CubeSat: SpaceTeamSat1
Preliminary Design I: System Architecture

February 2021

Version 3.1

Revision History

Revision	Date	Author(s)	Description
1.0	27.09.2020	SpaceTeamSat1	First draft: internal PDR
2.0	10.10.2020	SpaceTeamSat1	Implementations after internal PDR. Version sent to external reviewer.
2.1	17.10.2020	SpaceTeamSat1	Implementations before external PDR. EDU µC concept finished (but old EDU Reqs.). OBC not finished.
3.0	20.01.2021	SpaceTeamSat1	New mission design → Raspberry Pi approach.
3.1	07.02.2021	SpaceTeamSat1	Minor changes after PDR I: System Architecture.

Authors

Contribution	Name	Email
Project lead, COBC	Raphael Böckle	raphael.boeckle@spaceteam.at
Co-project lead, System Architecture	David Wagner	david.wagner@spaceteam.at
System Architecture, COBC	Jakob Riepler	jakob.riepler@spaceteam.at
System Architecture, COBC	Thomas Hirschbüchler	thomas.hirschbuechler@spaceteam.at
System Architecture	Patrick Kappl	patrick.kappl@spaceteam.at
System Architecture	Daniel Schloms	daniel.schloms@spaceteam.at
System Architecture	Paul Schmitt	paul.schmitt@spaceteam.at
EPS	David Freismuth	david.freismuth@spaceteam.at
EPS	Nikolas Thiel	nikolas.thiel@spaceteam.at
EDU	Jan Pac	jan.pac@spaceteam.at

Contents

1. Introduction 1
- 1.1. Mission Statement . . . 3
- 1.2. Mission Objectives . . . 3

2. System Requirements 4

3. System Architecture 6
- 3.1. Dataflow . . . 7
  - 3.1.1. Protocol . . . 7
  - 3.1.2. Beacons . . . 8
  - 3.1.3. Uplink . . . 8
  - 3.1.4. Downlink . . . 9
  - 3.1.5. Execution of EDU programs . . . 9
- 3.2. EPS – Electrical Power System . . . 11
  - 3.2.1. State of the Art . . . 11
  - 3.2.2. Architecture . . . 13
  - 3.2.3. Interfaces . . . 17
- 3.3. COBC – Communication Module and On-board Computer . . . 18
  - 3.3.1. RF Module . . . 19
  - 3.3.2. MCU – Microcontroller Unit . . . 19
  - 3.3.3. External Memory . . . 21
  - 3.3.4. Antenna Deployment Mechanism . . . 21
- 3.4. EDU – Education Module . . . 22
  - 3.4.1. Educational Importance . . . 22
  - 3.4.2. Architecture . . . 23
  - 3.4.3. Supporting Systems and Redundancy . . . 24

Bibliography 26

A. Appendix 28
- A.1. Power Budget . . . 28
- A.2. Project plan . . . 31
- A.3. Financial budget . . . 32

List of Acronyms

1U	One-Unit CubeSat ($10 \times 10 \times 10\,\mathrm{cm}^3$ and max. 1.33 kg)
ACK	Acknowledgment (signal)
ADCS	Attitude Determination and Control System
AHS	Allgemeinbildende Höhere Schule
ANT	Antenna System
AX.25	A data link layer protocol
BHS	Berufsbildende Höhere Schule
CAM	Camera module
CCSDS	A file transfer protocol
CDR	Critical Design Review
COM	Communication module
COTS	Commercial Off-The-Shelf
COBC	Communication Module and On-board Computer
CSI	Camera Serial Interface
DS	Deployment Switch
DT	Deployment Timer
EDU	Education module
eMMC	Embedded MultiMediaCard
EPS	Electrical Power System
ESA	European Space Agency
ESERO	European Space Education Resource Office[1]
FEC	Forward Error Correction
FFG	Österreichische Forschungsförderungsgesellschaft (Austrian Research Promotion Agency) [2]
FRAM	Ferroelectric Random Access Memory
FRR	Flight Readiness Review
FX.25	Protocol extension to AX.25
FYS	Fly-Your-Satellite programme of ESA [3]
GNSS	Global Navigation Satellite System
GS	Ground Station
HAF	High-altitude flight
ISS	International Space Station
LEO	Low Earth Orbit
MCU	Microcontroller Unit
MOSFET	Metal Oxide Semiconductor Field Effect Transistor
MPPT	Maximum Power Point Tracking
OBC	On-Board Computer
PCB	Printed Circuit Board
PD	Preliminary Design
PDR	Preliminary Design Review
SC	Solar Cells
RBF	Remove Before Flight pin
RF	Radio Frequency
RTC	Real-Time Clock
SatNOGS	Satellite Networked Open Ground Station
SRAM	Static Random Access Memory
STS1	SpaceTeamSat1 – name of CubeSat mission and CubeSat platform
SW	Software
UCI	Umbilical Cord Interface

1. Introduction

This document serves as the system architecture Preliminary Design (PD) for the CubeSat mission SpaceTeamSat1 (STS1), which is designed by the TU Wien Space Team [4]. It describes the system architecture and presents the basic concepts of the key components of the CubeSat mission. From a functional point of view, the Electrical Power System (EPS) and the Communication Module and On-board Computer (COBC) are the most important components of the CubeSat. Therefore, the main focus lies on these components as well as the dataflow of the complete CubeSat mission – including the Ground Station (GS) infrastructure. Since this document only serves as a PD for the system architecture, any solutions are subject to change once more design work is done and test data is acquired. It needs to be considered that this CubeSat mission is divided into various milestones and that the space flight part is not defined in detail yet:

- Definition of System Architecture (PDR I)
- Definition of subsystems (EPS, COBC and EDU) (individual PDRs)
- Development of subsystems (individual Critical Design Reviews (CDRs))
- Definition of High-altitude flight (HAF) challenges
- HAF to verify CubeSat platform (Flight Readiness Review (FRR))
- Implementation of insights of high-altitude flight
- Preparation of the CubeSat mission for space flight

In Section 1.1 and Section 1.2, the mission statement and the main objectives, on which the CubeSat mission is based on, are presented. In Chapter 2, the top high-level requirements and most relevant system requirements of the mission are stated and described in more detail. These requirements are based on the mission objectives and build the base for the mission concept and the design of the CubeSat. In Chapter 3, an overview of the CubeSat system architecture and its subsystems is given. First, the dataflow from the Earth-based infrastructure to the CubeSat is evaluated for up- and downlink. This information is then used to derive the basic concepts of the subsystems. At the end of the document, the bibliography and Appendix A with supporting documents are attached.

Main properties of the CubeSat mission SpaceTeamSat1

- Mission and CubeSat name: SpaceTeamSat1 (short: STS1)
- 1U ($10 \times 10 \times 10\,\mathrm{cm}^3$ and max. 1.33 kg) CubeSat platform
- STS1 shall operate in Low Earth Orbit (LEO) (350 km–500 km)
- STS1 shall be a lab for students of AHS and BHS
- STS1 shall enable access to the amateur radio community for students
- Educational mission objective:
  - Student teams compete in a space-related coding competition.
  - Python code which runs on the educational payload of the CubeSat (Raspberry Pi) platform.
  - Many possible software scenarios are possible. Therefore various sensors are available for read-out.
  - Gathered data is handed over to participating students for further evaluation of the received data.
  - Student teams shall present their acquired results and therefore gain important insights into space-related technologies.
- The education module (EDU) with its sensors is the educational payload of the CubeSat:
  - Temperature sensor
  - Magnetometers
  - Accelerometers
  - Gyroscopes
  - Dosimeter
  - GNSS module
  - Strain gauges
  - Brightness sensors
  - Cameras

1.1. Mission Statement

Excitement and enthusiasm for engineering and science are important features of a progressive society. They are a consequence of the urge to explore the universe, which is deeply rooted in human nature. Nowadays, space technologies enable humankind to satisfy this urge and reach out further than ever before. However, even though space technologies are deeply ingrained within pop culture and organisations like NASA, ESA, or SpaceX successfully conduct widely publicized missions, a hands-on approach to these topics seems out of reach for most people. Therefore, we want to provide an entry point into space technologies for students of secondary schools by giving them the opportunity to run their own software experiments on our self-developed CubeSat platform, which is "Made in Austria". We hope that this will broaden their educational horizon and thus inspire the next generation of space and science enthusiasts.

1.2. Mission Objectives

The mission objectives build the base for any space-related project as they give reasons why certain tasks need to be executed in space, which are mostly complex and costly. They state the relevance of the mission as well as its importance and allow the derivation of requirements and, later on, the design and operation of the CubeSat. The mission objectives for STS1 are separated into primary and secondary mission objectives, which indicate the relevance of the individual objectives.

Primary objectives:

1. To develop and build a working satellite
2. To operate our self-developed CubeSat
3. To give students a platform to run their self-developed software on a CubeSat
4. To inspire students to be part of a space project

Secondary objectives:

5. To take an image from space
6. To make the gathered data and insights available to the public and especially other CubeSat missions

2. System Requirements

The mission objectives build the base of any CubeSat mission, as described in Chapter 1, and are therefore used for the definition of the system requirements. The presented system requirements in this PD are high-level requirements with the goal of determining necessary subsystems which build the base of the CubeSat platform. All requirements shall be measurable and verifiable. Furthermore, requirements shall answer the "Why?" by specifying the "What?" and not addressing the "How?". For nomenclature reasons, we use the same wording to formulate requirements as in the Fly-Your-Satellite (FYS) design specification [3] by ESA:

- The word "shall" is used to express requirements that have to be met.
- The word "should" is used to express recommendations.
- The word "may" is used to express permissions.

The following system requirements are derived from the mission objectives:

1 The CubeSat shall have a system to generate power and distribute it to all other subsystems.
Note:
It shall be possible that the CubeSat can be powered solely by the solar cells, solely by the batteries, respectively both. The power distribution is only active in the operation phase of the CubeSat mission.

2 The CubeSat shall automatically charge the batteries as long as there is excess energy available.
Note:
Excess energy is the difference between the power provided by the solar cells and the power needed to power the subsystems of the CubeSat.

3 The CubeSat shall automatically transmit a beacon at least once every minute.
Note:
Obviously, this can only be done if enough power is provided.

4 The CubeSat shall be able to receive commands, when not transmitting data.
Note:
The CubeSat shall portion large amounts of data before transmission. After a certain amount of transmitted data the CubeSat has a transmission dead-time, which can be used to receive commands.

5 The CubeSat shall be able to deactivate all systems not necessary for communication once a corresponding command is received.
Note:
Relevant subsystems for communication are the EPS and the COBC.

6 The CubeSat shall operate according to a pre-defined state-chart.
Note:
Note that special attention needs to be paid for the launch phase.

7 The CubeSat shall have a single master, which is located on the COBC.
Note:
The master is the only subsystem that is able to initiate active communication with other subsystems on the CubeSat platform.

8 The primary ground station shall be able to send data and commands to the CubeSat. The primary ground station shall also be able to receive data from the CubeSat.
Note:
The communication capabilities shall be verified by a HAF, where the CubeSat shall be placed, for example, in an airplane or a hot air balloon. Additional GS' should be able to receive data from the CubeSat.

3. System Architecture

The following chapter discusses the basic concepts of the individual subsystems of STS1, namely the Electrical Power System (EPS), the Communication Module and On-board Computer (COBC) and the Education module (EDU). As the received and transmitted data build the base of the whole CubeSat platform, Section 3.1 describes the dataflow from the ground infrastructure to the CubeSat in more detail. In a next step the EPS – power generation and distribution – is investigated. Therefore, Section 3.2 discusses the conceptual design of the EPS in more detail. In a next step the COBC is presented in Section 3.3. The COBC is the main processing- and scheduling unit of the CubeSat and also handles the RF communication. Finally Section 3.4 describes the concept of the EDU. This subsystem is the educational payload of the CubeSat mission and therefore executes the software experiments that are provided by the individual student teams. The EDU is equipped with multiple sensors, including two cameras (CAM), to perform various experiments.

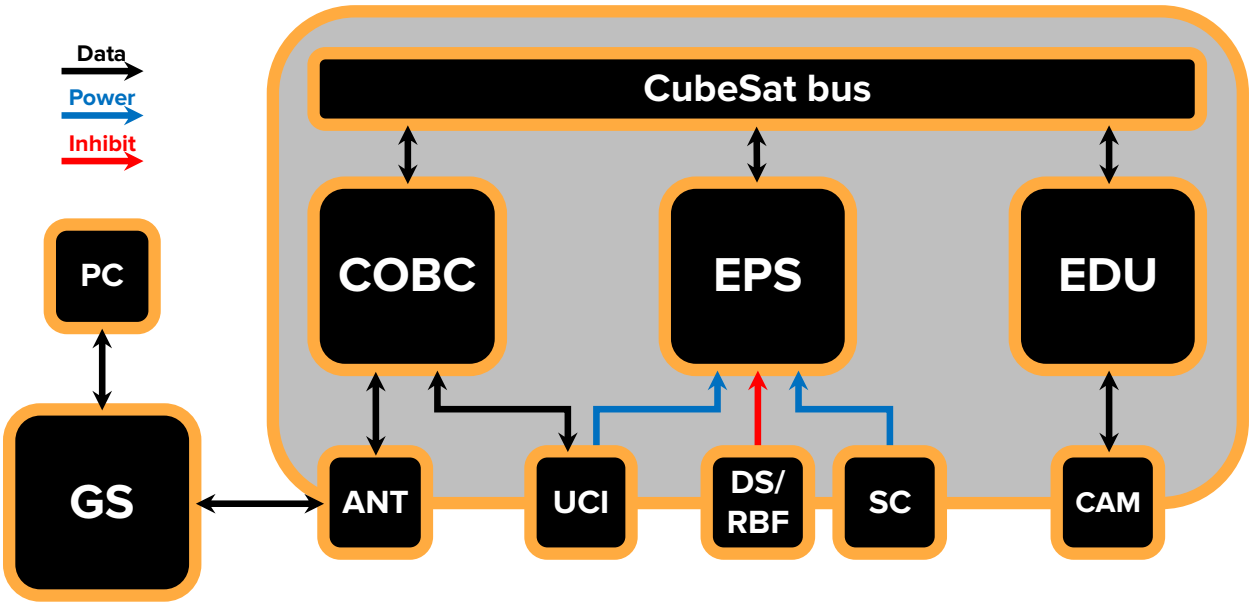

Figure 3.1.: Basic system architecture of STS1. The arrows determine the direction of data- (black), power- (blue) and inhibit-flow (red).

Figure 3.1 shows the basic system architecture of the CubeSat and the associated ground infrastructure. The Antenna System (ANT) is used to communicate with the COBC, which contains the RF module for receiving and transmitting data. The Solar Cells (SC) are directly connected to the EPS and are used to harvest electrical energy. The main Ground Station (GS) can be accessed from any PC on Earth via Internet. The Umbilical Cord Interface (UCI) allows us to re-program the COBC and charge batteries, which are operated by the EPS, after assembling the CubeSat. The Deployment Switches (DS) and the Remove-Before-Flight pin (RBF) ensure that the CubeSat is not powered in the safely stored configuration (e.g. in the chassis of the carrier rocket or in the extractor of the International Space Station (ISS)). Within the EDU module, the Camera module (CAM) can receive image acquisition commands.

3.1. Dataflow

This section discusses the flow of data and commands from the ground infrastructure to the CubeSat and its internal subsystems. The main GS with the amateur radio call sign OE3HMW (operated by Dr. Lars Mehnen) is primarily used to communicate with STS1. On the GS runs CouchDB [5], which buffers all commands and data until the CubeSat can be reached and the relevant data can be transmitted to the CubeSat respectively received. Furthermore, the GS uses Gpredict [6] to determine the timeframe when a RF communication link to the CubeSat can be established.

The following subsections discuss the basic deliberations of the dataflow processes. It needs to be considered that this approach is a preliminary design and needs iterations as further definitions of the hardware and software design are done. Therefore, a dataflow scenario game will be implemented, which allows us to simulate the dataflow by a pen-and-paper game procedure. Such a tool helps us to investigate any circumstances at an early stage of the development of the mission. Note that the following subsections present a starting point for these considerations and further definitions will be implemented at upcoming stages of the mission.

3.1.1. Protocol

As the RF communication should attract the amateur radio community, AX.25 (FX.25) or CCSDS protocols are considered to be used for communicating with the CubeSat. Both protocols provide similar functionality and have different strengths. As both protocols do not provide an application layer protocol, custom software will be necessary for packet decoding anyways, which dampens the "well known protocol"-argument by a bit. Currently, we are still evaluating which of those two options (or a mix of them) is best suited for the STS1 mission. The final decision will be made as soon as the hardware and software topology of the COBC is defined in a more confident matter.

AX.25

The amateur radio "standard" AX.25 based protocol stack mainly provides a well-known protocol and will probably aid with adoption in the amateur radio community. However, it was not built with space links in mind, which requires special considerations to support this highly asymmetric link topology.

CCSDS

The CCSDS protocol stack on the other hand was built specifically with space links and their quirks in mind, providing convolutional channel coding and Forward Error Correction (FEC) for improved error immunity.

Interoperability, Security Measures

To prevent others from hijacking (taking control of) STS1, only packages signed by the TU Wien Space Team are accepted by the CubeSat. This means that other interested amateur radio operators could establish an uplink as well, if the signature is shared. To increase the downlink capabilities, it is considered to cooperate with SatNOGS [7]. If so, the SatNOGS stations would increase the reception area of STS1. An additional self-built SatNOGS station would furthermore allow to create an additional educational platform for students and the TU Wien Space Team. The downlink is neither signed nor encrypted[1], enabling everyone in the amateur radio community to receive beacons and other data from STS1. This helps to increase the area of reception, which is especially useful when

downloading big amounts of data. However, it needs to be considered that a proper management of the acquired data needs to be implemented.

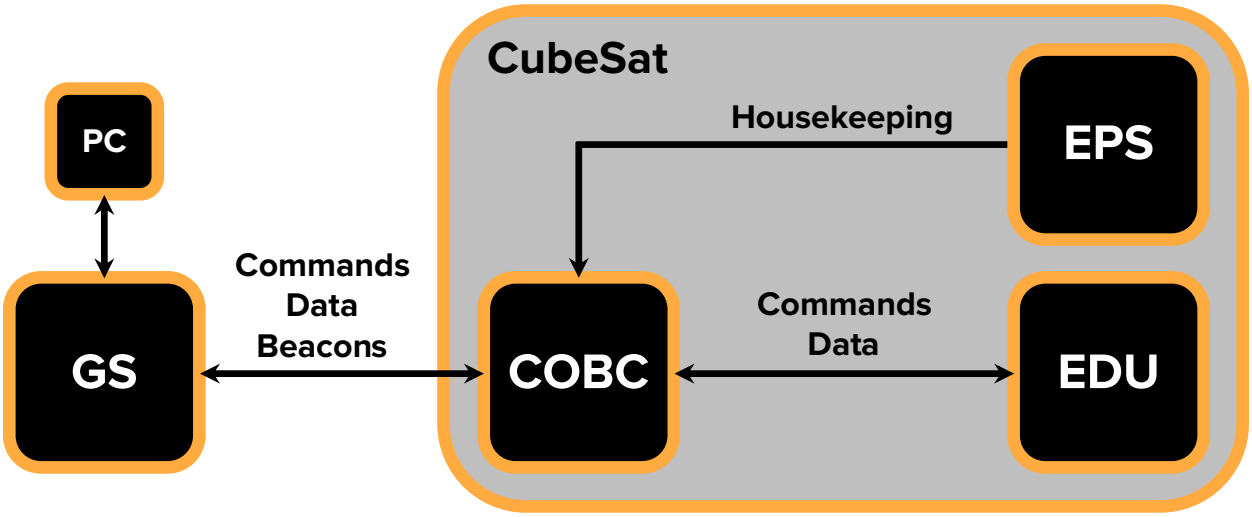

Figure 3.2.: High-level dataflow diagram of STS1. Additionally, the relevant datatypes are mentioned for each data-link between subsystems.

3.1.2. Beacons

The CubeSat shall periodically broadcast housekeeping data (beacons) at least once a minute. However, it is targeted to increase the number of beacons per minute significantly. For this, the Microcontroller Unit (MCU) on the COBC gathers housekeeping data from sensors which are located on the EPS and on the COBC. The data is subsequently stored on the external COBC memory. Before a transmission to the GS, the COBC prepares the beacon for a broadcast by the RF module. Additionally, the beacon data will be stored in the external COBC memory over long periods of time, so that a history of beacon data can be requested by an appropriate command from the GS. Thus, history data can be obtained and evaluated easily.

These beacons can be received by any GS in range of the CubeSat at the broadcasting time. They include telemetry, housekeeping and operation status data.

3.1.3. Uplink

The GS can send signed commands and data via an RF uplink to the RF module of the COBC. The MCU on the COBC interprets these commands and data, and reacts accordingly. Commands will always rely on some sort of acknowledgment. Note that, e.g. CCSDS already has an ACK mechanism implemented.

In the following, the different data types which can be transmitted by the GS to the CubeSat are listed:

COBC firmware update

If a COBC firmware update is transmitted to the CubeSat, the MCU stores it in the external memory of the COBC. Afterwards, a command to the CubeSat is able to trigger a reset of the MCU and

[1]Encryption is neither necessary, nor permitted by amateur radio regulations.

causes it to flash the new firmware stored in the external memory. A detailed concept for managing the firmware versions on the COBC needs to be considered during the COBC development. An initial approach is described in Section 3.3.

EDU programs

If the uploaded data is a program for the EDU module (Raspberry Pi), the uploaded data is stored in the external COBC memory. Note that an additional memory block dedicated for unique program IDs is available as well. The program IDs enable us to implement a queuing mechanism, which lists the program IDs which are executed sequentially. To schedule the execution of a new EDU program, first, the EDU program file has to be transmitted to the CubeSat. Afterwards, the EDU execution queue – consisting of program IDs and additional (not yet defined) parameters (i.e. relative timing) – is transmitted to the CubeSat. As soon as a matching EDU program and program ID are available – and enough power can be provided by the EPS to run the EDU – the COBC initiates the execution of the EDU program. The program ID list is executed sequentially till there are no more operation commands available.

EDU execution queue

If the uploaded data is an execution queue for the EDU programs, this data is stored on the external memory of the COBC. As described in the previous paragraph, this execution queue contains the information in which order to execute the EDU programs. The only possibility to modify the EDU queue from the GS is to overwrite it. Additionally, commands will be implemented to delete this execution queue. More detailed information is given in Section 3.3.2.

3.1.4. Downlink

All data generated by the CubeSat is stored on the external memory managed by the COBC. As outlined in section 3.1.2, the MCU automatically downlinks beacon data periodically, which can be captured by the amateur radio community all over the globe. All other data, including data generated by the EDU module will merely be downlinked after an appropriate command is sent to the CubeSat. As no dedicated details regarding the communication link is defined yet, no further processing steps from the GS to an individual PC can be given. However, this topic needs to be considered in the further development of the COBC and the satellite communication.

3.1.5. Execution of EDU programs

As mentioned previously, the execution of EDU programs depend on the EDU execution queue. For the execution of a EDU program, the program ID of the execution queue and the EDU program need to be compatible, i.e. the unique program ID can be “1a” and “1b”, for example, to be able to execute EDU program “1” two times. Additionally, the EPS needs to be able to provide enough power, which is signaled to the COBC by the EPS itself.

As soon as the program ID within the EDU execution queue and the EDU program are compatible and the power level provided by the EPS is within limits, the COBC starts the Raspberry Pi and transfers the correct EDU program to the Raspberry Pi for execution. The Raspberry Pi on the EDU module will merely execute one EDU program at a time. Therefore, no parallel executions will be possible. While the code is running, the Raspberry Pi has access to several sensors and two cameras to gather data. This data can be processed on the Raspberry Pi’s CPU and GPU. The (processed)

data is temporarily stored on the Raspberry Pi's internal memory. Note that basic limitations within the EDU execution queue need to be set to prevent infinity runs of EDU programs.

For permanent storage and transmission to ground, the data on the Raspberry Pi is transmitted to the external memory of the COBC. Within this operation, it needs to be ensured that the COBC still remains the master of the CubeSat platform. Therefore, the Raspberry Pi shall not send any commands to the MCU, rather an approach to save data to the external memory could be (similar to) the following: After a certain time (parameter given in the EDU queue), the COBC directly pulls data from the Raspberry Pi to its external memory. As this document only acts as PD, no further details are given here. A dataflow scenario games, as mentioned in the introduction of this Section, will help to further develop a reliable saving mechanism. Details regarding this topic will be given in upcoming reviews.

3.2. EPS – Electrical Power System

The Electrical Power System (EPS) is responsible for the continuous generation and provision of electrical power at a single, unregulated voltage level. Energy is harvested solely through solar cells. Excess energy is stored in batteries, in order to continue to power the satellite in orbit sections without light incidence. The EPS also contains system level safety features, like the Remove-Before-Flight pin (RBF), the Deployment Switches (DS) and a Deployment Timer (DT). All of these features shall prevent a premature activation, while the CubeSat is still mounted onto the launch vehicle. Housekeeping data like battery voltage, battery temperature, etc. is collected by the EPS, and made available to the COBC subsystem through a housekeeping data interface.

The following sections lay out state of the art of commercial EPS' (Section 3.2.1), the architecture of the proposed concept (Section 3.2.2) and the interfaces (Section 3.2.3) of the EPS.

3.2.1. State of the Art

As virtually any CubeSat hardware relies on electrical power, many commercial providers have gone through the effort of developing a ready-to-use Commercial Off-The-Shelf (COTS) EPS. In Table 3.1 a comparison of a selection of providers and the estimated STS1 requirements are listed. It can be seen that the commercial products are over-engineered in the majority of aspects. Especially when compared to the selection of available voltage rails and the capacity.

It has been decided to not implement a COTS EPS and rather develop the EPS in house for the following reasons:

1. Building and sharing of know-how
2. Reduction of mission cost
3. Experiences from previous CubeSat missions suggest, that the EPS should be as simple as possible, should contain as few features as possible and should not contain any microcontrollers. This circumstance is not applicable to most available COTS EPS.

	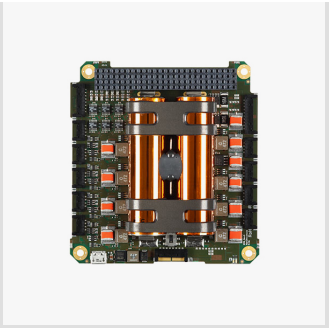		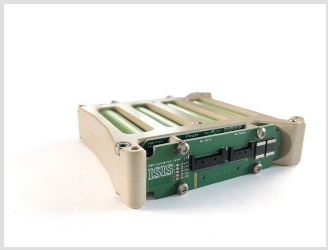	
	NanoAvionics[8]	**EnduroSat**[9]	**ISISpace**[10]	**Required for STS1**
Voltage Rails	BAT, 3.3 V, 5 V, configurable 3 V–18 V	BAT, 3.3 V, 5 V	BAT, 3.3 V, 5 V configurable	single voltage > 5 V
Capacity	11 W h	10 W h	22.5 W h	~ 2.33 W h
Power	20 W per rail	-	20 W per rail	~ 8.89 W
Mass	300 g	208 g	184 g	~ 300 g
Misc	RTC Software configurable	Serial Interfaces 6 General Purpose Outputs RBF and DS optional	Serial Interfaces FRAM based MCU Supervisor	No software DS Deployment Timer
Heaters	yes	yes	yes	no
Price	On Request	2.500€	3.300€	-

Table 3.1.: Comparison of the EPS' of a selection of providers and the estimated requirements of STS1. See Appendix A.1 for the source of the estimations.

3.2.2. Architecture

The system requirements 1 and 2, stated in Chapter 2, allow the deduction of the architecture depicted in Figure 3.4. The components and their functions are elaborated upon in the following list:

1. **External Charger**: Connects to the Umbilical Cord Interface (UCI) of the satellite and allows charging of the CubeSat batteries on the ground. It is possible to charge the batteries, even when the safety features of the CubeSat (i.e. DS, DT and RBF) are in effect. However, it is not possible to power other satellite subsystems via the external charger, while the safety features are in effect.
2. **Bus Gate**: Switch, that permanently connects the EPS-internal power bus to the system bus interface, once the deployment timer signal has been received.
3. **Battery Pack**: Contains battery cells that store excess energy generated by the solar cells. Allows operation of the satellite, when no energy is harvested by the solar cells.
4. **Temperature Sensor**: Measures the temperature of the battery pack, and distributes this information to the housekeeping block and to the battery controller.
5. **Solar Cell**: Allows generation of electrical energy in orbit.
6. **Deployment Switch**: Switches that are actuated, when the satellite is inserted in the launch device. When actuated, the power flow of solar cells and battery is interrupted.
7. **Remove Before Flight Pin**: While this pin is inserted into the satellite, power flow of battery and solar cells is interrupted.
8. **Voltage and Current Sensor**: Measures voltage and current readings. This information is collected by the housekeeping block. During the development it will be evaluated in detail which parameters are required.
9. **Battery Controller**: Controls charge- and discharge-rate of the battery pack. Monitors battery voltage level and temperature. Provides the "Battery OK" signal to the CubeSat Bus Interface.
10. **MPPT Controller**: Keeps the connected solar cell at its maximum point of power by utilizing a Maximum Point of Power Tracking mechanism (MPPT), in order to optimize energy generation. Provides a regulated voltage at its output.
11. **Deployment Timer**: Timer that is started once all DS are no longer actuated and the RBF has been removed. After expiration of the timer, an electrical signal is emitted.
12. **Housekeeping**: Collects data of the sensors on the EPS and provides this data via an interface.
13. **Umbilical Cord Interface (UCI)**: Interface to the umbilical cord connector. This connector allows ground based debugging and battery charging after assembly of the CubeSat.
14. **CubeSat Bus Interface**: Interface to the CubeSat system bus.

System requirement 1 demands, that either the solar cells or the battery must be able to drive the power bus, if the respective other part is currently not able to provide power. This circumstance implies four states of operation. These states and the respective conditions of battery and solar cells are depicted in Table 3.2. The battery state *operable* and *not operable* refer to the ability of the battery to provide electrical power. The state *not operable* could be caused by a low charge level. In this case, this state is temporary, until the charge level is high enough again. Another reason for the *not operable* state is an internal short of the battery. This fault implies, that the battery is stuck in the *not operable* state, and can not recover.

		Battery	
		Operable	Not operable
Solar cells	Generating	High power	Solar power only
	Not generating	Battery power only	No power

Table 3.2.: Combination of two possible states of battery and solar cells result in four possible states of operation.

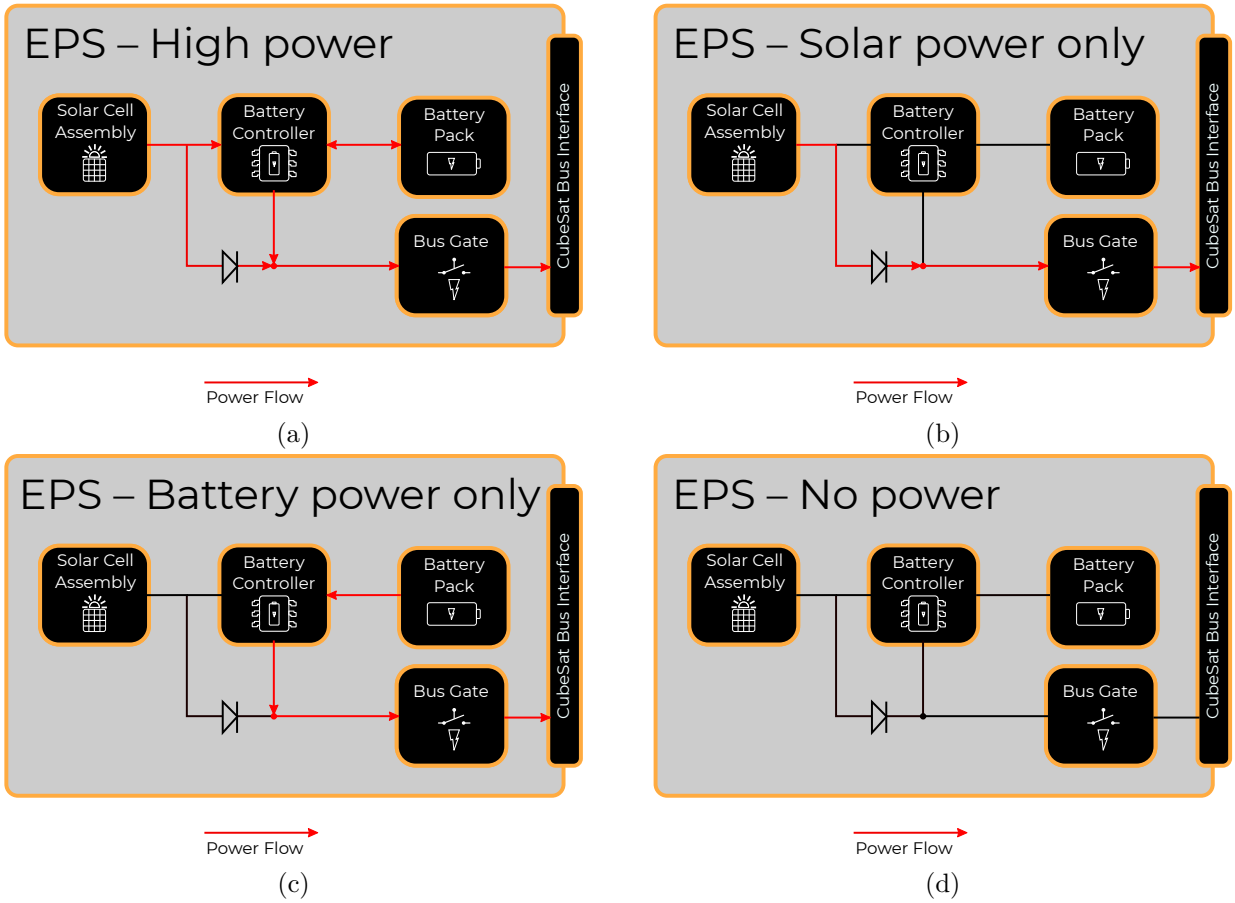

Figure 3.3.: The power flow in the EPS during the respective states of operation.

The definition of the transition between these operation states is not within the scope of this document. It shall be specified latest in the upcoming EPS-PDR. The transition specification shall be verified by a "dataflow" game, where participants assume the role of a CubeSat subsystem. The operation of the CubeSat is then simulated by the participants (see Chapter 3 for details).

High power In this state, the solar cells generate power. The battery is operable and is being discharged or charged, depending on the current power demand of the subsystems and solar cell output. The power bus is driven by both, solar cells and battery. See Figure 3.3a for reference.

Solar power only Only the solar cells provide power. This may be caused by a low battery charge level, or a permanent battery fault. The power bus is only driven by the solar cells and the battery charger is bypassed over a diode. This means that the provided voltage on the bus may be volatile as the light incidence on the solar cells varies strongly over the orbit. The "Battery OK" signal issues this state to other systems, so they may act accordingly. See Figure 3.3b for reference.

Battery power only The solar cells do not provide any energy. This may be the case, if the CubeSat traverses the shade of the earth, or the solar cell contacts corrode. The power bus is driven by the battery charger, and in extension by the battery. This continues as long as there is charge left, or the solar cells provide power again. See Figure 3.3c for reference.

No power Solar cells and battery do not provide any power. In this state the power bus is not driven. The EPS does not provide any energy. This results in no operation of the CubeSat. See Figure 3.3d for reference.

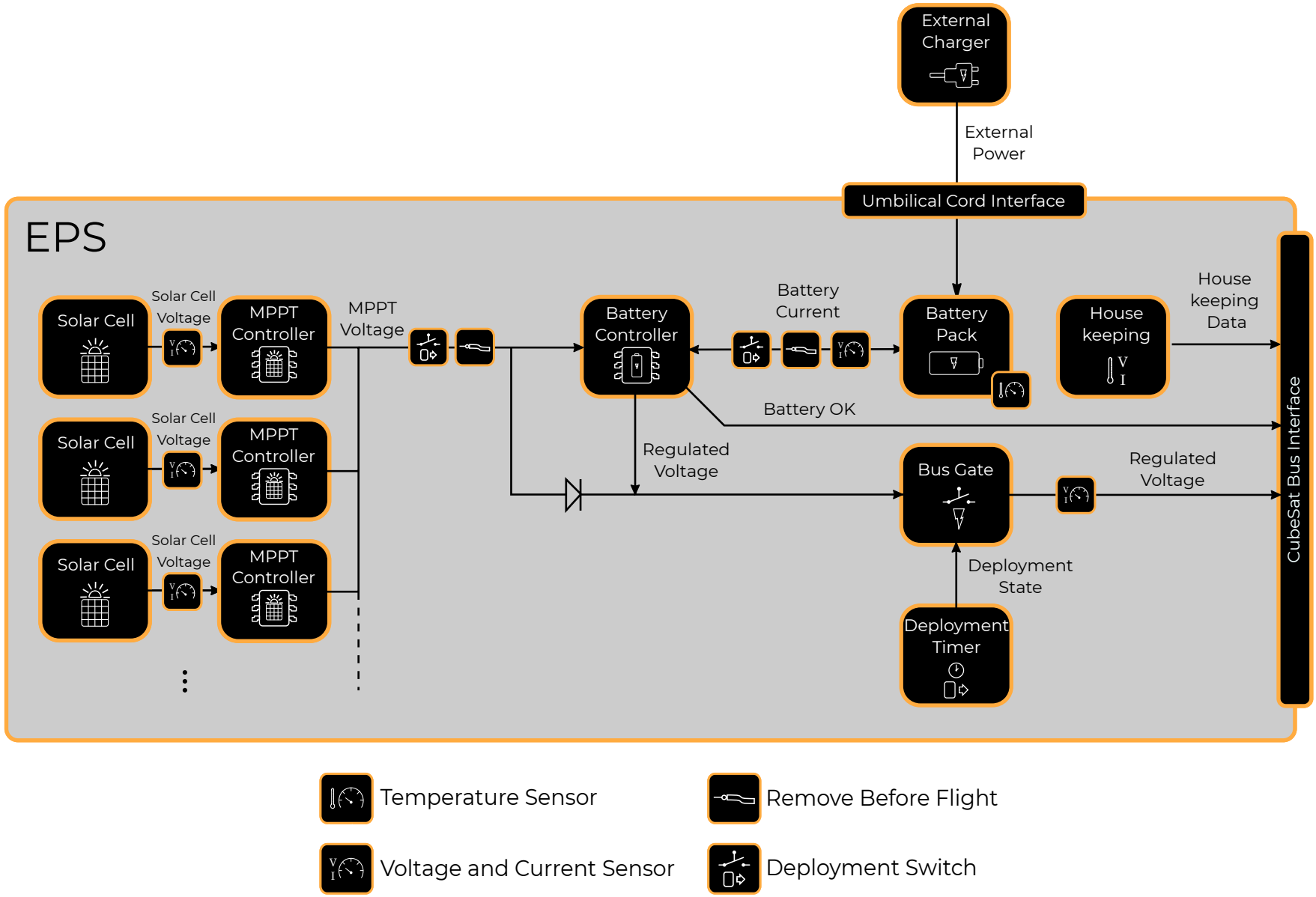

Figure 3.4.: The block diagram of the EPS.

3.2.3. Interfaces

The EPS has an interface to the CubeSat system bus, where it provides a single power bus, a "Battery OK" signal and the housekeeping data. The EPS does not guarantee a fixed voltage level on the power bus. Other subsystems therefore have to implement additional voltage regulators to power its individual components. Furthermore, the EPS connects to the UCI. Here, the external charging port is connected. A listing of EPS in- and outputs (from an EPS point of view) can be seen in table 3.3 and table 3.4.

EPS System Bus Interface pins		
Name	**Direction**	**Description**
Power Bus	Out	Provides power to other systems
Housekeeping data	In/Out	Interface for a serial communication protocol that allows polling of housekeeping data
Battery OK	Out	Signal, that indicates, whether the battery is currently able to provide power.

Table 3.3.: Pins which connect the EPS to the system bus.

EPS Umbilical Cord pins		
Name	**Direction**	**Description**
Cell X +	In	Positive terminal of battery cell X
Cell X −	In	Negative terminal of battery cell X

Table 3.4.: Pins which connect the EPS to the umbilical cord interface (UCI).

3.3. COBC – Communication Module and On-board Computer

The COBC of the CubeSat STS1 combines two classical subsystems which are key components in every satellite mission: the Communication module (COM) and the On-Board Computer (OBC). The COM is responsible for any received and transmitted data and the OBC mainly schedules any activities on the CubeSat. Additionally, the OBC acts as the master module of the CubeSat platform by managing memory access and operating the EDU. This also concludes that the COBC is always the first subsystem which is powered up if energy becomes available. Other CubeSat components (except the EPS) are then enabled sequentially by the COBC, this includes the power up of the EDU module. Figure 3.5 shows the main components of the COBC system.

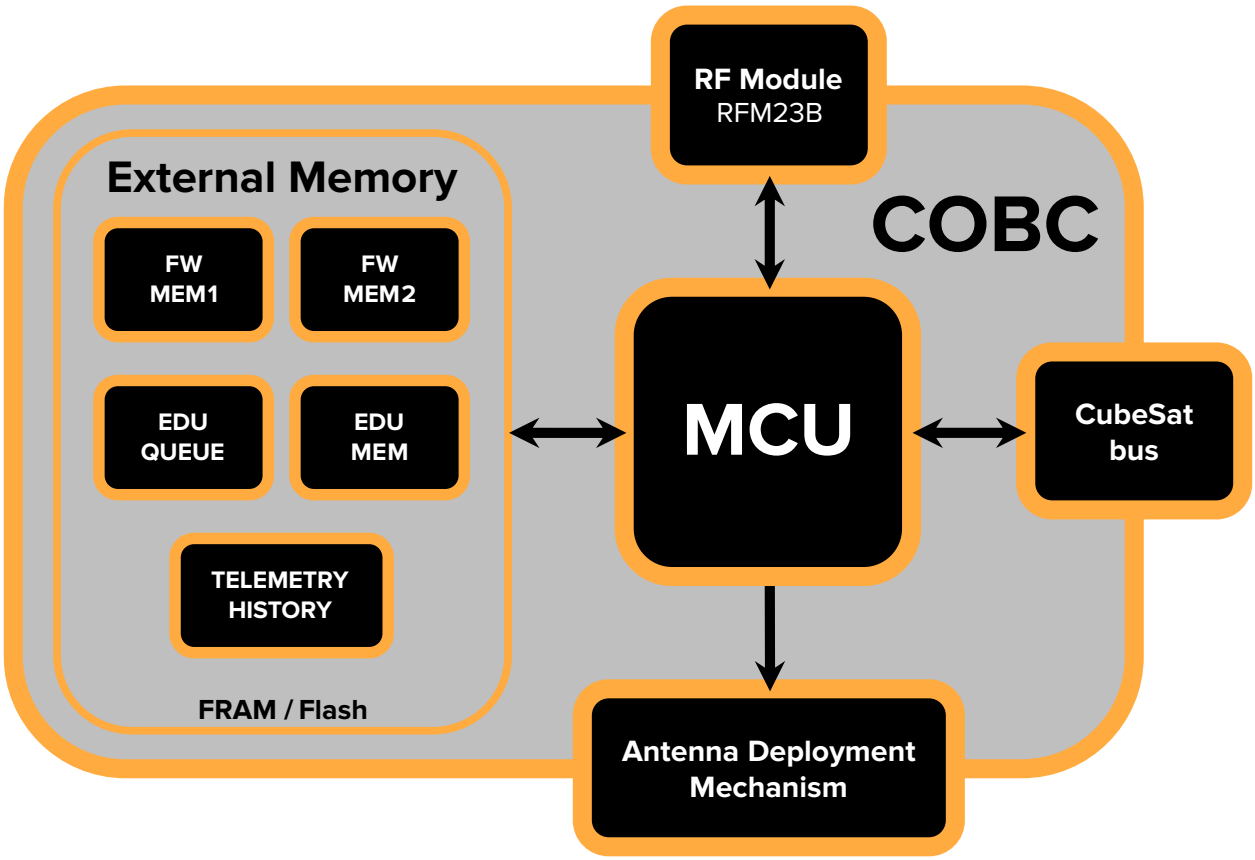

Figure 3.5.: Block diagram of the COBC with its main components. Furthermore, communication links, external interfaces, and the structure of the external memory are shown.

The Microcontroller Unit (MCU) is the COBC's core and handles the entire dataflow on board the CubeSat. This means analyzing incoming and preparing outgoing data from and to the RF module (presumably RFM23B [11]), managing all data access – read, write and erase – to the external memories – based on (proposed) FRAM and Flash – and handling data exchange with the EDU module. Details regarding the external memory on the COBC are given in Section 3.3.3. Note that the MCU also schedules the operation of the EDU module. Another important feature is that the antenna deployment mechanism is triggered by the MCU as well.

The proposed concept combines the approach of an individual COM and OBC subsystem. To determine if separate systems are necessary, the dataflow, which is described in section 3.1, was thoroughly discussed. The main argument is that the capabilities of this CubeSat mission do not require separate COM and OBC subsystems, as these functionalities can be combined on a single subsystem: the COBC. The COBC section is divided into four subsections: Section 3.3.1 describes the functionality of the RF module. Further, Section 3.3.2 gives an insight on the MCU itself, whereas the associated

external memories are discussed in Section 3.3.3. Finally, Section 3.3.4 describes the concept of the antenna deployment mechanism.

3.3.1. RF Module

The RF module is the active part of the RF front end, which receives and transmits a binary stream from and to the GS via an RF link and forwards these data to the MCU for further processing. The RF stream is modulated onto an amateur radio frequency band (70 cm) carrier. Note that the RF link works in half-duplex operation to reduce complexity in hardware and management (software implementation). To be more explicit, this prevents any switching of antennas/transceivers and means that the CubeSat cannot receive and transmit data at the same time. A custom cross-dipole (made of metal ruler tape) will be built and tuned to catch both polarization directions (via a power combiner). Additionally, a robust impedance matching is considered between the antenna and the transceiver.

As briefly introduced in Section 3.1, the whole RF link communication shall be amateur radio compliant. This means that the modulation scheme and filtering needs to be implemented according to amateur radio standards. Furthermore, it shall be possible to turn off the RF module by a command from the GS. This prevents the CubeSat from performing any further RF communication in the case that the used communication frequency is legally not usable anymore.

Although this document serves as a PD, it is considered to use the RFM23B [11], as this transceiver is already operational in Pegasus' "Stacy" communication module, which is a part of a former CubeSat project of the TU Wien Space Team. Furthermore, another CubeSat mission [12] successfully used the proposed RFM23B module as well. Nevertheless, proper investigations are required to verify the RFM23B for the use case of this CubeSat mission.

3.3.2. MCU – Microcontroller Unit

The MCU is the brain of the CubeSat. It handles all telemetry and housekeeping tasks required to keep the satellite operational. The functionality can be divided into the following main categories:

- System startup/initialization
- Communication
- System health data collection
- Scheduling of the operation time of the EDU

System startup/initialization

The first task of the MCU after deployment and power recovery is to initialize the internal peripherals and all external components which are connected to the MCU. If not explicitly deactivated, the antenna deployment mechanism is triggered (details can be found in Section 3.3.4). After a successfully verified antenna deployment[2], a command can be transmitted to the CubeSat to disable the antenna deployment mechanism. This ensures reduced power consumption during the regular boot operation of the CubeSat infrastructure. The deactivation of the antenna deployment mechanism is stored as a magic value in the COBC's external memory, including a checksum to prevent accidental deactivation due to random bit flips caused by radiation. Note that, any other execution of tasks (except the transmission of beacons) is triggered or managed from the GS, respectively. Since this document serves as a PD no dedicated assignment of data to FRAM or Flash is stated.

[2]Will be manually verified from the GS using metrics like signal strength.

Communication

The MCU also serves as the central communication hub of the CubeSat. It receives data from the RF and EDU modules and polls various sensor data – from sensors on the EPS and COBC – for the generation of the beacon data. The telemetry beacon is generated from said data in regular intervals (at least once a minute) and is sent to the RF module for transmission. It needs to be considered that the RF module merely receives packages and does not perform any integrity check. Therefore, the RF module just forwards the data to the COBC, where uplink data is first checked for data integrity and authenticity (Forward Error Correction (FEC) and signature) by the MCU. There are three possible types of uplink data: commands to be processed by the COBC, files (e.g. Python scripts or supporting data) meant for the EDU subsystem (including EDU execution queue files) and firmware updates for the COBC's MCU.

Commands are processed directly on the MCU as soon as resources for processing are available and downlink data is scheduled as soon as a command is successfully received. The scheduled downlink data is either a simple ACK or a response which in turn can require an ACK from the ground station to signal successful reception depending on the exact type of command and data. As already described in Section 3.1 a mechanism needs to be implemented to prevent any loops, which might occur due to issues with any acknowledgment mechanisms.

Files for the EDU subsystem are stored on the external memory on the COBC. Note that the MCU is capable to transfer these files to the EDU for the execution on the Raspberry Pi. This also concludes that the MCU acts as master and the EDU as slave. This is of major importance to avoid any conflicts of interest between these two processing units.

Firmware updates ensure that the MCU on the COBC itself can be updated. Therefore, the firmware version on the MCU and the latest firmware version on the external memory are compared. It needs to be ensured that a reliable concept is developed to conduct these tasks. A promising procedure, which inspires us, is the implementation, which is done on the Mars Rover Curiosity [13]. If the firmware version on the MCU is not equal to the firmware version on the external memory and an according command is received, the firmware version on the external memory gets flashed onto the MCU by resetting the MCU itself. Afterwards the MCU reboots with the new firmware. The Mars Rover Curiosity has an additional firmware image of the old firmware version as a fallback. Details will be described as soon as the COBC development progresses further.

In the other direction, things work similarly as the COBC can pull data into the external memory from the Raspberry Pi memory. These data can then be transmitted to the GS via a command sent from the GS to the CubeSat. As the CubeSat does not send any data (except for beacons) without an explicit request from the GS, situations where no uplink is possible due to a saturated downlink can be avoided. Downlink packets are queued in a priority-based queue as to not block beacons and commands with less important file transfers. For further protocol information see Section 3.1.

System health data collection

The third task of the MCU is to collect and store sensor data related to overall system health. This primarily consists of data generated by the EPS (e.g. bus current and voltage, solar cell status, battery voltage, charge/discharge current, battery state, temperatures, etc.) but also locally generated statistics like uplink packet FEC and re-transmission counts as well as signal strength, to name a few examples. Depending on the historical importance of data, it is either stored in (volatile) SRAM or (nonvolatile) FRAM. Apart from the regular telemetry packets, it is possible to request current and historical data directly. Long term data can be down sampled automatically after a certain amount of time to not loose old but not yet retrieved data in case the GS is unable to reach the satellite for extended time periods.

Scheduling EDU programs by queuing

As described in Section 3.1, the external memory contains an execution queue, which determines the order of operation of individual student software experiments on the EDU module. This allows us to keep an overview of the planned and already executed Python scripts. Furthermore, an organized handling of scripts and data is ensured. In the case of a power outage, the Python script will be executed at a later time as the queue remains in the external memory until the COBC obtains data from the EDU module. A more detailed definition will be done as soon as the development of the COBC progresses further. Additionally, this ensures that the MCU does not get corrupted by any rescheduling. Note that this also increases the level of reliability by reducing the complexity of the scheduling in comparison to a time-based mechanism.

3.3.3. External Memory

As shown in Figure 3.5, the external memory is divided into six main blocks: The FW MEM1 and FW MEM2 contain firmware updates for the COBC itself. Two separate memory blocks are necessary to implement the proposed concept which contains new firmware updates as well as an initial firmware version as a fallback. The EDU-relevant memory cells are EDU QUEUE and EDU MEM. EDU MEM contains the uploaded EDU programs and (processed) data which was generated by the execution of EDU programs, whereas EDU QUEUE contains the queuing procedures for student software experiments stored in EDU MEM. Finally, TELEMETRY HISTORY stores the telemetry and housekeeping data which is necessary for the beacon generation. All mentioned memory blocks are realized as FRAM or Flash. Details regarding its implementation will be done as the development of the COBC progresses further.

3.3.4. Antenna Deployment Mechanism

In the very beginning of the mission the antennas are wrapped around the CubeSat and held in place by a plastic string. This string must be severed in order for the antennas to deploy, making the CubeSat ready for RF communication. The antenna deployment mechanism accomplishes this by running a current through a resister which heats up by resistive heating and melts the string. Supplying power from the EPS to the resistor is done by a MOSFET which was already used in previous CubeSat missions, e.g. Pegasus [14].

The detailed management – activation at startup and deactivation by a telecommand – is explained in more detail in Section 3.3.2.

The following list gives a brief overview of some relevant features of the antenna deployment mechanism:

- According to first tests – where PLA zipties were melted – approximately 8.5 Ws–25 Ws are required.
- MOSFET: triggered by COBC
- The Antennas will also be tested in the undeployed position. Electrically isolating them from the frame may help reduce losses in this case.

3.4. EDU – Education Module

The education module (EDU) is the platform on which students can run their software experiments. It consists of a Raspberry Pi which has access to various sensors, including two cameras. The generated (and processed) data will be transferred to the COBC and stored on its external memory where it can be downloaded via a command. After a successful download, the data will be processed and handed over to the student teams to analyze it. The CubeSat is designed such that the EDU module is an independent payload, i.e. the CubeSat is fully operational without the EDU module or in the case of a complete failure of the module.

The EDU section is divided into three subsections: Section 3.4.1 gives an outline how the educational aspect of the mission fits into the global and national educational landscape. In Section 3.4.2 the different components of the EDU module and the data flow between them is discussed. Finally, Section 3.4.3 gives insights into thoughts about a supporting Attitude Determination and Control System (ADCS) and any redundancy concepts of the EDU module.

3.4.1. Educational Importance

Due to the increasing popularity of Commercial Off-The-Shelf (COTS) components in space-related missions, the number of missions with an educational[3] goal is steadily increasing. For example, in ESA's Astro Pi Challenge [15], students are able to run their code on a Raspberry Pi computer on-board the ISS. Furthermore, several educational CubeSat missions are currently in development or already in orbit, such as SelfieSat [16] or ArduSat [17].

Even though there are already established space-related educational missions, a hands-on approach seems out of reach for most people and thus we want to further lower the entry barrier for students in Austria and offer an approachable space education mission which will hopefully inspire the next generation of space and science enthusiasts. In comparison to the above-mentioned missions, STS1 will offer students an opportunity to develop their own software in close guidance with the TU Wien Space Team and run their code on an independent spacecraft in orbit around Earth. Furthermore, their code will be able to gather data during a prolonged period of time which allows more sophisticated software experiments as in comparison to ESA's Astro Pi Challenge where the runtime is limited to 3 hours. During the software development phase, students will be able to test their code on development kits which will further enhance their coding experience and give them the possibility of debugging their code. The supporting members of the TU Wien Space Team will be approachable during the whole development phase and can give valuable support to the students in their native language.

The EDU module of STS1 uses the popular Raspberry Pi platform [18]. This will further lower the entry barrier, both for teachers and students, as lots of resources about the platform are freely available on the Internet and some participants will already be familiar with the platform. In addition, we will try to keep the coding experience as accessible as possible by hosting workshops and providing libraries for the most common operations on the EDU module, i.e. for sensor readouts and standard commands for data processing. Furthermore, the Raspberry Pi platform will allow students to code their own software experiments in Python, which is nowadays one of the most popular [19] programming languages, especially for trending fields such as data science or machine learning. Thus, any familiarity with Python will greatly benefit the students in their later academic or professional careers.

Sensors under current consideration are: temperature sensor, magnetic field sensors, acceleration sensors, gyroscopes, GNSS receiver, cameras, strain gauges, brightness sensors and a radiation sensor. Possible student experiments with these sensors are for example mapping the radiation environment of

[3] In the following, every mission with a goal to spark interest in science and space is considered an 'educational' mission.

low Earth orbit (LEO) or determining how fast the CubeSat is spinning. Students who will participate in STS1's educational mission will gain hands-on experience in writing actual space software, which will advance their problem solving and teamwork skills. The whole mission will be held in form of a competition, where we expect to gain the support of Austrian space companies and space celebrities for honouring the winning teams. However, we will ensure that all participating student teams, not only the winning teams, will reach their software development goals.

As our educational mission seems in line with already well-established educational organisations in Austria, such as the Österreichische Forschungsförderungsgesellschaft FFG (Austrian Research Promotion Agency) [2] or the European Space Education Resource Office (ESERO) [1], we will get in contact with these and try to establish suitable cooperations, similar to how the TU Wien Space Team is already supporting ESERO's CanSat Challenge [20].

3.4.2. Architecture

Figure 3.6 shows the main components of the EDU module.

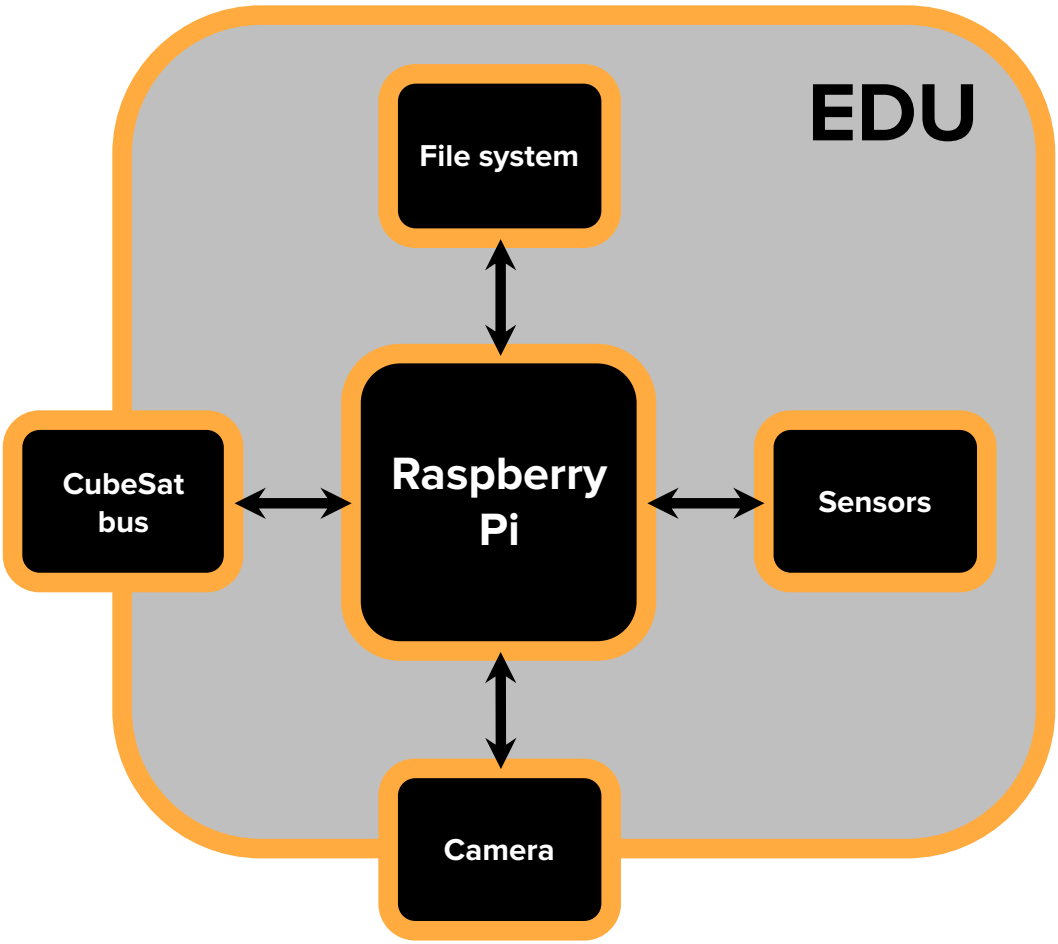

Figure 3.6.: Block diagram of the EDU module with its main components. The black arrows mark data communication links.

A Raspberry Pi Compute Module 3 will be used as processing unit in the development phase due to its excellent compromise between performance, power consumption and heat generation. Moreover, the Raspberry Pi Compute Module 3 allows the connection of two Raspberry Pi camera modules V2 [21], which was already successfully used to capture images of Earth from space on-board SSTL's DoT-1 satellite [22], via two distinct Camera Serial Interfaces (CSI). The Raspberry Pi will be equipped with

an eMMC Flash memory where the operating system and the file system for storage of EDU programs and EDU data will be located. The EPS will provide power for the EDU module. The necessary voltage levels will be generated on the EDU module. The connection to various sensors and the voltage generation will be enabled by a custom-made breakout board which will host the Raspberry Pi Compute Module in the CubeSat PCB stack.

We are confident that the chosen Raspberry Pi platform will be able to successfully operate in LEO for two reasons. First, different Raspberry Pi models were already successfully operated in space [23, 24]. Second, several radiation tests [25, 26, 27] suggest that different Raspberry Pi models can operate in radiation environments similar to LEO. If problems regarding the heat generation of the Raspberry Pi Compute Module 3 occur in the development phase, a change to use a Compute Module 1 will be considered. This change should be easily implementable as the Compute Module 1 and 3 have the same form factor and connections (same pin-out).

The COBC is able to turn on/off the entire EDU module and schedule the execution of EDU programs. The uploaded EDU programs will be transferred to the file system of the Raspberry Pi where they await the execution command from the COBC. If the COBC starts the execution of an EDU program, the Raspberry Pi can gather data from the connected sensors and cameras. This data can be processed on the Raspberry Pi's CPU and GPU. The (processed) data is temporarily stored on the Raspberry Pi's file system. For a permanent storage, the EDU program code has to include certain save commands which will save the (processed) data on the COBC's external memory[4]. With this method, the EDU data downlink is independent of the operation status of the EDU module, i.e. the data can also be downlinked when the EDU module is completely turned off. For a more detailed discussion of the EDU data flow, see Section 3.1.5.

3.4.3. Supporting Systems and Redundancy

In the previous CubeSat project of the TU Wien Space Team, DISCO One [28], the use of an Attitude Determination and Control System (ADCS) was planned to ensure a proper orientation of the CubeSat for a stable optical communication link. The ADCS of DISCO One consisted of several magnetometers and gyroscopes to sense the orientation of the satellite and a magnetic torquer board to be able to slow down the spinning of the spacecraft after launch.

STS1 will not use an ADCS. This decision leads to less power consumption, complexity, and a shortened development and testing phase. The only component on board STS1 which would greatly benefit from an ADCS is the camera module. However, the operational CubeSat DAVE [29] was able to capture an image of Earth with a cell phone camera without the use of an ADCS. Thus, we strive to replicate this success with rigorous testing of STS1's camera performance.

The use of two cameras, mounted on opposing sides of the CubeSat, with large field of views will give us the opportunity to image Earth in all possible orientations of the CubeSat. If STS1 is spinning after deployment, we will adjust the settings (brightness, exposure time,...) of the camera to be able to acquire a sharp image. This procedure will be extensively tested in the upcoming development phase. If the initial spinning of the CubeSat exceeds the capabilities of our camera module to acquire a sharp image, we will focus on the operation of the other systems and sensors until the satellite has slowed down its rotation by atmospheric drag. Even if the rotation speed of STS1 is too fast for a sharp image acquisition over the entire lifetime, no primary mission objective is compromised by the decision of not using an ADCS. Besides the camera, all other systems of STS1 (including the sensors of the EDU module) will be able to operate nominally without an ADCS.

[4]The COBC is always the master system regarding COBC and EDU communication. This includes the above-mentioned saving operation.

Traditional space missions often use redundant components to lower the risk of failure of certain components or whole (sub)systems. However, the incorporation of redundancy always comes with an increased complexity of the entire system and thus needs significant development and testing time in order to be able to benefit from the incorporated redundancy. Thus, we decided to not integrate the EDU module in a redundant manner for two main reasons. First, STS1 is the first self-developed CubeSat mission of the TU Wien Space Team, which means that already a significant amount of development time is needed to establish a reliably working CubeSat platform – even without incorporating redundant systems. Second, in the former preliminary concept of STS1, the redundant systems were heavily criticized by external reviewers. Therefore, our goal is to develop a working CubeSat hardware early on and start with systematic (automatic) testing of the individual CubeSat subsystems and the whole CubeSat stack as early as possible.

Even though major systems will not be implemented in a redundant manner, we have concerns that the eMMC memory of the Raspberry Pi platform is vulnerable to the radiation environment in LEO. Thus, we are investigating the usage of a system that would allow the Raspberry Pi to boot from multiple eMMCs. The firmware that runs on the Raspberry Pi will not be able to receive a firmware update as the needed functionality is limited. Additionally, extensive testing of the used EDU platform will be performed on ground before launch. Therefore, the only critical error handling scheme of the EDU module is the restart via a command of the COBC and potentially a switchover to a redundant eMMC memory.

Bibliography

[1] *ESERO Austria – European Space Education Resource Office*. https://ars.electronica.art/esero/de/. (accessed Jan 16, 2021).

[2] *Österreichische Forschungsförderungsgesellschaft FFG (Austrian Research Promotion Agency)*. https://www.ffg.at/. (accessed Oct 10, 2020).

[3] *ESA: Fly Your Satellite! programme*. https://www.esa.int/Education/CubeSats_-_Fly_Your_Satellite/Fly_Your_Satellite!_programme. (accessed Sept 24, 2020).

[4] *TU Wien Space Team*. https://spaceteam.at/. (accessed Feb 07, 2021).

[5] *Apache CouchDB – A NoSQL database*. https://couchdb.apache.org/. (accessed Jan 18, 2021).

[6] *Gpredict – A real-time satellite tracking and orbit prediction application*. http://gpredict.oz9aec.net/. (accessed Jan 15, 2021).

[7] *SatNOGS – Open Source global network of satellite ground-stations*. https://satnogs.org/. (accessed Jan 18, 2021).

[8] *NanoAvionivcs CubeSat Electrical Power System EPS*. https://nanoavionics.com/cubesat-components/cubesat-electrical-power-system-eps/. (accessed Jan 20, 2021).

[9] *Endurosat EPS I PLUS*. https://www.endurosat.com/cubesat-store/cubesat-power-modules/eps-power-module-i-plus/. (accessed Jan 20, 2021).

[10] *ISISPACE iEPS Electrical Power System*. https://www.isispace.nl/product/ieps-electrical-power-system/. (accessed Jan 20, 2021).

[11] *RFM23B RF module from HopeRF*. http://www.hoperf.com/modules/rf_transceiver/RFM23BW.html. (accessed Jan 10, 2021).

[12] S. Hansen et al. "Airglow-CubeSat with Orientation Control by Aerospike Puff-jets". In: http://www.utahspacegrant.com/wordpress/wp-content/uploads/2013/01/6.-Stewart-Hansen.pdf. 2012.

[13] *The Mars Rover On-board Computer*. https://media.ccc.de/v/35c3-9783-the_mars_rover_on-board_computer. (accessed Feb 05, 2021).

[14] *Pegasus – A CubeSat project by the TU Wien Space Team, FH Wiener Neustadt and the Space Tech Group*. https://spaceteam.at/cubesats/cubesat/?lang=en. (accessed Sept 06, 2020).

[15] *Website of ESA's Astro Pi Challenge*. https://astro-pi.org/. (accessed Oct 10, 2020).

[16] *The SelfieSat – Selfies From Space. A CubeSat project by students of the Norwegian University of Science and Technology (NTNU)*. https://www.orbitntnu.com/current_mission/. (accessed Sept 06, 2020).

[17] *ArduSat – Your Arduino Experiment in Space*. https://www.kickstarter.com/projects/575960623/ardusat-your-arduino-experiment-in-space. (accessed Jan 16, 2021).

[18] *Website of Raspberry Pi Computer*. https://www.raspberrypi.org/. (accessed Oct 10, 2020).

[19] *TIOBE Programming Community Index*. https://www.tiobe.com/tiobe-index//. (accessed Jan 16, 2021).

[20] *Website of ESERO's CanSat Challenge*. https://ars.electronica.art/esero/de/projects/cansat/. (accessed Oct 10, 2020).

[21] *Raspberry Pi Camera Module V2*. https://www.raspberrypi.org/products/camera-module-v2/. (accessed Sept 06, 2020).

[22] *SSTL Releases Raspberry Pi Camera Image and Video*. https://www.satellitetoday.com/imagery-and-sensing/2019/09/05/sstl-releases-raspberry-pi-camera-image-and-video/. (accessed Sept 06, 2020).

[23] *Raspberry Pi in space! – Raspberry Pi Blog entry*. https://www.raspberrypi.org/blog/raspberry-pi-in-space/. (accessed Jan 16, 2021).

[24] *Compute Module CubeSats – Raspberry Pi Blog entry*. https://www.raspberrypi.org/blog/compute-module-cubesats/. (accessed Jan 16, 2021).

[25] Decena et al. *Radiation Damage Threshold of Satellite COTS Components: Raspberry Pi Zero for Opal CubeSat – Utah State University*. https://digitalcommons.usu.edu/cgi/viewcontent.cgi?article=1851&context=researchweek. (accessed Jan 16, 2021).

[26] Daniel P. Violette. *Arduino/Raspberry Pi: Hobbyist Hardware and Radiation Total Dose Degradation – NASA Goddard Space Flight Center (GSFC) and University of Connecticut (UCONN)*. https://nepp.nasa.gov/workshops/eeesmallmissions/talks/11%20-%20THUR/1430%20-%202014-561-%20Violette-Final-Pres-EEE-TN17486%20v2.pdf. (accessed Jan 16, 2021).

[27] Oscar Gutiérrez et al. "TID characterization of COTS parts using radiotherapy linear accelerators". In: *IEICE Electronics Express* 16.7 (2019), pp. 20190077–20190077. DOI: 10.1587/elex.16.20190077.

[28] *DISCO One – A CubeSat concept by the TU Wien Space Team*. https://spaceteam.at/cubesats/disco-one-concept/?lang=en. (accessed Sept 06, 2020).

[29] *DAVE (Damping and Vibrations Experiment) by PolySat from California Polytechnic State University*. https://calpolynews.calpoly.edu/news_releases/2018/November/CubeSat_Earth_Image and https://mustangnews.net/cubesat-dave-sends-back-first-high-res-photo-of-earth/. (accessed Sept 02, 2020).

A. Appendix

A.1. Power Budget

STS1 Power Budget

Solar Array Sizing

Key:
- User Input
- Calculation result
- Constant

Silicon Solar Array		
Mission Life Time	0,5	years
Power Input	1358	W/m²
Efficiency	25 %	
Power Out (Po)	339,50	W/m²
Inheret Degradation (Id)	0,77	
Array Degradation (3.75%/yr)	0,98	

Power Generated / panel					
Incidence Angle	0	30	45	60	
Power Generated (BOL)	0,94	0,82	0,67	0,47	W
Power Generated (EOL)	0,92	0,80	0,65	0,46	W

Panel Dimensions		
Width	0,06	m
Length	0,06	m
Area	0,0036	m²

Power Generated		
	Power	
Corner points to sun	1,38	W
Face points to sun	0,92	W
Edge points to sun	1,31	W
Mean	**1,20**	**W**

STS1 Power Budget
Power Budget

Key:
User Input
Calculation result
Constant

Percent of time in Sun	60.52 %	percent
Initial Orbit Period	5507.4	sec
Initial Orbit Period	1.53	hours

Quick Summary - Power Usage		
Day Power	1.1	W
Night Power	1.0	W
Orbit Average	0.82	W

	Voltage	Component Power (W)	Day/Night/ Both	Day Power (W)	Day Duty Cycle	Day Avg Power (W)	Night Power (W)	Night Duty Cycle	Night Avg Power (W)	Orbit Avg Power(W)	Peak Power(W)
EDU		**3.42**		**3.42**		**0.70**	**2.62**		**0.69**	**0.70**	3.42
Camera	3.3	0.80	Day	0.80	0.01	0.01	0.00	0.00	0.00	0.00	
Raspberry Pi active	3.3	2.40	Both	2.40	0.20	0.48	2.40	0.20	0.48	0.48	
Sensors (Temp, gyrp, accel,...)	3.3	0.01	Both	0.01	0.20	0.00	0.01	0.20	0.00	0.00	
GPS	3.3	0.18	Both	0.18	1.00	0.18	0.18	1.00	0.18	0.18	
Dosimeter	3.3	0.03	Both	0.03	1.00	0.03	0.03	1.00	0.03	0.03	
OBC		**0.12**		**0.12**		**0.06**	**0.12**		**0.06**	**0.06**	0.12
Processor	5	0.05	Both	0.05	1.00	0.05	0.05	1.00	0.05	0.05	
Sensors	3.3	0.01	Both	0.01	0.20	0.00	0.01	0.20	0.00	0.00	
Memory (2x FRAM, 1x Flash)	3.3	0.06	Both	0.06	0.20	0.01	0.06	0.20	0.01	0.01	
COM		**2.88**		**2.88**		**0.04**	**2.88**		**0.07**	**0.05**	2.88
S/C Receiver (Active)	2.5	0.13	Both	0.13	0.02	0.00	0.13	0.02	0.00	0.00	
S/C Transmitter (Active)	2.5	2.75	Both	2.75	0.02	0.04	2.75	0.02	0.07	0.05	
EPS		**0.00**		**0.00**		**0.00**	**0.00**		**0.00**	**0.00**	0.00
	0	0.00	Day	0.00	0.00	0.00	0.00	0.00	0.00	0.00	
Total Avg Power						**0.81**			**0.83**	**0.82**	**6.42**
Power Design Margin		5 %				0.04			0.04	0.04	0.32
Subtotal Power						**0.85**			**0.87**	**0.86**	**6.74**
Charging and Conversion Losses						**0.25**			**0.17**	**0.22**	**2.48**
Battery Charging (Efficiency 90%)		0.75				0.08			0.00	0.05	0.94
Power Conversion (80%)						0.17			0.17	0.17	1.54
						DAY			NIGHT	Avg Power	Peak Power
Power Required (W)						**1.09**			**1.04**	**0.82**	**8.89**
Power Available (W)						**1.20**			**0.00**	**0.73**	
Power Balance (W)						**0.11**			**-1.04**	**-0.09**	

STS1 Power Budget

Battery Sizing

Battery parameters		
Bus Voltage	3,2	V
Orbit Period	1,53	h
Mission Life	4383	h
Charge/Discharge Cycles	**2865**	cycles
Eclipse Time	0,60	h
Power Consumption at night	1,041	W
Depth of Discharge Max	30 %	percent
Discharge Efficiency	90 %	percent
Consumend energy per eclipse	0,70	Wh
Total Capacity	**2,33**	Wh

Battery Charging		
Charge Energy	0,70	Wh
Orbit Day Length	0,93	h
Charge Power	**0,75**	W

Cell Comparison	SAFT VHAA 1200	SAFT SL18650	
Cell Voltage	1,2	3,6	V
Cell Capacity	1,3	1,66	Ah
Cell Energy	1,56	5,976	Wh
Min # Cells	2	1	cells
Battery Pack Properties (Required)			
# Battery Packs Required	1	1	Battery Packs
# Cells Per Battery Pack	3	1	cells
Nominal Voltage/Pack	3,6	3,6	V
Total Battery Pack Capacity	4,68	6	Wh
DOD Typical	15 %	12 %	percent

A.2. Project plan

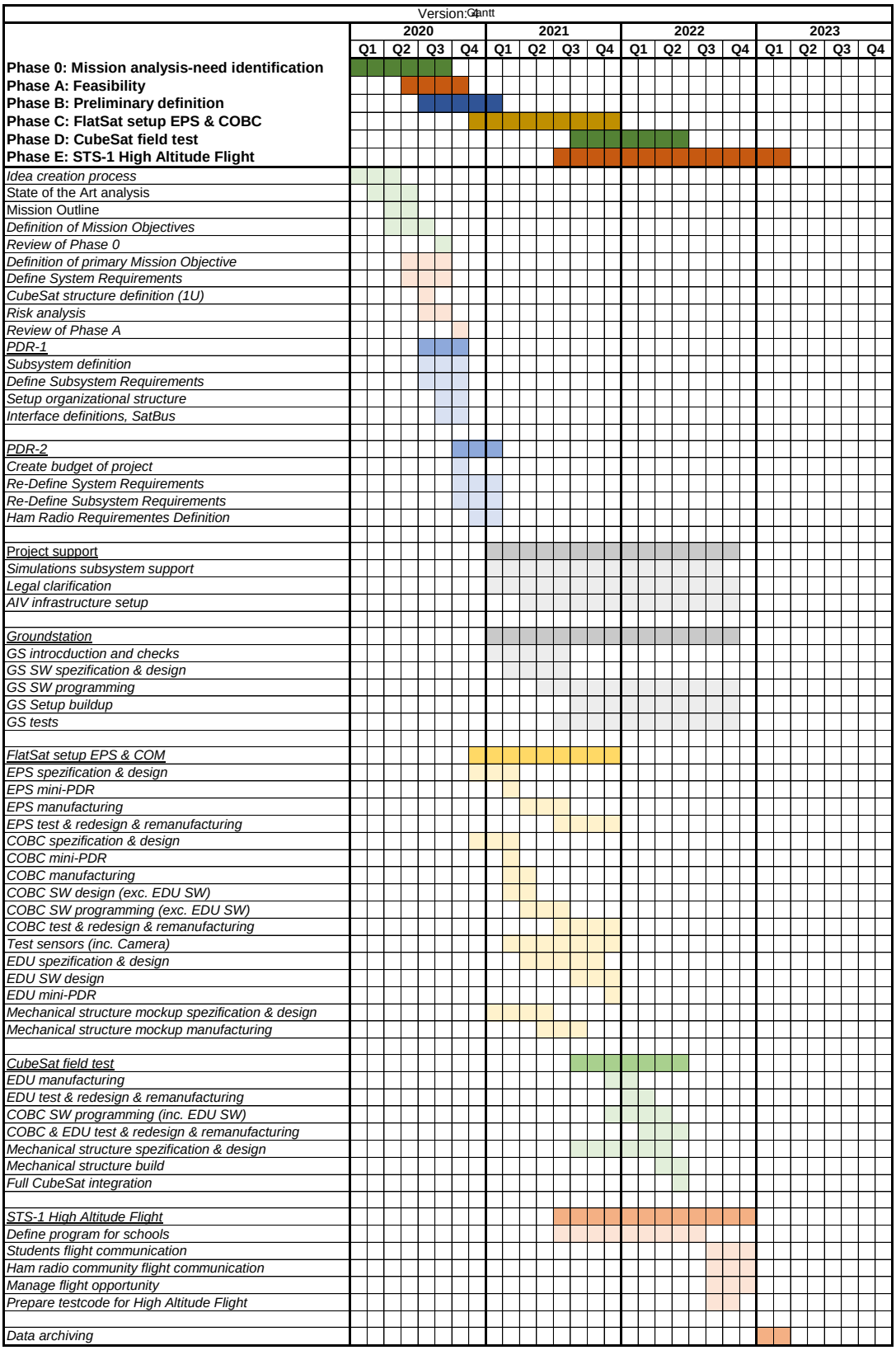

Figure A.1.: Gantt chart for the high-altitude flight (HAF) of STS1.

A.3. Financial budget

CubeSat STS1 – Budget 2021				
Name	**Description**	**Price**	**Pieces**	**Costs**
Engineering hardware	Electronic support equipment (Programmer, Debugging stuff, Analyzer)	500.00 €	1	500.00 €
	Spare/replacement components (Cables, Antennas, Connectors)	650.00 €	1	650.00 €
				1 150.00 €
EPS	Fully equipped PCBs	500.00 €	1	500.00 €
	Batteries	250.00 €	1	250.00 €
	Solar cells	250.00 €	8	2 000.00 €
				2 750.00 €
COBC	Fully equipped PCBs	500.00 €	1	500.00 €
	RF modules	250.00 €	1	250.00 €
				750.00 €
AIV	Fully equipped testing PCBs (Break-out boards for our CubeSat stack)	500.00 €	1	500.00 €
				500.00 €
			Sum	**5 150.00 €**

Figure A.2.: Financial budget for STS1 for 2021: HW development of the EPS, COBC and the FlatSat.