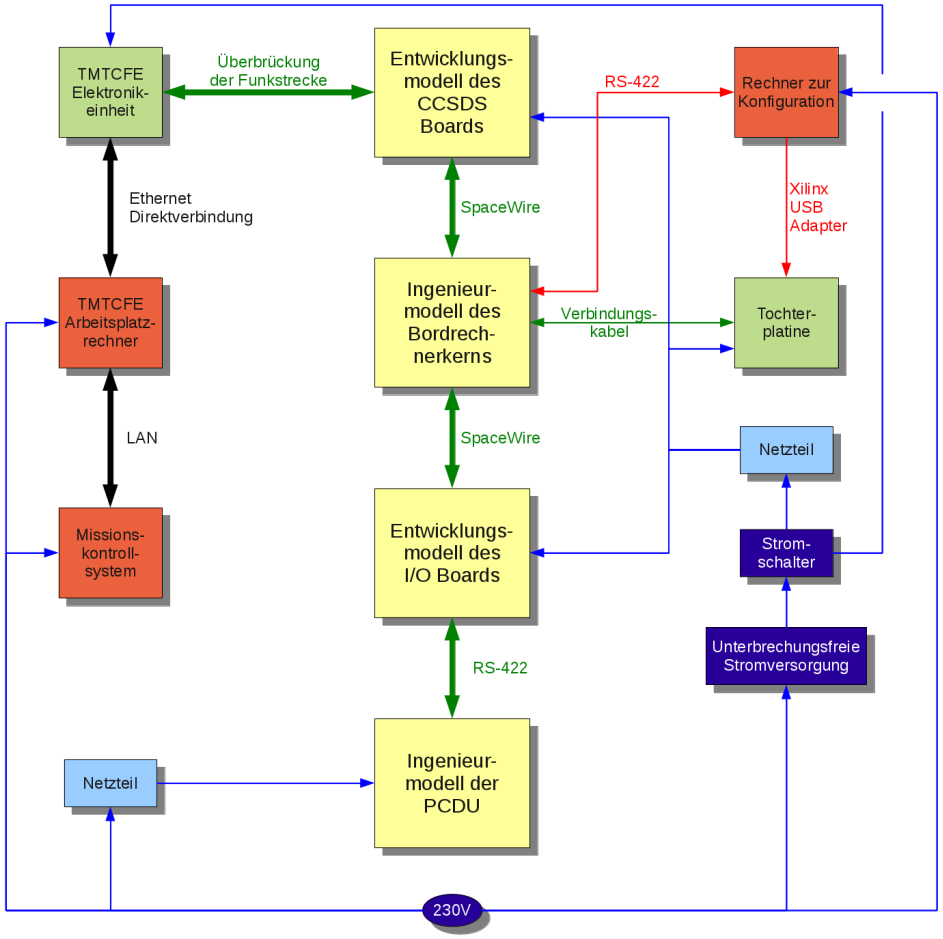


Abbildung 5.28: Verifikationsumgebung mit der realen Kommandierungskette

Die Kommunikation zwischen SCOS-2000 und CCSDS Board funktionierte erneut so wie in Abbildung 5.9 dargestellt. Die Kommunikation zwischen CCSDS Board und Bordkomponenten ist in Abbildung 5.29 dargestellt. Wie auch in Konfiguration 3 holt sich der Bordrechnerkern ein Telekommando vom CCSDS Board ab und prozessiert es zunächst. Anschließend wird ein Schreibkommando gesendet, nun aber nicht an das simulierte, sondern an das reale I/O Board. Da dieses auf einem FPGA und nicht auf einer CPU aufsetzt, kann es zeitgleich Daten senden. Deswegen kann nicht bestimmt werden, ob die Bestätigung des Empfangs oder die Weiterleitung des Schreibkommandos generell zuerst erfolgt. Im Diagramm sind beide mit Nummer 4 versehen. Das Schreibkommando wird nun von der adressierten

Bordkomponente prozessiert. Wird eine Antwort wie beispielsweise ein Sensorwert angefordert, so wird diese nun von der Bordkomponente zurück ans I/O Board geschrieben. Die Antwort muss sich der Bordrechnerkern mit einem Lesekommando vom I/O Board abholen. Da auf Seite der Bordkomponente zwischen dem Empfang des Schreibkommandos und dem Erzeugen einer Antwort typischerweise mindestens einige Millisekunden vergehen, muss das in der Bordsoftware durch das in Kapitel 3.2 vorgestellte Scheduling berücksichtigt werden. Schließlich kann die Antwort im Bordrechnerkern prozessiert werden, woraus Telemetriedaten resultieren können, die dann wieder auf das CCSDS Board geschrieben werden.

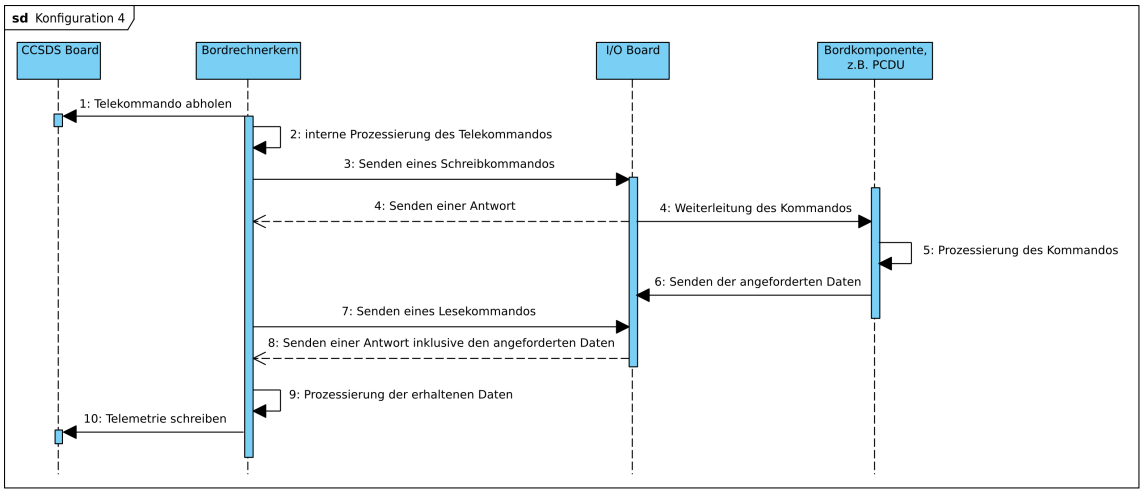


Abbildung 5.29: UML Sequenzdiagramm für Konfiguration 4

Mit den damit durchgeführten Tests wurde gezeigt, dass die Schnittstellen des Bordrechners mit denen der Bordkomponenten sowie die zur Kommunikation verwendeten Protokolle kompatibel sind. Diese Tests konnten noch nicht für alle Bordkomponenten abgeschlossen werden, da auf dem Entwicklungsmodell des I/O Boards nicht alle vorgesehenen Schnittstellen vorhanden waren. Auch in der Bordsoftware waren noch nicht alle dafür benötigten Elemente vorhanden. Anhand der bereits getesteten PCDU wird nachfolgend dargestellt, wie solche Tests prinzipiell ablaufen.

Wie in Abbildung 5.30 dargestellt, wurde dafür zunächst das Ingenieurmodell der PCDU mit der vom Hersteller gelieferten Testsoftware vom Verantwortlichen des Subsystems getestet. Dabei wurde die PCDU mit einem herkömmlichen Laptop verbunden. Ein spezielles Kabel ermöglicht dabei die Umsetzung der USB-Schnittstelle des Laptops auf die RS-422 Schnittstelle der PCDU. Auf diese Weise können alle für die PCDU vorgesehenen Kommandos gesendet und entsprechend Telemetrie empfangen werden.

Je nach Komponente ist auf andere Art und Weise zu überprüfen, ob die Kommandos tatsächlich ausgeführt wurden. Bei der PCDU konnte dies relativ einfach

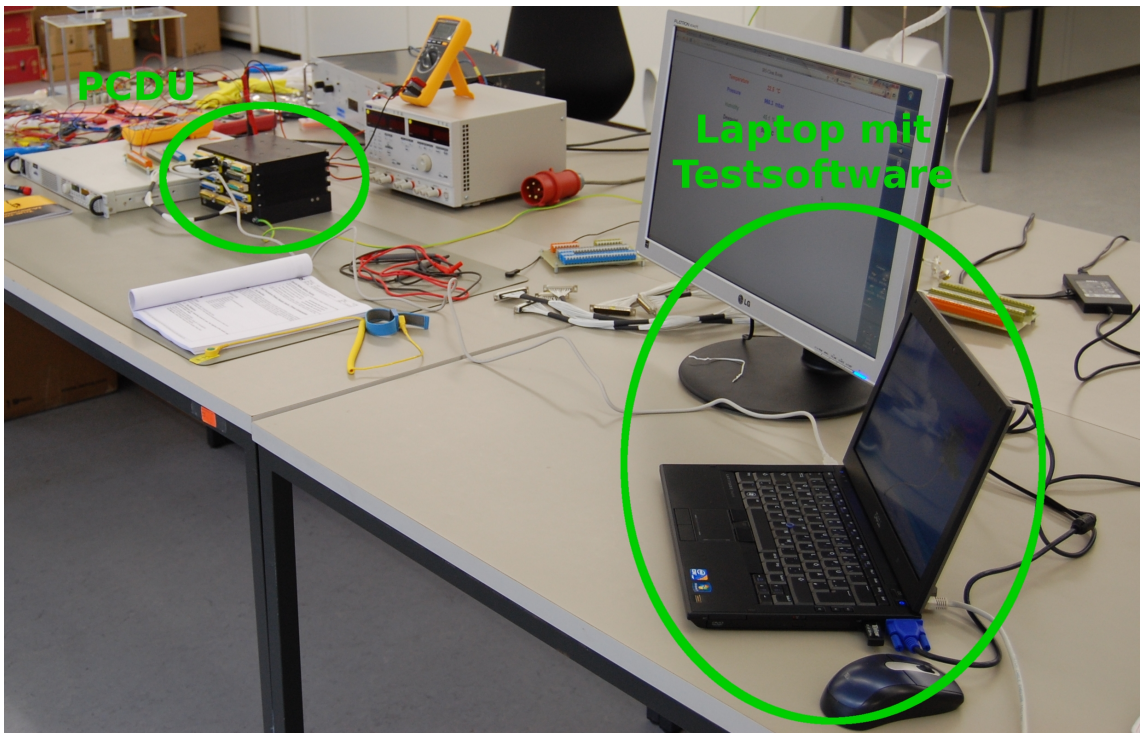


Abbildung 5.30: Test der PCDU auf Komponentenebene

durchgeführt werden, da die Kommandos darauf beschränkt sind Stromschalter zu betätigen und Sicherungen zurück zu stellen. Es musste also überprüft werden, ob die an den entsprechenden Ausgängen von der PCDU zur Verfügung gestellten Spannungen konform zu den gesendeten Kommandos waren. Dafür wurden Kabelklemmen mit den Ausgängen der PCDU durch passende Stecker verbunden. Eine dieser Kabelklemmen ist in Abbildung 5.31 dargestellt. Daran konnten mit einem Multimeter die anliegenden Spannungen gemessen werden.

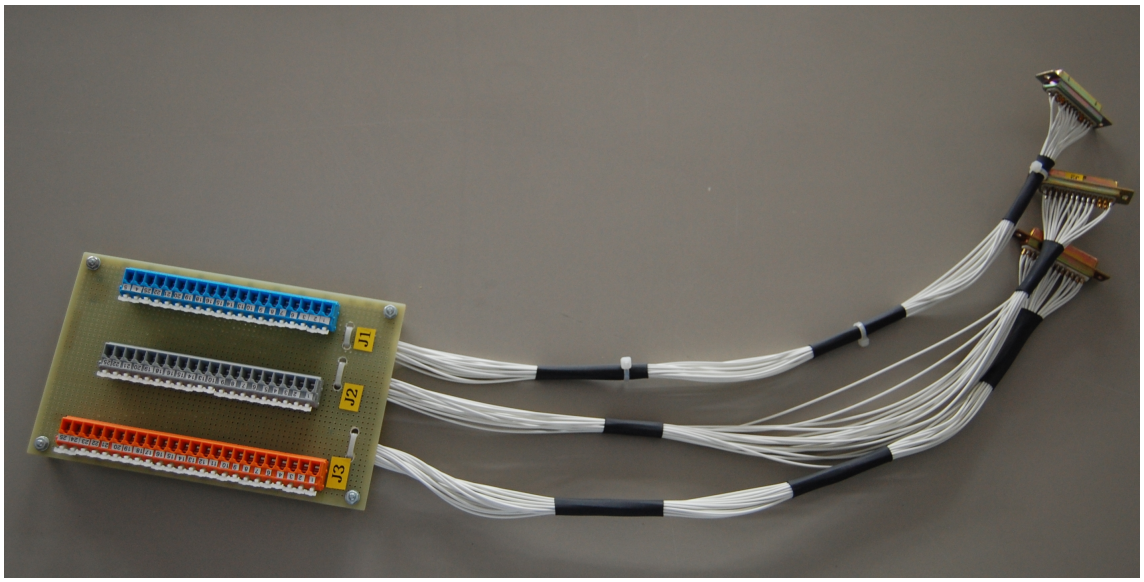


Abbildung 5.31: Kabelklemme für PCDU Tests

Da jedoch sowohl die PCDU wie auch die Testsoftware vom Hersteller entwickelt wurden, war im Rahmen dieser Arbeit zu testen, ob die PCDU die vom IRS zur Verfügung gestellten Spezifikationen an der Schnittstelle zum Bordrechner erfüllt. Abbildung 5.32 zeigt den Test der regulären Kommandos, bei dem die PCDU über das I/O Board mit dem Bordrechnerkern verbunden wurde. In dieser Konfiguration konnte auch die Telemetrie der PCDU an den Bordrechnerkern getestet werden. Alle Tests wurden in einem Dokument schrittweise definiert und festgehalten, analog zum entsprechenden Dokument für die oben beschriebene Integration. Durch diese Tests konnten Fehler in der PCDU gefunden werden. So waren zum Beispiel Prüfsummen aufgrund von Ungenauigkeiten in der Spezifikation nicht korrekt definiert und Telemetrie fehlerhaft. Nach der Rückmeldung an den Hersteller, dem Erhalt einer überarbeiteten Software für den Mikrocontroller der PCDU und der Installation derselben mussten die Tests nochmal gefahren werden und waren erfolgreich.

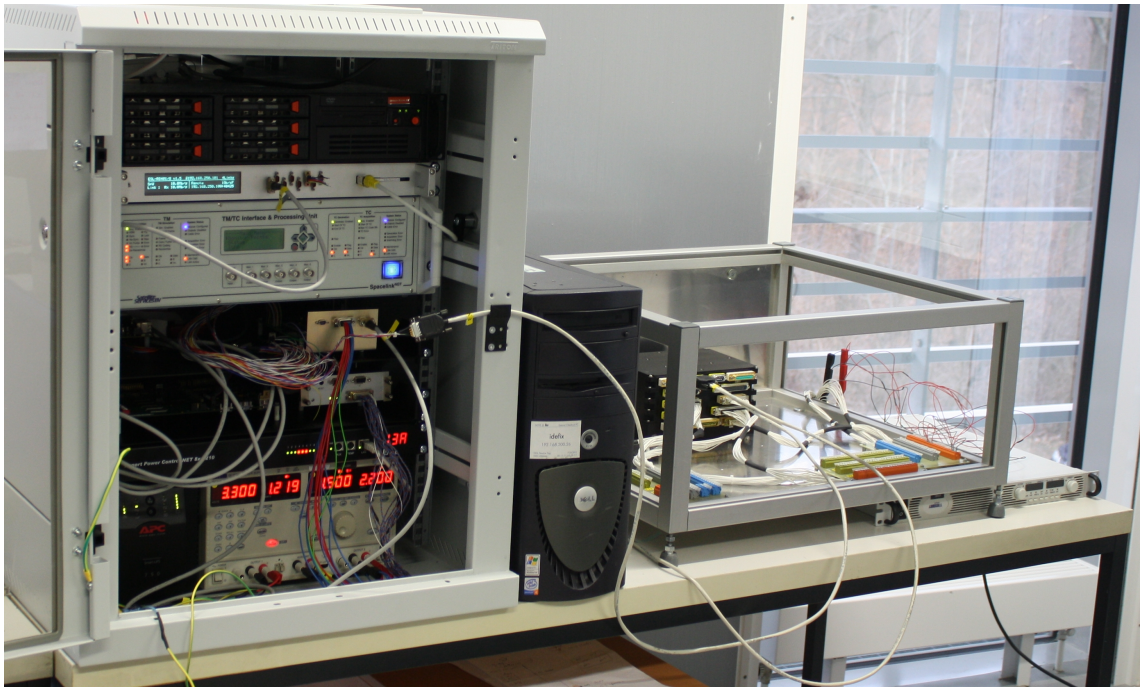
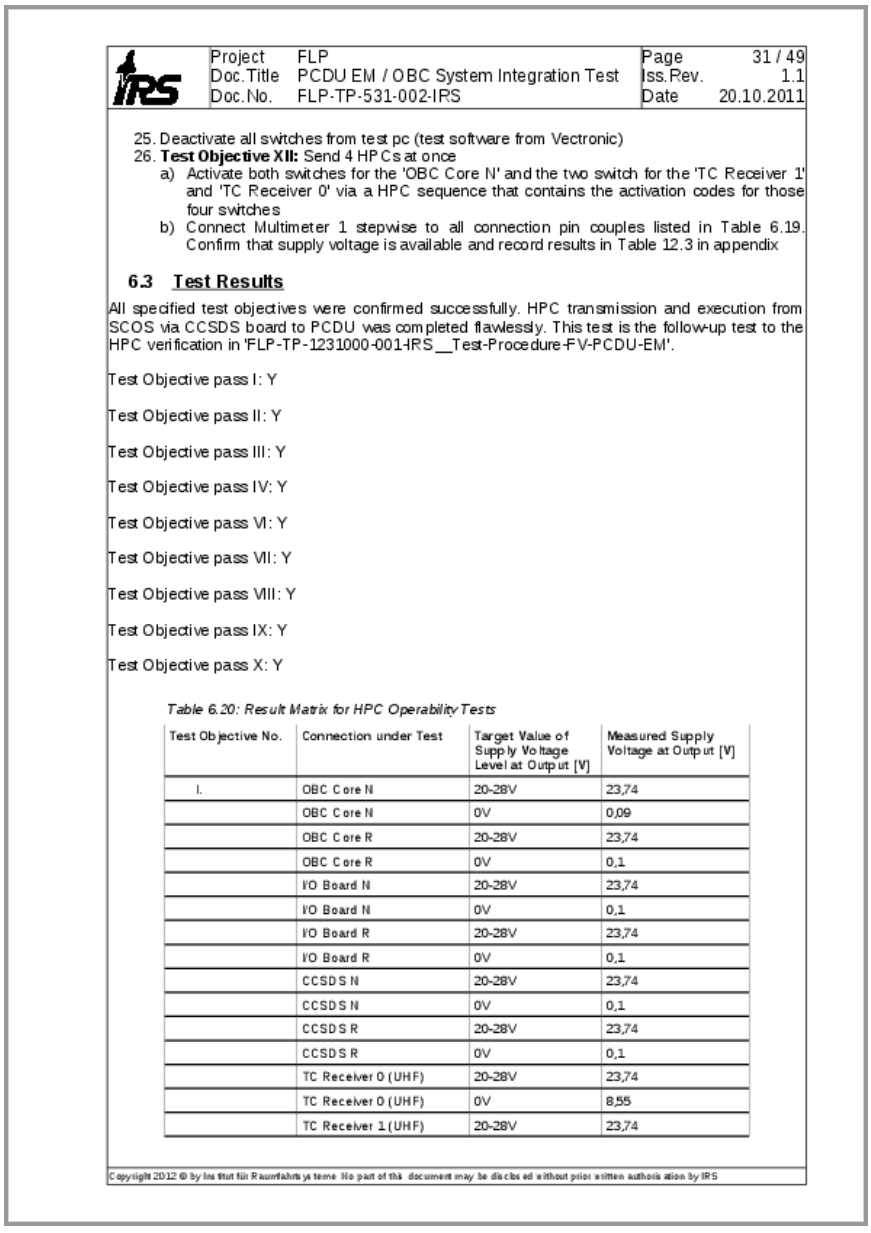


Abbildung 5.32: Test der Kommunikation des Bordrechners mit der PCDU

Sämtliche für diesen Test relevanten Details sowie die dazugehörigen Ergebnisse wurden in Zusammenarbeit mit dem für die PCDU verantwortlichen Kollegen in einer Testprozedur festgehalten. Die dort zu findenden Informationen gehen über die Beschreibungen in diesem Kapitel hinaus. Unter anderem wurde jedes Gerät und jeder Aufbau nicht nur erläutert, sondern auch mit Fotografien ergänzt. Dadurch lässt sich bei der erneuten Durchführung dieses Tests schnell erfassen, ob die dafür verwendeten Geräte gleich sind. Auch Sicherheitswarnungen zur Durchführung des Tests, sowohl für Menschen wie auch für die verwendeten Geräte, sind in dieser

Prozedur enthalten. Kriterien für einen erfolgreichen Abschluss der Tests wurden definiert. Sämtliche an der PCDU gemessene Spannungen wurden notiert und in die Prozedur eingetragen. Abbildung 5.33 zeigt einen Auszug dieser Prozedur. Es konnten alle zur Verfügung stehenden Kommandos und auch die Telemetrie erfolgreich verifiziert werden. Damit wurde gezeigt, dass die Abstimmung der Entwicklung des Bordrechners mit der PCDU erfolgreich war und das I/O Board wie spezifiziert funktioniert.



Project	FLP	Page	31 / 49
Doc.Title	PCDU EM / OBC System Integration Test	Iss.Rev.	1.1
Doc.No.	FLP-TP-531-002-IRS	Date	20.10.2011

25. Deactivate all switches from test pc (test software from Vectronic)
26. **Test Objective XII:** Send 4 HPCs at once
    a) Activate both switches for the 'OBC Core N' and the two switch for the 'TC Receiver 1' and 'TC Receiver 0' via a HPC sequence that contains the activation codes for those four switches
    b) Connect Multimeter 1 stepwise to all connection pin couples listed in Table 6.19. Confirm that supply voltage is available and record results in Table 12.3 in appendix

6.3 Test Results

All specified test objectives were confirmed successfully. HPC transmission and execution from SCOS via CCSDS board to PCDU was completed flawlessly. This test is the follow-up test to the HPC verification in 'FLP-TP-1231000-001-IRS__Test-Procedure-FV-PCDU-EM'.

Test Objective pass I: Y

Test Objective pass II: Y

Test Objective pass III: Y

Test Objective pass IV: Y

Test Objective pass VI: Y

Test Objective pass VII: Y

Test Objective pass VIII: Y

Test Objective pass IX: Y

Test Objective pass X: Y

Table 6.20: Result Matrix for HPC Operability Tests

Test Objective No.	Connection under Test	Target Value of Supply Voltage Level at Output [V]	Measured Supply Voltage at Output [V]
I.	OBC Core N	20-28V	23,74
	OBC Core N	0V	0,09
	OBC Core R	20-28V	23,74
	OBC Core R	0V	0,1
	I/O Board N	20-28V	23,74
	I/O Board N	0V	0,1
	I/O Board R	20-28V	23,74
	I/O Board R	0V	0,1
	CCSDS N	20-28V	23,74
	CCSDS N	0V	0,1
	CCSDS R	20-28V	23,74
	CCSDS R	0V	0,1
	TC Receiver 0 (UHF)	20-28V	23,74
	TC Receiver 0 (UHF)	0V	8,55
	TC Receiver 1 (UHF)	20-28V	23,74

Abbildung 5.33: Testprozedur für die Interaktion zwischen Bordrechner und PCDU

Alle weiteren an das I/O Board angeschlossene Bordkomponenten müssen auf gleiche Weise im späteren Flatsat verifiziert werden. Dabei kann auf die Testprozedur und gemachte Erfahrungen zurück gegriffen werden. Hierfür soll auch diese Dissertation als Hilfestellung dienen.

Mit dieser Konfiguration wurde also die Kommunikation zwischen Bordrechnerkern und Bordkomponenten über das I/O Board getestet. Dafür wurde die PCDU als Bordkomponente gewählt. Die korrekte Funktionalität des I/O Boards wurde verifiziert. Ungenauigkeiten in der vom IRS bereit gestellten Spezifikation für die PCDU wurden identifiziert und behoben. Außerdem wurde gezeigt, dass das zur Ansteuerung des I/O Boards gewählte Remote Memory Access Protokoll und die dafür spezifizierten Adressen korrekt sind. Die daraus vom I/O Board entpackten Protokolle wurden über eine RS-422 Schnittstelle korrekt an die PCDU weitergeleitet. Auch das Abholen von Komponententelemetrie auf dem I/O Board wurde verifiziert. Die Kommunikationstests zwischen Bordrechnerkern und PCDU konnten erfolgreich abgeschlossen werden.

5.6 Verifikation hochprioritärer Telekommandos

Die fünfte Konfiguration ist in Abbildung 5.34 zu sehen. Die PCDU wurde hier direkt an das CCSDS Board angeschlossen, am Bordrechnerkern und I/O Board vorbei. Damit konnte sowohl verifiziert werden, ob die in Kapitel 4.5.3 beschriebenen HPCs korrekt vom CCSDS Board an die PCDU weitergeleitet werden, wie auch ob sie korrekt im Missionskontrollsystem definiert wurden und auf dem Weg zur PCDU keine Konvertierungsfehler auftreten. Außerdem wurden die für diese Kommunikationsstrecke vorgesehenen Schnittstellen am CCSDS Board und an der PCDU verifiziert.

Die Kommandierung vom Missionskontrollsystem zum CCSDS Board funktioniert wie in Konfiguration 2, siehe Abbildung 5.9. Abbildung 5.35 zeigt den weiteren Weg eines Telekommandos vom CCSDS Board aus. Da es sich hier um HPCs handelt, werden die Daten des Telekommandopakets in diesem Fall vom CCSDS Board entpackt und direkt als UART 8N1 über die RS-422 Leitung an die PCDU gesendet. Telemetrie wird über diese Verbindung jedoch nicht gesendet.

Das dabei verwendete Protokoll war in einem ersten Test aufgrund einer Ungenauigkeit der dafür spezifizierten Prüfsumme nicht kompatibel, was ein Beispiel für die Notwendigkeit dieser Tests ist. Nach der Änderung der Prüfsumme in der PCDU waren diese Tests schließlich erfolgreich. Im Rahmen der Tests wurden im Missionskontrollsystem die dafür nötigen Kommandos definiert. Sie können für den Betrieb des Satelliten genau so verwendet werden. Auch die hier durchgeführten Arbeitsschritte, die verwendeten Geräte und die detaillierten Testergebnisse wurden in die im vorangehenden Kapitel erwähnte Testprozedur eingetragen.

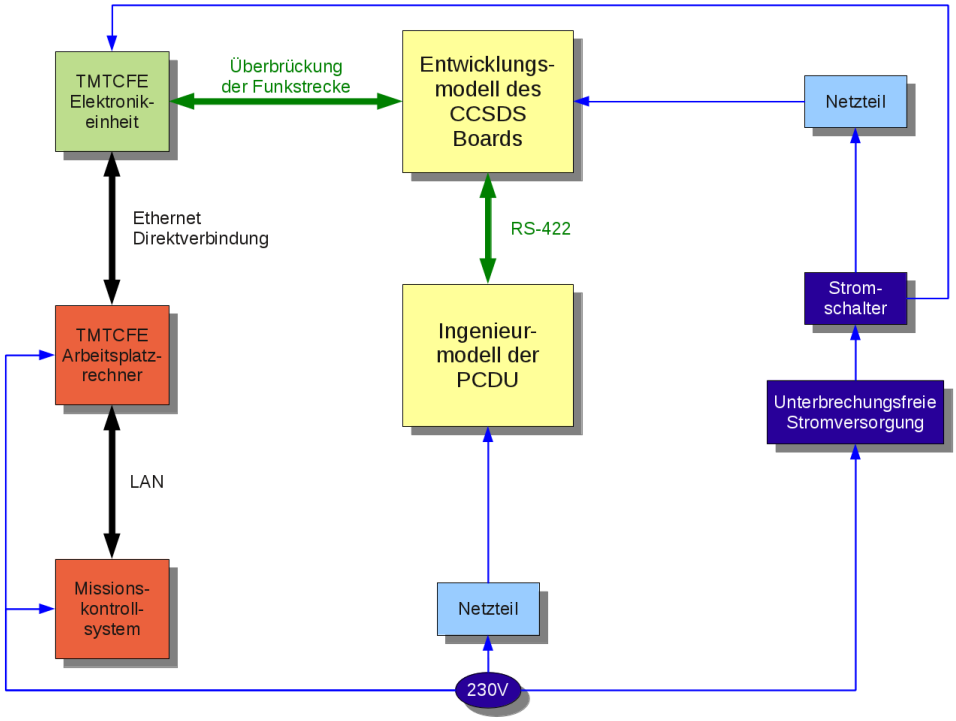


Abbildung 5.34: Verifikationsumgebung für Notfallkommandos an die PCDU

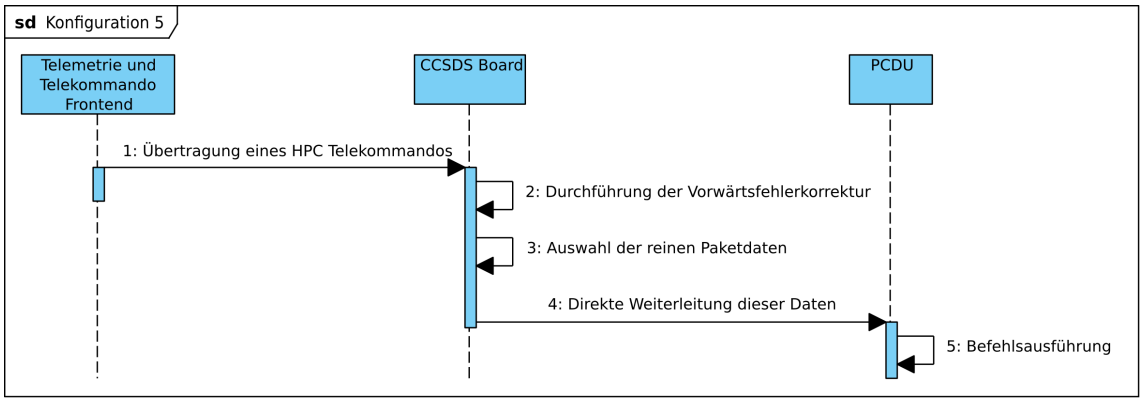


Abbildung 5.35: UML Sequenzdiagramm für Konfiguration 5

Es wurden also alle zur Verfügung stehenden hochprioritären Kommandos erfolgreich getestet. Dabei wurden wieder Ungenauigkeiten in den am IRS aufgestellten Spezifikationen gefunden und behoben. Die im Missionskontrollsystem definierten Telekommandos können auch im späteren Betrieb verwendet werden.

5.7 Zusammenfassung der durchgeführten Tests

Es wurde also erfolgreich getestet,

- dass der Bordrechnerkern konfiguriert werden kann,
- die Elemente des Bordrechners zueinander kompatibel sind,

- der Bordrechner mit dem Missionskontrollsystem kompatibel ist,
- mit Bordkomponenten kommuniziert werden kann,
- hochprioritäre Kommandos korrekt an die PCDU versendet und dort verarbeitet werden und
- eine Simulationsinfrastruktur zur Verifikation der Bordsoftware genutzt werden kann.

Die Tests decken also nicht nur die am Bordrechner zur Verfügung stehenden Schnittstellen vollständig ab, sondern beinhalten auch den Aufbau der Bodenstations- und Verifikationsinfrastruktur.

5.8 Geplante Verifikation von Sender und Empfänger

Für die Verifikation von Sender und Empfänger wurden im Rahmen dieser Arbeit drei weitere Konfigurationen vollständig geplant, konnten aber aufgrund des Entwicklungsstands bei Sender und Empfänger noch nicht genutzt werden. Sobald diese Geräte zur Verfügung stehen, können die Tests aber unter Zuhilfenahme dieser Dissertation schneller und effizienter durchgeführt werden.

Die Nummerierung der Konfigurationen aus den vorangegangenen Kapiteln wurde fortgesetzt. Konfiguration 6 und 7 dienen dazu, die für den Satelliten vorgesehenen Sender und Empfänger zunächst auf ihre Kompatibilität zum CCSDS Board zu verifizieren. In diesen Konfigurationen wird noch keine Funkstrecke genutzt. Um ein moduliertes Signal zu generieren oder zu analysieren, werden ein Signalgenerator und ein Signalanalysator verwendet. Abbildung 5.36 zeigt den prinzipiellen Aufbau. Der Aufbau ist für beide Konfigurationen fast gleich. Entweder wird ein Sender mit einem Signalanalysator oder ein Empfänger mit einem Signalgenerator an das CCSDS Board angeschlossen. Auf diese Weise wird vom Sender zwar ein Signal moduliert, dieser ist jedoch nicht an eine Antenne, sondern mit einem Koaxialkabel an den Signalanalysator angeschlossen, der das Signal wieder demoduliert. Umgekehrt kann mit einem Signalgenerator ein Signal moduliert und in den Empfänger eingespeist werden.

Im Gegensatz zum Aufbau unterscheiden sich die Abläufe in beiden Konfigurationen. Abbildung 5.37 ist das Sequenzdiagramm für Konfiguration 6, welche zur Verifikation des Senders dient. Der Verifikationsingenieur aktiviert über den Bordrech-

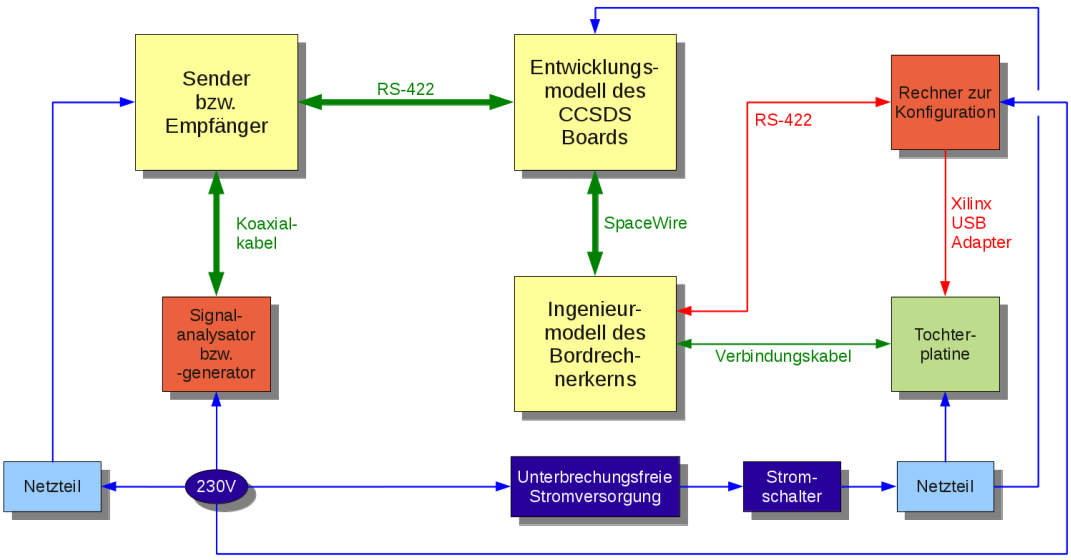


Abbildung 5.36: Verifikationsumgebung für Sender und Empfänger

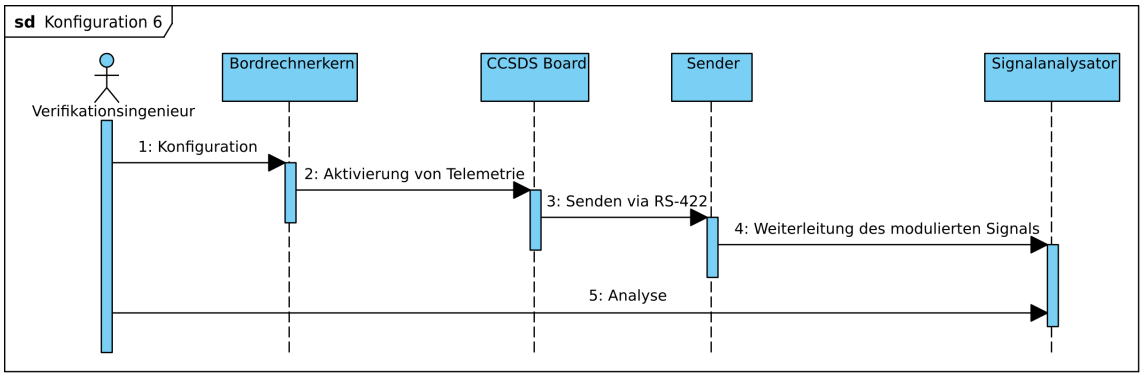


Abbildung 5.37: UML Sequenzdiagramm für Konfiguration 6

nerkern das Senden von leeren Telemetriedaten vom CCSDS Board. Solche Telemetriedaten werden im Englischen und im Fachjargon “Idle Telemetry” genannt. Auf Paketebene werden dabei vom CCSDS Board Zufallsdaten erzeugt. Die Transferebene erhält dagegen reale Informationen, unter anderem einen Zähler, der die eindeutige Identifikation des Transfer Frames ermöglicht. Nach der Demodulation des Signals am Signalanalysator kann ein Idle Frame unter Verwendung dieser Informationen auf seine Korrektheit hin untersucht werden. Damit kann festgestellt werden, ob die Schnittstellen von CCSDS Board und Sender kompatibel sind und ob die Signalmodulation korrekt funktioniert. Es ist zu beachten, dass die am Signalanalysator erhaltenen Daten vom CCSDS Board um die Information für eine CADU erweitert wurden. Dementsprechend sind nach Abbildung 3.7 die ersten 4 Bytes und die letzten 160 Bytes zu ignorieren.

Bei der in Abbildung 5.38 dargestellten Konfiguration 7 zur Verifikation des Emp-

fängers gestaltet sich der Aufbau etwas aufwändiger. Zunächst muss eine gültige und mit Konfiguration 2 erfolgreich gesendete CLTU in den Signalgenerator eingespeist werden. Dieser moduliert nun die zur Verfügung gestellten Daten, welche vom Empfänger demoduliert werden. Damit nun das CCSDS Board die Daten korrekt lesen kann, müssen zwei RS-422 Signalleitungen, die im reellen Betrieb Auskunft über den Status der Funkverbindung[7] geben, entsprechend stimuliert werden. Anschließend muss der Bordrechner mit einer Software gestartet werden, die vom CCSDS Board liest. Nun kann analysiert werden, ob die in den Signalgenerator eingespeisten Informationen denen auf dem CCSDS Board entsprechen. Auch hier ist zu beachten, dass das CCSDS Board eine Vorwärtsfehlerkorrektur durchführt. Daher sind nach Abbildung 3.6 die ersten 2 und die letzten 8 Bytes, sowie im übrig bleibenden Teil jedes achte Byte des mit dem Signalgenerator eingespeisten Pakets beim Vergleich zu ignorieren.

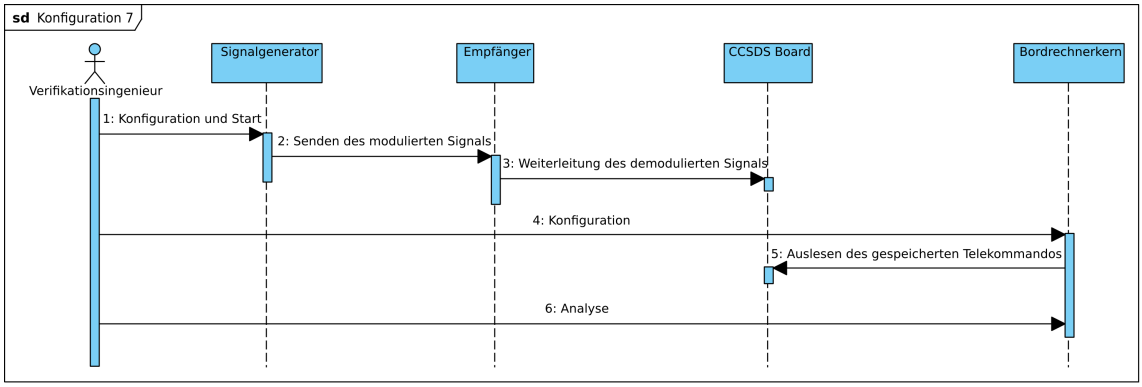


Abbildung 5.38: UML Sequenzdiagramm für Konfiguration 7

Zuletzt kann Konfiguration 8 dazu genutzt werden, die vollständige Kommandierungskette mit der Funkstrecke zu verifizieren. Dazu wird an das Telemetrie und Telekommando Frontend ein Modulator/Demodulator durch ein Überbrückungskabel angeschlossen. Dieser Modulator/Demodulator erzeugt eine Frequenz von 70 MHz. Diese Frequenz ist bei der Signalübertragung mit einem Kabel weniger störanfällig als die höherfrequente Übertragungsfrequenz. In der Nähe der bodenseitigen Antenne wird daraus mit Hilfe eines Frequenzwandlers eine Hochfrequenz erzeugt. Da die Leistung des Sendesignals hier noch reduziert ist, kann der Empfänger mit einem Überbrückungskabel angeschlossen werden. Wäre die Leistung zu hoch, würde der Empfänger zerstört. Im Betrieb wird entsprechend das hochenergetische Signal

[7]Es handelt sich um die Informationen, ob erstens ein Signal empfangen wird und zweitens das Verhältnis Signal zu Störung nicht unter einem entsprechenden Schwellenwert liegt, bei dem die Vorwärtsfehlerkorrektur nicht mehr funktioniert. Nur wenn beide Bedingungen erfüllt werden und dadurch auf beiden RS-422 Signalleitungen ein entsprechender Status übertragen wird, beginnt das CCSDS Board zu lesen.

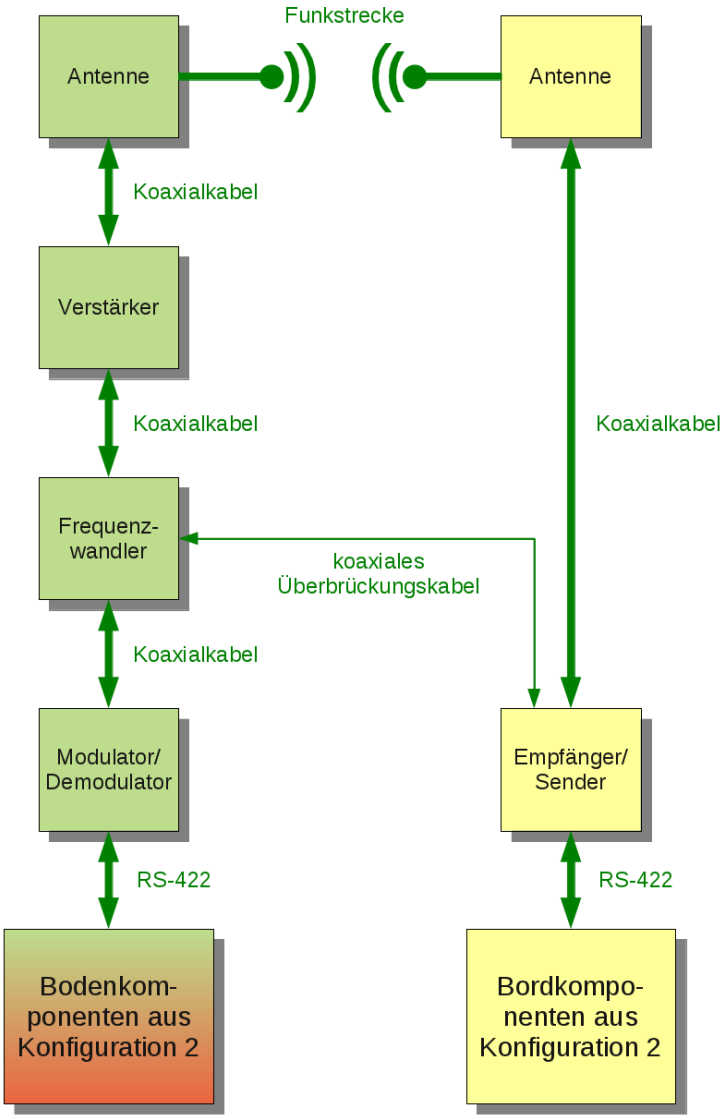


Abbildung 5.39: Verifikation mit der Komamndierungskette inklusive Funkstrecke

durch den Übertragungsweg abgeschwächt. In einem letzten Schritt kann schließlich bodenseitig eine Verstärkerstufe mit Antenne und bordseitig eine Antenne an die vorhandene Infrastruktur angeschlossen werden. Die gesamte Infrastruktur ist in Abbildung 5.39 dargestellt. Die beiden unteren Elemente sind in Abbildung 5.9 zu finden und werden aus Gründen der Darstellbarkeit hier zusammengefasst. Um nun sowohl die atmosphärische Abschwächung zu berücksichtigen, als auch die Ausrichtgenauigkeit der Antenne zu verifizieren, muss die Funkstrecke durch einen Relaissatelliten hergestellt werden. Das Telekommando wird also an einen Satelliten gesendet, der dieses empfängt und unverändert weiterleitet, in diesem Fall an den sich noch am Boden befindenden Kleinsatelliten Flying Laptop.

6 Weitere Testergebnisse

Im Rahmen dieser Arbeit konnten Verifikationsergebnisse über die in Kapitel 5 beschriebenen Kompatibilitätstests hinaus erzielt werden. Zum einen handelt es sich dabei um die Verifikation der in Matlab/Simulink vorhandenen Lageregelungsalgorithmen zur Überprüfung der funktionalen Korrektheit der genutzten Simulationsumgebung (vgl. Kapitel 4.4.3), zum anderen konnte ein erster in die Programmiersprache C++ konvertierter Lageregelungsmodus auf dem beschriebenen Bordrechner verifiziert werden. Nachfolgend werden beide Fälle beschrieben und Ergebnisse dargestellt. Es kommt jeweils die in Kapitel 4.4 beschriebene Simulationsumgebung zum Einsatz. Bei den Lageregelungsalgorithmen aus Matlab/Simulink handelt es sich um eine Controller in the Loop Konfiguration, beim Aufbau mit dem Bordrechner um Hardware in the Loop.

- In Kapitel 6.1 wird dargestellt, welche Ergebnisse sich durch Anbindung der Matlab/Simulink Lageregelungsalgorithmen an die genutzte Simulationsumgebung ergaben. Diese Ergebnisse wurden mit denen reiner Matlab/Simulink Simulationen verglichen. Damit wurden die am IRS entwickelten Simulationsmodelle verifiziert.

- In Kapitel 6.2 werden anschließend Ergebnisse dargestellt, die sich durch Nutzung eines solchen Lageregelungsalgorithmus auf dem echten Bordrechner bei der Anbindung an die Simulationsumgebung ergaben. Dabei konnten die Ergebnisse aus Kapitel 6.1 als Referenzergebnisse zum Vergleich genutzt werden. Da die Ergebnisse übereinstimmten, kann davon ausgegangen werden, dass der in Kapitel 4 beschriebene Prüfstand auch für die Verifikation von zusammengesetzter Bordsoftware genutzt werden kann, und nicht nur für die Verifikation einzelner Teile der Software, wie in Kapitel 5.4 beschrieben. Außerdem zeigte sich so, dass der Prüfstand keine Latenzprobleme hat und auch unter Last funktioniert. Damit konnte der Prüfstand selbst erfolgreich getestet werden.

6.1 Controller in the Loop

Die in Kapitel 4.4.3 beschriebene Konfiguration zur Verifikation der funktionalen Korrektheit der Simulationsumgebung konnte zugleich für die Verifikation von Lageregelungsalgorithmen verwendet werden. Nachfolgend werden die Ergebnisse von Verifikationsläufen einiger Modi des Lageregelungssystems exemplarisch dargestellt. Dabei werden die Ergebnisse aus reinen Matlab/Simulink Simulationen denen gegenüber gestellt, die sich mit der in Kapitel 4.4.3 beschriebenen Konfiguration der Anbindung der Lagereglungsalgorithmen an die in Kapitel 4.4.2 erläuterte Simulationsumgebung ergeben haben.

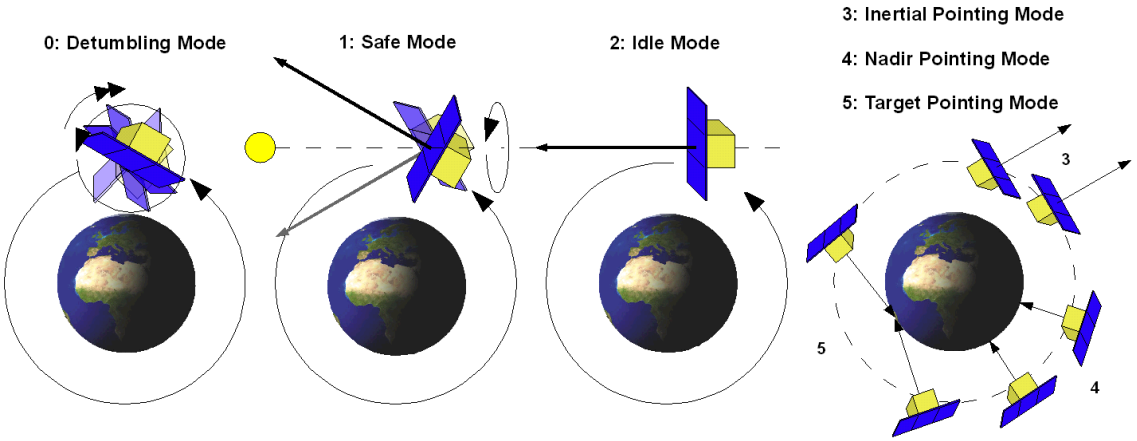


Abbildung 6.1: Lageregelungsmodi des Flying Laptop [Grillmayer 2010]

Abbildung 6.1 ist ein Überblick der Lageregelungsmodi des Kleinsatelliten Flying Laptop.

- Der Detumble Mode baut hohe Drehraten ab, wie sie beispielsweise nach der Trennung vom Träger bestehen.
- Der Safe Mode ist ein Notfallmodus, der den Satellit um seine stabile Hauptträgheitsachse mit den Solarpaneelen grob zur Sonne hin aufspinnt und dabei nur zwei Arten von Sensoren und eine Art von Aktuator benötigt.
- Der Idle Mode hingegen ist ein dreiachsenstabilisierter Modus mit genauer Ausrichtung der Solarpaneele zur Sonne.
- Die drei Pointing Modi ermöglichen die Ausrichtung des Satelliten auf einen Punkt. Das kann beim Inertial Pointing Mode ein Punkt in unendlich hoher Entfernung, beim Nadir Pointing Mode der Erdmittelpunkt und beim Target Pointing Mode jeder beliebige andere Punkt sein. Da es beim Target Pointing sinnvoll ist, einen Punkt auf der Erdoberfläche zu wählen und damit der

Abstand zum Ziel am geringsten ist, entstehen hier die höchsten Drehraten. Damit ist auch der Anspruch an das Lageregelungssystem am größten. Die Drehgeschwindigkeit des Satelliten muss sich hier ständig ändern.

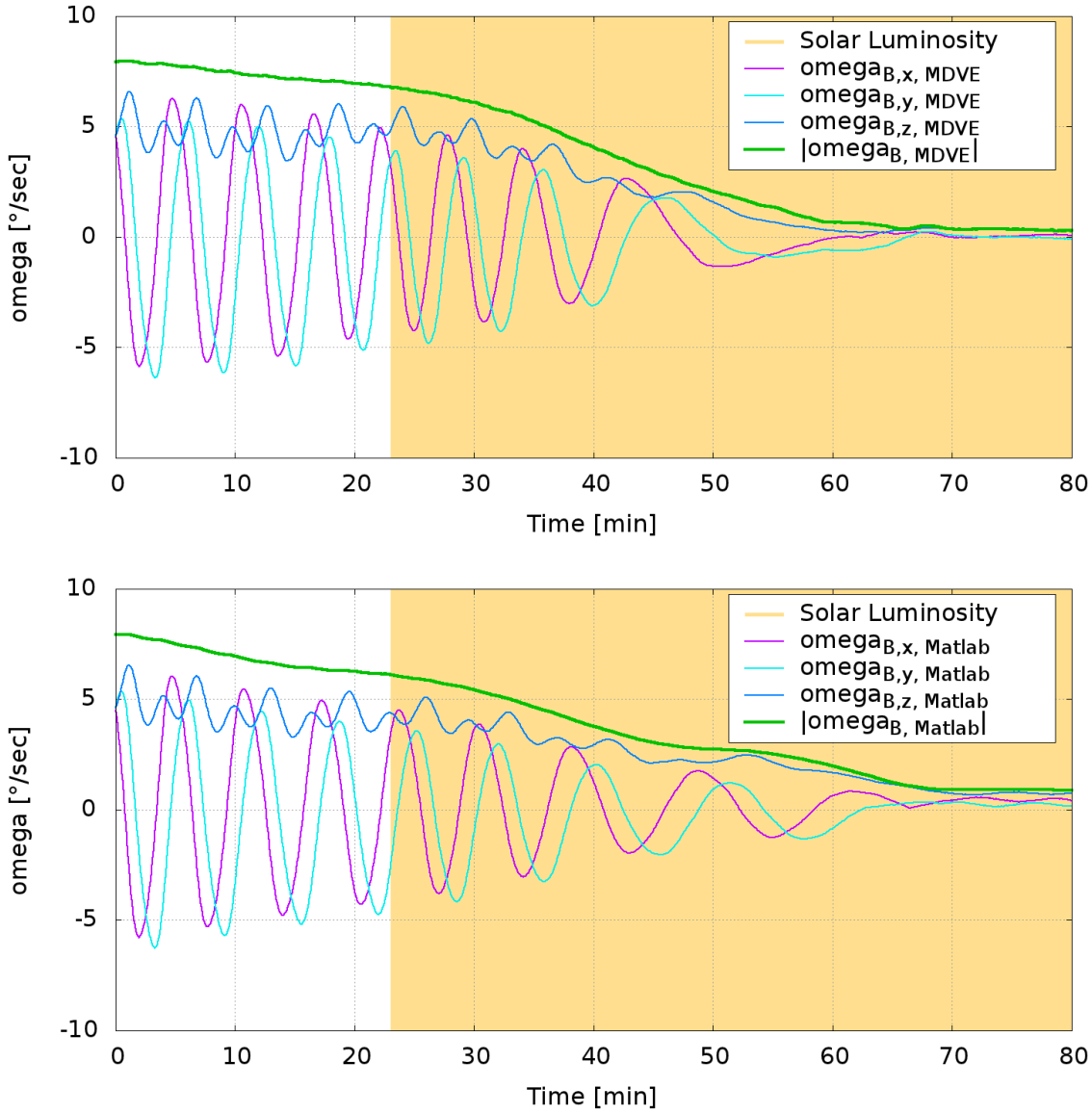


Abbildung 6.2: Ergebnisse für den Detumble Mode [Winter 2011]

Der in [Winter 2011] durchgeführte Vergleich zeigt, dass sich alle Lageregelungsalgorithmen auf beiden Simulationsumgebungen stabil verhalten. Für den Detumble und Safe Mode waren die Ergebnisse qualitativ übereinstimmend, was essentiell ist, da es sich hier um die wichtigsten Modi handelt. Aber auch die Ergebnisse aller weiteren Modi waren adäquat. Abbildung 6.2 zeigt die Ergebnisse für den Detumble Mode, oben in der im Rahmen dieser Arbeit verwendeten Simulationsumgebung und unten in der reinen Matlab/Simulink Simulationsumgebung. Es ist zu sehen,

dass die Zeit zur Reduzierung der Drehrate in beiden Fällen etwa gleich schnell war. Außerdem wurde auch die Drehrate um die einzelnen Achsen ähnlich abgebaut.

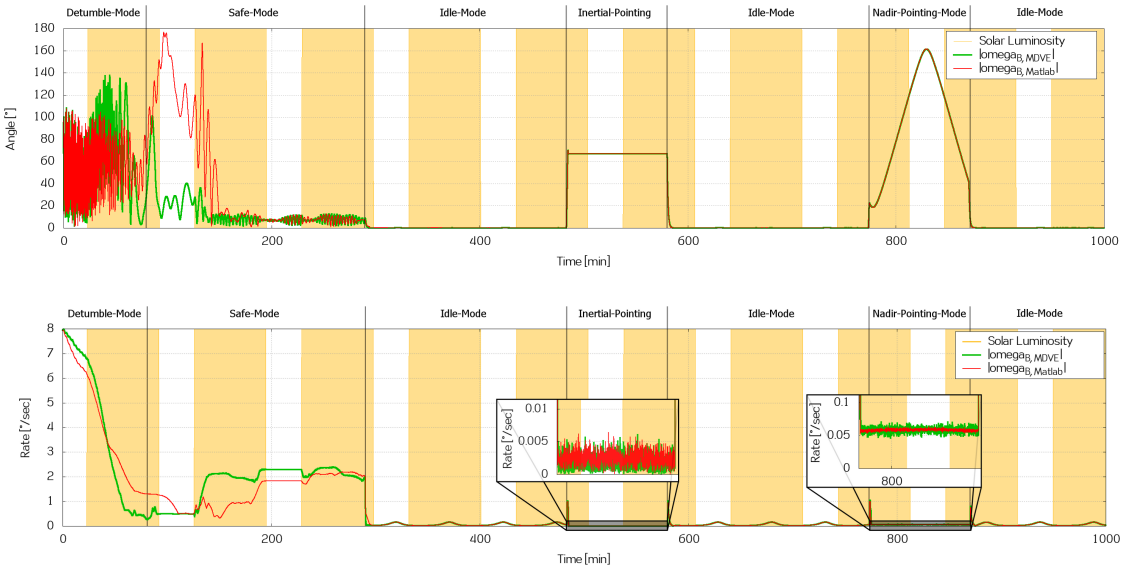


Abbildung 6.3: Ergebnisse für verschiedene Modi

Alle weiteren Modi mit Ausnahme des Target Pointing Modes werden in Abbildung 6.3 dargestellt. Oben zu sehen ist die Lage zur Sonne in beiden Simulationsumgebungen, unten die jeweilige Drehrate. Ohne auf Lageregelungsdetails einzugehen, kann festgestellt werden,

- dass die Ergebnisse in beiden Simulationsumgebungen sehr ähnlich sind.
- Daher kann davon ausgegangen werden, dass beide Simulationsumgebungen frei von schwerwiegenden Fehlern sind.
- Außerdem wurde sichergestellt, dass die Modetransitionen korrekt funktionieren.

Es sei festgehalten, dass die Controller in the Loop Konfiguration bereits zu einem frühen Zeitpunkt in der Entwicklung, noch bevor jegliche Bordrechnerhardware zur Verfügung stand, Schlüsse auf die Simulationsmodelle, aber auch auf die entwickelten Regelalgorithmen zuließ. So können die wichtigsten Modi Detumble Mode, Safe Mode und Idle Mode als robust betrachtet werden. Das Verhalten der Pointing Modi ist prinzipiell korrekt, jedoch sind die Regelalgorithmen hierfür noch zu erweitern. Mehr dazu ist in [Winter 2011] zu finden.

6.2 Hardware in the Loop

Wie bereits in Kapitel 5.4 beschrieben, wurde die Bordsoftware mit den simulierten Komponenten erfolgreich verbunden. Dadurch war es im Rahmen dieser Arbeit erstmals möglich, Lageregelungsmodi von der Bordrechnerhardware aus zu verifizieren. Insbesondere bei der Konvertierung der Regelalgorithmen von Matlab/Simulink in bordsoftwarekonformen C/C++ Quelltext konnten und können erhebliche Fehler entstehen. Auch die Anbindung an die Datenbank der Bordsoftware war eine erhebliche Herausforderung für die dafür zuständigen Kollegen. Dieses Kapitel stellt exemplarisch die Verifikation des Detumble Mode auf der echten Bordrechnerhardware dar und zeigt damit, dass der Prüfstand auch unter Last korrekt funktioniert.

Abbildung 6.4 zeigt die Ergebnisse des Detumble Modes vom Bordrechner aus bei der Nutzung des in Abbildung 5.1 zu sehenden Prüfstands. Im Vergleich mit Abbildung 6.2 oben, dem direkten Anschluss des Matlab/Simulink Regelalgorithmus an die Simulationsumgebung, ergibt sich ein ähnliches Verhalten. Die Dauer, um die Drehrate abzubauen, ist etwa gleich lang. Auch der Verlauf der Gesamtdrehrate weißt große Ähnlichkeiten auf.

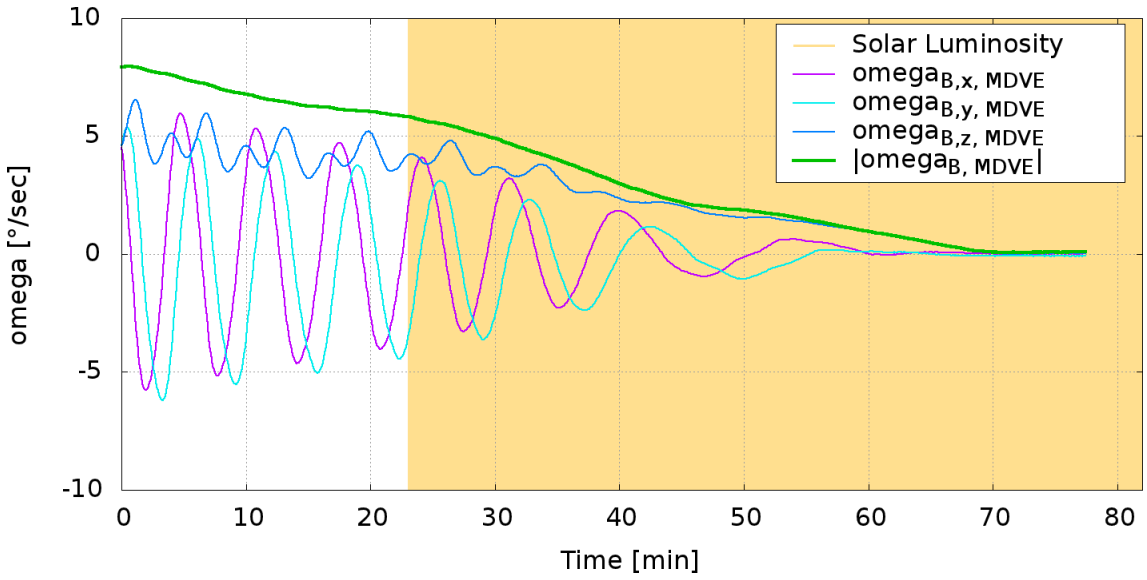


Abbildung 6.4: Ergebnisse für den Detumble Mode

Somit hat sich gezeigt, dass

- die entwickelte Anbindung einer Simulationsumgebung funktionsfähig ist und zeitliche Randbedingungen erfüllt,
- die Simulationsumgebung auch unter Last funktioniert und

- die in Kapitel 5.1 beschriebene Konfiguration 3 für die funktionale Verifikation der Bordsoftware genutzt werden konnte und kann.

Damit konnten unter Verwendung des aufgebauten Prüfstands nicht nur die in Kapitel 5 der vorliegenden Dissertation beschriebenen Tests durchgeführt werden. Es wurde auch gezeigt, dass sich dieser Prüfstand für anstehende Aufgaben ohne Einschränkung eignet.

7 Diskussion

7.1 Erkenntnisse

Aus den durchgeführten Tests konnte abgeleitet werden, dass das vorgestellte Bordrechnerkonzept die funktionalen Anforderungen an einen Satellitenbordrechner erfüllt. Auch die Schnittstellen haben sich als kompatibel erwiesen. Bei Planung, Durchführung und Auswertung der Tests wurden jedoch Sachverhalte beobachtet, die auch in der weiteren Entwicklung des Bordrechners und des Kleinsatelliten Flying Laptop eine Rolle spielen können.

Dazu gehört, dass beim Wechsel des Testaufbaus oder einzelner Verifikationskonfigurationen innerhalb eines Aufbaus Probleme auftreten können. Eine Planung, die nicht detailliert genug war, konnte beispielsweise dazu führen, dass die für einen Aufbau nötigen Teile wie Kabel oder Stecker anderswo in Verwendung waren. Wurde dies erst kurz vor Fertigstellung des entsprechenden Aufbaus festgestellt, so konnte das zu einigen Stunden bis einigen Tagen Verzögerung hinsichtlich der Durchführbarkeit der Tests führen. Auch die Entnahme von Teilen und Komponenten aus einem Aufbau und deren rechtzeitige Reintegration konnte Probleme verursachen. Es war darauf zu achten, dass alle Randbedingungen exakt wiederhergestellt wurden, wozu unter anderem die korrekte Verwendung von Steckverbindungen gleicher Art und auch die Erdung der Teile gehört. Hier konnten Prozeduren helfen, die die nötigen Schritte beschreiben. Gerade diese so simpel klingenden Notwendigkeiten waren für die zügige Durchführung der Tests enorm wichtig, da sie oft als Kleinigkeiten betrachtet wurden und auch bei einer einem solchen Sachverhalt folgenden Fehlersuche zunächst oft vernachlässigt wurden. Es hat sich also heraus gestellt, dass eine detaillierte Planung wichtig für solche Tests ist. Oft hat die Anfertigung von Diagrammen dabei geholfen, der nötigen Detailtreue gerecht zu werden.

Eine weitere Erkenntnis, die vor allem bei künftigen Projekten helfen kann, ergibt sich aus der Spezifikation von Funktionen und Bauteilen. Hier waren Details oft ungenau oder fehlend, was ebenso zu Verzögerungen führen konnte. Hier sei als Beispiel die ungenaue Spezifikation der Prüfsumme für hochprioritäre Kommandos

vom CCSDS Board zur PCDU genannt, siehe Kapitel 5.6. Erst nach dem nicht erfolgreichen Test fiel auf, dass in der Spezifikation Angaben wie Startwerte für die zur Überprüfung genutzte Polynomdivision und die Art der Formatierung der berechneten Prüfsumme fehlten. Auch hier wurde eine Verzögerung durch die beim Hersteller in Auftrag gegebene Korrektur und den erneuten Test verursacht. Gerade in einem auch zur Ausbildung genutzten Projekt wie dem Kleinsatelliten Flying Laptop ist daher verstärkt auf die detaillierte Spezifikation von in Auftrag gegebenen Bauteilen hinzuweisen. Solche Beispiele zeigen aber auch deutlich, wie wichtig die in dieser Arbeit beschriebenen Tests waren.

Ebenfalls ein Problem konnte die Konfigurierbarkeit von Bauteilen und Geräten darstellen. Zwar war sie in vielen Fällen notwendig, um Geräte oder Protokolle kompatibel zu halten, jedoch mussten die entsprechenden Einstellungen oft zuerst einmal gesetzt werden. Beispielsweise wurde nach Anschluss des CCSDS Boards an das Telemetrie und Telekommando Frontend scheinbar willkürlich ein Teil der Daten inkorrekt übertragen. Gleiche Fehler ließen sich nicht wiederherstellen. Nach einer viele Stunden andauernden Suche konnte schließlich festgestellt werden, dass es eine Inkonsistenz in den Einstellungen der Datenübertragung gab. Wie in Abbildung 7.1 dargestellt, wurden auf Seiten des Telemetrie und Telekommando Frontends die Daten auf der fallenden Flanke des Clock-Signals gesendet. Auf Seiten des CCSDS Boards wurden sie hingegen auf der steigenden Flanke gelesen. Somit wurde dort am Übergang zweier Bits und damit quasi zufällig das richtige oder falsche Bit gelesen. Nach dem Umstellen auf die steigende Flanke am Telemetrie und Telekommando Frontend funktionierte die Übertragung korrekt. Es lässt sich also festhalten, dass eine höhere Konfigurierbarkeit immer mehr Flexibilität aber auch mehr Arbeitsaufwand bedeutete.

Zuletzt sei die Personalplanung zur Durchführbarkeit der Tests genannt. So musste sichergestellt werden, dass für die Durchführung der Tests an jeder dafür zu besetzenden Stelle ein Mitarbeiter oder Student zur Verfügung steht, der für die entsprechende Aufgabe qualifiziert ist. Hier hat die rechtzeitige Einarbeitung in Aufgaben eine wesentliche Rolle gespielt. Nur durch die Streuung von Kompetenzen und die dadurch für das Projekt entstehende Flexibilität konnte die Verifikation der PCDU in Zusammenspiel mit dem Bordrechner zügig durchgeführt werden.

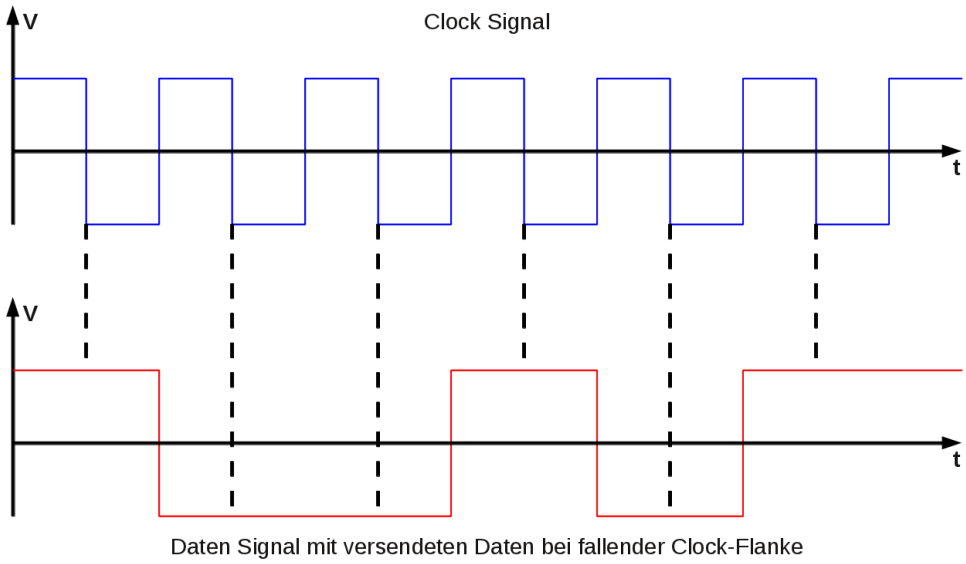


Abbildung 7.1: Datensignal bei fallender Flanke des Clocksignals

7.2 Bewertung

Die durchgeführten Tests haben eine Verifikation des Zusammenspiels der von verschiedenen Personen konzipierten und von unterschiedlichen Herstellern gelieferten Bordkomponenten ermöglicht. Unstimmigkeiten in der Kommunikation einzelner Bordkomponenten sowie zwischen Boden- und Bordseite konnten rechtzeitig entdeckt und behoben werden. Zur Kommunikation gehört die richtige Verwendung der Schnittstellen selbst, aber auch die jeweils darauf genutzten Protokolle. Auch die auf den Bordkomponenten laufende Software oder entsprechende Hardwareschaltungen zählen dazu.

Das Konzept der Anbindung einer Simulationsumgebung mit simuliertem I/O Board zum Verzicht auf VME-Karten sowie die Verwendung eines frei zur Verfügung stehenden Echtzeit-Linux Betriebssystems haben sich für die Bordsoftwareverifikation als erfolgreich erwiesen. Wie in Kapitel 6.2 beschrieben, konnten und können Regler damit ohne Einschränkungen verifiziert werden. Die verwendete Simulationsumgebung selbst konnte wie in Kapitel 6.1 dargestellt durch Vergleich mit einer anderen Simulationsumgebung und unabhängig aufgebauten Modellen verifiziert werden.

Aussagen über die interne Korrektheit von Bordkomponenten und entsprechender Software können mit den durchgeführten Tests jedoch nicht getroffen werden. Das ist jedoch besonders für am Institut entwickelte Hard- und Software notwendig. Dafür mussten und müssen außerhalb des Rahmens dieser Arbeit spezielle Methoden entwickelt und angewandt werden.

7.3 Verwendbarkeit

Der im Rahmen dieser Arbeit entwickelte und aufgebaute Prüfstand ermöglicht weit mehr als die oben dargestellten Tests. Er verbindet die Entwicklung der Bordsoftware mit dem operationellen Design des Satelliten. Es wird fortlaufend getestet, ob die im Missionskontrollsystem definierten Telemetrie- und Telekommandopakete mit den Elementen der Bordsoftware funktional übereinstimmen. Dafür wurden im Prüfstand bereits entsprechende Datenbanken verknüpft, um sowohl die Datenkonsistenz zu gewähren wie auch die Implementierung solcher Pakete zu vereinfachen. Die so aufgebaute Telemetrie- und Telekommandodatenbank kann später gemeinsam mit den in dieser Arbeit beschriebenen Elementen der Bodenstation in derselben weiterverwendet werden. Der gesamte Prüfstand kann ohne Änderungen in einen Flatsat integriert werden. In diesem sind nur noch die Bordkomponenten an das I/O Board anzuschließen, um Kommunikationstests durchführen zu können. Nach Lieferung der Flugmodelle des Bordrechners können außerdem die bislang verwendeten Entwicklungs- und Flugmodelle einfach ausgetauscht werden. Auch für die zur Telekommunikation ausgewählten Modulatoren, Verstärker, Antennen, Sender und Empfänger muss kein eigenständiger Prüfstand mehr aufgebaut werden. Da nach dem Start des Satelliten Entwicklungs- und Ingenieurmodelle am Boden bleiben, können sie weiter genutzt werden. Der Prüfstand ermöglicht damit durch Verwendung der Simulationsumgebung Tests vor der Übertragung veränderter und aktualisierter Bordsoftware an den Satelliten. Außerdem können Operatoren trainiert werden, indem mit den Elementen der Bodenstation die Simulationsumgebung angesteuert wird.

Der aufgebaute Prüfstand ermöglicht also die

- Koordination unterschiedlicher Entwicklungsbereiche,
- die Verifikation sämtlicher an den Bordrechner angeschlossenen Komponenten,
- die Weiterverwendbarkeit der aufgebauten Infrastruktur sowie
- Tests und Ausbildung nach dem Start des Satelliten.

Damit kann abschließend festgestellt werden, dass dieser Prüfstand wie in Abbildung 7.2 dargestellt zu einem Sternpunkt bei der Entwicklung des Kleinsatelliten Flying Laptop geworden ist.

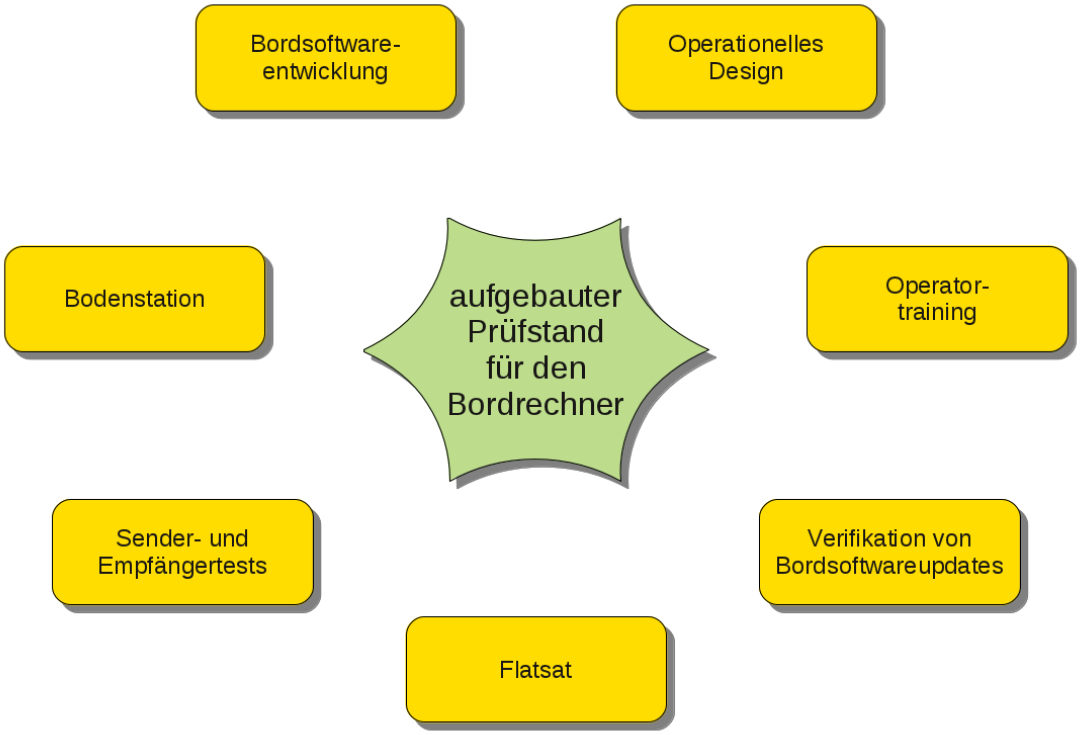


Abbildung 7.2: Prüfstand und zusammenhängende Entwicklungen

8 Zusammenfassung und Ausblick

Diese Arbeit stellt dar, wie Hardware und Software eines modernen Kleinsatellitenbordrechners sinnvoll, effizient und erfolgreich auf die Kompatibilität zu den Bordkomponenten, zur Bodenstationsinfrastruktur und zu einer Simulationsumgebung getestet wurden. Bei dem zu verifizierenden Bordrechner handelte es sich um ein LEON3-FT basierendes und modular aufgebautes System, das in Zusammenarbeit mit den industriellen Partnern Aeroflex Colorado Springs, Aeroflex Gaisler und 4Links entstand. Es ermöglicht die Kommunikation zum Boden mit Protokollen nach CCSDS. Als Plattform für diesen Bordrechner dient der vom Institut für Raumfahrtsysteme entwickelte Kleinsatellit Flying Laptop. Durch die beschriebenen Tests wurde gezeigt, dass sich dieser Bordrechner auch für eine Verwendung in kommerziellen Projekten eignet.

Zur Durchführung solcher Tests musste vom Verfasser dieser Dissertation zunächst eine geeignete Umgebung ausgewählt und angepasst werden. Anschließend konnten sukzessive die Bodenstationsinfrastruktur, der Bordrechner sowie weitere Bordkomponenten in diese Umgebung integriert werden. Dabei wurde stets darauf geachtet, dass der Testaufbau korrekt geerdet ist, um Beeinträchtigungen oder Beschädigungen zu vermeiden. Aufgrund US-amerikanischer Exportbeschränkungen war der Bordrechner gegen ungewollten manuellen Zugriff zu sichern. Trotzdem sollte die volle Konfigurierbarkeit gewährleistet sein. Bei der Verbindung der Geräte mussten zur Verfügung stehende Schnittstellen sowie die darauf laufenden Kommunikationsprotokolle zunächst analysiert werden. Danach konnten die Schnittstellen verbunden und in Betrieb genommen werden.

Nun konnten verschiedene Verifikationskonfigurationen aufgebaut und genutzt werden. In einem ersten Schritt wurde der Bordrechnerkern in Betrieb genommen. Das schließt das korrekte Setzen von Registern ebenso ein, wie das Aufspielen einer ersten Software sowie die Inbetriebnahme der Schnittstellen. In zwei weiteren Konfigurationen wurde der Bordrechnerkern mit der Bodenstationsinfrastruktur und simulierten Satellitenkomponenten verbunden, was Bordsoftwaretests in Echtzeit ermöglicht. Dazu mussten Elemente der Bodenstationsinfrastruktur konzipiert, auf-

gebaut und in Betrieb genommen werden. Es wurden professionelle Systeme wie das ESA-Missionskontrollsystem SCOS-2000, die Prozedursteuerung MOIS der Firma Rhea und ein Telemetrie- und Telekommandofrontend der Firma Satellite Services eingesetzt. Diese Elemente wurden Stück für Stück verbunden und schließlich an den Bordrechner angeschlossen. Dabei war jedes Mal zu testen, ob die durchgeführten Konvertierungen den Anforderungen der verwendeten Standards entsprechen. Auf Bordseite wurden die simulierten Komponenten mit dem Bordrechner verbunden. Auch hier waren Konvertierungen zu testen. Ein Konzept wurde entwickelt und realisiert, um die zeitlichen Randbedingungen der Schnittstellen einhalten zu können. Um Fehler im Simulator so weit wie möglich ausschließen zu können, wurde die verwendete Simulationsumgebung mit einer anderen, unabhängig aufgebauten Simulationsumgebung verglichen. In einer weiteren Konfiguration wurde mit der PCDU erstmals eine reale Bordkomponente mit dem Bordrechner verbunden. Mit diesem Testaufbau konnten dedizierte Telekommandos vom Boden am Bordrechnerkern vorbei geleitet werden. Diese Art von Telekommandos sind nötig, um den Bordrechnerkern kalt redundant betreiben zu können und um die rechnende Einheit vom Boden aus bestimmen zu können. Für diese Konfiguration wurden außerdem die entsprechenden Telekommandos in SCOS-2000 definiert. Die letzte im Rahmen dieser Arbeit genutzte Konfiguration war schließlich die Kommandierung der PCDU mit herkömmlichen Kommandos. Hierfür wurde die PCDU mit einer anderen Schnittstelle des Bordrechners verbunden als im vorangegangenen Aufbau.

Mit diesen Verifikationskonfigurationen konnte der Verfasser zeigen, dass

- das Bordrechnerkonzept funktionsfähig ist,
- es sich mit einer professionellen Bodenstationsinfrastruktur kommandieren lässt,
- sämtliche Bordrechnerschnittstellen zu den Bordkomponenten kompatibel sind,
- Bordkomponenten damit über die vorgesehenen Wege kommandiert werden und Telemetrie zurück senden können,
- anstatt der realen Bordkomponenten Modelle in einer Simulationsumgebung verwendet werden können, was den Test operationeller Szenarien erst ermöglicht und
- damit die Bordsoftware vollständig und effizient verifiziert werden kann.

In den Ergebnissen der Arbeit wurde zunächst dargestellt, wie die genutzte Simulationsumgebung verifziert wurde. Damit konnten noch existierende Fehler in der gesamten Umgebung entdeckt und behoben werden, was die Wahrscheinlichkeit von Fehlern auf ein Mindestmaß reduzierte. Fehler in Spezifikationen der Hersteller konnten damit allerdings nicht ermittelt werden. Anschließend wurde gezeigt, welche Ergebnisse sich für den Anschluss des Bordrechners an diese Umgebung ergaben. Durch Darstellung eines Lageregelungsmodus konnte gezeigt werden, dass auch der Bordrechner mit den Komponentenmodellen und der Simulationsumgebung korrekt interagieren kann. Damit wurde bewiesen, dass die Bordsoftware auf diese Weise funktional verifiziert werden kann. In einem nächsten Schritt wurde die Kompatibilität des Bordrechners mit weiterer Infrastruktur wie dem Missionskontrollsystem erläutert. Auch diese Tests waren erfolgreich. Zuletzt wurde dargestellt, wie mit diesem Testaufbau sogar Fehler in der Kommunikation mit Bordkomponenten gefunden und behoben werden konnten.

Damit erwiesen sich die in dieser Arbeit durchgeführten Hardware- und Software-Kompatibilitätstests als notwendig und erfolgreich. Mit dem aufgebauten Prüfstand kann nun die Bordsoftware fertig entwickelt und funktional verifiziert werden. Dabei können auch weitere in dieser Arbeit entwickelte Methoden wie die automatische Dokumentation von Verifikationsläufen genutzt werden. Pläne für die funktionale Verifikation der Bordsoftware sind zu erweitern. Die Flugmodelle des Bordrechners, ein dazu gehörender Bordrechnerkern ist in Abbildung 8.1 dargestellt, sind ebenfalls auf ihre Kompatibilität zu testen, was durch die entwickelten Tests und Methoden beschleunigt wird.

Parallel zur Entwicklung der Bordsoftware müssen Konzepte zum Betrieb des Satelliten erarbeitet werden. Ein Konzept für die Verknüpfung der Datenbanken des Missionskontrollsystems und der Bordsoftware wird dazu beitragen, die Daten konsistent zu halten und sie nur einmal eingeben zu müssen. Damit sind dann weitere Telekommandos und Telemetrie für den Kleinsatelliten Flying Laptop zu definieren und zu integrieren. Mit diesen können die ausgearbeiteten Konzepte durch die Erstellung von Flugprozeduren in der Prozedursteuerung realisiert werden. Die Simulationsumgebung ermöglicht den Test der Flugprozeduren. Auch die Funkausrüstung kann, wie in dieser Arbeit beschrieben, mit dem realisierten Aufbau auf einfache Art und Weise getestet werden.

Diese Arbeit kann des Weiteren als Grundlage für den Aufbau eines Flatsats verwendet werden. Nicht nur die Mobilität und Fernsteuerung tragen hierzu bei. Vom Missionskontrollsystem bis hin zum Bordrechner ist die gesamte Komman-

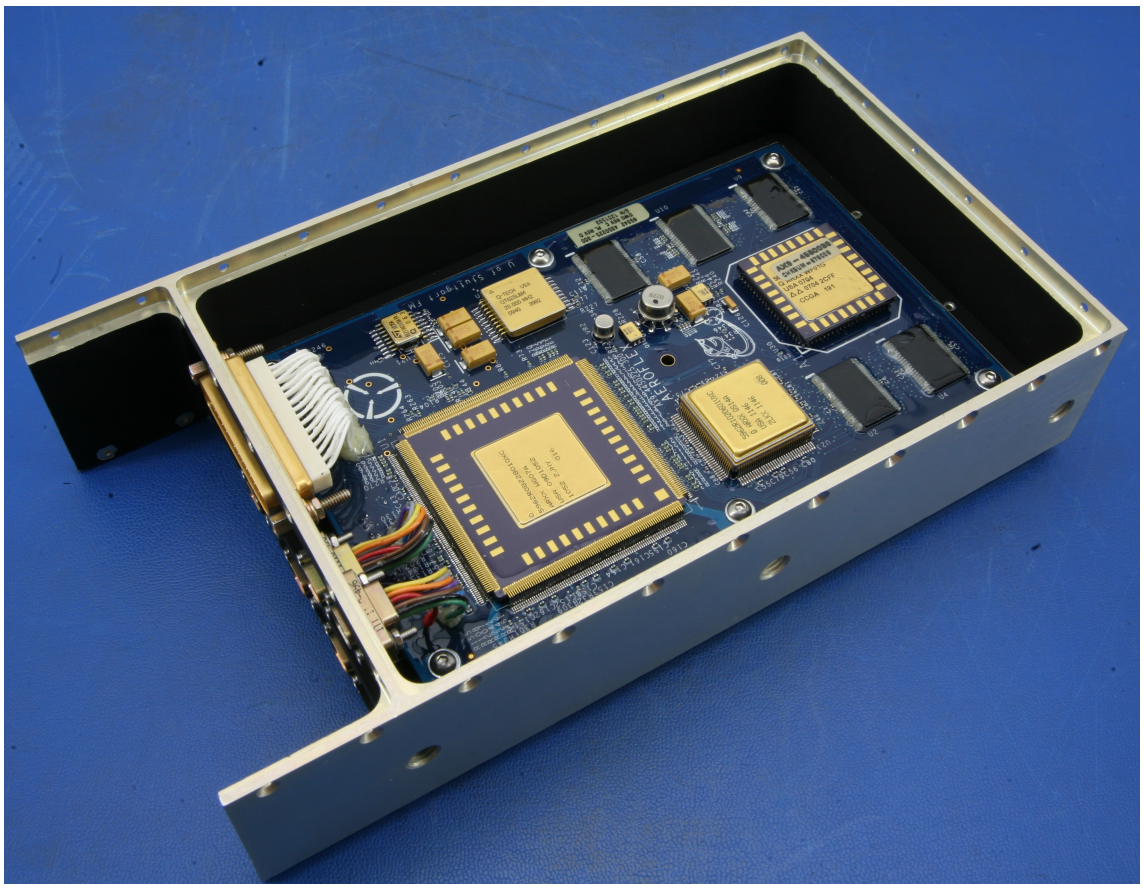


Abbildung 8.1: Flugmodell eines Bordrechnerkerns ©Aeroflex

dierungskette funktionsfähig und getestet. Für einen Flatsat sind ausschließlich die Bordkomponenten anzuschließen und korrekt zu erden. Sobald die PCDU zur Versorgung erster Bordkomponenten eingesetzt oder mit den Solarpaneelen verbunden wird, darf diese nicht mehr durch ein Netzgerät gespeist werden. Stattdessen ist eine Batterie zu wählen, da sich eine solche für den Ausgleich schwankender Lasten besser eignet und sowohl einen Lade- wie auch einen Entladevorgang ermöglicht.

Nach dem Start des Flying Laptops kann der Testaufbau durch Verwendung der Ingenieurmodelle des Bordrechners und der Komponentenmodelle mit der simulierten Weltraumumgebung außerdem für die funktionale Verifikation von Aktualisierungen der Bordsoftware verwendet werden. Da das Hochladen einer fehlerhaften Software zum Versagen des gesamten Satelliten führen kann, sind solche Verifikationsläufe essentiell, um möglichen Schaden abzuwenden. Operatoren können durch Kommandierung dieses Aufbaus vom Missionskontrollsystem aus ausgebildet werden, ohne den realen Satelliten ansteuern zu müssen. Gerade im universitären Bereich ist diese Möglichkeit für Ausbildungs- und Demonstrationszwecke hervorragend geeignet. Auch innerhalb von Lehrveranstaltungen kann davon Gebrauch gemacht werden.

Abschließend lässt sich feststellen, dass die in dieser Arbeit durchgeführten Tests verschiedene Disziplinen bei der Entwicklung des Flying Laptop verknüpft haben. Es wurde also ein Sternpunkt für die Entwicklung, die Verifikation und den Bau des Satelliten geschaffen. Dabei werden industrielle Standards und Anforderungen mit der Ausbildung von Studenten und Absolventen verbunden, was letztendlich einen Mehrwert sowohl für die Industrie wie auch für die Ausbildung bietet.

9 Epilog

Diese Dissertation wurde während meiner Arbeit am Institut für Raumfahrtsysteme in den Jahren 2009 bis 2012 angefertigt. Dabei haben viele Menschen auf unterschiedliche Art und Weise zum Gelingen beigetragen.

Besonderer Dank gilt meinem Doktorvater Prof. Dr. rer. nat. Hans-Peter Röser, der mir viel Eigenverantwortung eingeräumt und mich gleichzeitig in allen Hinsichten unterstützt hat. Durch die Etablierung des Stuttgarter Kleinsatellitenprogramms, die engen Kontakte zur Industrie sowie den Bau des Raumfahrtzentrums Baden-Württemberg hat er vorbildliche Rahmenbedingungen für Forschung und Lehre geschaffen. Dadurch konnte mitunter diese Dissertation entstehen.

Auch den Mitberichtern Prof. Dr.-Ing. Rudolf Benz und Prof. Dr.-Ing. Roger Förstner möchte ich herzlich danken. Das schließt nicht nur die Betreuung dieser Dissertation, sondern auch die hervorragenden Gespräche und Tipps darüber hinaus explizit mit ein. Ich freue mich darauf, auch künftig mit beiden in regelmäßigem Kontakt zu bleiben, um Erfahrungen austauschen zu können.

Ebenfalls großer Dank gilt Prof. Dr.-Ing. Jens Eickhoff, der mir während der Arbeit am Institut für Raumfahrtsysteme viele hilfreiche Tipps bei technischen Fragestellungen geben konnte. Die zahlreichen und vielseitigen Diskussionen waren ein wichtiger Aspekt bei der Entstehung dieser Dissertation. Auch für alle Gespräche darüber hinaus möchte ich mich ausdrücklich bedanken. Die Zusammenarbeit habe ich immer als äußerst fruchtbar und motivierend erlebt.

Dem Kleinsatellitenteam des Instituts für Raumfahrtsysteme möchte ich für die gute Zusammenarbeit danken. Die täglichen Begegnungen haben mich vor allem als Mensch sehr bereichert und geformt. Auch die gemeinsame Arbeit auf technischem Gebiet war immer vielseitig und tiefgehend. Für die letzte Etappe auf dem Weg zum Start des ersten Stuttgarter Kleinsatelliten wünsche ich dem Team alles Gute und viel Erfolg!

Stefan Pflüger, Jürgen Freund, Sebastian Winter, Uli Mohr und Nico Bucher, die diese Dissertation mit den Ergebnissen ihrer Studien- und Diplomarbeiten indirekt unterstützt haben, möchte ich ebenfalls herzlich danken. Der Dank gilt insbesondere

für die vielen fruchtbaren Diskussionen, die gemeinsam entwickelten Ideen, die vielen guten Gespräche und die enorme bewältigte Arbeitslast. Die Zusammenarbeit hat mir sehr viel Freude gemacht und wird mir immer eine schöne Erinnerung sein.

Für die gemeinsame Arbeit an mehreren wissenschaftlichen Veröffentlichungen möchte ich mich außerdem bei Dr.-Ing. Toshinori Kuwahara, Claas Ziemke, Rouven Witt, Ivan Kossev und Alexander Brandt herzlich bedanken. Die wissenschaftliche Arbeit war immer sehr bereichernd für mich. Bei den Konferenzbesuchen durfte ich immer viel Neues erfahren.

Die Kindheit ist Wiege für Bildung und Wissen. Meine Eltern haben daher einen erheblichen Anteil auf dem Weg zu dieser Dissertation. Sie haben mich als Kind in jeglicher Hinsicht gefördert: Ihre Erziehung hat sowohl meine Selbstständigkeit wie auch mein kritisches Denkvermögen extrem geprägt. Bei Fragen hatten und haben sie immer ein offenes Ohr. Daher gilt auch ihnen, besonders im Zusammenhang mit dieser Dissertation, großer Dank.

Meiner Frau Annette möchte ich für ihre Geduld, ihr Verständnis für meine Arbeit, die Entbehrung vieler Stunden sowie für ihre Unterstützung von ganzem Herzen danken. Ohne ihre liebevolle Begleitung über die Jahre hinweg wäre diese Dissertation nicht zustande gekommen.

Literaturverzeichnis

Uwe Baumgarten. *Betriebssysteme: Eine Einführung.* Oldenbourg Wissenschaftsverlag, München, 2007. ISBN 978-3486582116.

Alexander Brandt. *Development of Functional Models for Communication and Thermal Hardware and a Thermal Model for the Flying Laptop Microsatellite Simulation.* Diplomarbeit, Institut für Raumfahrtsysteme, Universität Stuttgart, 2007.

Nico Bucher. *Kommandierung realer Bordkomponenten des Kleinsatelliten Flying Laptop mit dem Missionskontrollsystem.* Studienarbeit, Institut für Raumfahrtsysteme, Universität Stuttgart, 2011.

CCSDS. *Consultative Committee for Space Data Systems 132.0-B-1: TM Space Data Link Protocol.* Washington, DC, USA, 2003a.

CCSDS. *Consultative Committee for Space Data Systems 133.0-B-1: Space Packet Protocol.* Washington, DC, USA, 2003b.

CCSDS. *Consultative Committee for Space Data Systems 230.1-G-1: TC Synchronization and Channel Coding - Summary of Concept and Rationale.* Washington, DC, USA, 2006a.

CCSDS. *Consultative Committee for Space Data Systems 130.1-G-1: TM Synchronization and Channel Coding - Summary of Concept and Rationale.* Washington, DC, USA, 2006b.

CCSDS. *Consultative Committee for Space Data Systems 232.0-B-2: TC Space Data Link Protocol.* Washington, DC, USA, 2010.

Enrico Del Re. *Satellite communications and navigation systems.* Springer, New York City, USA, 2008. ISBN 978-0387475240.

DLR. *Deutsches Zentrum für Luft- und Raumfahrt, ECSS-E-10A: Raumfahrttechnik - Systemtechnik (System Engineering).* Köln, 1996.

DLR. *Deutsches Zentrum für Luft- und Raumfahrt, ECSS-E-10-02A: Raumfahrttechnik, Verifizierung.* Köln, 1998.

ECSS. *European Cooperation for Space Standardization: Ground systems and operations - Telemetry and telecommand packet utilization.* Noordwijk, die Niederlande, 2003.

ECSS. *European Cooperation for Space Standardization: SpaceWire - Links, nodes, routers and networks.* Noordwijk, die Niederlande, 2008.

Jens Eickhoff. *Simulating spacecraft systems.* Springer, Heidelberg, 2009. ISBN 978-3642012761.

Jens Eickhoff. *Onboard Computers, Onboard Software and Satellite Operations.* Springer, Heidelberg, 2011. ISBN 978-3642251696.

Jens Eickhoff, Albert Falke und Hans-Peter Röser. *Model-Based Design and Verification – State of the Art from Galileo Constellation down to Small University Satellites.* Valencia, Spanien, 2006.

Jens Eickhoff, Michael Fritz, Rouven Witt, Ivan Kossev, Mario Kobald und Alexander Brandt. *OpenSimKit - An open-source System Simulation Environment applied to Space Projects.* Madrid, Spanien, 2010a.

Jens Eickhoff, Hans-Peter Röser, Dave Stevenson und Sandi Habinc. *University Satellite Featuring Latest OBC Core & Payload Data Processing Technologies.* Budapest, Ungarn, 2010b.

Jens Eickhoff, Barry Cook, Paul Walker, Sandi Habinc, Rouven Witt und Hans-Peter Röser. *Common board design for the OBC I/O unit and the OBC CCSDS unit of the Stuttgart University Satellite Flying Laptop.* Malta, 2011.

Albert Falke. *Modell-basierte Systemsimulation eines Kleinsatelliten mit einem FPGA-basierten On-board Computer.* Institut für Raumfahrtsysteme, Universität Stuttgart, 2009.

Jürgen Freund. *Selection and configuration of a real-time operating system for the system simulation environment and integration of a SpaceWire interface.* Studienarbeit, Institut für Raumfahrtsysteme, Universität Stuttgart, 2011.

Michael Fritz. *Simulated operations of the microsatellite Flying Laptop with a Mission Operation System.* Studienarbeit, Institut für Raumfahrtsysteme, Universität Stuttgart, 2008.

Michael Fritz. *Aktualisierung und Optimierung eines Thermalmodells des Kleinsatelliten Flying Laptop.* Diplomarbeit, Institut für Thermodynamik der Luft- und Raumfahrt, Universität Stuttgart, 2009.

Michael Fritz, Albert Falke, Toshinori Kuwahara, Hans-Peter Röser, Steve Pearson, Andrew Witts und Jens Eickhoff. *A commercial Procedure Execution Engine completing the command chain of a University Satellite Simulation Infrastructure.* Elsevier, Oxford, Vereinigtes Königreich, 2010a.

Michael Fritz, Hans-Peter Röser, Jens Eickhoff und Simon Reid. *Low Cost Control and Simulation Environment for the Flying Laptop, a University Microsatellite.* Huntsville, Alabama, USA, 2010b.

Aeroflex Gaisler. *CCSDS TM / TC and SpaceWire FPGA - Data Sheet and User's Manual.* Göteborg, Schweden, 2011.

Jiri Gaisler. *A Portable and Fault-Tolerant Microprocessor Based on the SPARC V8 Architecture.* Washington, DC, USA, 2002.

Kristof Gantois, Frederic Teston, Oliver Montenbruck, Pierrik Vuilleumier, Pieter Van Den Braembussche und Markus Markgraf. *Proba-2 Mission and new Technologies Overview.* Domus de Maria, Italien, 2006.

Michael Gook. *PC Hardware Interfaces: A Developer's Reference.* A-List LLC, Wayne, Pennsylvania, USA, 2004. ISBN 978-1931769297.

Georg Grillmayer. *An FPGA based Attitude Control System for the Micro-Satellite Flying Laptop.* Verlag Dr. Hut, München, 2010.

Georg Grillmayer, Michael Lengowski, Sebastian Walz, Hans-Peter Röser, Felix Huber, Maria von Schönermark und Thomas Wegmann. *Flying Laptop - Micro-Satellite of the University of Stuttgart for Earth Observation and Technology Demonstration.* Vancouver, Kanada, 2004.

Michael Jones, Catherine Lannes, Erik Soerensen und Jean Marie Pfennings. *The Network Control and Telemetry Routing System – A Systematic Approach to Ground Station Interfacing.* Tokio, Japan, 1998.

Tomas Kalibera, Pavel Parizek, Ghaith Haddad, Gary Leavens und Jan Vitek. *Challenge Benchmarks for Verification of Real-time Programs.* New York City, USA, 2010.

Kushal Koolwal. *Myths and Realities of Real-Time Linux Software Systems.* Dresden, 2009.

Toshinori Kuwahara. *FPGA-based Reconfigurable On-board Computing Systems for Space Applications.* Institut für Raumfahrtsysteme, Universität Stuttgart, 2010.

Michael Lengowski, Hans-Peter Röser, Richard Haarmann, Ulrich Beyermann und Gregor Gebel. *Mechanical Design of the Micro-Satellite Flying Laptop.* Berlin, 2007.

Wilfried Ley. *Handbook of space technology.* Wiley, Chichester, 2009. ISBN 978-0470697399.

Ulrich Mohr. *Inbetriebnahme des On-Board Computer Engineering Models des Kleinsatelliten Flying Laptop.* Studienarbeit, Institut für Raumfahrtsysteme, Universität Stuttgart, 2011.

NASA. *National Aeronautics and Space Administration: Software Management Guidebook.* Washington, DC, USA, 1996.

Javier Noguero, Gonzalo García Julian und Theresa W. Beech. *Mission Control System for Earth Observation Missions Based on SCOS-2000.* Big Sky, Montana, USA, 2005.

Stefan Pflüger. *Erstellung und Verifizierung von Komponentenmodellen und externen Anbindungen für die Entwicklung der Onboard-Software des Kleinsatelliten Flying Laptop in einer Systemsimulationsumgebung.* Diplomarbeit, Institut für Raumfahrtsysteme, Universität Stuttgart, 2010.

Werner Roddeck. *Einführung in die Mechatronik.* Teubner, Wiesbaden, 3. überarbeitete und ergänzte Auflage, 2006. ISBN 978-3835100718.

Klaus Schilling. *Networked Distributed Pico-Satellite Systems for Earth Observation and Telecommunication Applications.* Samara, Russland, 2009.

Neal Stollon. *On-chip Instrumentation: Design and Debug for Systems on Chip.* Springer, New York City, USA, 2011. ISBN 978-1441975621.

Martin Sweeting. *Small Satellites & Moore's Law: Implications for Earth Observations.* Berlin, 2009.

James Wertz. *Space mission engineering: the new SMAD.* Microcosm Press, 1. Auflage, Hawthorne, Kalifornien, USA, 2011. ISBN 978-1881883159.

Sebastian Winter. *Modellbasierter Test von Lageregelungsalgorithmen des Kleinsatelliten Flying Laptop*. Studienarbeit, Institut für Raumfahrtsysteme, Universität Stuttgart, 2011.

Pamela Zave. *The Operational versus the Conventional Approach to Software Development*. New York City, USA, 1984.

Oliver Zeile. *Entwicklung einer Simulationsumgebung und robuster Algorithmen für das Lage- und Orbitkontrollsystem der Kleinsatelliten Flying Laptop und PERSEUS*. Verlag Dr. Hut, München, 2012.

Daixun Zheng, Tanya Vladimirova und Martin Sweeting. *A CCSDS-Based Communication System for a Single Chip On-Board Computer*. Laurel, Maryland, USA, 2002.

Claas Ziemke, Michael Fritz, Ivan Kossev, Alexander Brandt, Jens Eickhoff und Hans-Peter Röser. *An open-source system simulation framework for small satellite development based on OpenSimKit, QEMU and SystemC*. Berlin, 2011.

Anhang

A Das OSI Schichtenmodell

Das OSI-Schichtenmodell[1] ist ein von der Internationalen Organisation für Normung (ISO) entwickeltes Referenzmodell für Kommunikationsprotokolle, das 1983 veröffentlicht wurde. Im Text der vorliegenden Dissertation wird dazu sowohl im Rahmen der Kommunikation nach CCSDS wie auch bei den Hardwareschnittstellen Bezug genommen. Das OSI-Schichtenmodell besteht aus sieben Schichten. Jeder Schicht ist dabei ein Datenformat und eine Funktion zugeordnet. Abbildung A.1 stellt die sieben Schichten schematisch dar.

Das OSI-Schichtenmodell		
Datenformat	**Schichtname**	**Funktion**
Daten	7. Anwendungen	Netzwerkschnittstelle zur Anwendung
	6. Darstellung	Datenkonvertierung
	5. Sitzung	Handhabung der Verbindung zweier Rechner
Segmente	4. Transport	Segmentierung und Stauvermeidung
Pakete	3. Vermittlung	Logische Adressierung
Übertragunsblöcke	2. Sicherung	Physikalische Adressierung
Bits	1. Bitübertragung	Signalübertragung

Abbildung A.1: Das OSI-Schichtenmodell

[1] aus dem Englischen: Open Systems Interconnection Model

Die Anwendungsschicht beschreibt, wie Anwendungen Zugriff auf das Netzwerk erhalten. Die Darstellungsschicht konvertiert systemabhängige Daten in ein systemunabhängiges Format. Dies schließt auch Konzepte wie Verschlüsselung oder Datenkompression mit ein. Auf der Sitzungsschicht wird die Kommunikation zwischen zwei Geräten gehandhabt. Dabei wird auch dafür gesorgt, dass Zusammenbrüche der Verbindung verhindert werden. Schließlich wird der Datenstrom auf der Transportschicht segmentiert. Damit werden die oberen drei Schichten von den Eigenschaften des Kommunikationsnetzwerks entkoppelt. In der Vermittlungsschicht werden die Daten adressiert. Damit wird eine Weiterleitung der Daten auf ihrem Weg zum Ziel möglich ohne die oben liegenden Schichten zu analysieren. Auf dem Übertragungsweg entstehende Fehler können auf der Sicherungsschicht erkannt werden, beispielsweise durch Checksummen. Oft ist auch eine Fehlerkorrektur vorgesehen. Der Bitübertragungsschicht ist die physikalische Übertragung der Daten zugewiesen. Das schließt auch die Administration des physikalischen Übertragungswegs ein.

B Hardwareschnittstellen

Wie in [Gook 2004] beschrieben, können über Hardwareschnittstellen Informationen zwischen Geräten ausgetauscht werden. Die Information kann dabei auf verschiedene Arten übertragen werden.

- Analoge Informationen sind kontinuierlich in Zeit und Größe.
- Diskrete Informationen beinhalten eine bestimmte Anzahl an Zuständen und sind damit wertdiskret. Oft handelt es sich dabei um zwei Zustände, es können aber auch mehr Zustände sein. Beispielsweise könnte es sich um die Information handeln, ob ein Gerät an oder aus ist. Weitere Zustände wären zum Beispiel, wenn der Status nicht bekannt ist oder sich das Gerät in einem bestimmten Modus befindet.
- Digitale Informationen sind eine spezielle Form der diskreten Information. Mehrere diskrete Informationen können als digitale Information betrachtet werden. Eine analoge Information kann in einem bestimmten Bereich quantisiert und als digitale Information übertragen werden. Die Information ist dadurch nicht mehr wertkontinuierlich.

Um Informationen übertragen zu können, müssen sie nach [Gook 2004] außerdem als Signal vorliegen. Auch hier gibt es analoge, diskrete und digitale Signale. Der Typ des Signals kann dabei ein anderer sein als die damit übertragene Information. Zum Beispiel wandelt ein Modem[2] ein digitales Signal eines Rechners in ein analoges zur Übertragung um und umgekehrt. Die Information ist dabei immer digital. Der zu wählende Typ des Signals hängt unter anderem von der Länge des Übertragungswegs, der Übertragungsrate, der Sicherheit, den Kosten und des Leistungsverbrauchs ab.

In Rechnern werden digitale Informationen meist durch Aneinanderreihung diskreter Signale mit den beiden Stati hoch und tief übertragen und als Binärsignal verarbeitet. Eines dieser diskreten Signale wird dabei Bit genannt. Grundsätzlich gibt es zwei Ansätze, um eine Reihe von Bits zu übertragen.

- Der erste ist die Übertragung über parallele Schnittstellen. Da Prozessoren in aller Regel nur eine bestimme Menge an Bits verarbeiten, werden hier mehrere Bits zeitgleich über verschiedene Leitungen übertragen. Als Beispiel sei

[2] Modem ist ein Kunstwort und durch Aneinanderreihung der Worte Modulator und Demodulator entstanden.

die Centronics-Schnittstelle zur gleichzeitigen Übertragung von 8 Bits über 8 Leitungen genannt, wie sie früher in Druckern zum Einsatz kam.

- Alternativ kommen serielle Schnittstellen zum Einsatz. Dabei werden alle Bits über eine Signalleitung nacheinander übertragen. Da moderne Schnittstellen immer höhere Geschwindigkeiten mit Verlässligkeit in geforderter Höhe bieten, lösen serielle Schnittstellen oft parallele ab, wie das im Falle der Drucker geschehen ist.

Ein weiteres Merkmal der Signalübertragung ist die Symmetrie.

- Unsymmetrische Signale übertragen das Signal einadrig, eine zweite Ader wird dabei als Referenzerdung genutzt. Dadurch kann das Signal durch Störungen von außen beeinflusst werden. Diese Übertragungsart eignet sich daher nur für kurze Distanzen.

- Symmetrische Signale übertragen Signale über zwei Adern. Bei differentieller Übertragung, einer Form der symmetrischen Übertragung, wird das Signal positiv und negativ über zwei verdrillte Adernpaare übertragen. Durch Subtraktion auf der Empfängerseite können so Störungen weitgehend eliminiert werden. Dadurch werden größere Übertragungsdistanzen möglich.

Zuletzt sei die Eigenschaft der Synchronisation genannt. Bei synchronen Signalübertragungen wird die Taktfrequenz der Übertragung abgeglichen. Dies kann passieren, indem Sender und Empfänger denselben Taktgeber verwenden oder indem der Sender dem Empfänger seine Taktfrequenz durch eine dedizierte Leitung mitteilt. Bei asynchronen Übertragungen findet kein Abgleich statt. Hier muss auf andere Weiße sichergestellt werden, dass die Übertragung nicht durch einen zeitlichen Drift des Sender- und Empfängertakts beeinträchtigt wird.

UART[3] ist ein elektronisches Bauelement, das die asynchrone, serielle Übertragung von Daten ermöglicht. Dafür muss beim Sender die gleiche Taktfrequenz eingestellt sein wie beim Empfänger. Um den Drift der beiden Taktgeber zu handhaben, können in einem Paket fünf bis neun Bit gesendet werden. Der Start der Übertragung wird mit einem Startbit gekennzeichnet, das Ende mit einem Stopbit. Ein optionales Paritätsbit zur Fehlererkennung kann an die Daten angehängt werden. Dafür muss der Empfänger ebenfalls wissen, welche Anzahl an Datenbits gesendet wird und ob ein Paritätsbit existiert. Beim Flying Laptop kommen oft Datenpakete

[3]Universal Asynchronous Receiver Transmitter, zu Deutsch etwa: universeller asynchrones Empfänger und Sender

mit 8 Datenbits und keinem Paritätsbit zum Einsatz. Dieses Format heißt 8N1 und ist in Abbildung A.2 abgebildet. Da UART nicht an einen Übertragungsstandard gekoppelt ist, kann einerseits ein unsymmetrisches Signal wie RS-232, andererseits ein symmetrisches Signal wie RS-422 oder RS-485 genutzt werden.

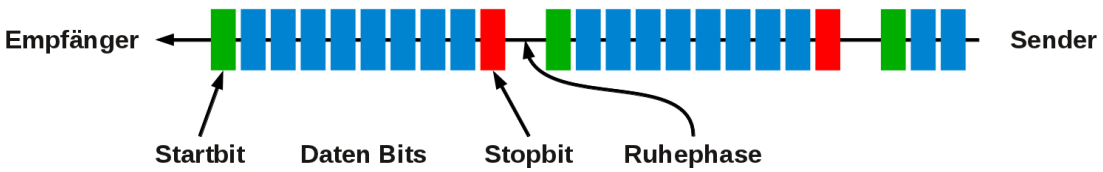


Abbildung A.2: UART Datenübertragung

C Datenübertragung innerhalb des Flying Laptop

Beim Kleinsatelliten Flying Laptop kommen ausschließlich serielle Übertragungsarten zum Einsatz. Die wichtigsten davon werden hier kurz umrissen.

- Aufgrund der hohen geforderten Toleranz gegen elektromagnetische Störungen werden bei Raumfahrzeugen in aller Regel die differentiellen Schnittstellen **RS-422** und **RS-485** zur Übertragung von seriellen Daten verwendet. Diese Schnittstellen sind fast identisch und können miteinander verbunden werden. Es kommen drei verschiedene Übertragungskonfigurationen zum Einsatz:
  - Eine sogenannte Simplex-Verbindung besteht aus einem Sender und einem oder mehreren Empfängern. Charakteristisch für diese Konfiguration ist die unidirektionale Übertragung.
  - Die Vollduplex-Verbindung verbindet zwei Geräte und ermöglicht zeitgleich beidseitiges Senden und Empfangen. Es werden zwei Adernpaare sowie je ein Sender und Empfänger benötigt. Ein Adernpaar verbindet den Sender des einen mit dem Empfänger des anderen Geräts und das zweite Adernpaar wird für die gegenläufige Strecke genutzt.
  - Eine Halbduplex-Verbindung ermöglicht den Zusammenschluss mehrerer Schnittstellen. Dazu werden sowohl Sender wie auch Empfänger mit dem gleichen Adernpaar verbunden. Damit ist zeitgleiches Senden und Empfangen nicht mehr möglich. Eine Halbduplex-Verbindung kann nur mit RS-485 realisiert werden.
- Bei **LVDS**[4] handelt es sich um einen Standard, der beim Flying Laptop nicht in dieser Form zur Anwendung kommt, aber die Grundlage für drei verwendete Schnittstellen bildet. Diese differentielle Schnittstelle ermöglicht durch die Verwendung von Niedrigspannung höhere Datenübertragungsraten. Als Nachteil ist die dadurch ansteigende Störanfälligkeit gegen elektromagnetische Wellen zu nennen. Eine entsprechende Leiterbahnführung und spezielle Kabel können diese Störanfälligkeit stark einschränken, schlagen sich aber in der Komplexität und damit den entstehenden Kosten nieder. LVDS verbindet zwei Geräte miteinander.
- **M-LVDS** ermöglicht den Zusammenschluss mehrerer Schnittstellen und wird beim Flying Laptop hauptsächlich auf dem Nutzlastrechner eingesetzt. Allerdings werden beim Flying Laptop damit nie mehr als zwei Geräte miteinander

[4] Low-Voltage Differential Signaling, zu Deutsch: differentielles Niedrigspannungssignal

verbunden. Die Verbindung zwischen Nutzlastrechner und Bordrechner besteht unter anderem aus solchen M-LVDS Verbindungen, aber auch die Kamerasysteme werden mit dieser Schnittstelle angesteuert.

- **LVDM** ist ein Standard der Firma Texas Instruments und ermöglicht ebenfalls LVDS Multipunkt-Verbindungen. Dabei existieren leichte Abweichungen zum M-LVDS Standard, zum Beispiel ist der Ausgangsstrom unterschiedlich.

- **SpaceWire** ist ein von der European Cooperation for Space Standardization[5] spezifizierter Standard zur synchronen Hochgeschwindigkeitsdatenübertragung [ECSS 2008]. Mit Übertragungsgeschwindigkeiten von bis zu 400 Megabit pro Sekunde können Daten gleichzeitig gesendet und empfangen werden. SpaceWire verwendet ebenfalls Niedrigspannung. Mit SpaceWire sind nur Punkt zu Punkt Verbindungen möglich, allerdings können Daten systematisch weitergeleitet werden. Die hohen Datenübertragungsraten setzen verdrillte und einwandfrei geschirmte Übertragungskabel voraus, um elektromagnetische Störungen zu unterbinden.

[5] zu Deutsch: Europäische Kooperation für Raumfahrtnormung

D Modellphilosophien

Die Modellphilosophie beschreibt die bei der Verifikation und Qualifikation von Raumfahrzeugen benötigten Prototypen. Die Modellphilosophie versucht den Verifikationsanforderungen gerecht zu werden und dabei den Aufwand so weit eingeschränkt wie möglich zu halten. Sie kann sich sowohl auf Komponenten- wie auch Subsystem- oder Systemebene beziehen. Es kommen unterschiedliche Modellphilosophien zum Einsatz. Sie lassen sich nach [DLR 1998] in drei Kategorien einordnen:

- Die erste Kategorie wird Prototyp-Philosophie genannt. Hier wird ein Prototyp gebaut, der nur zur Verifikation und Qualifikation eingesetzt wird und später nicht Teil des Raumfahrzeugs ist. Diese Philosophie bietet den Vorteil, dass das Versagensrisiko durch ausgiebigere Tests sinkt. Außerdem können mehrere Modelle simultan verwendet werden, was zu einer zeitlichen Beschleunigung der Prozesse führen kann. Ferner können Prototypen für Tests auf höheren Ebenen wie der Subsystem- oder Systemebene verwendet werden. Mit Prototypen können auch nach dem Start des Raumfahrzeugs noch Tests am Boden durchgeführt werden, was besonders bei auftauchenden Fehlern von Vorteil sein kann. Die Qualifikation wird in aller Regel mit dem Prototyp durchgeführt, die Abnahme hingegen mit dem fliegenden Exemplar, das in der Fachsprache Flugmodell genannt wird. Typischerweise kommt diese Philosophie bei Missionen zum Einsatz, die komplex gestaltet sind und neue Technologien an Bord mitführen. Auch besondere Anforderungen können zu solchen Prototypen führen. Der Bau von Prototypen verursacht allerdings höhere Kosten.

- Der Prototyp-Philosophie steht die Protoflug-Philosophie gegenüber. Bei diesem Ansatz gibt es keine Prototypen, Verifikation und Qualifikation werden mit den Teilen und Komponenten durchgeführt, die später Teil des Raumfahrzeugs sind. Dadurch besteht ein erhöhtes Versagensrisiko und es kann immer nur an und mit einem Exemplar gearbeitet werden. Am Flugmodell werden Qualifikation und Abnahme durchgeführt. Diese Philosophie kommt in aller Regel dann zum Einsatz, wenn es sich um Standardbauteile für die Raumfahrt handelt, also die Teile schon auf mehreren Missionen erfolgreich eingesetzt wurden und deren Komplexität nicht zu hoch ist. Mit dieser Philosophie können die Kosten niedrig gehalten werden.

- Ein Kompromiss zwischen beiden Kategorien ist die Hybridphilosophie. Hier kommt wie bei der Protoflug-Philosophie auch das Flugmodell zum Einsatz,

jedoch sind die daran durchzuführenden Tests weniger ausgiebig. Kritische Randbedingungen werden dabei auf spezielle Modelle angewandt. Mit dem Flugmodell wird also nur ein Teil der Qualifikationstests durchgeführt. Hinsichtlich Versagensrisiko und Kosten liegt diese Philosophie zwischen den beiden anderen.

Die Modellphilosophie auf Systemebene kann dabei eine andere sein als auf Komponentenebene. Nach [DLR 1996] ergibt sich die Modellphilosophie aus der Verifikationsstrategie, dem Entwicklungsstatus, dem Integrations- und Testprogramm sowie den programmatischen Randbedingungen wie beispielsweise Zeitplan und Kosten. Prototypen und Flugmodelle bilden gemeinsam die sogenannten Modelle. Beim Kleinsatelliten Flying Laptop kommen die folgenden, ebenfalls in [DLR 1998] beschriebenen, Modelle zum Einsatz:

- Das Flugmodell ist das Modell, welches für den Einsatz auf dem Raumfahrzeug vorgesehen ist. An diesem findet der Abnahmetest statt, es wird also überprüft, ob es einsatzfähig ist.
- Die Attrappe ist eine geometrische Konfiguration, mit der das Integrationsverfahren und die Unterbringung von Teilen überprüft werden können. Dabei spielen die für die Attrappe verwendeten Werkstoffe oft keine Rolle.
- Das Entwicklungsmodell ist ein funktionales Modell, bei dem die Größe und die verwendeten Bauteile oft vom Flugmodell abweichen. Es wird eingesetzt, um die Durchführbarkeit eines Designs zu bestätigen.
- Das Ingenieurmodell entspricht in Form, Aufbau und Funktionalität dem Flugmodell. Allerdings werden Abweichungen hinsichtlich Redundanz und hochverlässlichen Teilen oft zugelassen. Es dient der funktionalen Qualifikation.
- Am Qualifikationsmodell werden Qualifikationstests mit Extrembedingungen durchgeführt. Dabei kann es sich um Thermal-Vakuumtests, mechanische Tests wie Vibrationstests, Schocktests, Modalanalysen und Tests mit akustischen Lasten sowie elektromagnetische Verträglichkeitstests handeln. Bei Anwendung der Prototyp-Philosophie ist das Ingenieurmodell oft gleichzeitig Qualifikationsmodell.
- Das Protoflugmodell ist ein Flugmodell, das gleichzeitig als Qualifikationsmodell dient.

- Das Struktur-Thermal-Modell dient der Qualifikation der Struktur und des Thermalmodells. Außerdem werden die dafür verwendeten mathematischen Modelle validiert. Dieses Modell existiert nur auf der Systemebene.

- Mathematische und daraus abgeleitete numerische Modelle kommen auf Rechnern durch die Verwendung entsprechender Simulatoren zum Einsatz.

Durch Anwendung dieser Modelle in unterschiedlichen Kombinationen kommen beim Flying Laptop gleich mehrere Philosophien zum Einsatz. Beispielsweise kommt beim Bordrechnerkern die Prototyp-Philosophie mit Ingenieurmodell und Flugmodell zum Tragen, da die Kosten dafür gedeckt sind und damit ein Exemplar am Boden bleibt, auf dem Patches der Bordsoftware mit angeschlossenem Simulator vor dem Hochladen getestet werden können. Bei den Peripherieplatinen wird ebenfalls diese Philosophie angewandt, jedoch mit zusätzlichem Entwicklungsmodell, da der Hersteller zum ersten Mal ein solches Produkt fertigt. Bei den Reaktionsrädern und den Kameras der Sternsensoren wird dagegen die Protoflug-Philosophie eingesetzt, da Ingenieurmodelle nicht bezahlbar waren. Die Hybridphilosophie findet sich auf Systemebene wieder. So wird für den kompletten Satelliten zwar ein Protoflugmodell genutzt, jedoch finden zusätzliche Tests mit einem Struktur-Thermal-Modell und einer Attrappe statt.

E Unified und System Modeling Language

Die Unified Modeling Language[6], kurz UML, wurde 1997 als Standard offiziell eingeführt, um den Aufbau von objektorientierter Software in Diagrammen darstellen zu können. Dabei werden unterschiedliche Diagrammtypen verwendet. Die Strukturdiagramme beschreiben die Struktur der objektorientierten Software, also beispielsweise Klassen oder Abhängigkeiten in der Software. Die Verhaltensdiagramme beschreiben das Verhalten der Software wie Zustandsautomaten oder Anwendungsfälle. Aus der Unified Modeling Language hat sich die System Modeling Language[7], kurz SysML, entwickelt, die 2007 als Standard veröffentlicht wurde. Die meisten SysML-Diagramme sind wiederverwendete UML-Diagramme. Beide Sprachen ermöglichen die Verknüpfung der erstellten Diagramme mit Anwendungssoftware. So kann zum Beispiel aus UML Quelltext erzeugt und umgekehrt Quelltext als UML Diagramm dargestellt werden.

Das im Rahmen dieser Arbeit verwendete Sequenzdiagramm zählt zu den Verhaltensdiagrammen und gehört zu UML wie auch zu SysML. Es ermöglicht die exakte Darstellung zeitlicher Abläufe in Software und Systemen sowie die dabei stattfindenden Interaktionen zwischen einzelnen Elementen.

[6] zu Deutsch etwa: Vereinheitlichte Modellierungssprache
[7] zu Deutsch etwa: Systembeschreibungssprache

Lebenslauf

Persönliche Daten

Name:	Michael Wolfgang Fritz
Geburtsdatum:	4. Februar 1984
Geburtsort:	Stuttgart
Staatsangehörigkeit:	deutsch
Familienstand:	verheiratet

Bildungsweg

2003 - 2009	Studium der Luft- und Raumfahrttechnik mit den Vertiefungsrichtungen Raumfahrtsysteme und Thermodynamik an der Universität Stuttgart
1994 - 2003	Albert-Schweitzer-Gymnasium Leonberg
1990 - 1994	Grundschule Friolzheim

Berufserfahrung

seit 2012	Entwicklungsingenieur bei der Astrium GmbH - Satellites
2009 - 2012	Wissenschaftlicher Mitarbeiter am Institut für Raumfahrtsysteme
2007 - 2009	Studentische Hilfskraft am Institut für Raumfahrtsysteme
2005 - 2006	Praktikant bei Behr GmbH & Co. KG

Mitgliedschaften

seit 2010	Verein deutscher Ingenieure
seit 2010	Deutsche Gesellschaft für Luft- und Raumfahrt

Veröffentlichungen

13 Beiträge zu wissenschaftlichen Konferenzen

1 Beitrag in einer wissenschaftlichen Zeitschrift

2 Buchbeiträge