POLITECNICO DI TORINO

Master's Degree in Computer Engineering - Embedded Systems

Master's Degree Thesis

Design, development and verification of the Electric Propulsion Interface System for the CubeSat Test Platform

Supervisors

Prof. Sabrina CORPINO

Prof. Fabrizio STESINA

Candidate

Nicolò BIANCO

Academic Year 2021/2022
April 2022

Summary

The present thesis was born as part of a joint project between the CubeSat PoliTo Team of Politecnico di Torino and the European Space Agency (ESA). The goal of this collaboration is to continue the development of an All Electric Cubesat Test Platform able to perform several types of tests on miniaturized electric propulsion systems. The desired outcome is to assert the mutual interaction between the platform itself and the electric propulsion systems in order to characterize their static and dynamic behavior in different conditions. The collaboration has started with the detailed study of the state-of-the-art of the project, which already was at its second iteration. Once identified the main weaknesses and improvement opportunities the third iteration has began following a modified V-model where new functional and non-functional requirements have been listed, and a new design for the platform has born. The main addition from the previous iterations is a sub-system completely dedicated to interface the EP System, and to characterize its thermal, electrical and electromagnetic behavior inside a CubeSat, which takes the name of Electric Propulsion Interface System (EPIS). The driving requirement during the design has been the flexibility to host different EP Systems in terms of data communication, electrical supply and mechanical integration. While the final design was still in definition, the software development started and preliminary implementation tests has been performed using development boards and communication mock-ups. Each unit has been tested alone and integrated following a bottom-up AIV plan. The complete CTP has been validated through a rigorous Full Functional Test campaign, at first in laboratory conditions, and finally fully integrated with the T4I's REGULUS Propulsion System. A first test campaign was performed in T4I's facilities at Padua to verify the correct behavior of the two systems integrated, while a final commissioning test campaign was performed at the Electric Propulsion Laboratory (EPL) at ESA-ESTEC in Noordwijk, Holland.

Table of Contents

List of Tables VII

List of Figures VIII

Abbreviations XI

1 Introduction, Motivations and Goals 2
- 1.1 Introduction to Cubesats . . . 2
- 1.2 Motivations . . . 2
- 1.3 Goals . . . 4

2 Project review, Methodology and Design 6
- 2.1 The ESA-µProp project . . . 6
  - 2.1.1 Overview . . . 6
  - 2.1.2 Improvement opportunities . . . 9
  - 2.1.3 Objectives and Design principles . . . 9
- 2.2 ESA-µProp 3 . . . 12
  - 2.2.1 Project Management . . . 12
  - 2.2.2 High level requirements . . . 12
  - 2.2.3 Measurements of interest . . . 13
  - 2.2.4 Functional architecture and product tree . . . 17
  - 2.2.5 Basic Avionics . . . 20
  - 2.2.6 Electric Propulsion Interface System . . . 29
  - 2.2.7 Physical Layout . . . 35
- 2.3 Software architectural design . . . 36
  - 2.3.1 Operative Modes . . . 37
  - 2.3.2 Commands' definition . . . 39
  - 2.3.3 Data and Command budget . . . 41
  - 2.3.4 Hardware Architectural definition . . . 44
  - 2.3.5 Software Architectural definition . . . 51

3 Software Implementation and Testing 56
3.1 Development and Unit Testing . . . 56
3.1.1 Set-up of working environment . . . 56
3.1.2 Set-up interfaces . . . 58
3.1.3 Data storage software units . . . 69
3.1.4 Communication software units . . . 71
3.1.5 Data acquisition software unit . . . 74
3.1.6 Power management software unit . . . 74
3.1.7 EP System software unit . . . 77
3.1.8 Time Scheduling Analysis . . . 78
3.2 CTP Integration and Testing . . . 80
3.2.1 EPIS Software Integration . . . 80
3.2.2 EPIS Electrical Integration and Test . . . 83
3.2.3 Avionics Electrical Integration and Test . . . 85
3.2.4 CTP Integration and Full Functional Test . . . 88
3.3 EP System Integration . . . 94

4 Environmental Test Campaign and Results 96
4.1 Test campaign at CISAS-UniPD . . . 97
4.1.1 Test Success Criteria . . . 97
4.1.2 List of Items . . . 98
4.1.3 Test Procedure . . . 98
4.1.4 Results . . . 99
4.1.5 Anomalies . . . 103
4.1.6 Conclusion . . . 104
4.2 Test campaign at EPL-ESTEC . . . 105
4.2.1 Test Success Criteria . . . 107
4.2.2 List of Items . . . 107
4.2.3 Test Procedure . . . 107
4.2.4 Thermal Short Results . . . 109
4.2.5 Long Endurance @ 20W Results . . . 113
4.2.6 Long Endurance @ 30W Results . . . 117
4.2.7 Anomalies and discussions . . . 121

5 Conclusions 122
5.1 Interfaces . . . 122
5.2 Electrical power consumption . . . 123
5.3 EMC/EMI . . . 123
5.4 Thermal environment . . . 124
5.5 Conclusions . . . 124

Bibliography 126

List of Tables

2.1 High level requirements (1) . . . 14
2.2 High level requirements (2) . . . 15
2.3 High level requirements (3) . . . 16
2.4 List of measurable parameters . . . 17
2.5 F/E matrix - subsystem level . . . 18
2.6 List of main components of EPS . . . 24
2.7 List of main components of CDH . . . 27
2.8 List of main components of COMSYS . . . 29
2.9 F/E matrix of DL - component level . . . 32
2.10 CTP commands definition . . . 40
2.11 CM4 GPIO function assignment . . . 45
2.12 List of main components of EPIS-DL . . . 48
2.13 List of main components of EPIS-Power . . . 50

3.1 CM4 UARTs assignment . . . 62
3.2 Set-up time execution time . . . 70
3.3 File dimensions during time in bytes . . . 71
3.4 Save data in memory execution time . . . 71
3.5 CRC32 execution time . . . 72
3.6 Sending packets to CDH execution time . . . 73
3.7 Receiving packets to CDH execution time . . . 74
3.8 Data Acquisition, Validation and Processing execution time . . . 74
3.9 Digital Potentiometer communication execution time . . . 77
3.10 prop_config.txt file format . . . 77
3.11 Configuration file loading execution time . . . 78
3.12 Communication with EP System execution time . . . 78
3.13 Loop execution time . . . 79
3.14 EPIS-DL firmware flags . . . 82
3.15 REGULUS EP System Performance . . . 95

List of Figures

1.1 Total nanosatellites and CubeSats launched . . . 3

2.1 ESA-µProp2 CTP Block Diagram . . . 7
2.2 V-Model Process . . . 13
2.3 CTP functions tree . . . 18
2.4 On-board interfaces . . . 19
2.5 CTP product tree . . . 19
2.6 Test Architecture . . . 20
2.7 ESA-µProp2 Basic Avionics Thermal Desktop simulation of steady state. HPMS is located at the bottom of the stack. . . . 21
2.8 EPS Block Scheme . . . 23
2.9 EPS board . . . 23
2.10 CDH Block Scheme . . . 26
2.11 CDH board . . . 26
2.12 COMSYS board . . . 28
2.13 Basic Avionics and Battery Packs . . . 29
2.14 ESA-µProp3 Functional Tree . . . 30
2.15 EPIS-Power Functional Tree . . . 34
2.16 F/E matrix of EPIS-Power - component level . . . 34
2.17 ESA-µProp3 CTP Physical Layout . . . 36
2.18 Testing phases . . . 38
2.19 Operative Modes . . . 39
2.20 EPIS-DL Rx packet structure . . . 41
2.21 EPIS-DL Tx packet structure . . . 41
2.22 Configuration parameters . . . 42
2.23 List of EPIS-DL telemetry data . . . 43
2.24 EPIS-DL Block Scheme . . . 46
2.25 EPIS-DL Board Layout . . . 47
2.26 EPIS-Power Block Scheme . . . 49
2.27 EPIS-Power Board Layout . . . 49
2.28 EPIS-DL Software static model . . . 51

2.29 EPIS-DL Software dynamic model 52

3.1 Connection between CM4IO and Host Computer 59
3.2 UART interfaces testing 62
3.3 I2C interfaces testing 64
3.4 SPI interfaces testing 66
3.5 GPIOs testing 68
3.6 Set-up time test procedure 70
3.7 CRC32 test procedure 73
3.8 Data Acquisition, Validation and Processing test procedure 75
3.9 Digital Potentiometer communication testing 76
3.10 Worst case scenario execution time 79
3.11 AIV plan 81
3.12 EPIS-DL Source code structure 81
3.13 EPIS boards integrated (Thermistors, LISN and RF boards missing) 84
3.14 Avionics Electrical Integration and Test block scheme 87
3.15 CTP ready for FFT 90
3.16 EPIS post-process log analysis (1) 92
3.17 EPIS post-process log analysis (2) 93
3.18 REGULUS integrated inside CTP Propulsion Module 95

4.1 CTP commissioning test plan 96
4.2 CTP interfaces with CISAS's Vacuum Chamber 97
4.3 Full Functional Test flow activity 99
4.4 EP System Supply Bus 100
4.5 EPIS-DL Propulsion Module temperatures 101
4.6 EPIS-DL Avionics Module temperatures 102
4.7 LISN and RF units 103
4.8 REGULUS Command Response 105
4.9 CTP interfaces with EPL-SPF 106
4.10 Thermal test flow activity 107
4.11 EP System Supply Bus - Thermal Short 109
4.12 EPIS-DL Recorded temperatures - Thermal Short (1) 110
4.13 EPIS-DL Recorded temperatures - Thermal Short (2) 111
4.14 LISN and RF units - Short Thermal 112
4.15 EP System Supply Bus - Long Endurance @ 20W 113
4.16 EPIS-DL Recorded temperatures - Long Endurance @ 20W (1) . . 114
4.17 EPIS-DL Recorded temperatures - Long Endurance @ 20W (2) . . 115
4.18 LISN and RF units - Long Endurance @ 20W 116
4.19 EP System Supply Bus - Long Endurance @ 30W 117
4.20 EPIS-DL Recorded temperatures - Long Endurance @ 30W (1) . . 118

4.21 EPIS-DL Recorded temperatures - Long Endurance @ 30W (2) . . 119
4.22 LISN and RF units - Long Endurance @ 30W 120

Abbreviations

ADC Analog to Digital converter

AIV Assembly Integration and Verification

BCR Battery Charge Regulator

CAN Controller Area Network

CDH Command and Data Handling

COMSYS Communication System

COTS Commerical-Off-The-Shelf

CTP Cubesat Test Platform

DAC Digital to Analog Converter

EMC Electro-Magnetic Compatibility

EMI Electro-Magnetic Interference

EPIS Electric Propulsion Interface System

EPL ESA Propulsion Laboratory

EP Electric Propulsion

EPS Electrical Power System

EQM Electrical Qualification Model

FFT Full Functional Test

GPIO General Purpose Input Output

GSE Ground Support Equipment

GSS Ground Support System

GUI Graphic User Interface

HL Hardline

I2C Inter-Integrated Circuit

LISN Line Impedence Stabilization Network

LS Loads Switch

MEMS Micro Electro-Mechanical System

MET Mission Elapsed Time

NTC Negative Temperature Coefficient

PC Personal Computer

POCC Payload Operations Control Centre

PoliTO Politecnico di Torino

RBT Remove Before Test

RF Radio Frequency

RFT Reduced Functional Test

RS Recommended Standard

RTOS Real Time Operating System

RX Reception

SoW Statement of Work

SPF Small Plasma Facility

SPI Serial Peripheral Interface

TNC Terminal Node Control

TX Transmission

UART Universal Asynchronous Receiver-Transmitter

Chapter 1

Introduction, Motivations and Goals

1.1 Introduction to Cubesats

In the last five years the usage of nano satellites for space missions has been subject to significant growth, with an **increase of 876%** if compared with the previous 20 years.

They represent an emerging opportunity to reach a broad set of mission goals, including scientific experiments, technology demonstration, debris removal, communications and even interplanetary exploration thanks to their low development cost and fast delivery. These characteristics derive from the modular technology on which the CubeSats are based on: spacecrafts composed by multiples of standardized dimensions units 10x10x10 cm and maximum 1.1 kg, hence the name "Cube".

The main feature of these standard-sized spacecrafts is the use of highly modular on-board systems composed by Commercial-Off-The-Shelf (COTS) components and equipment. In this framework, miniaturized propulsion systems greatly increase the range of mission concept achievable with multi-unit CubeSats (6U+) in terms of orbit change and raising, station keeping, orbit maintenance against disturbances, formation flying, proximity operations and de-orbit.

1.2 Motivations

However, there is still a gap between the actual CubeSat capabilities and the lack of knowledge about the real interaction among a propulsion system and the CubeSat technology. To support this thesis from year 1998 only 109 nano satellites (6.2%)

that were actually launched were equipped with propulsion modules, and even less uses all electric solutions.

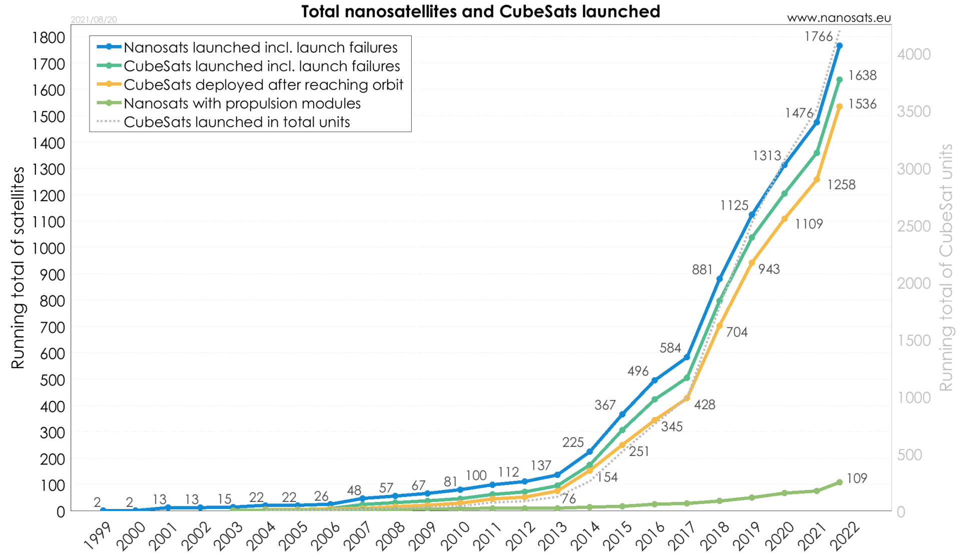


Figure 1.1: Total nanosatellites and CubeSats launched

Electric propulsion systems utilize on-board power to generate thrust. If compared to other propulsion systems (cold gas, chemical) they generate less thrust but higher specific impulse (i.e. how efficiently an engine creates thrust). Three types of EP systems can be found [1]:

- Electrothermal: use electrical power to heat a gas, which is accelerated through a supersonic nozzle, transferring propellant enthalpy into kinetic energy;
- Electromagnetic: use a combination of electric and magnetic fields to accelerate plasma;
- Electrostatic: use magnetic fields to ionize propellant, while thrust is produced through electrostatic acceleration of the plasma.

At the sub-system level, many concepts have been recently developed and relevant test campaigns in representative environment have been conducted, but as mentioned before the greatest challenge in this context is the integrability with actual nano satellites technology and the difficulty to complete an effective test campaign, which generates the gap for the propulsion solutions, slowing down the introduction of this technology in the space domain, missions and market [2].

The same considerations can be applied in the other way: standard CubeSats avionics and mechanical sub-systems are not compliant with most of the EP

systems requirements in terms of power consumption, operating voltage, mechanical interface, electro-magnetic field emissions and thermal fluxes.

Just to make analyze the state of the art of CubeSat, e-st@r-II can be taken as an example. **e-st@r-II** is an 1U CubeSat developed by the CubeSat Team of Politecnico di Torino, launched in April 2016 and currently in Low Earth Orbit, which is equipped with an Electrical Power System capable of delivering as peak power only 2.5W when transmitting, while an Electric Propulsion system during the thrust phase would require at least a power in the range of 30-100W.

1.3 Goals

The present thesis was born as part of a joint project between the CubeSat PoliTo Team and the European Space Agency to design, develop and maintain an All Electric CubeSat Test Platform (CTP) [3]. The ultimate goal of the platform is the assessment of mutual impact between a miniaturized Electric Propulsion system and the CubeSat technology.

To do so, a system dedicated to interface an EP system called **Electric Propulsion Interface System (EPIS)** has been required, which must provide the interfaces of CTP towards the EP system, the instruments and devices to measure the parameters for assessing the mutual interactions between CTP and EP system. This means that a complete engineering process is required to develop the system, composed by:

- Specifications and Requirements analysis;
- Design of the functionalities;
- Design of the electronic boards;
- Development of the software units;
- Verification and Validation of the integrated system.

The main design driver through all the project has been the **flexibility** to host as many EP systems as possible, with the minimum effort for the CTP operator. This would allow simpler test campaigns and faster availability of the results.

Chapter 2

Project review, Methodology and Design

2.1 The ESA-µProp project

2.1.1 Overview

In 2017, Politecnico di Torino has led an experimental research program, sponsored by ESA, that aims to assess the effects on the nano satellites operations and the interactions between miniaturized EP system and a CubeSat platform. The project took the name of ESA-µProp (micro propulsor) and the main output of the first two iterations was a 6U Cubesat Test Platform (CTP) based on CubeSat technology designed to test at system-level a variety of electric propulsion systems. The design was focused on the capability to interface the majority of the European EP systems [2] in terms of power supplying, command and data exchange, mechanical and fluidics connections, and to gather information on the thermal environment, radio-frequency emissions, electromagnetic interference, electrical power consumption, chemical contamination of the platform surfaces and external elements such as solar panels.

The platform features an Al-alloy 6U structure, which contains

- Electric Propulsion System (1-4U)
- On-Board Avionics (1U)
- Battery Box (1U)

The avionics systems are in-house developed electronic boards representative of the CubeSats technology. The avionics box is constituted by the Propulsion Interface System (PIS) that manages the data and power exchanges with the

propulsion system and a basic CubeSat module made of the Command and Data Handling board, the Electrical Power System (PCDU and battery, without solar panels), and the communication module (UHF for housekeeping and experiment data transmission). The boards are connected through a 104-pins bus connector in order to electrical power supplying and data exchange and mounted on stack of four bars that fix the avionics box to the primary structure. The interface with GSS and GSE (battery packs recharging ports, hardline connectors and remove before test switches) are located on the +Y face, while the electrical and data interfaces towards the EP system are located on the other side of the box.

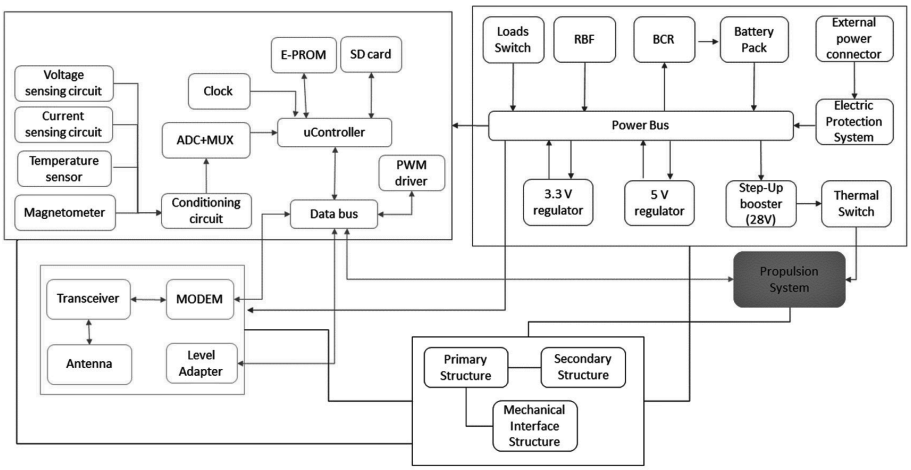


Figure 2.1: ESA-µProp2 CTP Block Diagram

The Propulsion Interface System (PIS) provides the interfaces of CTP towards the propulsion system and the instruments and devices to measure the parameters for assessing the interactions between the CTP and the propulsion system. Two main parts constitute PIS: the Data Logging System (DLS) and the High-Power Management System (HPMS). Data Logging (DLS) acquires many measurements through an electronics board. This board is part of the Avionics stack and mainly hosts the conditioning RC filters and amplifiers and two 16-channel Analog-to-Digital Converter (ADC). The outputs of the ADC are passed to the CDH board via an SPI bus.

Sixteen temperature sensors (twelve NTCs and four PT1000s) provide temperatures of the Propulsion Box in the range [-20; 120] degC and [-40; 350] degC, respectively. The PT1000 sensors are located near to the thruster and on the propulsion system case while the NTC are fixed on the bulkhead , on the faces (+Y, -Y, Z, -Z), four inside Propulsion box close to the propulsion system and four on the external surfaces. The other twelve NTCs are installed in the avionics box

and the battery packs.

All these sensors help to characterize the thermal environment, mainly inside the CTP and in its proximity. Eight micro-RF circuits measure the radiated emissions. These circuits are based on simple antennas, a RF receiver made of a LTC5507 [4] chip, and pass-band filters calibrated in order to allow the frequency 1–10 MHz, 20–50 MHz, 100–200 MHz and 400–500 MHz. All the components are mounted on dedicated PCB sized 20x30 mm. Moreover, the pass band filter can be tuned on other ranges by changing few cheap, passive components, conferring high flexibility to these elements. Two boards for each range of frequencies are available and all the boards are installed on the bulkhead (four on the side towards the Service module and four towards the Propulsion Box).

The conducted emissions are measured using a dedicated circuit called Line Impedance Stability Network (LISN): it enables the evaluation of the radiative environment generated by the propulsion system along the power supplier line when it absorbs current. This circuit is again based on the LTC5507 RF receiver and measures the noise on the absorbed with high accuracy (1 mA) and with a sample rate of 1 Hz. A surface mounted MEMS triaxial magnetometer [5] belongs to the Command and Data Handling board. The output of the sensor is sent on the I2C bus. It allows us to monitor the Magnetic Field variations inside the CTP.

The High-Power Management System (HPMS) is made of a board (on the avionics box) and two PS battery packs that stay in the second unit of the service module. The HPMS board hosts the Step-up circuit: it receives in input voltages in a range [10.5; 12] V from the PS-battery packs and provides in output 12 V regulated (with a minimum accepted level of 11.7 V) with a maximum current of 5A. This circuit is based on a switching DC/DC booster that has an efficiency of conversion higher than 95%. The board has also the battery recharger for PS battery packs: the BCR are the core of this element and can receive up to 32 W (16 V is the expected input voltage and max 2A has charging current) in order to minimize the recharging time (less than 6 hours for a complete recharge). PS battery packs provide up to 5.2 Ah at 12 V and their output is controlled by dedicated 'Remove Before Test' switches.

Protection circuits prevent over current, over voltage, or short circuits on the power bus; refresh fuses are added to each line, especially for the connection towards the propulsion system. Acquisition circuits gather the measurements of the voltage and current and temperature for PS battery packs, step-up circuits, and the consumption on the 3.3 V and 5 V power bus lines. The output of the ADC is connected to the 104-pins bus to provide the sampled values to the SPI bus. These measurements allow for the estimation of the power consumption in any phase of the test, for different modes of operations of the propulsion system.

2.1.2 Improvement opportunities

The first two phases of the project had very different objectives:

- **First phase (12 months: February 2018 - February 2019):** demonstrate the feasability of a test platform to host electric propulsion systems, including the design and manufacturing of a very low cost prototype of a platform (CTP);
- **Second phase (20 months: February 2019 - October 2020):** assess the mutual effects of electric propulsion systems and CTP by integrating the CTP with EP system, performing a test campaign with a selected EP system in vacuum and identify a set of procedures for the Assembly, Integration and Verification (AIV) plan.

In this context the main target stakeholders were only the electric propulsion systems developers.

In order to extend the capabilities of the CTP the design has been revisited not only to host different EP systems, but also to test other CubeSats avionics. In this way the objective is to perform a complete verification campaign of an All Electric CubeSat by having a one-stop testing facility with hardware and software needed to conduct functional, mission and environmental tests up to qualification.

The design process started from the review and expansion of the previous objectives and requirements. Starting from there, a new set of design principles have been found.

2.1.3 Objectives and Design principles

Two objectives of the new project iteration have been identified:

- **OBJ1:** To design and build a prototype CubeSat Test Platform (CTP) based on COTS technology suitable for hosting and handling miniaturised EP systems
- **OBJ2:** To define a procedure for testing the integrated CTP/EP system in a relevant environment (@ESA/ESTEC EPL)

From these a set of design drivers has been stemmed, supporting the design, implementation and V&V processes of the project.

Safety. The CTP shall be compliant with safety requirements. In particular, the following aspects shall be considered, at platform level:

- Leakage and contamination protection: protection of the CTP from chemical residuals and plume released by the thruster
- Over-voltage and over-current protection: protection of the electrical loads from undesired changes of voltage and current
- Thermal protection: isolation of the parts that may provoke changes of the temperatures (over-heating/under-heating) inside the platform
- Electro-magnetic interferences protection: protection of CTP and propulsion system from mutual and external electromagnetic interference

Safety critical functions/items must be identified and managed. Fault avoidance (high-quality parts and appropriate design margins), and fault tolerance (redundancy) approaches shall be implemented at least for safety-critical functions. Notwithstanding the intrinsic safety level of the CTP, the test operators shall be able to take control of the test at any time.

Reliability. The CTP shall guarantee the test execution with a certain level of reliability, i.e. must be able to execute the test despite failures occurring in the system. In principle, at least one system failure should be tolerated (i.e. no single point failure shall exist) for a set of identified functions. Possible solutions are:

- relevant data of the experiment shall be saved and stored in two or more independent items
- the EP system shall be commanded by two independent lines, if possible

An onboard autonomous Failure Detection Isolation and Recovery (FDIR) system would well serve the purpose of enhancing reliability (and safety) of the CTP and can be considered for implementation in the design.

Autonomy. The CTP is intended for use in a test facility with adequate GSE and skilled operators to conduct the test. However, a high-degree of autonomy of the CTP can be foreseen in order to increase the value of the project in terms of fidelity (with respect to a real case – CubeSat in orbit), and to reduce the effort and workload of operators, thus pursuing cost reduction of CubeSat test campaign. Regarding autonomy, two main extreme options exist:

1. the CTP is fully autonomous and has full authority over the test run
2. operators have total control over the test, and the CPT only performs commanded functions and returns data as required

Hybrid solutions between these extreme options can be implemented in order to optimise test execution and to comply with safety requirements. Main constraint (apart from safety issues and decision-making capabilities) for a full autonomous test is related to the need of electrical power supply from external source, as no solar panels nor solar simulator are available for the test. The limitation due to this aspect is strongly linked with the duration of the test (for short duration test sessions, the onboard battery can be enough to run the entire test, while long duration tests will need external intervention). However, the CTP design will give room/consider for future upgrade, e.g. considering the possibility to add solar panels.

Flexibility. The CTP is intended to host a variety of miniaturised propulsion systems and to run a wide range of tests. With this regard, the design of the CTP shall consider the capability of

- handling (wide) ranges of electrical powers and operative voltages
- managing different communication protocols
- managing different data and commands
- providing mechanical interfaces (between the platform and the electric propulsion system) able to adapt to the specific system of interest

A modular architecture can well serve the purpose of flexibility and shall be pursued throughout the design process.

Accessibility. The CTP shall guarantee easy physical access to onboard systems during all test phases (set-up/preparation, run, post-test), and for maintenance activities between subsequent test sessions. From the mechanical point of view, it is required that the structure of the CTP features access ports, and/or it is an "open" structure (truss-like) with easy mounting panels. The CTP shall also be interfaced with GSE through hard-lines for data and commands exchange for all test phases.

Cost. The CTP shall make use of low-cost technology and processes (i.e. COTS components).

2.2 ESA-µProp 3

The ESA-µProp3 version of the CTP includes some key new features from the previous ones which have been designed to be fully compatible with the old design. These features are mainly driven by the Reliability and Flexibility design drivers: the platform shall be able to continue the test execution even if a failure occurs. The Reliability property has been fulfilled using the Duplication With Comparison (DWC) fault detection technique at the component level (i.e. two temperature sensors measuring the same component). Flexibility was already partially satisfied - the new version guarantees the complete compatibility with most of the EP Systems on the European market[2].

A complete overview of the new design is described in the next sections.

2.2.1 Project Management

Project management has been a key asset during the entire duration of the project itself. At first, the typical Waterfall process model has been adopted: this helped to fix some high-level requirements and design drivers that would have been used later in the development.

However, even if this model has a clear discipline and process control, it is often very difficult to apply in its pure form, even more so within an educational project. In the case of this project for example, while the physical electronic boards were not yet available, the hardware and software development and testing already started on specifically-designed development board. This approach has allowed to sensibly reduce the elapsed amount of time between the architectural design and coding phases and the testing phases. This topic will be described in more detail in the next chapter.

In this context the adopted process model is a modified V-Model, where the sub-systems integration testing is iterated a number of times: at first with just development hardware/software, and finally with the physical hardware, as soon as it is available.

It is important to remember that this project has been in collaboration with the European Space Agency, and for this reason at each major phase of the project (i.e. Milestones) a Progress Meeting has been performed where design solutions and possible implementations have been analyzed and negotiated.

2.2.2 High level requirements

Stemming from the objectives and the design drivers the following high level requirements have been derived for the definition of the platform architecture.

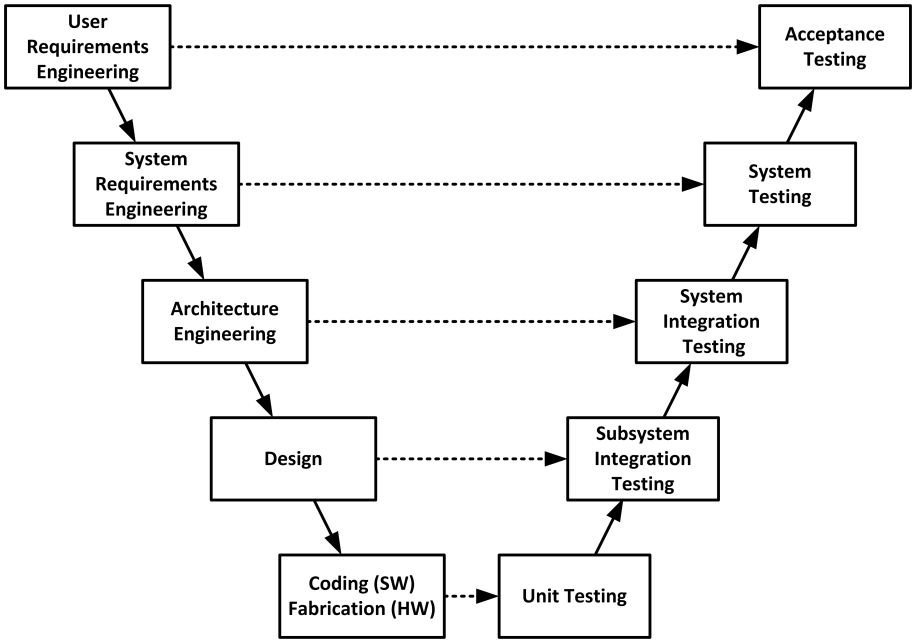


Figure 2.2: V-Model Process

Tables 2.2, 2.3 and **??** contain foreach high level requirement its unique identificator and the backward traceability.

2.2.3 Measurements of interest

Electric propulsion system's impact could be assessed along several metrics, including thrust levels, mass, power consumption, specific impulse, total impulse per unit system wet mass ratio, total impulse per unit system volume ratio, and thrust to power ratio. In general, a test campaign fulfilling the above-mentioned objectives and requirements would include the measurement of several parameters, at subsystem and system level.

Four sets of parameters have been identified to characterising EP System:

- thrust performance
- mass flow and mass variation
- electrical and diagnostics performance
- other parameters (specific to the single type of EP system)

ID	Requirement Text	Traceability
MIS-010	A CubeSat Test Platform (CTP) shall be developed	SoW
MIS-020	CTP shall integrate and test a miniaturized electric propulsion (EP) system	SoW
MIS-030	CTP shall assess the mutual effects of EP system and CTP	OBJ1
MIS-040	CTP shall operate in (thermo-) vacuum environment (i.e. Small Plasma Platform of EPL/ESA-ESTEC, or equivalent)	OBJ2
MIS-050	A set of procedures for the verification of CubeSat with EP System shall be delivered at the end of the project	OBJ3
MIS-060	CTP shall assess the following mutual interaction between CTP and EP system in terms of: • Generated Thermal Environment: effect of EP system operation on temperature distribution all over the spacecraft and at critical equipment (e.g. batteries) • Generated Electro-magnetic: electro-magnetic interference • Power consumption	MIS-40
MIS-070	CTP shall provide the measurement in terms of: • heat flow • electrical and diagnostics performance • other parameters (specific to the single type of EP System)	MIS-40
MIS-080	CTP design and development shall preferably use CubeSat technology and Commercial Off The Shelf components	Driver - Cost

Table 2.1: High level requirements (1)

ID	Requirement Text	Traceability
MIS-090	CTP shall include high-grade items for safety critical technology	Driver - Safety
MIS-100	Only complete EP Systems shall be considered (stand-alone thrusters are not object of the test). Complete EP Systems include (at least): thruster, tank, PPU, PFS	SoW
MIS-110	CTP shall help access and maintenance during of assembly, integration and set-up, and post test activities	Driver - Accessibility
MIS-120	ESA-uProp third program shall cost less than 100k €	SoW
MIS-130	The documentation shall be provided according to the schedule and the milestone specified in SoW	SoW
MIS-140	CTP (with EP system) shall be designed, manufactured, integrated and tested before 28/02/2022	SoW
MIS-150	CTP (without EP system) shall be designed, manufactured, integrated and tested before 31/01/2022	SoW
MIS-160	Test campaign in EPL facility shall be concluded before 28/02/2022	SoW
MIS-170	CTP shall operate without causing catastrophic and critical damages	Driver - Safety
MIS-180	CTP shall operate nominally after one major failure occurrence	Driver - Reliability
MIS-190	CTP shall stop to operate after two major failures occurrence without causing damages	Driver - Reliability
MIS-200	CTP shall test PS with at least 3 independent inhibits to activation	Driver - Safety
MIS-210	CTP shall test EP systems demanding less than 120 Wh	Driver - Safety
MIS-220	CTP shall operate without the intervention of operators	Driver - Autonomy
MIS-230	EP system shall be commanded by the Operators through the ground support equipment and the CTP	Driver - Autonomy

Table 2.2: High level requirements (2)

Other parameters can be identified to monitor the functionalities and performance of the on-board subsystems. Range, accuracy and sampling rate are subject to change upon specific EP system of interest and test requirements. Table 2.4 summarizes the parameters that can be monitored and measured during a test, highlighting:

- the instrument(s) required for the measurement
- the range and the accuracy of the measurement
- the sampling time required for post processing analysis
- the object under test: only the EP system, the CTP, or the integrated CTP+EP system
- the test point, i.e. where the measurement is taken
- the method for evaluation of the parameter: post-processing (PP) and/or real time (RT)

Range, accuracy and sampling rate are subject to change upon specific EP system of interest and test requirements.

ID	**Requirement Text**	**Traceability**
MIS-240	CTP shall exchange information with operators	Driver - Safety
MIS-250	CTP shall manage communication with the EP system identified in TN02 - User Manual	Driver - Flexibility
MIS-260	CTP shall deliver the electrical power for the EP systems identified in TN02 - User Manual	Driver - Flexibility
MIS-270	CTP shall provide a mechanical interface (between the platform and the propulsion system) able to adapt to the specific system of interest	Driver - Flexibility

Table 2.3: High level requirements (3)

Parameter	Instrument	Range	Accuracy	Sampling Rate[Hz]	Test Object	Test Point	Note
Thrust							
Thrust [mN]	Torsional thrust balance	[10 uN - 1mN]	1 uN	1	EP+CTP	GSE	RT + PP
Specific Impulse [s] and effective specific impulse [s]	Indirect from thrust measurement	[0 – 5000] s	1 s	-	EP	GSE	PP
Impulse bit [μN/s]	Indirect from thrust measurement	[0 - 5000] μN/s	1 s	-	EP	GSE	PP
Mass							
Mass flow rate [mg/s]	Mass flow sensors	[1 - 10] mg/s	0,1 mg/s	1	EP	GSE	RT + PP
Propellant mass variation per area [g/cm^2]	Quartz Crystal Microbalance	[0 - 10] g/cm^2	0,1 g/cm^2	1	EP	GSE	PP + RT
Propellant mass consumption [g]	Balance	[0 - 1000] g	10 g	1	EP	CTP+GSE	PP
Electrical and diagnostics							
Current [mA]	Amperometer (GSE), current sensing (CTP)	[0 - 10000] mA	10 mA	1	EP+CTP	CTP	RT + PP
Voltage [V]	Voltmeter (GSE), voltage sensing (CTP)	[0 - 50] V	10 mV	1	EP+CTP	CTP	RT + PP
Temperatures [°C]	Thermocouples (GSE), thermistors (CTP)	[-20 - +100] °C	0.1 °C	1	EP+CTP	GSE + CTP	RT + PP
Magnetic fields [T]	Magnetic Field Mapper (GSE), Magnetometer (CTP)	[-10^-5 - +10^-5] T	10^-5	1	EP	GSE + CTP	RT + PP
Electro-Magnetic Field – Radiated generation [dB/Hz]	RF circuits (CTP)	[-34 - 14] dBm	0.1 dBm	1	EP+CTP	CTP	RT+PP
Electro-Magnetic Field – Conducted generation [dB/Hz]	RF circuits (CTP)	[-34 - 14] dBm	0.1 dBm	1	EP+CTP	CTP	RT+PP

Table 2.4: List of measurable parameters

2.2.4 Functional architecture and product tree

The high-level functions tree for the CubeSat Test Platform is then illustrated in Figure 2.3: this represents the functional architecture of the CTP. The functions have been drawn from analysis of high-level requirements, constraints and considering main design drivers. Up to 3 levels of decomposition for each main function have been analyzed:

Table 2.5 reports the Functions/Equipment matrix which helps to identify and assign each function to a specific on-board subsystem. Electric Propulsion Interface System is the main test object. A Data Logger (EPIS-DL) system gather all the information to assess the mutual impact between EP system and CTP while the Power board (EPIS-PW) electrically supplies the EP System. The Structure hosts all the subsystems including the EP system and protects the CTP from the environment. Electrical power is managed by the Electrical Power System (EPS), on-board operations are managed by the Command and Data Handling System (CDH) and all the communication of information to/from the CTP towards the GSS is guaranteed by a Communication System (COMSYS). The product architecture is obtained through development of functions/equipment (F/E) matrix at several

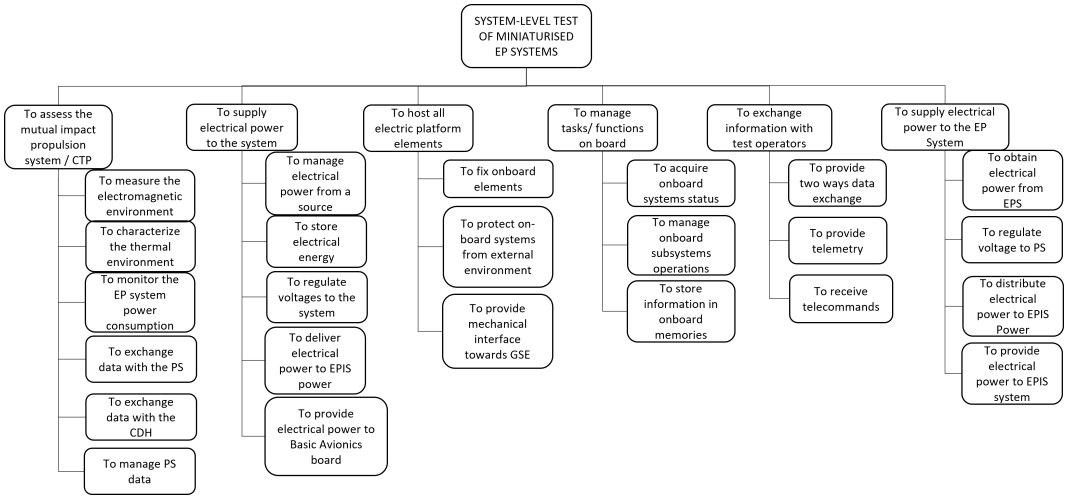


Figure 2.3: CTP functions tree

levels of decomposition. Figure 2.5 shows the product tree for the CTP with 3 levels of decomposition, i.e. subsystem level. A detailed description of each product is proposed in the next section.

	EPIS-DL	**EPIS-PW**	**Structure**	**CDH**	**EPS**	**COMSYS**
To assess the mutual impact EP system / CTP	**X**					
To supply electrical power to propulsion system		**X**				
To host All-Electric platform elements			**X**			
To operate the all-electric platform				**X**		
To supply electrical power					**X**	
To exchange information with the operators						**X**

Table 2.5: F/E matrix - subsystem level

The internal configuration of the CTP is shown in Figure 2.4: on the left the Basic Avionics are stacked together and connected on the same electric bus, which is composed by both power and data lines. On the center the EPIS stack is connected to the Basic Avionics with an unregulated power line and a communication line. Finally the EP System is connected to the EPIS subsystem with a regulated power line and a selectable data line.

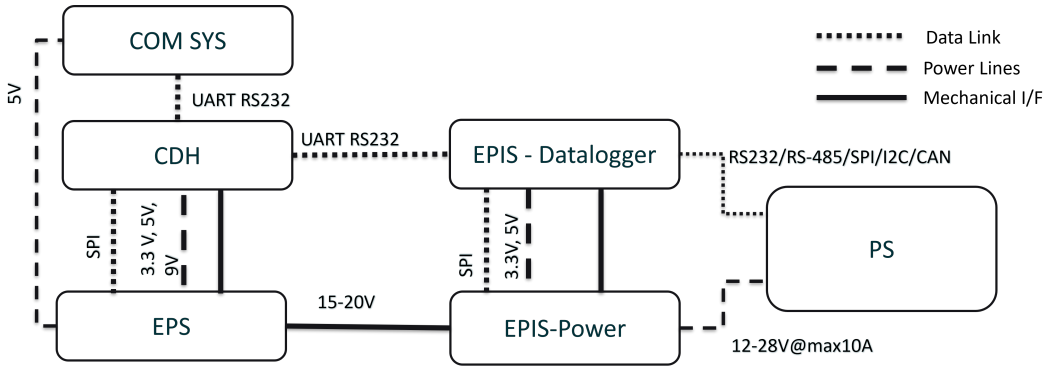


Figure 2.4: On-board interfaces

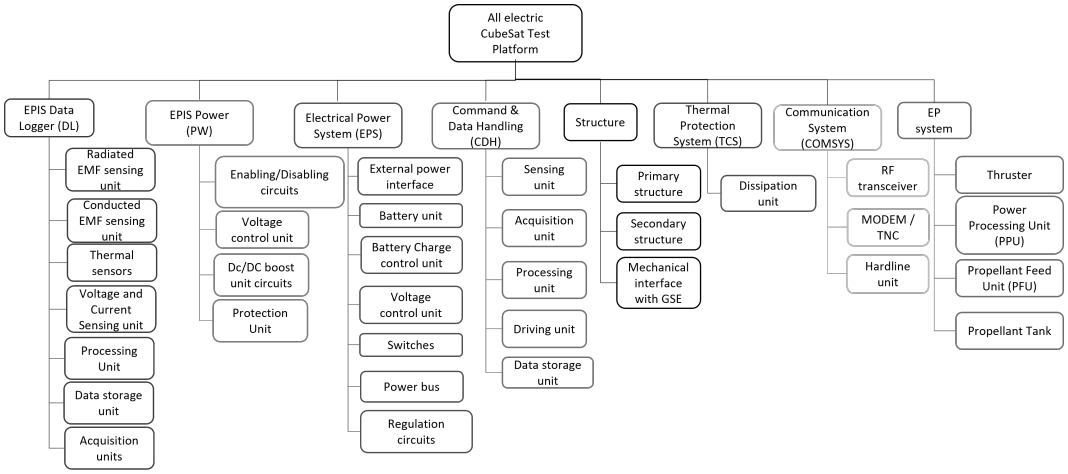
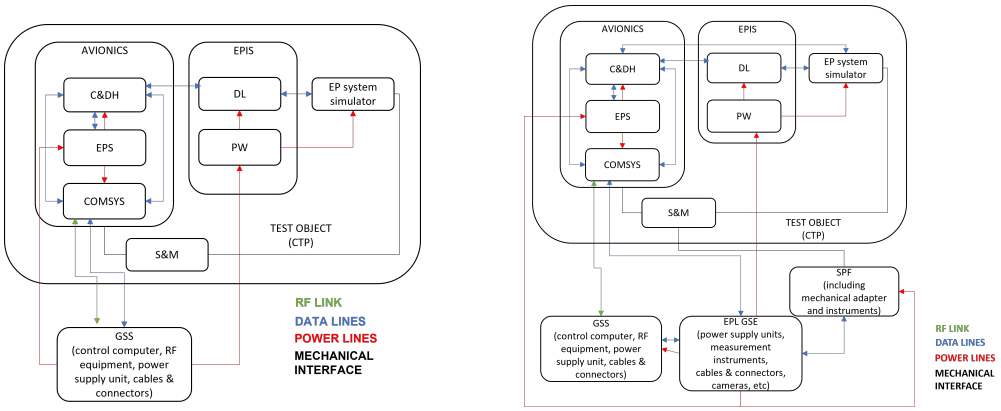


Figure 2.5: CTP product tree

Figure 2.6 depicts the functional block diagrams of the architecture for two possible tests of the CTP. The diagrams illustrate the connections (command and data lines, command and data RF link, power connections, and mechanical interfaces) between single subsystems within the platform and with external equipment. The first case refers to the functional architecture of test to be carried out before integration of the propulsion system in the platform. In this case, the EP system is

simulated through a load representative of the propulsion system of interest and/or with a communication mock-up. This test would be done at PoliTO facilities in ambient conditions, with the main objective of verifying functional and operative requirements and validate the platform design. The second diagram shows the functional architecture of the final test, to be carried out at ESTEC in relevant environment with the test object including the EP system. Should the propulsion system be not available, the same approach used for the test at PoliTO can be pursued, to verify the platform design in a relevant environment.

In the next sections all the sub-systems composing the CTP will be described from a physical point of view, but even if an in-depth functional analysis has been conducted for every sub-system composing the CTP, for the means of this thesis only the Electric Propulsion Interface System will be covered.



(a) @ PoliTo Facilities **(b)** @ ESA-EPL

Figure 2.6: Test Architecture

2.2.5 Basic Avionics

In the previous designs the space reserved for the CTP avionics was 1U, this meant that all the electronic boards had to be stacked one on top of the other. The main problem with this solution was the heat management: the High Power Management System (HPMS) board, responsible of providing the required power to the EP system, would generate such high temperatures almost at the end of the operating temperature range of the board. In this operating conditions the subsystem would undergo a degradation of performance, but most likely, due to high temperatures, it would irreversibly damage the board itself and also affect near boards [6].

This was one of the two reasons that led to separate and re-design the avionics in two different units of 1U each: the Basic Avionics and the Electric Propulsion Interface System. The second reason is to be able to host and test other CubeSats

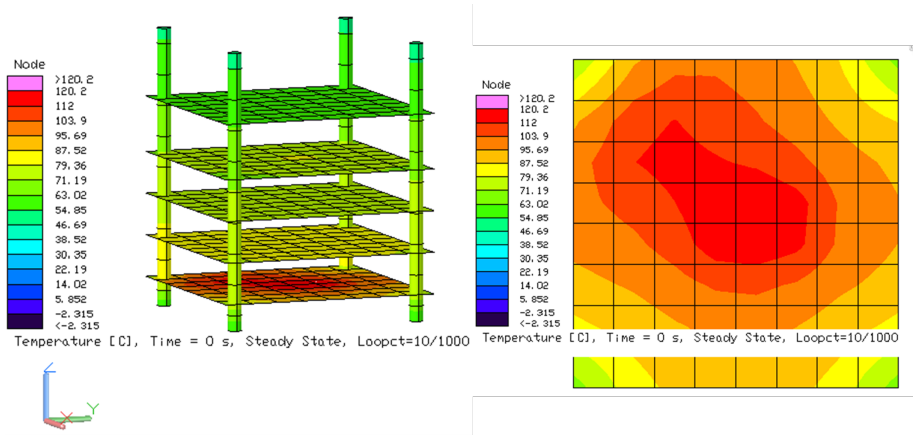


Figure 2.7: ESA-µProp2 Basic Avionics Thermal Desktop simulation of steady state. HPMS is located at the bottom of the stack.

avionics. This would expand the CTP capabilities to perform a complete verification campaign of an all electric CubeSat by having a one-stop testing facility with hardware and software needed to conduct functional, mission and environmental tests up to qualification.

In this new design Basic Avionics is composed by three boards:

- Electric Power System - EPS
- Command and Data Handling - CDH
- Communication System - COMSYS

2.2.5.1 Electric Power System

The Electric Power System (EPS) is the subsystem responsible for the supply of regulated power to the Basic Avionics bus. It is composed by the EPS board and two batteries. The EPS board is part of the Basic Avionics box while the batteries are contained in the Battery box. The EPS board is composed by the following units:

- **Battery Recharge Unit.** BCR circuits are responsible for recharging the two batteries and balancing the recharge currents. They can receive up to 36W (18V@2A) and recharge batteries in less than 6 hours when CTP is in Dormant State.
- **Voltage Regulators.** Power coming from the batteries is regulated by this circuits which can provide 3.3V@200mA, 5V@1A and 9V@1A on three dedicated power buses.

- **Protection Unit.** On-board circuits are protected by different units. A Load Switch (LS) allows the operator to control the activation and deactivation of the CTP, which directly cuts the connection between the battery bus to the regulators. On the same line a fuse is placed to avoid over-current failures.
- **Remove Before Test.** Two Remove Before Test (RBT) switches separate the Battery packs from the recharging circuits. Both RBTs and LS allow the CTP to be switched in the Dormant State.
- **Acquisition Unit.** The acquisition unit is composed by one 24-bit Sigma-Delta ADC and its conditioning circuits. It gathers the measurements of voltage, current and temperature of the different power buses and batteries. Communication with acquisition unit is achieved through a dedicated SPI bus on the 104-pins bus by the CDH micro-controller.
- **Connectors.** EPS board is equipped with two 2x26 pins female connectors (P1-P2) that compose the 104-pins bus. Two 1x2 high power connectors (P3-P4) are used by the batteries, while other two 1x2 high power connectors (P5-P6) are used to recharge from the outside the two batteries. One 1x2 high power connector (P7) is used to supply unregulated power to the EPIS-Power board. One 1x4 male connector (P8) is used to connect two thermistors for the batteries.

The Battery packs are two packs of 8 cylindrical AA-size Li-Ion cells. These packs provide a nominal voltage of 14.4 V@12Ah. The batteries are connected to the EPS board with cables and dedicated temperature sensors are mounted on their surface, in order to keep track of the heat generated by them.

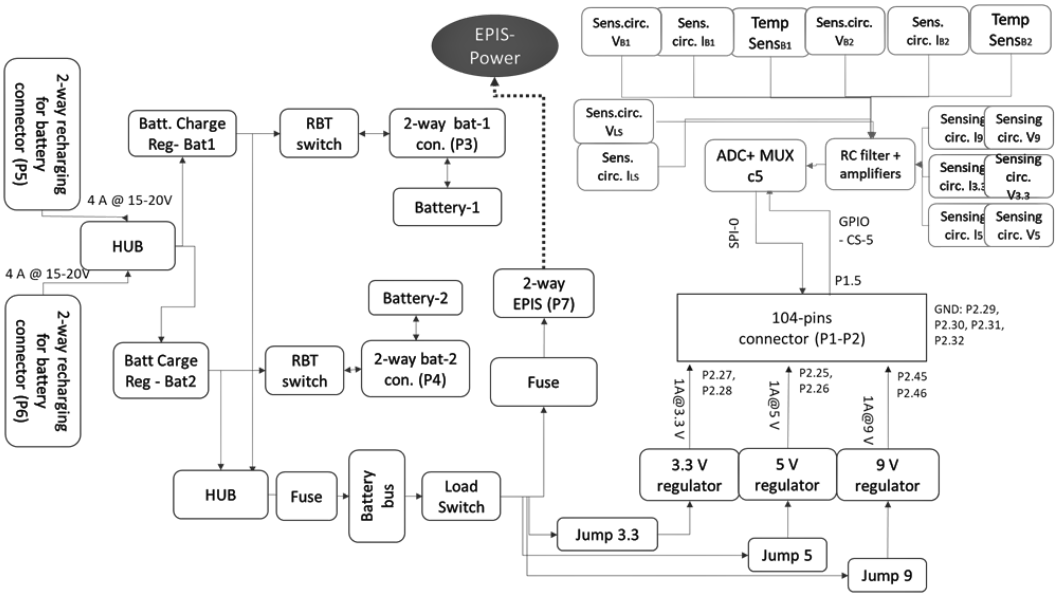
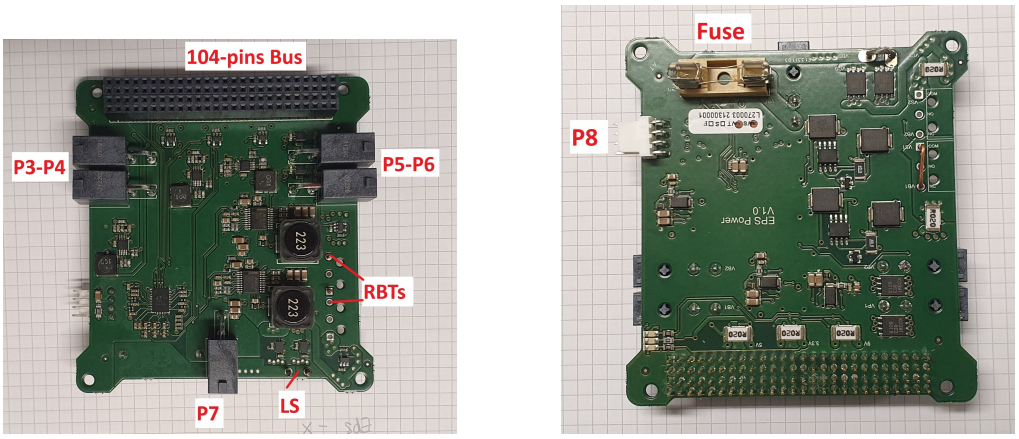


Figure 2.8: EPS Block Scheme



(a) -X face **(b)** +X face

Figure 2.9: EPS board

	Equipment	**Component**	**Name**
Battery packs	Battery pack	Cell/assembled cells	Rechargeable battery, 12000mAh @ 14.4V
EPS - board	Recharging Unit	Battery recharging	BCR MAX745EAP+-ND + resistors, inductance, capacitors
	External source interface	Protection elements	Refreshing Fuses, SMD Bourns, 15mΩ max
	Regulation Unit	Voltage Regulators	Reg Buck 3.3 V, Reg Buck 5V, Reg Buck 9V TPS57160
	Protection system	Fuse	Fuse 15A, Diode LTC4357CDCB
	Power distribution	Switches	2-positions switches
	Other parts	Printed board	96x90 mm - Double face printed board
	Other parts	Other components	Passive electronic components, amplifiers

Table 2.6: List of main components of EPS

2.2.5.2 Command and Data Handling

The Command and Data Handling system (CDH) is the logic core of the Basic Avionics: it controls the CTP-related functions and hosts the On-Board Computer System (OBCS). It is composed by the following units:

- **Micro-controller.** It is the RD129 provided by ELPA s.a.s. - Linux Embedded 2.6.32 version is the Real Time Operating System (RTOS) that guarantees a soft real-time (± 10 micro-seconds), enough to accomplish all the software tasks without losses of synchronization. Firmware is customized on the specific application of ESA-µProp: UART, I2C, SPI, USB drivers are installed, timer and GPIO pin settings are available, and watchdog is enabled. It is mounted on the CDH board with two 60-pins SMD connectors. It holds the control of the data bus through the 104-pins connector. Application software of the CDH system is written in C/C++ language, cross-complied for ARM-9 architecture micro-controller, and the binary file can be loaded through the micro-SD card. It is also equipped with a small 64MB Flash Memory, large enough to store the firmware and application files.
- **Acquisition Unit.** The acquisition unit is composed by one 24-bit Sigma-Delta ADC and its conditioning circuits. It gathers the temperature measurements of the different sub-systems composing the CTP. Communication with acquisition unit is achieved through a dedicated SPI bus, which is shared with EPS Acquisition Unit.
- **Magnetometer.** A Micro Electro-Mechanical Systems (MEMS) magnetometer is used to log the magnetic field around the CTP. It communicates with CDH microcontroller via I2C.
- **SD Card Slot.** SD CARD is the main memory for data storage, and it is inserted in the dedicated slot.
- **Connectors.** CDH board is equipped with two 2x26 pins female connectors (P1-P2) that compose the 104-pins bus, one 2x20 pins male header connector for the external thermoresistor, one 2x10 pins male header previously used for communication with the EP system, a 1x6 pins male header for serial communication with the Ground Support System (two RS232 serial interfaces).

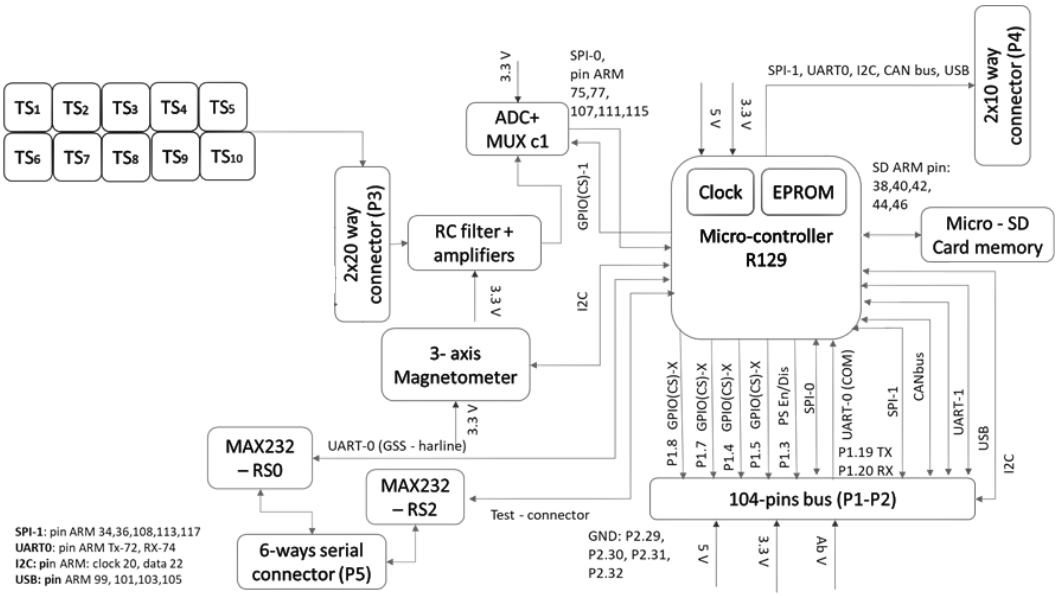
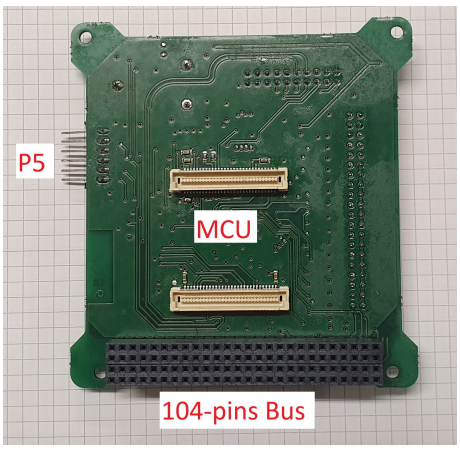
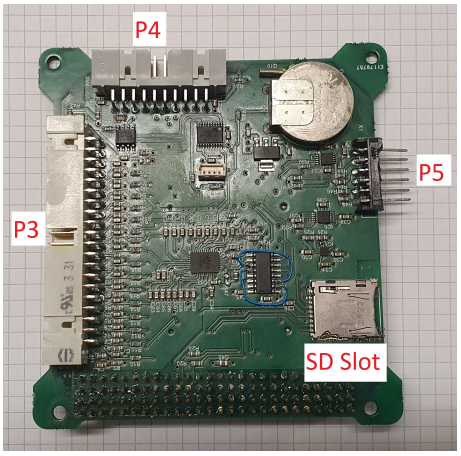


Figure 2.10: CDH Block Scheme



(a) -X face



(b) +X face

Figure 2.11: CDH board

Equipment	**Component**	**Name**
Sensing Unit	Temperature sensor	Vishay NTCLE203E3103FB0, 100mW [-40, +125] °C
	Magnetometer	3-Axis Magnetometer BM1422AGMV
	Other sensing circuits for current and voltage	Amplifiers and passive components
Acquisition Unit	Conditioning circuits	Amplifiers and passive components
	ADC+MUX	24-Bit Sigma-Delta ADC LTC2449IUHF
Processing Unit	Micro-controller + EPROM	RD129
	SD Card	SD Kingston SDHC 8 GB
Communication Unit	Level Adapter	MAX232
Other parts	Printed board	96x90 mm - Double face printed board
	Other components	Cables, connectors, amplifiers and passive components

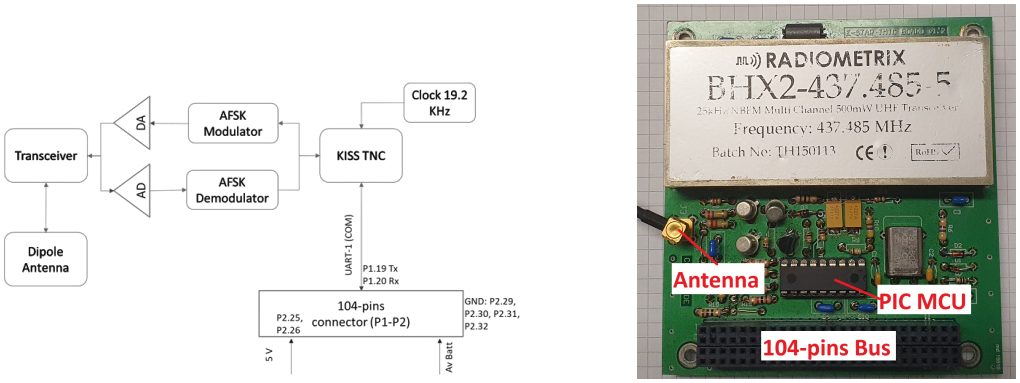
Table 2.7: List of main components of CDH

2.2.5.3 Communication System

The Communication System (COMSYS) is constituted by a PCB that hosts TNC/MODEM, transceiver, signal adapter and the antenna mounted on the structure, outside CTP. All TNC and modem functions are implemented by a Microchip PIC16F88 microcontroller, which interface via RS232 lines to the CDH on one side and via analogue lines to the radio on the other. The RS232 interface is native in the microcontroller; the digital to analog conversion has been performed by a simple resistors array commanded by four digital outputs of the microcontroller. The TNC functions, i.e. the protocol management, has been re-written to allow processing packets in KISS (Keep It Simple Stupid) format to/from CDH and in AX.25 format on the radio channel. A crystal oscillator provides the clock signal for the micro-processor. Data rate of the lines is 1200 bps. A COTS UHF radiomodule provided by Radiometrix Ltd. (model BHX2) [7] is the full-duplex transceiver that offers a 500mW RF power output. A double aluminium shielding protects the transceiver.

Antenna is a dipole antenna with about 2 dB of gain and wide beamwidth. It is easy to be assembled through a SMA connector with the COMSYS board and mechanically connected with the structure. EPS provides power through 3.3 V and 5 V lines via 104-pin bus for all the COMSYS components. Total current consumptions of about 400 mA in Tx and 50 mA in RX are estimated.

This sub-system is the same communication system used on the e-St@r CubeSats developed by the CubeSat Polito Team.

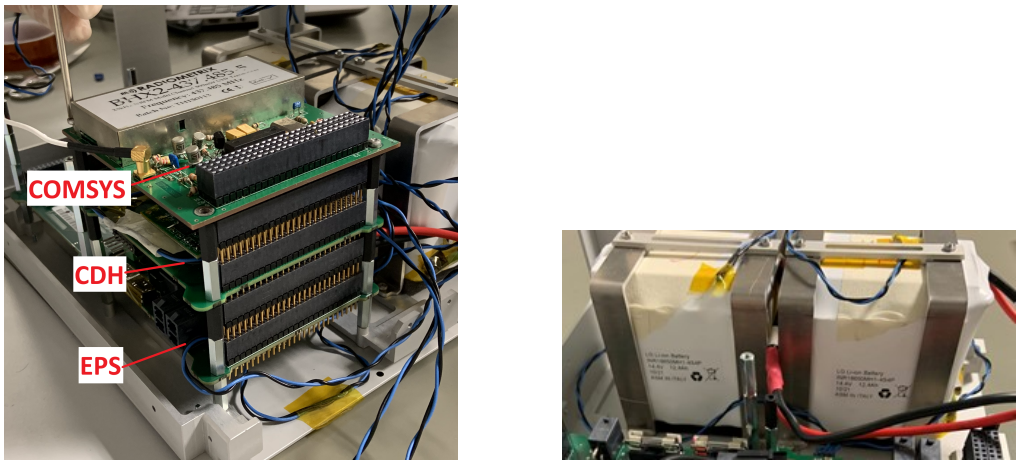


(a) Block scheme **(b)** Board -X face

Figure 2.12: COMSYS board

Equipment	Component	Name
RF Unit	Radiomodule	Radiometrix - BHX2
	TNC	Microchip PIC16F88-I/SO PIC 8 bit
	AFSK MODEM	ADC and DAC
	Antenna	In house developed
Other	Printed board	96x90 mm - Double face printed board
	Other components	passive electronic components, oscillator 19MHz, cables, connectors.

Table 2.8: List of main components of COMSYS



(a) Basic Avionics **(b)** Battery Packs

Figure 2.13: Basic Avionics and Battery Packs

2.2.6 Electric Propulsion Interface System

While the previous boards and their functions were already designed, implemented and available at the start of ESA-µProp3, the Electric Propulsion Interface System has been designed from scratch.

Electric Propulsion Interface System (EPIS) is the system that provides the interfaces of CTP towards the EP system and the instruments and devices to measure the parameters for assessing the mutual interactions between CTP and EP system. Two main parts constitute EPIS: the DataLogger or Logic (DL) which hosts the core logic, and the Power(PW) responsible for electrically supplying the EP system.

2.2.6.1 EPIS-DataLogger

EPIS DataLogger deals with the measurements of the electromagnetic environments, the thermal environment and the power consumption, and provides the digital interfaces towards the EP system. The functional tree of the Data Logger is reported in Figure 2.14.

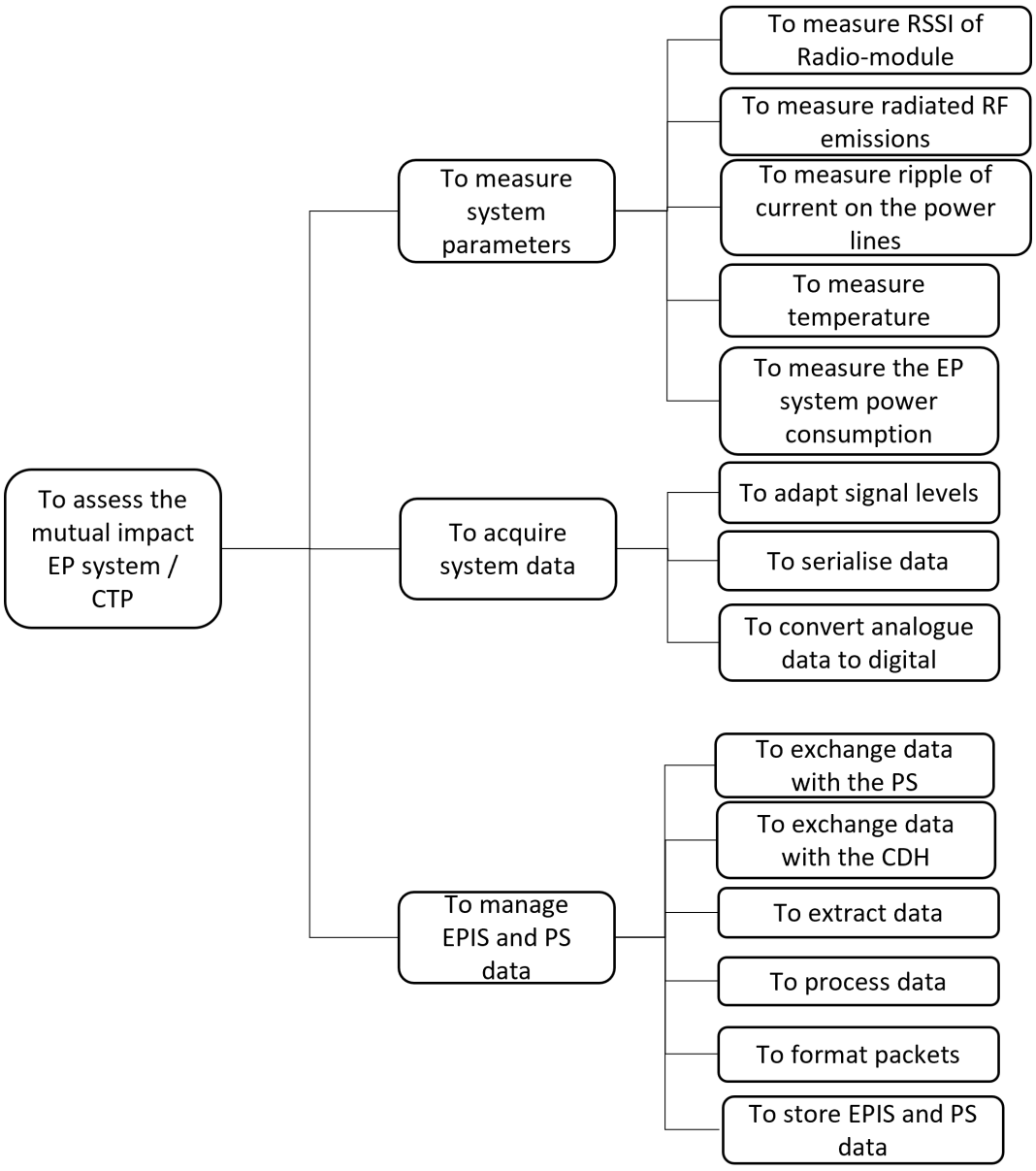


Figure 2.14: ESA-µProp3 Functional Tree

The measurements of the electromagnetic field (EMF) are characterized by:

- **RSSI (Received Signal Strength Indicator) of the radio-module.** RSSI indicates strength of the signal received by the radio-module. This value allows identifying the level of noise around the reception components. This value can change if other sources of RF signal are around the receiver. Radio-modules, in general, provides a signal in output that is proportional to the S/N ratio in input the receiver. The filtered signals pass through a conditioning circuits and an ADC and it is sent to the CPU.

- **RF emissions strength.** each device (e.g. the EP system) generates an electromagnetic field measurable in different ranges of frequencies: it means to quantify the strength of the signals at different frequency. The measurement of this radiated environment is based on the strength of emission at different frequencies is measured through RF equipment and a stage of filtering. The filtered signals pass through a conditioning circuits and an ADC and it is sent to the CPU.

- **Current ripple on the electrical interfaces between CTP and EP system.** EP system is a load that can generate narrow and high rate variations that impact on the generator functionality. The measurement of current with high sampling rate assesses these variations in time. A Line Impedence Stabilization Network (LISN) circuit based on low-pass filters placed in the line the supplying line generates a signal proportional to the current variations. This signal is acquired and passed to the CPU.

The assessment of the thermal environment is characterized by a wide set of measurements of the temperatures in different points of the CTP. Temperature sensors reports the thermal behaviours both at system level and on specific parts and equipment. A group of sensors is dedicated to the avionics system for redundancy, while another one to the EP system. The power consumption of the EP system is evaluated in time measuring the supplied voltage and the absorbed current. The EPIS-DataLogger exchanges data and commands with the CDH and the propulsion system. The communication with the CDH is a serial connection based on the RS232 protocol, while towards the hosted propulsion system is guaranteed by different protocols such as I2C, SPI, RS232, RS485 and CAN BUS. Going into details, EPIS-DL can pass commands (both self-generated and from GSS) and send all the telemetries related to propulsion system (i.e sent by the propulsion system) and the CTP information abovementioned in this paragraph.

The EPIS-DataLogger is composed by a micro-controller, the Acquisition Unit, the EP System Communication Unit and different communication interfaces to communicate with the MCU as well as different connectors.

	LISN circuits	RF reception circuits	Temperature sensor	Voltage sensing circuit	Current sensing circuit	Conditioning circuit	ADC + MUX	CPU	Memories
To measure RSSI of Radio-module	X								
To measure radiated RF emissions		X							
To measure ripple of current on the power lines					X				
To measure temperature			X						
To measure PS electrical power				X					
To adapt signal levels						X			
To serialise signals							X		
To convert analogue data to digital							X		
To exchange data with the PS								X	
To exchange data with the CDH								X	
To extract data								X	
To process data								X	
To format packets								X	
To store EPIS and PS data									X

Table 2.9: F/E matrix of DL - component level

- **Micro-controller.** The MCU is the core of the EPIS sub-system since it controls the communication with EP system, with CDH and the Acquisition Unit. A customized version of Debian-Linux is the selected OS due to its flexibility and reliability. A customized kernel allows the configuration of different interfaces over the GPIOs simply modifying a configuration file before the boot of the OS. The firmware of the EPIS-DataLogger is written in C/C++ language, cross-compiled for ARM-v8 architecture micro-controller or directly compiled on the board, while the binary file can be loaded through the micro-SD card or sent over Ethernet with SCP.

- **EP System Communication Unit.** The EP System Communication Unit consists of different communication interfaces in order to be as much flexible as possible. Supported communication protocols are I2C (Single-Master or Multi-Master), SPI, RS232, RS485, CANbus. To configure the right interface a dedicated configuration file is necessary. A RS232 adapter is used to allow the communication between the EPIS-Logic and the CDH.

- **Acquisition Unit.** The Acquisition Unit consists of one ADC dedicated

to acquiring the thermal status of the EP system and CTP sub-systems with 12 NTC thermistor, and 4 PT1000 thermistor, the EMF status with 2 set of 4 different RF-sensors, each one responsible of measuring a different band section (1-10 MHz, 20-50MHz, 50-120 MHz, 400-500 MHz). One set is placed on the -X side of the Bulkhead (facing the EP system) while the other set is placed on the +X side of the Bulkhead (facing the Service Module). Also, a Line Impedance Noise Sensor (LISN) is placed between the EP System power supply port and the actual EP System and mounted on the +X side of the Bulkhead.

- **Connectors.** EPIS-DL board is equipped with one 2x26 pins female connectors (P1) that composes the 52-pins bus, two 2x20 pins male header connector for the external thermoresistors (P4) and RF sensors (P5), one 2x10 pins male header for communication with the EP system, one Ethernet connector (P2) communication with the GSS (over SSH) and one 1x4 pins male header for serial communication RS232 with CDH (P6).

2.2.6.2 EPIS-Power

The EPIS-Power is the sub-system responsible for the supply of regulated power to the EP System and EPIS-DataLogger. It receives unregulated power from EPS board directly from the batteries, and is able to deliver a voltage between 12V and 28V with a maximum power delivery of 120W to the EP system, and supply the on-board circuits that are working at 3.3V and 5V through two power buses. The functional tree of EPIS-Power is reported in Figure 2.15.

Energy delivered by EPS is regulated by two buck converter, and a programmable TrimDAC is used to set the output voltage of a DC-DC Buck-Boost converter in the range between 12V-28V. Two enable signals are used to close the connection with the EP System, which are active HIGH. In case of loss of communication with the EPIS-DL micro-controller, the output is disabled and EP System is automatically switched off.

The EPIS-Power is composed by the Regulation unit, the DC-DC Buck-Boost converter, the protection circuits and the Acquisition Unit.

- **Regulation Unit.** Power coming from the batteries is regulated by this circuits which can provide 3.3V@1A and 5V@3A on the two power buses.

- **Protection Unit.** Two different protection solutions are implemented: one fuse is used to isolate the unregulated power coming from EPS to the on-board circuits, another fuse is used to isolate the regulated power from the DC-DC Buck-Boost converter to the EP System.

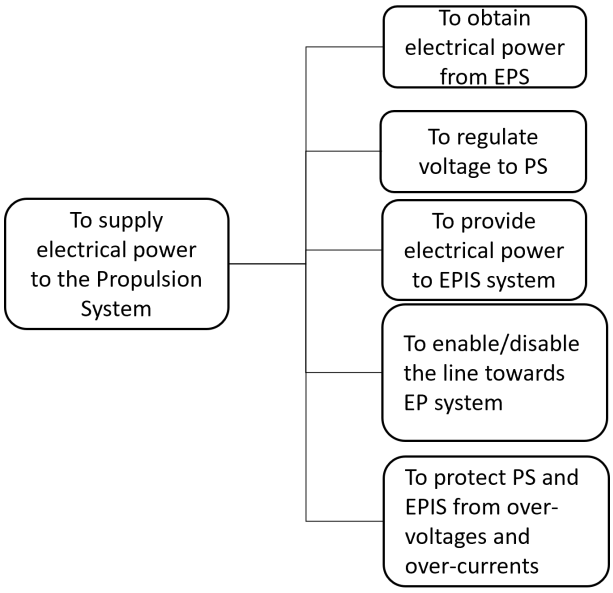


Figure 2.15: EPIS-Power Functional Tree

	External source interface	Boost regulator	Voltage regulators unit	Enabling/disabling circuit	Protection Circuit
To obtain electrical power from the EPS	X				
To provide regulated voltage to PS		X			
To provide power to EPIS			X		
To enable/disable the line towards EP system				X	
To protect EP system and EPIS from over-voltages and overcurrent					X

Figure 2.16: F/E matrix of EPIS-Power - component level

- **Buck-Boost converter.** A DC-DC Buck-Boost converter is used to regulate the input power from EPS in a range between 12V-28V with a maximum power consumption of 120W. The circuit is enabled/disabled by two logic signals coming from the EPIS-DL micro-controller. The output voltage is set by programming a Digital Potentiometer through a SPI line with the MCU.
- **Acquisition Unit.** The acquisition unit is composed by one ADC with at least 16 channels and its conditioning circuits. It gathers the measurements of voltage and current of the different power buses and output circuits plus two on-board temperature sensors. Communication with acquisition unit is achieved through a SPI bus on the 52-pins bus by the EPIS-DL MCU.

- **Connectors.** EPIS-Power board is equipped with one 2x26 pins female connector(P1) that compose the 52-pins bus. One connector (P3) is used to retrieve unregulated power from the EPS, while another connector (P2) is used to supply the EP system.

2.2.7 Physical Layout

The physical layout of the CTP is characterized by two main parts contained in the 12U structure:

- The Propulsion System box (PS box) is a module with dimension up to 8 U where the EP system under test finds place. The adaptability of the module is guaranteed by a bulkhead used to fix the PS to the structure and to interface the EP system with the Service Module.

- The Service Module bos contains the spacecraft bus and each equipment that supports the tests. The avionics boards are located in the Avionics box: EPIS DL and EPIS Power are mounted on a stack of four bars that fix the box to the primary structure. Similarly, Basics Avionics (i.e. CDH, EPS and COM SYS boards) are mounted on a stack of four bars that fix the them box to the primary structure. The external interfaces (battery packs recharging ports, hardline connectors, and switches) are available on the -Y face. The electrical and data interfaces towards the EP System are located along +X. The entire Avionics box occupies a 2U on the -Y side of box. The side +X/-Y is allocated to the Battery packs. They stay in supports made of peek material. Figure 2.17 shows the battery packs on the right, the Avionics box on the left. The bulkhead divides the service module from the PS box.

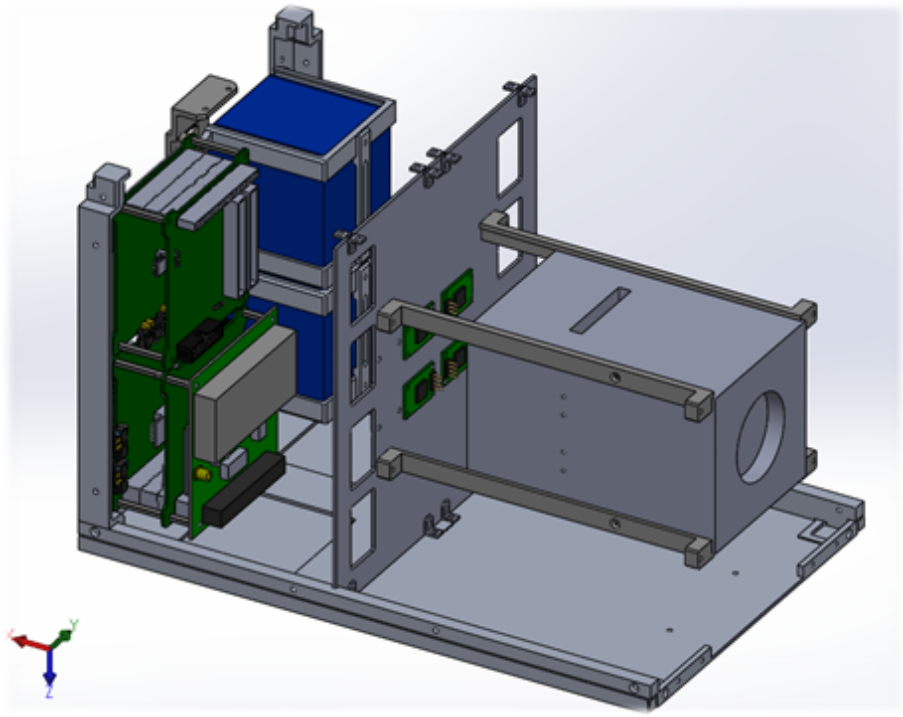


Figure 2.17: ESA-µProp3 CTP Physical Layout

2.3 Software architectural design

The functional requirements analysis clearly identifies the role of each sub-system, as well as their interactions inside the platform. In the context of this project, it is responsibility of each domain expert to expand the relevant blocks of the functional tree.

Regarding the software development, it is possible to state that the CDH and EPIS-DL subsystems are the ones that more relates to software coding and software-hardware interfacing. Since the CDH software was been already developed, tested and verified, the main goal of this activity has been to correctly translate the functional blocks of the functions tree of EPIS-DL to a software product. To brief up the content of the tree, three main functions have been assigned to EPIS-DL:

- **Monitor.** Monitoring consists of the measurement on board parameters such as voltages, currents, temperatures and the acquisition of these parameters through the conditioning of the sensing signals, their serialization and conversion of signals from analogue to digital.
- **Control.** Control consists of the data and command handling, the management of the on board tasks and failures. Data handling foresees extracting,

processing and formatting the acquired data. Command handling passes through the reception, validation and decoding of the commands from the CTP master (i.e. CDH).

- **Characterize.** Characterization consists of the measurement of the mutual impact between the EP system and CTP sub-systems. This include the measurement of temperatures, electromagnetic environment, voltage and currents absorbed by the EP system.

The Function/Equipment matrix of EPIS-DL summarizes the relation between each assigned function and the related equipment. In this way the required software-related components can be found in a quick and effective way. For example, it can be noticed that:

- to store EPIS and EP system's data at least one memory is required. This could be an external memory storage (SD Card) or the internal one (eMMC Flash);
- sensors or sensing circuits such as: voltage, current, temperature, electromagnetic environment are required;
- considering the high demand in terms of acquisition from sensors, proper sensing and acquisition circuits shall be considered: multiple-channel Analog to Digital Converters (ADC) are required.
- multiple communication buses shall be considered: one dedicated to exchange data with CDH, one to collect and command the ADCs, and considering the high flexibility required in terms of EP System selection, multiple buses to communicate with the EP systems, which shall be physically concurrently available and easily configurable by software.

2.3.1 Operative Modes

By analyzing a possible test scenario for the platform, it is possible to figure out which may be the potential operative modes of the software. The current operative mode is normally selected by the master CDH system, but in some particular case the EPIS-DL could change the operative mode of CTP. A scheme of an experiment scenario is sketched in Table 2.18:

Since the test scenario evolves into different phases, also the software behaviour has to change subsequently. This imposes the following operative modes:

- **Dormant mode:** CTP and subsystems are switched off; no operations are required by the software in this mode;

Phase	Sub-Phase	Starting event	Ending event	Operative mode
Integration	Integration	CTP and PS delivered at EPL and checked separately	CTP and PS integrated and checked	Dormant/CTP/PS
Set up	Installation of CTP in SPF	CTP, SPF and GSE checked	CTP installed in SPF	Dormant
	Pre-experiment checks	Functional check test of CTP, SPF and GSE	CTP, SPF and GSE completed checks	CTP/PS
	SPF activation	SPF door closed and activated	SPF reaches the operative conditions	Dormant
Execution	CTP activation	CTP commanded ON	CTP checks completed	CTP
	PS activation	PS commanded ON	PS checks completed	PS
	Test sequence execution	Test started (Thruster ON)	Test stop (Thruster OFF)	Burst
	CTP deactivation	CTP checks started	CTP commanded OFF	PS/CTP
Conclusion	SPF deactivation	SPF switched OFF	SPF checks completed	Dormant
	CTP disassembly	SPF door open	CTP out of SPF	Dormant
Analysis	Data collection	Data collected from CTP	Data processing completed	Dormant
	Data analysis	Data available to users	Experiment results available	Dormant
	CTP checks	CTP available for test	CTP checked	CTP
CTP stowage	CTP stowage	End of data processing	Next test session	Dormant

Figure 2.18: Testing phases

- **Basic mode (CTP):** CTP basic avionics and EPIS are active, while EP system is off. In this mode the software must collect data and telemetry from the CTP avionics and communicate with the COMSYS;
- **PS mode:** CTP basic avionics and EPIS are active, EP system is switched on while thruster is off. EPIS-Power Buck-Boost converter must be turned on. Typically in this mode the EP system is switching on and preparing to burst. The software must perform the same operations of Basic mode but also must collect telemetry and communicate with EP system;
- **Burst mode:** CTP basic avionics and EPIS are active, EP system is switched on, thruster is active. The software shall perform the same operations of PS mode, but some EP system - specific commands could be hard-programmed inside the software to be automatically sent.

Transitions among the different operative modes can be performed in three different ways:

- Commanded, by issuing a command from the GSS;

- Autonomously, executed by the software if specific conditions occour;
- Manually, by the operators.

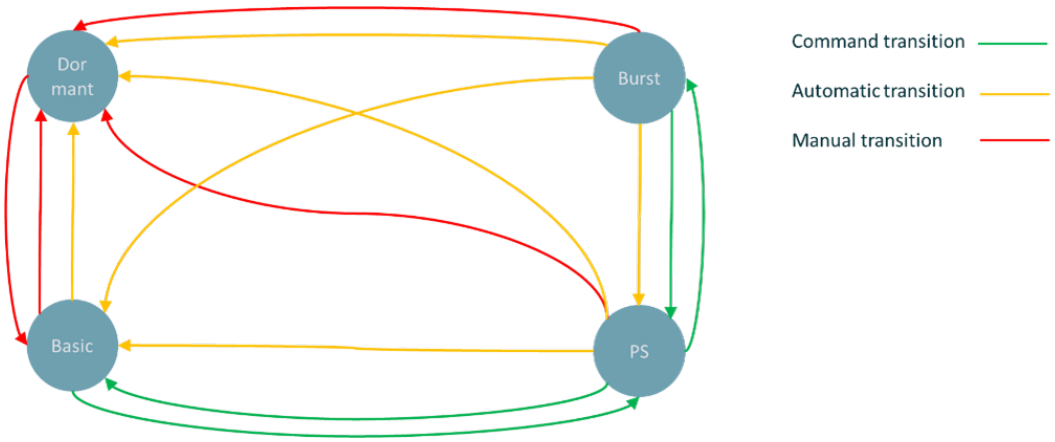


Figure 2.19: Operative Modes

2.3.2 Commands' definition

From the mission scenario, a list of commands that the on-board software shall handle is defined. These commands follow a simple nomenclature scheme: if the command is targeted to both CDH and EPIS-DL boards, it starts with *C*, if it targets only the EPIS-DL board, it starts with *D*. Furthermore, some commands require a parameter, while others not. These commands are:

- **C01 - Nominal.** Switches the internal operative mode to Basic.
- **C02 - PS On.** Switches the internal operative mode to Propulsion System On. In this operative mode the EP System power supply bus is activated, with a standard voltage of 12.0V, which can be further modified by a specific command.
- **C03 - Burst.** Switches the internal operative mode to Burst.
- **C04 - Power Off.** The software execution is interrupted.
- **C05 - Reboot.** The software execution is interrupted, and the operative system is rebooted.
- **C06 - Change Time.** The internal Mission Elapsed Time (MET) is set to the timestamp specified by the parameter.

- **C07 - Change Log.** The internal log file name is set to the name specified by the parameter.
- **D01 - Change EPs interface.** Change the software interface port with the EP system specified by the parameter.
- **D02 - Change EPs interface parameter.** Change the software interface speed with the EP system to the one specified by the parameter.
- **D03 - Change EPs supply voltage.** Change the voltage supplied to EP system to the one specified by the parameter.

A command, when sent to the CTP, is composed by the Command Identifier and an optional parameter.

Command Name	**Command ID**	**Parameter**	**Size [bytes]**	**Example**
Nominal	C01	None	3	C01
PS On	C02	None	3	C02
Burst	C03	None	3	C03
Power Off	C04	None	3	C04
Reboot	C05	None	3	C05
Change Time	C06	ddhhmmss	11	C0600000000
Change Log	C07	String	Up to 30	C07log_file.txt
Change EPs IF	D01	char	4	D01I
Change EPs IF param	D02	7 unsigned int	10	D0201000000
Change EPs V	D03	Float	7	D0312.0

Table 2.10: CTP commands definition

Two commands need to be further explained: the first is **D01 - Change EPs interface**. This command receives one char as parameter that specifies which interface is used by the EP System among the following:

- I for I2C;
- S for SPI;
- C for CANBus;
- U for UART-RS232;
- V for UART-RS485.

The command **D02 - Change EPs interface parameter** is used to set the communication speed (bitrate) of the interface. It is specified by a 7-digits unsigned integer, sufficient to cover all the supported speeds of the specified communication protocols.

It is important to underline that this list does not include any EP System-specific commands. This argument will be covered in a further section.

2.3.3 Data and Command budget

The EPIS-DL system interfaces with the GSS only through CDH board. This meant that the packets structure had to be carefully designed in order to not overload CDH serial port. All the CTP-related commands occupy at most 11 bytes, while analyzing the previous iterations of the project, an EP System command could be as long as 200 bytes.

Two different packets have been defined: one **from CDH to EPIS-DL** (C2D) containing the command, CDH status and more; a second one **from EPIS-DL to CDH** (D2C) containing the acquired measures, EPIS status and the EP System telemetry.

header	time	status	opmode	command	crc32	closer
3 bytes	4 bytes	2 bytes	1 byte	200 bytes	4 bytes	3 bytes
217 bytes						

Figure 2.20: EPIS-DL Rx packet structure

header	time	status	saved_pack_number	opmode	last_comm	ps_telemetry	dl_telemetry	crc32	closer
3 bytes	4 bytes	2 bytes	4 bytes	1 byte	1 byte	70 bytes	68 bytes	4 bytes	3 bytes
160 bytes									

Figure 2.21: EPIS-DL Tx packet structure

Packets are encapsulated between a header and a closer in order to clearly identify the packet frame. Inside the header the direction of the communication is encoded:

- header = "C2D" for packets from CDH to EPIS-DL;
- header = "D2C" for packets from EPIS-DL to CDH,

while the closer is always the string "END". A Cyclic Redundancy Check code of 32 bits (CRC32) is computed for each packet and appended to it to detect accidental changes in the data. At each end point the received CRC32 is compared to the self-computed one: if it differs the packet is discarded.

Parameter	Label	Nominal range	Number of bytes
MET	time	[0-2678400]	4
Extra status	status	[0-65535]	2
Saved data packet number	saved_pack_number	[0 - 4294836225]	4
Operative Mode	opmode	[1 - 3]	1
Last Command received	last_comm	[1 - 255]	1

Figure 2.22: Configuration parameters

Parameter	Label	Nominal range	Number of bytes
Bus 3.3V Voltage	bus3V_volt	[3,2 - 3,4] V	2
Bus 5V Voltage	bus5V_volt	[4,9 - 5,1] V	2
Bus 3.3V Current	bus3V_curr	[0 - 1]A	2
Bus 5V Current	bus5V_curr	[0 - 1]A	2
Bus EP System Voltage	bus12V_volt	[0 - 28] V	2
Bus EP System Current	bus12V_curr	[0 - 6] A	2
Bus EPS-EPIS Voltage	busPS_volt	[14 - 16,8]V	2
Bus EPS-EPIS Current	busPS_curr	[0,1 - 6] A	2
EPIS-Power Temperature	epis_temp	[0 - 60] °C	2
Battery 1 Temperature	bat1_temp	[0 - 60] °C	2
Battery 2 Temperature	bat2_temp	[0 - 60] °C	2
BCR Temperature	bcr_temp	[0 - 60] °C	2
COMSYS Temperature	comm_temp	[0 - 60] °C	2
CDH Temperature	cdh_temp	[0 - 60] °C	2
EPS Temperature	eps_temp	[0 - 60] °C	2
EPIS-DL Temperature	dl_temp	[0 - 60] °C	2
Extra-Temperature 1	extra_temp1	[0 - 80] °C	2
Extra-Temperature 2	extra_temp2	[0 - 80] °C	2
Extra-Temperature 3	extra_temp3	[0 - 80] °C	2
Extra-Temperature 4	extra_temp4	[0 - 80] °C	2
Extra-Temperature 5	extra_temp5	[0 - 80] °C	2
Extra-Temperature 6	extra_temp6	[0 - 80] °C	2
Extra-Temperature 7	extra_temp7	[0 - 80] °C	2
Extra-Temperature 8	extra_temp8	[0 - 80] °C	2
Extra-Temperature 9	extra_temp9	[0 - 80] °C	2
LISN	lisn	[-24,0 - 22,0] dBm	2
RF sensor 1	rf1	[-24,0 - 22,0] dBm	2
RF sensor 2	rf2	[-24,0 - 22,0] dBm	2
RF sensor 3	rf3	[-24,0 - 22,0] dBm	2
RF sensor 4	rf4	[-24,0 - 22,0] dBm	2
RF sensor 5	rf5	[-24,0 - 22,0] dBm	2
RF sensor 6	rf6	[-24,0 - 22,0] dBm	2
RF sensor 7	rf7	[-24,0 - 22,0] dBm	2
RF sensor 8	rf8	[-24,0 - 22,0] dBm	2

Figure 2.23: List of EPIS-DL telemetry data

2.3.4 Hardware Architectural definition

2.3.4.1 EPIS-DataLogger

Once the overall requirements for the EPIS sub-system have been completed, the first step has been to choose an adequate MCU for the EPIS-Datalogger board. The Compute Module 4 (CM4) by The Raspberry Pi Foundation [8] has been selected. This System-on-Module (SoM) is equipped with processor, memory, eMMC Flash and supporting power circuitry, and is configured with Raspberry Pi OS, which is a free operating system based on Debian, optimised for the Raspberry Pi hardware, and with a custom Linux Kernel 5.13. This MCU has been selected because of its high flexibility in terms of communication ports, programmable peripherals and overall performances. Along with the MCU a dedicated development board Compute Module 4 I/O board (CM4IO) [9] has been used to start the development and testing phase before the operational boards were available. The main parameters of the SoM are listed:

- Broadcom BCM2711 Quad core Cortex-A72 (ARM v8) 64-bit SoC @ 1.5GHz [10]
- 1GB LPDDR4-3200 SDRAM with ECC
- 0 or 8GB eMMC Flash Memory (depending on the SoM used)
- 0 °C to +85 °C temperature tolerant

The module offers 28 General Purpose Input Output (GPIO) pins, which can be easily programmed in order to have the following peripherals available:

- Up to 5 UART
- Up to 6 I2C (5 Masters and 1 Slave)
- Up to 5 SPI
- 1 SDIO

and many more that are not necessary for this project.

Each GPIO can be programmed in 6 alternative function, which can be found in the BCM2711's datasheet. Based on the requirements and the peripherals available, each GPIO has been assigned to a specific function (only the used GPIO are listed). In some cases a GPIO can be mapped to more than one function: its behaviour can be easily managed via software.

GPIO	Function	Description
0	TXD2	UART Tx for communication with CDH
1	RXD2	UART Rx for communication with CDH
2	SDA1	I2C Master Data Line for communication with EP System
3	SCL1	I2C Master Clock Line for communication with EP System
4	TXD3	UART Tx for communication with EP System
5	RXD3	UART Rx for communication with EP System
7	SPI0_CE1_N	SPI Chip Select for CANBus controller for communication with EP System
8	SPI0_CE0_N	SPI Chip Select for communication with EP System
9	SPI0_MISO	SPI Master In Slave Out for communication with EP System
10	SPI0_MOSI	SPI Master Out Slave In for communication with EP System
	BSCSL SDA	I2C Slave Data Line for communication with EP System
11	SPI0_SCLK	SPI Clock Line for communication with EP System
	BSCSL SCL	I2C Slave Clock Line for communication with EP System
14	TXD1	UART Tx for communication with EP System
15	RXD1	UART Rx for communication with EP System
16	SPI1_CE2_N	SPI Chip Select for communication with EPIS-Power ADC
17	SPI1_CE1_N	SPI Chip Select for communication with On-Board ADC
18	SPI1_CE0_N	SPI Chip Select for communication with On-Board ADC
19	SPI1_MISO	SPI Master In Slave Out for communication with ADCs
20	SPI1_MOSI	SPI Master Out Slave In for communication with ADCs
21	SPI1_SCLK	SPI Clock Line for communication with ADCs
24	Output	Enable 1 for the DC-DC Buck-Boost converter on EPIS-Power
25	Output	Enable 2 for the DC-DC Buck-Boost converter on EPIS-Power
26	SPI1_CE3_N	SPI Chip Select for communication with TrimDAC on EPIS-Power

Table 2.11: CM4 GPIO function assignment

A first sketch of peripherals distribution to the external interfaces has been devised in figure 2.24.

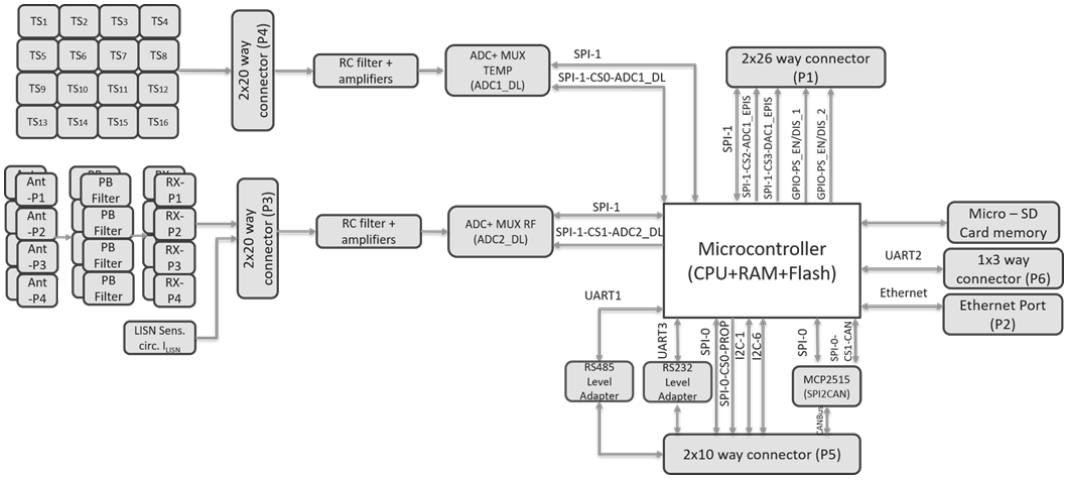


Figure 2.24: EPIS-DL Block Scheme

From the figure it is possible to notice:

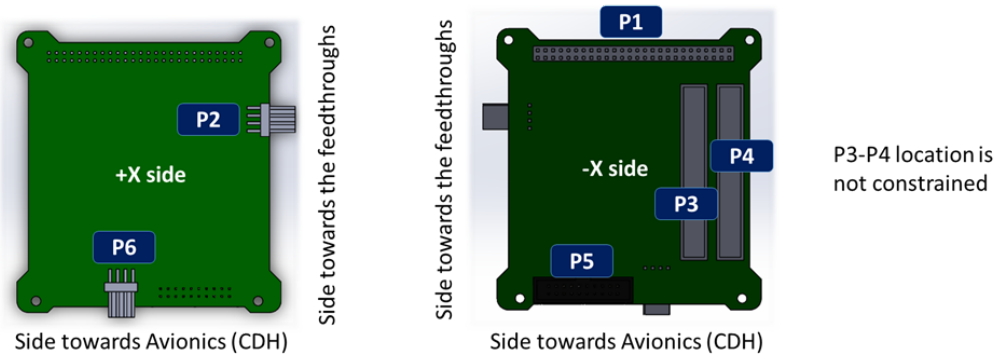
- One UART line has been dedicated to the communication with CDH (UART2);
- Two UART lines have been dedicated to the communication with EP System. One line for the RS232 protocol (UART3) while the other for RS485 protocol (UART1);
- Two I2C lines have been dedicated to the communication with EP System. One Master (I2C1) and one Slave (BSCSL);
- One SPI line has been dedicated to the communication with EP System (SPI0). The same line controls a CANBus controller to communicate with EP System;
- One SPI line has been dedicated to the acquisition of data from ADCs (SPI1);
- One Ethernet line has been dedicated to communicate with the micro-controller via SSH.

This block diagram also contains all the circuitry necessary to adapt the communication interfaces to the desired one. In fact:

- **UART1** requires a RS485 level adapter;
- **UART2 and UART3** require a RS232 level adapter;
- **CANBus** requires a standalone CAN controller controlled via SPI;
- **SPI0 and SPI1** require a 5V level adapter;

- **ADCs** input signals require filters and amplifiers.

A design of the board layout has been also sketched:



P1: 2x26 connector Towards EPIS-Power (SPI-1)
P2: Debug line of the processor (Ethernet)
P3: interfaces with 8x RF sensing circuit and 1 x LISN circuit
P4: interfaces with Temperature sensors (12 x NTC + 4 x PT1000)
P5: DIGITAL interfaces (i2c-1,i2c-6, spi0, uart1-rs485, uart3-rs232, can bus) towards PS
P6: DIGITAL interfaces (uart2) towards Avionics

Figure 2.25: EPIS-DL Board Layout

The actual EPIS-DL hardware schematic, with the real components has been designed and sent to an external supplier for production.

Since one of the main objective of this sub-system is to acquire a great number of analog sensor measurements, it has been decided to equip it with two dedicated Analog to Digital Converter with a balanced tradeoff between conversion speed and conversion precision: the 24-Bit Sigma-Delta ADC LTC2449 by Linear Technology has been selected. This ADC offers up to 16 acquisition channels and can be controlled via SPI. Two ADCs have been installed on EPIS-DataLogger, while a third one on EPIS-Power.

Equipment	Component	Name
DataLogger	Micro-controller	Raspberry CM4
	External memory	SD Card
	Conditioning circuit	Amplifiers and passive electronic components
	ADC+MUX	24-Bit Sigma-Delta ADC LTC2449IUHF
	Sensing circuits for current and voltage	Amplifiers and passive electronic components
RF Sensing board	Antenna	STILO
	RF Receiver	LTC55xxx
	Band pass filters	Passive electronic components
Other sensors	Temperature sensors	PT1000, NTC
	Current sensing circuit	Amplifiers and passive electronic components
Other	Printed Board	96x90 mm - Double face printed board
	Other components	Cables, connectors, amplifiers and passive electronic components

Table 2.12: List of main components of EPIS-DL

2.3.4.2 EPIS-Power

The overall design for EPIS-Power has been very similar to EPS board since most of the functions it provides are equivalent to it, but the extra feature of this board is the DC-DC Buck/Boost converter which can be set to provide a voltage in the range between 12V-28V with a maximum current of 10A, in total max 120W. The converter output is enabled enabled by two GPIOs output from EPIS-DL, and the voltage is set by programming a digital potentiometer via SPI, also from EPIS-DL. The potentiometer changes a resistance value in the feedback loop of the converter, which subsequently modifies the output voltage. A first sketch of the different blocks composing EPIS-Power is devised in Figure 2.26, as well as the possible layout of the board in Figure 2.27.

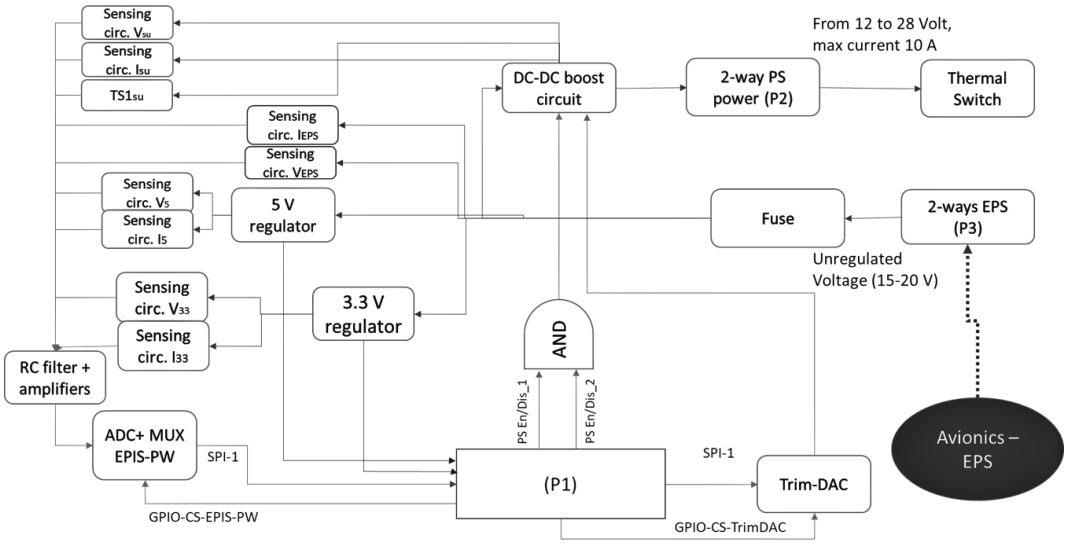


Figure 2.26: EPIS-Power Block Scheme

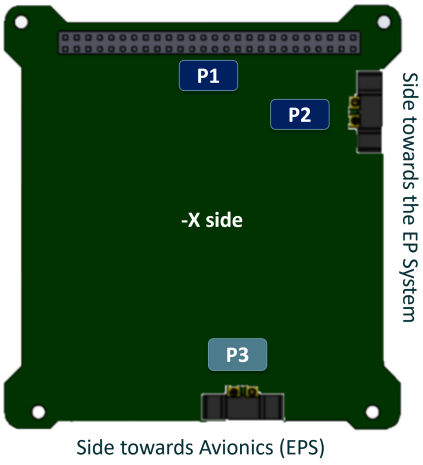


Figure 2.27: EPIS-Power Board Layout

Equipment	**Component**	**Name**
EPIS-Power	Protection elements	Refreshing Fuses, SMD Bourns, 15mΩ max
	DC-DC Buck/Boost regulator	LT3790EFEPBF
	Digital Potentiometer	AD5141BCPZ10-RL7
	Thermal switches	Honeywell - 250 Series
	Fuses	Fuse 15 A - FSV12120V
	ADC+MUX	LT2449
Other	Printed board	96x90 mm - double face printed board
	Other components	passive electronic components, amplifiers

Table 2.13: List of main components of EPIS-Power

2.3.5 Software Architectural definition

Once the hardware physical architecture has been clarified, the software logical model can be explained more easily. In this section architectural and logical strategies will be described only, while development details on coding will be addressed in the next chapter. First a static model will be presented, then its dynamic behaviour will be carefully studied, specifying its real-time behaviour, reusability, safety and reliability.

2.3.5.1 Software Static model

MAIN thread	MASTER thread	SLAVE thread

Initialization
• Set-up peripherals (SPI, UARTs, Watchdog)
• Set-up configurations (time, propulsion configurations)
• Initialize TrimDAC
• Start MASTER thread
• Start SLAVE thread
	Loop
	• Read packet from CDH
	• Validate command from CDH
	• Execute command
	  ○ EPIS related command
	  ○ Send command to EP System
	• Acquire data from ADCs
	• Process data
	• Save data and time in memory
	• Send data to CDH
		Loop
		• Enable the listener for EP System
		• Update global data structures

Figure 2.28: EPIS-DL Software static model

From table 2.28 it is possible to notice the static structure of the software. It composed of:

- **Main thread**: main function of the software which is executed at the start. It sets the UART-RS232 peripheral required to communicate with the CDH, as well as the SPI bus to control the acquisition unit. TrimDAC is set with a fixed value in order to have an EP System supply voltage of 12.0V. Then the propulsion configurations are loaded, and the specified communication port is set-up. Finally, two threads are started: Master thread and Slave thread;
- **Master thread**: this thread is composed by a periodic 1 second task dedicated to communicate with CDH, send commands to EP System, acquire measurements from the acquisition unit and store the acquired data;
- **Slave thread**: this thread is dedicated to listen the communication port with EP System, in order to receive its telemetry as soon as it is sent. This telemetry is stored inside a global data structure, accessible to the Master thread.

2.3.5.2 Software Dynamic model

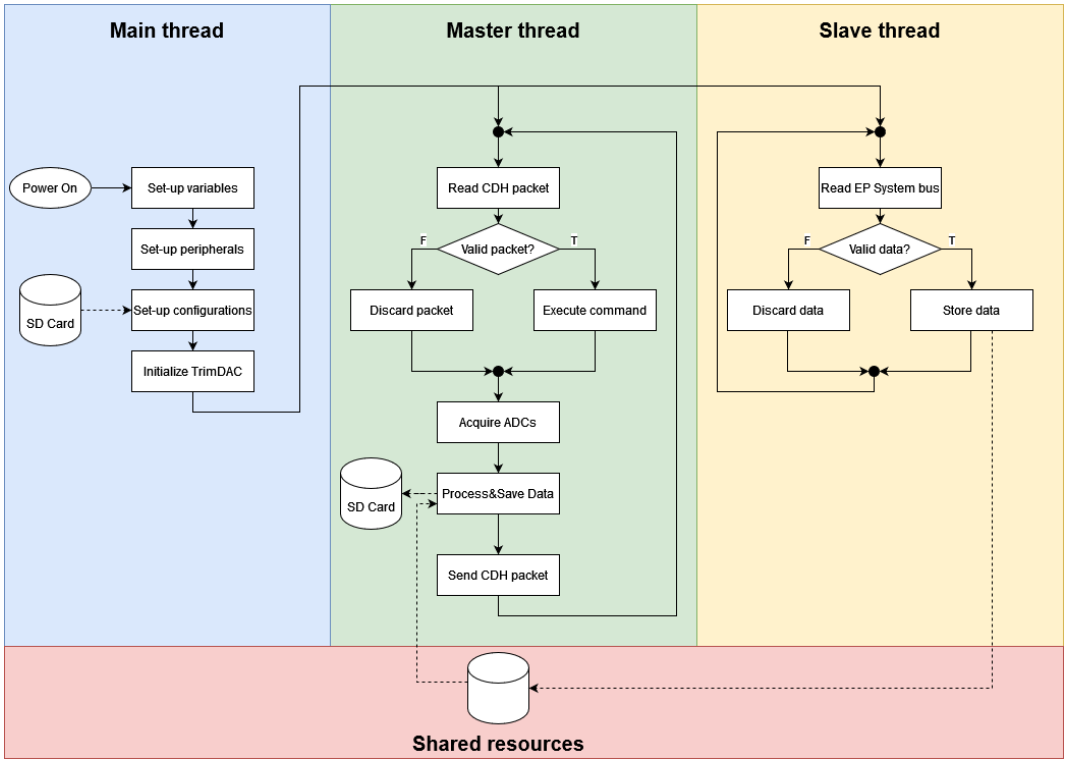


Figure 2.29: EPIS-DL Software dynamic model

The dynamic software flow is presented in Figure 2.29 and each block will be deeply analyzed in the next chapter. Here a brief explanation will be presented.
Main thread:

1. **Set-up variables:** every variable and structure used later in the software are now statically allocated and initialized;

2. **Set-up peripherals:** the communication port UART-RS232 with CDH is set and opened, as well as the SPI bus to control the ADCs and TrimDAC;

3. **Set-up configurations:** the system loads the last saved mission time from three different files and the EP System parameters from its dedicated file; the specified EP System communication port is set-up and opened;

4. **Initialize TrimDAC:** the DC-DC buck/boost converter output voltage that supplies the EP System is controlled via a digital potentiometer (TrimDAC) over SPI. The corresponding resistance value such to have the supply voltage

specified in the EP System configuration file is sent to the TrimDAC (note: the DC-DC converter is not yet enabled);

Master thread:

1. **Read CDH packet and validation:** the serial line is read and the data found is stored into a buffer. A packet is found when its header and closer are found inside the buffer. The validation process consists of a CRC32 check: if the validation fails the buffer is discarded, else if a command is present inside the packet it is executed;

2. **Execute command:** if it is a CTP-related command from Table 2.10 the execution is performed, otherwise in case of an EP System command the sequence of bytes is just relayed over its communication bus, with no further elaborations;

3. **Acquire ADCs:** the two ADCs on the EPIS-DL and the one on EPIS-Power are read, one channel at a time; the 24-bits data is then trimmed to 16-bits to better optimize the conversion uncertainty;

4. **Process & Save Data and Time:** the data structures referring to the telemetry are updated with the new acquisitions (both from ADCs and EP System), and saved on the SD Card memory, as well as the time word;

5. **Send CDH packet:** the telemetry is put inside the communication packet to CDH, and the corresponding CRC32 is computed. Finally the packet is sent over UART-RS232.

6. The loop is restarted from point 1. after one second from its start.

Slave thread:

1. **Read EP System bus:** the line is read and the data found is stored into a buffer. If something is found (more than 0 bytes) the data is stored in a global shared structure, accessible to the Master thread;

2. The loop is restarted from point 1. after 1us.

2.3.5.3 Software Real-Time Model

As described in the dynamic model the two threads consist of two periodic tasks of respectively 1s and 1us, which seems a reasonable timing strategy when dealing with the technologies involved in this project. This means that commands issued from the CDH to the EPIS-DL (hence from the GSS) are executed within 1 second. Considering these type of deadlines it is possible to assert that the software operates

in **soft real-time**, so even if the timing is not precise the system continues to operate nominally. The timing and scheduling of the system is achieved through different solutions:

- **POSIX Threads** (pthread library) used to manage the different threads and their task's scheduling;
- **POSIX Semaphores** used to guarantee the mutual exclusion between the shared global resources;

2.3.5.4 Flexibility

One of the main design driver for this project has been the flexibility of the system to adapt to a new EP System with the minimum amount of changes. Of course for the mechanical installation almost every EP System (if not standardized) would require a dedicated mechanical interface with the CTP (different brackets placement, different nozzle panel size, etc.), but from the software point of view the required changes have been reduced to the minimum. In fact the adopted strategy requires the CTP operator only to:

- Load both on the SD Card and on the GSS software an user-friendly text-based configuration file for the EP System. This file specifies the communication protocol, the required supply voltage and finally the list of supported commands by the EP System;
- Add minor changes to the EPIS-DL firmware to be finely tailored to the EP System, since each one could require specific interface controls (i.e. autonomous periodic commands).

2.3.5.5 Safety and Reliability

The following strategies have been considered to increase the safety and reliability of the platform. Note that these strategies cover only the autonomous on-board solutions, but not the manual approaches which would require the CTP operator to send commands or even to cut the power using the Load Switch.

- **EP System supply bus:** the EP System supply bus is supplied by the DC-DC buck/boost converter, which is activated only when both of its enabling signals are HIGH. The state of these signals is updated at each cycle: if the communication with CDH is lost for more than 5 seconds these signals are turned LOW. Also, in case of MCU failure the two lines are pulled down by two resistors guaranteeing a fail-safe behaviour;

- **Fuses:** two fuses are placed between the EPS-EPIS supply line and between EPIS-EP System line. This guarantees that in case of an over-current event on the EP System side CTP avionics are adequately protected, but also viceversa;
- **TMR:** for storing the time word, Triple Module Redundancy is used.

Chapter 3

Software Implementation and Testing

The design, development and testing of the software units started months before the operational hardware was completed and delivered. This made necessary to organize the testing in different phases, each one characterised by a different hardware configuration. Initially only the Development Boards plus some hardware mock-ups were available, when the EPIS boards were available the integrated system was tested.

3.1 Development and Unit Testing

3.1.1 Set-up of working environment

The first phase was oriented to get some experience with the new Compute Module 4 (CM4) processor and its Development Board (CM4IO). As said before, for this phase two different development boards have been used:

- **RD126 DevBoard** with RD129 CPU to mimic the CDH board;
- **CM4IO DevBoard**[9] with CM4 CPU to mimic the EPIS-DataLogger board.

In order to control the RD129 CPU (CDH) its UART1 debug port has been used in combination with a RS232-to-USB converter. This allowed the direct control of the operative systems using the serial terminal application **minicom** [11]. At the same time to control the CM4 CPU (EPIS-DL) a simple Ethernet connection has been used in combination with **SSH** [12]. Further details on the implementation will be given in the next section.

Before addressing the software development it is worth to describe how the Compute Module 4 was configured, as well as the overall working environment.

3.1.1.1 Installing Raspberry Pi OS Lite

The selected Raspberry Pi OS version was *Lite*, which comes with a minimal set of packages and no graphical interfaces nor desktop environment. This means Pi OS Lite requires very low amount of RAM and CPU usage if compared to the complete version. Also, the selected Compute Module 4 was the Lite version, which comes with no eMMC memory installed on the board; for this reason the OS had to be installed on a SD Card. The main steps to flash the SD Card are summarized:

1. Download from its website the Raspberry Pi Imager [13];
2. insert the SD Card into the host machine;
3. run Raspberry Pi Imager;
4. select as Operating System *Raspberry Pi OS Lite (32-bit)*;
5. press CTRL+SHIFT+X to open the *Advanced Options*;
6. enable SSH;
7. select as *Storage* the SD Card;
8. press *Write*;
9. wait for the download and installation of the image;
10. press *Continue*.

The SD Card is now ready to be installed on the CM4IO DevBoard, and the OS ready to be tested. To communicate with the CM4 there were two feasible options:

- **UART** communication through a TTL-to-USB converter;
- **SSH** communication through Ethernet.

The second option was chosen due to multiple benefits like the possibility to use a standard and simple connector between host and target machine and easy file transfers using SCP [14], which reduced the complexity of the development and testing phases.

3.1.1.2 Installing IDE and cross-compiler on Linux PC

The software development has been carried on a personal Linux PC, where different software packages have been installed to correctly develop, cross-compile and transfer the firmware to the target machine. The procedure is here listed:

1. A PC with Linux 64-bit environment is required;
2. On a terminal type "sudo apt-get install libc6-armel-cross libc6-dev-armel-cross binutils-arm-linux-gnueabi libncurses5-dev build-essential bison flex libssl-dev bc"; press Enter;

3. type "sudo apt-get install gcc-arm-linux-gnueabihf"; press Enter;

The cross-compilation toolchain is now ready, and in order to compile a C application the binary **arm-linux-gnueabihf-gcc** must be used instead of the classical **gcc** command. It is worth to note that it is also possible to directly compile the firmware on the board itself.

The software development was entirely carried on the Visual Studio Code [15], a source code editor by Microsoft. It does not come with compiler but can be expanded with plug-ins that enhance the development process like syntax highlighting, intelligent code completion and much more.

3.1.2 Set-up interfaces

Since the EPIS-DataLogger was designed to host different types of interfaces, the first suite of test was oriented to assert the correct behavior of the interfaces and the communication between the two microprocessors.

3.1.2.1 Ethernet

The first interface that required testing was the Ethernet port, since it will be used during all the development phase as debug port and to access the Operating System. To allow a direct LAN connection between the host PC and the target CM4IO a Dynamic Host Configuration Protocol (DHCP) [16] server service is needed to ran on the host machine. This service will automatically configure a Local Area Network and assign the IP addresses: to accomplish this a Bash script was written. The hardware interconnection simply consists of an Ethernet IEEE 802.3 cable, while the software connection is performed using the Secure Shell (SSH).

Test Success Criteria The host machine is able to succesfully control the target OS via SSH. This means that:

- the LAN is correctly set-up;
- the physical interconnection is established;
- the two machines are in the same LAN;
- host is able to connect through SSH to target;
- host is able to transfer files/directories through SCP to and from target.

List of items

- Personal Computer with Linux system;
- CM4IO board with CM4 CPU installed;
- Flashed SD Card inserted in CM4IO board;
- Ethernet cable;

Test Procedure for SSH

1. Check that on the SD Card for the CM4 the file */boot/ssh* is present. This will enable the SSH service directly at boot;
2. connect an Ethernet cable to CM4IO Ethernet Port;
3. connect the Ethernet cable to Host PC Ethernet Port;
4. open a terminal on the Host PC;
5. type "sh workstation_dhcp.sh" to execute the script; it will create a LAN in the range /24 and assign a static IP address to the host machine;
6. power up the CM4IO; wait 30 seconds;
7. type "nmap 192.168.0.0-254"; the nmap tool will explore the LAN and will eventually find the CM4IO IP address;
8. type "ssh pi@<address>" specifying the address found in the previous step; the pre-configured password is *raspberry*.
9. type "sudo nano /etc/dhcpcd.conf"; a configuration file is now opened and modificable;
10. at the bottom of the file add

    - interface eth0
    - static ip_address=192.168.0.3/24
    - static routers=192.168.0.1
    - static domain_name_servers=192.168.0.1

    this will set 192.168.0.3 as static IP of the target machine on the *eth0* network interface;
11. reboot the CM4 typing "sudo reboot";

Once the procedure is terminated the Host Computer has direct control of the Operative System running on the CM4 CPU: this will allow further tests.

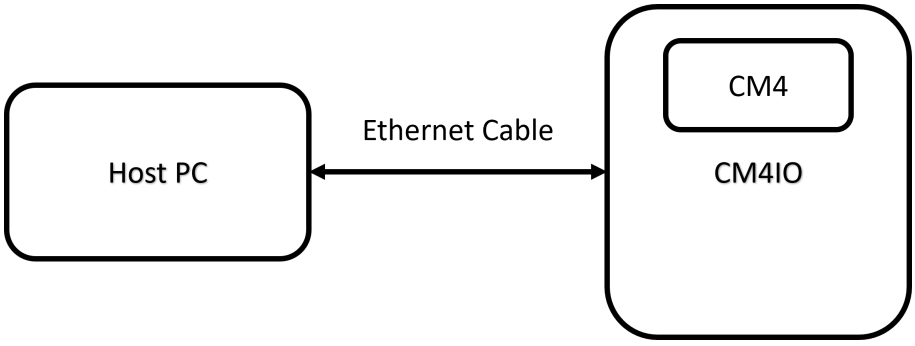


Figure 3.1: Connection between CM4IO and Host Computer

Every time a file (binary or source files) has required to be moved between the target and host machine the Secure Copy (SCP) protocol is used.

Test procedure for SCP

1. Set-up the hardware interconnections as the previous procedure;
2. open a terminal on the host machine;
3. type "scp -r <source(s)> pi@192.168.0.3:<dest>"; this will copy all the files/directories specified by <source(s)> path in <dest> path.

Note: all the following tests require Ethernet connection to control the CM4, so in the list of items and testing procedure the Ethernet link is skipped but the same set-up is maintained.

3.1.2.2 UART

The Compute Module 4 offers a lot of UART (Universal Asynchronous Receiver-Transmitter) ports, but from the specification only 3 were needed:

1. To communicate with the CDH board
2. To communicate with the ePS over RS232
3. To communicate with the ePS over RS485

To accomplish the requirements **UART1** was assigned to RS485 serial communication with EP System, **UART2** to RS232 serial communication with CDH and finally **UART3** to RS232 serial communication with EP System.

At this moment only the Transistor Transfer Level (TTL) signals were available since no level adapters were installed on the DevBoards, but the communication packet were already designed, so the test was conducted to check not only the transmission of the packets, but also their correct decoding. The physical connection between the two DevBoards is shown in Figure 3.2. To test the different ports the same procedure was used where the selected UART is configured as 115200/8N1, meaning that

- 115200 bit/s as baudrate
- 8 bits of data for each frame
- N to disable the parity bit
- 1 bit of Stop
- no software/hardware flow control.

Test Success Criteria All three UART lines work correctly. This means:

- The physical interconnection is established;
- the correct serial configuration is set-up;
- packets are transmitted from CM4IO to RD126;

- packets are received by CM4IO from RD126;
- packets are validated;
- no major losses in the communication.

List of Items

- Two Personal Computer with Linux systems;
- Three female-to-female jumpers;
- RD126 board with RD129 CPU installed;
- CM4IO board with CM4 CPU installed;
- Flashed SD Card inserted in CM4IO board;

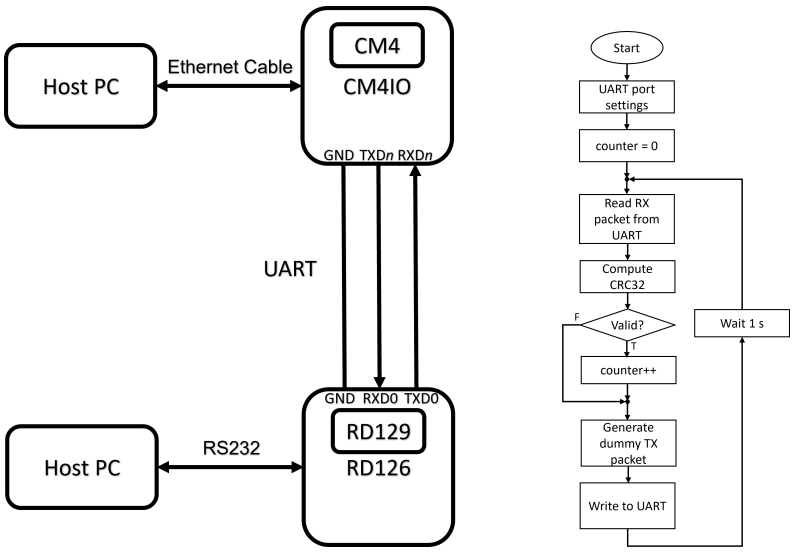
Test Procedure

1. Connect the two DevBoards with three Female-to-Female jumpers (TX, RX, GND);
2. Connect the two DevBoards to the Host computers;
3. Power up the DevBoards;
4. Check that in the Boot Configuration file of CM4, located in */boot/config.txt* the following lines are present:
   dtoverlay=uart1
   dtoverlay=uart2
   dtoverlay=uart3
   This specifies to the Operative System to load the device driver and device tree for the three UARTs, and assign them to predefined pins of the CPU (which are available on the DevBoard);
5. Reboot the CM4 to make sure the device drivers were loaded correctly;
6. Launch the test application *test_uart <N>* where N specifies the port number;
7. Assert the correct reception and decode of the packets on the RD129 by looking at its Host Computer;
8. Assert the correct reception and decode of the packets on the CM4 by looking at its Host Computer.

Test software structure The test starts by setting up the specified UART port, which in the Linux File System are represented by a specific File Descriptor:

Then, an unsigned integer variable *counter* is initialized to zero; this variable will keep track of how many valid packets are received during the test. After that the software enters into an infinite loop where:

- 500 bytes are read from the UART RX buffer;



(a) Connection between CM4IO and RD126

(b) Test flowchart

Figure 3.2: UART interfaces testing

UART port	TX GPIO	RX GPIO	File Descriptor
UART1	14	15	/dev/serial0
UART2	0	1	/dev/ttyAMA1
UART3	4	5	/dev/ttyAMA2

Table 3.1: CM4 UARTs assignment

- the buffer is scanned to find a header and a closer, which would indicate the presence of a packet (packet's structures are defined in **Data and Command budget** section of Chapter 2);
- the CRC32 of the packet is computed and compared to the included one in the packet: if it is valid *counter* is incremented.
- a dummy packet with random payload is generated and sent to the UART TX buffer;
- the software is put into sleep for one second; then the loop restarts.

The same software was let run on both CPUs with minor changes in order to simulate the constant communication between the two.

Results All three UART interfaces worked as expected, with no loss of packets over a period of 60 minutes. In fact, by looking at how many packets were sent by the RD129 and how many were received by CM4, the overall throughput was

100%.
All the code developed and tested for this unit has been wrote into two files: source code inside **uart.c** and header file inside **uart.h**.

3.1.2.3 I2C Master and Slave

The Compute Module 4 offers up to 6 different I2C (Inter Integrated Circuit) ports that are usable only in single master configuration, and one port that can act as an I2C Slave. From specifications an I2C Multi-master port was needed but unfortunately the CM4 CPU does not support this configuration. The proposed solution requires one I2C Master and I2C Slave peripherals connected on the same bus. For this reason the **I2C1** was selected as Master, while the **BSCSL** (Broadcom Serial Control Slave) peripheral was selected as Slave. In order to test both functionalities the Master sends a packet on the bus that should be received by the Slave, which address was previously set.

Test Success Criteria The two I2C lines work correctly. This means:

- The physical interconnection is established;
- the correct communication configuration is set-up;
- strings sent from the Master to the Slave are correctly received by the Slave.

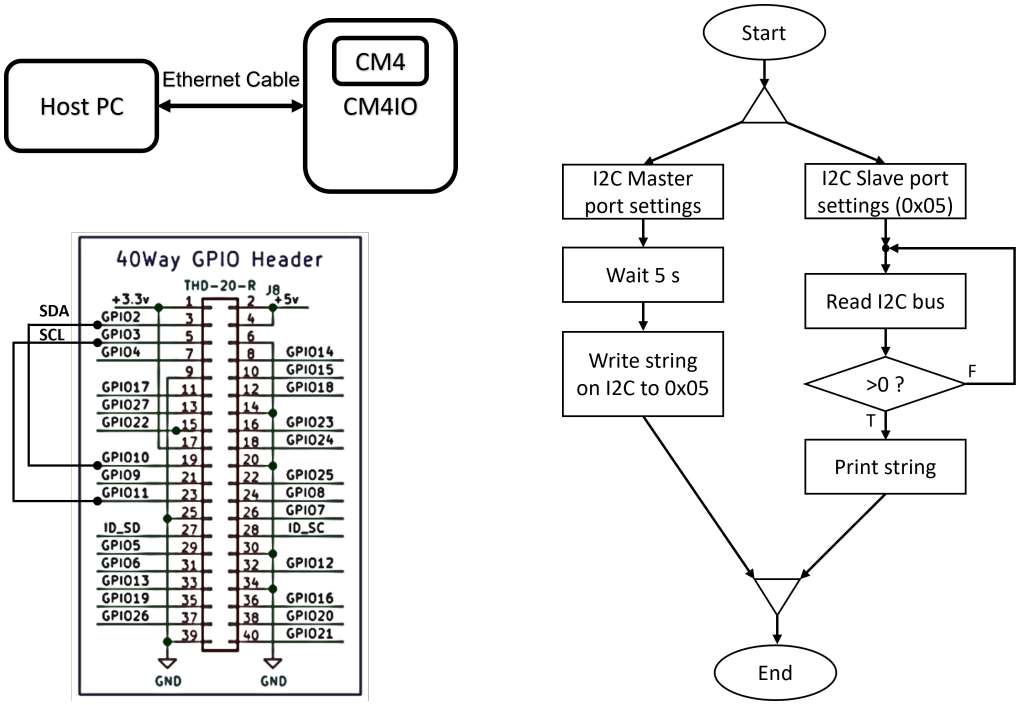
List of items

- Personal Computer with Linux system;
- Three female-to-female jumpers;
- CM4IO board with CM4 CPU installed;
- Flashed SD Card inserted in CM4IO board.

Test Procedure

1. Connect the I2C-1 Master Port to the BSCSL Port using two Female-to-Female jumpers (SDA, SCL);
2. Connect the CM4IO to the Host Computer;
3. Power up the CM4IO;
4. Check that in the Boot Configuration file of CM4, located in */boot/config.txt* the following lines are present:
   dtoverlay=i2c1
   This specifies to the Operative System to load the device driver and device tree for the I2C-1, and assign them to predefined pins of the CPU (which are available on the DevBoard)

5. Reboot the CM4 to make sure the device drivers are loaded correctly;
6. Launch the test application *test_i2c_master_slave*, which will connect to the bus an I2C Slave device witch address 0x05, while an I2C Master device will send a string of variable bytes to the 0x05 address.



(a) CM4IO GPIO Header **(b)** Test flowchart

Figure 3.3: I2C interfaces testing

Test software structure The software is composed by two parallel threads that are started at the same time: Master thread and Slave thread.

The **Master thread**:

- Open the I2C1 port with a bitrate of 100 kbit/s;
- Wait 5 seconds, the time needed for the Slave to set-up;
- Send on the I2C bus a string to the address 0x05.

The **Slave thread**:

- Open the BSCSL port with a bitrate of 100 kbit/s and set its address to 0x05;

- The BSCSL control registers are read to know when data is ready in the RX FIFO buffer;
- If a non-zero length string is found, it is printed on screen.

Results Different strings of various lengths were tested with a limit of 70 bytes, and the peripherals worked as expected with no communication losses. The 70 bytes limit was imposed since no EP System known would require higher data transfers during the test operations.
All the code developed and tested for this unit has been wrote into two files: source code inside **i2c.c** and header file inside **i2c.h**.

3.1.2.4 SPI

The Compute Module 4 offers up to 6 different SPI (Serial Peripheral Interface) ports, but from specification only two were needed: one to communicate with the ADCs and TrimDAC of the EPIS boards, and one to communicate with EP System. The selected SPI ports are **SPI0** for communication with EP System and **SPI1** for the other purposes. Both ports were tested using a logic board equipped with two ADCs used to log temperatures and RF signals. The test consisted in sending a specific command to each ADC to gather the value read from each of the 16 channels of the chip. If the communication worked fine a series of raw values on 16-bits are printed on screen, otherwise a all ones value is printed.

Test Success Criteria The two SPI lines work correctly. This means:

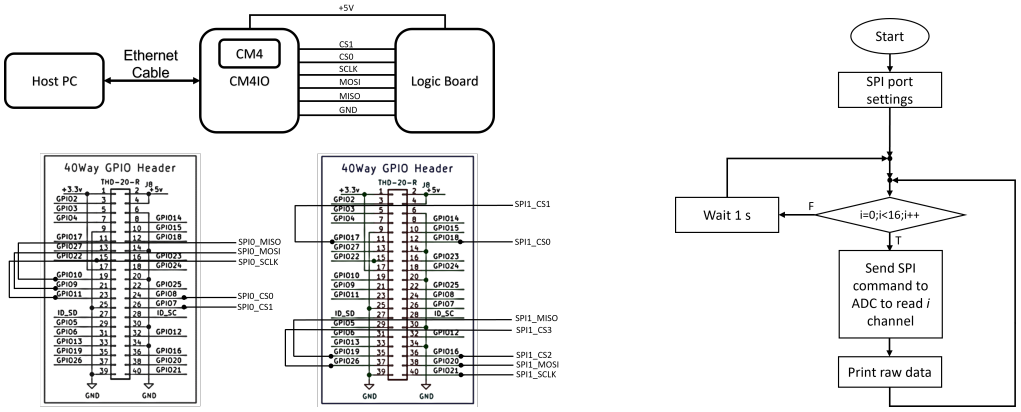
- The physical interconnection is established;
- the correct communication configuration is set-up;
- commands to the ADCs are correctly sent and executed;
- responses from the ADCs are correctly received and interpreted.

List of items

- Personal Computer with Linux system;
- External logic board equipped with ADCs
- Seven female-to-male jumpers;
- One NTC thermistor;
- CM4IO board with CM4 CPU installed;
- Flashed SD Card inserted in CM4IO board.

Test Procedure

1. Connect the SPI_N (where N specifies the port number) signals of the CM4IO to the external logic board using 7 Female-to-Male jumpers (+5V, SCLK, MISO, MOSI, GND, CS0, CS1);
2. Connect the CM4IO to the Host Computer;
3. Power up the CM4IO;
4. Check that in the Boot Configuration file of CM4, located in */boot/config.txt* the following lines are present:
   dtoverlay=spi0-2cs
   dtoverlay=spi1-4cs
   This specifies to the Operative System to load the device driver and device tree for the SPI0 and SPI1, and assign them to predefined pins of the CPU (which are available on the DevBoard);
5. Reboot the CM4 to make sure the device drivers are loaded correctly;
6. Launch the test application *test_spi_<N>*;
7. Assert the correct communication with the ADCs by looking at the non-ones values printed on screen.



(a) CM4IO GPIO Header **(b)** Test flowchart

Figure 3.4: SPI interfaces testing

Test software structure The test consists of sending commands over SPI to the two ADCs installed on the logic board, and printing on screen the raw data acquired.

- The SPI port is configured and opened to work with:

  - CPOL (Clock Polarity) = 0;
  - CPHA (Clock Phase) = 0;
  - Bitrate 100 kbit/s;
  - Data frames of 8 bits;
- a loop between i=0 and i=15 starts;
- at each iteration the command to read a specific channel of one ADC is sent on the SPI bus;
- the ADC responds with 32 bits that include some metadata [4] plus 24 bits of the acquisition which is truncated to 16-bits and printed on screen;
- when the loop ends the software is put in sleep for 1 second, then it restarts to acquire.

Results SPI interfacing with two ADC LTC2449 correctly works. This is demonstrated by the fact that values different from all ones were printed. In fact since the SPI data lines are HIGH during idle, if the ADCs were not responding the read data would be all ones, or $2^{16} - 1 = 65535$.

To further confirm the correct behaviour, a NTC thermistor was connected on some channels of the ADCs and a different value was printed in different temperature conditions: when put close to a heat source the raw data increased, while when left close to a cooler metal the value decreased.
All the code developed and tested for this unit has been wrote into two files: source code inside **spi.c** and header file inside **spi.h**.

3.1.2.5 GPIOs

In order to enable the DC-DC Buck/Boost converter, two GPIOs are required. This test aims to verify that both signals can be driven by the on-board software. To do so, the internal *raspi-gpio* util is used: it allows to set the GPIO's function, pull-up/down resistors and logic state. An oscilloscope has been used to verify the correct behaviour of the pins.

Test Success Criteria

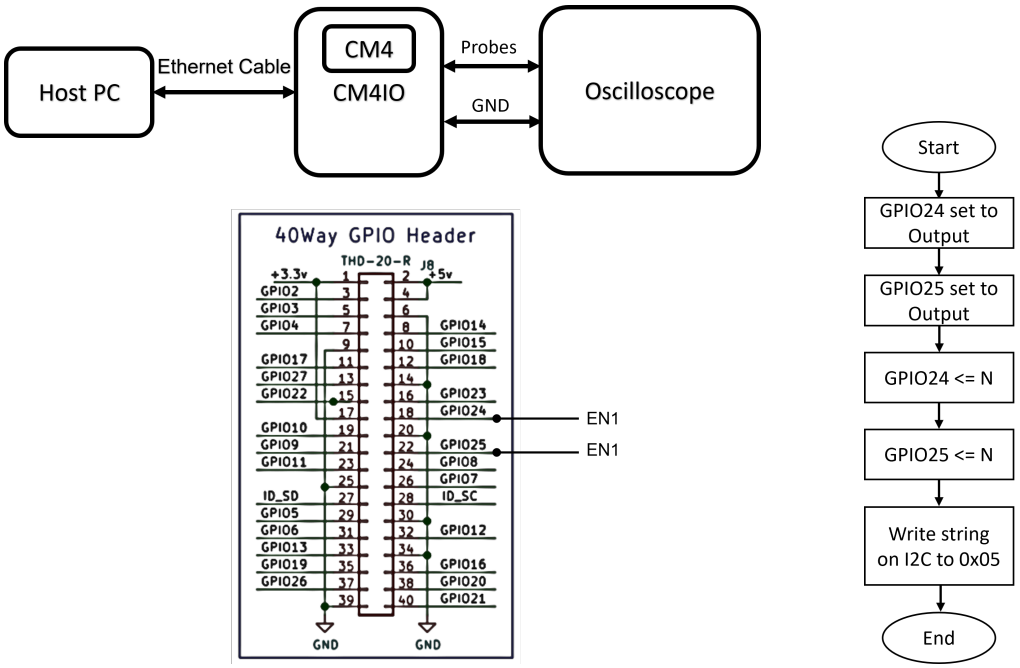
- GPIO signal can be software driven;
- the expected voltage is found on the physical pin.

List of Items

- Personal Computer with Linux system;
- Oscilloscope;
- CM4IO board with CM4 CPU installed;
- Flashed SD Card inserted in CM4IO board.

Test Procedure

1. Connect two probes to the oscilloscope input channels;
2. Connect the two probes to GPIO 24 and 25;
3. Connect the CM4IO to the Host Computer;
4. Power up the CM4IO;
5. Launch the test application *test_gpio <N>*, where N is the logical value to set the GPIOs: if 0, the voltage should be 0V, while if 1, the voltage should be 3.3V.



(a) CM4IO GPIO Header

(b) Test flowchart

Figure 3.5: GPIOs testing

Results The GPIOs behaved as expected, bringing the voltage up to 3.3V when set to the HIGH state, and switching to 0V when set to LOW. The two pins have been tested together since they will drive a 2-input AND port on the EPIS-Power (Figure 2.26). All the code developed and tested for this unit has been wrote into two files: source code inside **gpio.c** and header file inside **gpio.h**.

3.1.3 Data storage software units

In this section are described all the software units that shall interact with the mass storage memory. As explained in the previous chapter, information redundancy strategy has been adopted when possible to increase the reliability and safety of the stored data.

3.1.3.1 Set-up time

This software unit is in charge of loading the initial time of the mission. This value is stored inside a file called "time.txt". This method allows to recover the last mission time saved in case of a software reboot. Triple Modular Redundancy (TMR) has been applied to this unit, so the time word is actually saved in three different files. When the software starts these files are loaded and the saved time words compared: if at least two are equivalent the mission time restarts from that point, otherwise the three files are cleared and the mission time restarts from zero.

Test Success Criteria

- The unit correctly choose the most reliable time word.

List of Items

- Personal Computer with Linux system;
- CM4IO board with CM4 CPU installed;
- Flashed SD Card inserted in CM4IO board.

Test Procedure

1. Connect the CM4IO to the Host Computer;
2. Power up the CM4IO;
3. Create a folder "time" typing "mkdir time";
4. Create three time files, two containing the same integer number;
5. type "./time_test" to launch the test application.

Results After the test all three files contain the same time word, indicating that the software unit worked as expected. A further test could be tried where each file has a different time word. The expected result would be that after the test all three files contain "0" as mission time. Overall the unit takes more time if compared to other software components simply because it needs to compare three files together. All the code developed and tested for this unit has been wrote into two files: source code inside **time.c** and header file inside **time.h**.

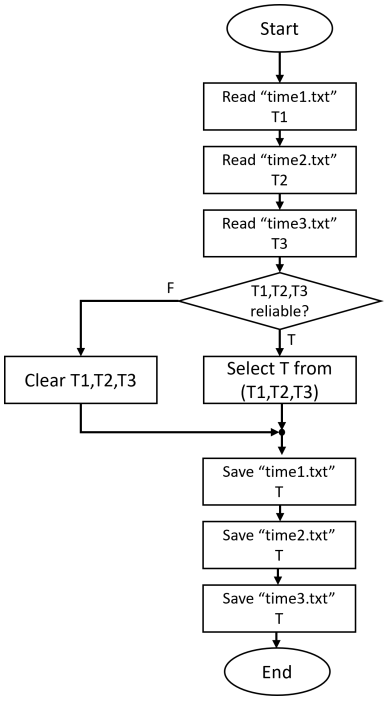


Figure 3.6: Set-up time test procedure

Min[ms]	Mean[ms]	Max[ms]
2.082	7.865	169.290

Table 3.2: Set-up time execution time

3.1.3.2 Save data in memory

All the communication packets to be sent to CDH are also stored in a local file to have a further level of reliability in case of failures. This file is called "log.txt" and at each iteration a new packet is appended to it. It is worth assessing two aspects of this software unit:

- the size of this file, which will constantly increase over time;
- the execution time, since this unit will be periodically executed every second.

The comparison between Time files and Log file is also interesting, because the first are constant in dimensions, while the second increases. The overall memory demand is represented in Table 3.3.

Thus the expected memory usage after 1 hour of operations is approximately 576 KB. These values are compatible considering the available storage memory of 8 GB.

File Name	Size after 1 s	Size after 1 m	Size after 1 h
time1.txt	4	4	4
time2.txt	4	4	4
time3.txt	4	4	4
log.txt	160	9600	576000
Total	172	9612	576012

Table 3.3: File dimensions during time in bytes

Test Success Criteria

- The item correctly appends a packet to the log file.

List of Items

- Personal Computer with Linux system;
- CM4IO board with CM4 CPU installed;
- Flashed SD Card inserted in CM4IO board.

Test Procedure

1. Connect the CM4IO to the Host Computer;
2. Power up the CM4IO;
3. type "./log_test" to launch the test application.

Results The unit was let run for one hour, in which all the expected packets were saved in the file. In order to verify the correct content of the file, a decode application was developed which converts the log to a Comma Separated Values (CSV) file, from which the analysis can be done. All the code developed and tested for this unit has been wrote into two files: source code inside **log.c** and header file inside **log.h**.

Min[ms]	Mean[ms]	Max[ms]
0.071	0.083	0.822

Table 3.4: Save data in memory execution time

3.1.4 Communication software units

Here all the units regarding the communication between the EPIS-DL and CDH will be covered. Since this part of the software was already tested during the UART interfaces tests only a brief comments on the results will be presented.

3.1.4.1 CRC32

The Cyclic Redundant Code implemented is not used for error correction, but only for data integrity checks, and is based on the following generator polynomial: $g(x) = x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$. This unit will be executed every time a new CDH packet is received, to check its integrity, and every time a packet is generated, appending the resulted CRC32 at the end of the packet. Finally, the generated code has been compared to the one computed by an online tool in order to validate the result.

Test Success Criteria

- The unit correctly generates the CRC32.

List of Items

- Personal Computer with Linux system;
- CM4IO board with CM4 CPU installed;
- Flashed SD Card inserted in CM4IO board;
- CRC32 calculator website.

Test Procedure

1. Connect the CM4IO to the Host Computer;
2. Power up the CM4IO;
3. Type "./crc32_test" to launch the test application;
4. Compare the computed CRC32 with its reference.

Results The unit successfully generated the Cyclic Redundant Code on 4 bytes every time. The execution time was good enough to apply this unit in multiple phases of the firmware execution.

Min[ms]	Mean[ms]	Max[ms]
0.036	0.040	0.142

Table 3.5: CRC32 execution time

3.1.4.2 Sending packets to CDH

This software unit was already tested during the UART interfaces test and for this reason only the timing results will be discussed.

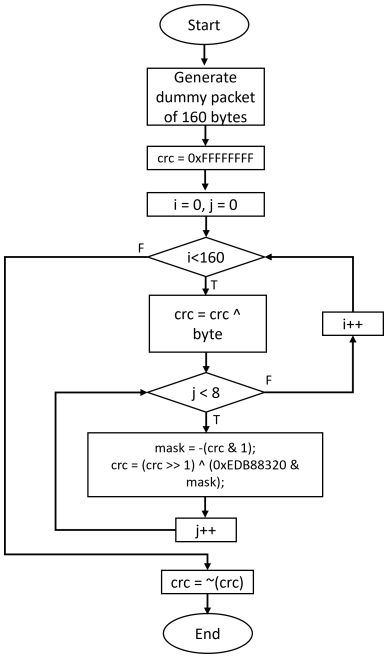


Figure 3.7: CRC32 test procedure

Results In summary everything this unit does is to copy some structs (Figure 2.21) inside a buffer and send it to the UART TX port, which could be seen as a file.

Min[ms]	Mean[ms]	Max[ms]
0.014	0.017	0.044

Table 3.6: Sending packets to CDH execution time

3.1.4.3 Receiving packets from CDH

This software unit was already tested during the UART interfaces test and for this reason only the timing results will be discussed.

Results The reception unit does multiple things: reads from the UART RX port up to 500 Bytes, performs a linear search inside this buffer to find the extremities of a packet (Figure 2.20) and finally copies the packet inside a well formatted struct. For this reason the execution time is significantly greater than the transmission unit.

Min[ms]	Mean[ms]	Max[ms]
0.097	0.100	0.228

Table 3.7: Receiving packets to CDH execution time

3.1.5 Data acquisition software unit

This unit deals with the acquisition from the different Analog to Digital converters of EPIS. The test set-up, as well as the items required were the same as described under SPI interfaces tests.

3.1.5.1 Data Acquisition, Validation and Processing

Test Success Criteria

- commands to the ADCs are correctly sent and executed;
- responses from the ADCs are correctly received, validated and processed.

Test Procedure

1. Connect the CM4IO to the Host Computer;
2. Power up the CM4IO;
3. Type "./adc_test" to launch the test application;
4. Assert the correct communication with the ADCs by looking at the non-zerovalues printed on screen.

Results As previously verified the SPI communication with ADCs works correctly. This unit would be one of the greatest in terms of computation time needed, since it needs to send a 32-bits command and read a 32-bits response for each of the 48 available channels.

Min[ms]	Mean[ms]	Max[ms]
57.699	58.486	69.433

Table 3.8: Data Acquisition, Validation and Processing execution time

3.1.6 Power management software unit

Almost every aspect of the power management is performed by analog hardware that does not require digital commands nor software interfaces. The only unit

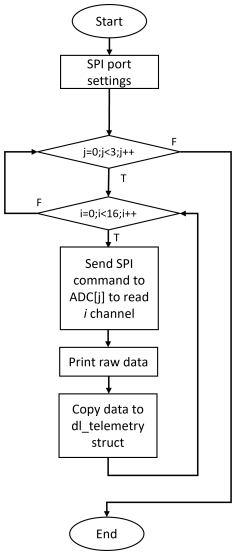


Figure 3.8: Data Acquisition, Validation and Processing test procedure

that requires control is the DC-DC Buck/Boost converter on EPIS-Power, which is enabled by two GPIOs and its output value set by changing a resistance value, achieved with a programmable digital potentiometer.

This test required the real EPIS-Power board to be available, since no board mock-ups were available.

3.1.6.1 Digital Potentiometer communication

This unit was developed in order to be as much user-friendly as possible, since it requires as input a floating point value in the range [12.0 - 28.0]. The conversion from output voltage to the potentiometer value is done at run-time. This impacts the overall performances, but allows an easier command structure. For this test the voltage value has been locked to 12.0V, and the converter enabled for five seconds.

Test Success Criteria

- The unit correctly sets the digital potentiometer resistance value;
- The unit correctly sets the output voltage.

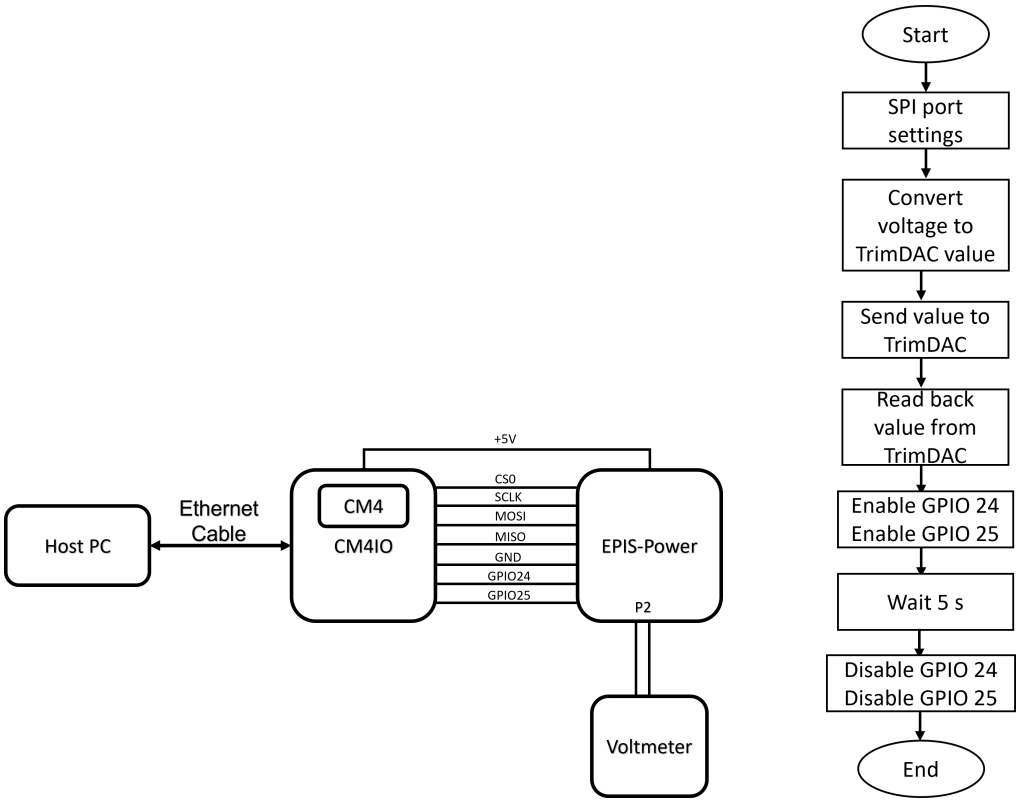
List of Items

- Personal Computer with Linux system;
- CM4IO board with CM4 CPU installed;
- Flashed SD Card inserted in CM4IO board;

- EPIS-Power board;
- Voltmeter.

Test Procedure

1. Connect the CM4IO to the Host Computer;
2. Power up the CM4IO;
3. Type "./dac_test" to launch the test application;
4. Check the output voltage on EPIS-Power with the voltmeter.



(a) Test set-up **(b)** Test flowchart

Figure 3.9: Digital Potentiometer communication testing

Results The unit always set the correct voltage value with an uncertainty of $\pm$ 0.1V. The conversion requires a lot of execution time due to its floating point operations, but it is important to note that is executed only when requested by the GSS operator, typically once for every test.

Min[ms]	Mean[ms]	Max[ms]
163.059	163.154	164.539

Table 3.9: Digital Potentiometer communication execution time

3.1.7 EP System software unit

3.1.7.1 EP System configuration file

This configuration file has been designed to reduce the CTP-related overhead when executing a test. The file must be written following this format:

<interface_type>	<bitrate>
<supply_voltage>	<num_of_cmds>

Table 3.10: prop_config.txt file format

where each parameter must be written following the same formats as explained in Table 2.10 and spaced using tabs. Every EP System will have its own configuration file.

Test Success Criteria

- The parameters are correctly loaded into the software.

List of Items

- Personal Computer with Linux system;
- CM4IO board with CM4 CPU installed;
- Flashed SD Card inserted in CM4IO board.

Test Procedure

1. Connect the CM4IO to the Host Computer;
2. Power up the CM4IO;
3. Type "nano prop_config.txt" to create or modify the EP System configuration file;
4. Type "./prop_config_test" to launch the test application;

Results The file has always been correctly decoded and its parameters loaded into the software.

Min[ms]	Mean[ms]	Max[ms]
9.033	10.749	21.649

Table 3.11: Configuration file loading execution time

3.1.7.2 Communication with EP System

Since the targeted EP System utilizes I2C Multi-Master as communication protocol, this software unit was already tested during the I2C interfaces test and for this reason only the timing results will be discussed.

Results All the strings tested were correctly sent and received by the slave thread. The execution time is compatible with the communication bitrate.

Min[ms]	Mean[ms]	Max[ms]
0.216	0.345	0.568

Table 3.12: Communication with EP System execution time

3.1.8 Time Scheduling Analysis

3.1.8.1 Loop Timing

This unit is in charge of computing how much time has elapsed from the start of the loop, and therefore put the master thread in the sleep state in order to reach 1 second. To verify its accuracy, the software is let run for one hour and the final execution time is compared with an external chronometer.

Test Success Criteria

- The unit correctly puts the thread into sleep state;
- The unit timing is verified with an external chronometer.

List of Items

- Personal Computer with Linux system;
- CM4IO board with CM4 CPU installed;
- Flashed SD Card inserted in CM4IO board;
- Chronometer application on Personal Computer.

Test Procedure

1. Connect the CM4IO to the Host Computer;
2. Power up the CM4IO;
3. Type "./loop_test" to launch the test application;

Results The unit successfully executed the loop with a period of 1 second. Considering the test execution of 1 hour, the unit was behind of around 5 seconds which is acceptable considering the simple implementation and the scope of the test.

Min[ms]	Mean[ms]	Max[ms]
0.083	0.099	0.156

Table 3.13: Loop execution time

3.1.8.2 Worst case scenario

Considering the worst case scenario in terms of execution time, the overall time required to execute all the software units is about **404 ms**, which is fairly under the 1 second limit and means that for more than the 50% of the time the CPU is doing nothing.

Since the communication with the EP System is handled by a parallel thread, its execution time was not included in the abovementioned analysis, even if it would not majorly impact the total execution time.

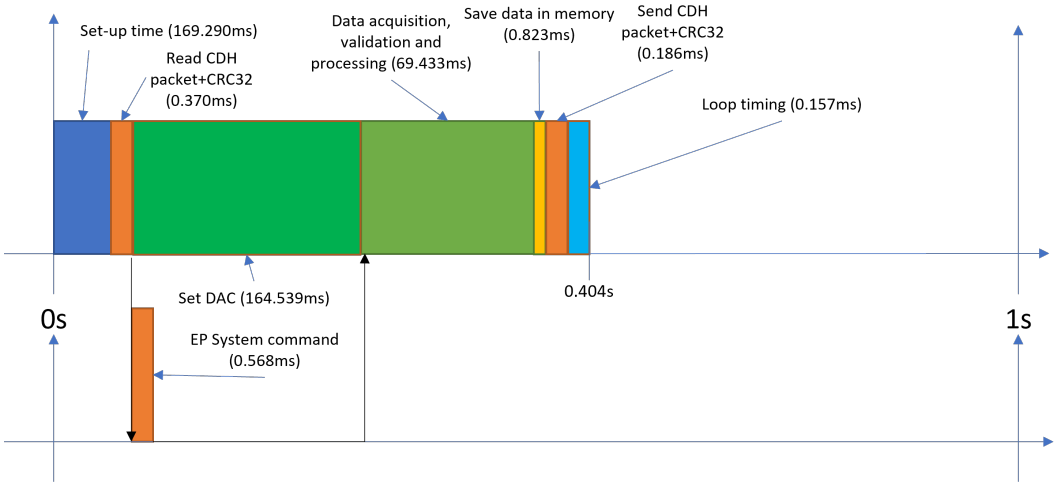


Figure 3.10: Worst case scenario execution time

3.2 CTP Integration and Testing

Once all the software units have been developed and tested, they have been integrated to form the complete EPIS firmware which would also need to be tested. As it is possible to notice, the integration and testing process followed a **bottom-up approach**, where at first the single units have been tested alone (with some drivers or mock-ups if needed), then integrated and tested together.
The intended Assembly, Integration and Verification (AIV) for the CTP is represented in Figure 3.11. The tests that will be covered in this sections are the following:

- **CTP-EQM-EPIS01 - EPIS Software, Electrical Integration and Test:** the two EPIS boards are electrically integrated and tested standalone in order to verify the functionalities of the system under ambient conditions;
- **CTP-EQM-FFT01 - Avionics Electrical Integration and Test:** the Basic Avionics and EPIS system are electrically integrated to verify all that all CTP functionalities are verified under ambient conditions;
- **CTP-EQM-FFT02 - CTP Integration and Full Functional Test:** all the systems except the EP System are mechanically and electrically integrated in the CTP structure, and their functionalities are verified under ambient conditions.

3.2.1 EPIS Software Integration

All the different software units were now ready to be stacked together to compose the final EPIS firmware. Figure 3.12 shows the source code files structure. Following the dynamic software model designed in the previous chapter the remaining tasks are to write down the C main function in **main.c** and integrate all the different software units to work together.

It is worth to note that at this point an **EP System was selected** (more information about this system in the next section) to perform the commissioning test of the platform. This meant that the firmware integration focused in improving and optimizing the software interface between EPIS and this specific EP System, which as communication interface used **I2C Multi-Master**.

The process of software integration resulted into composing the main function with:

- interface peripherals set-up of UART, SPI and I2C Multi-Master of section 3.1.2;

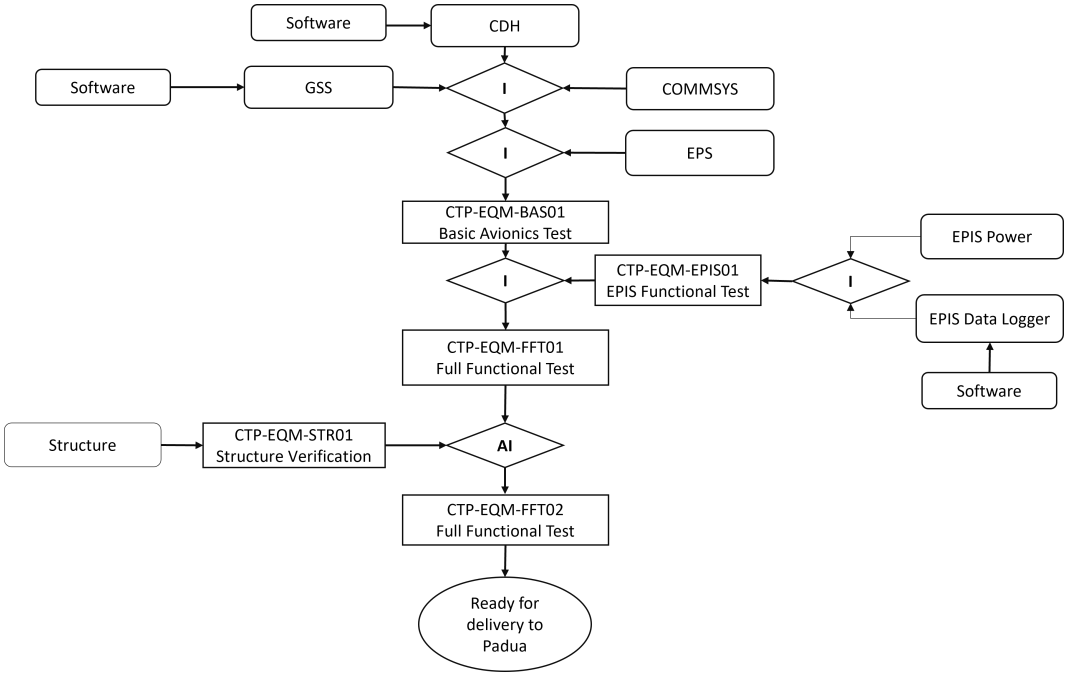


Figure 3.11: AIV plan

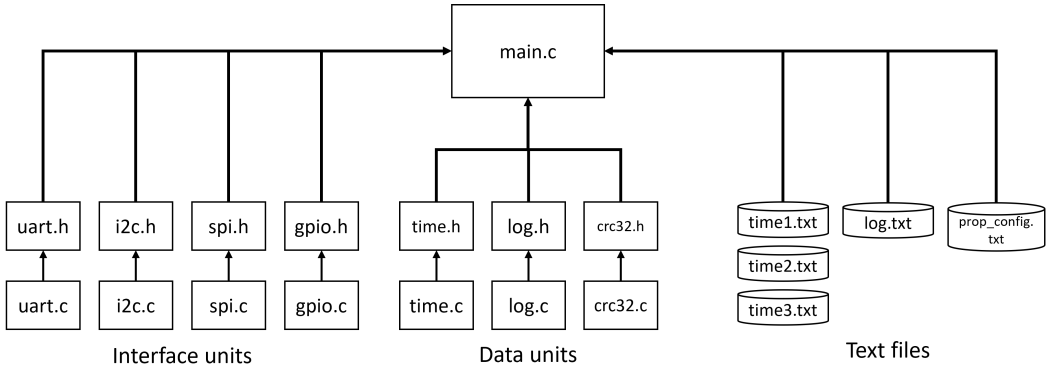


Figure 3.12: EPIS-DL Source code structure

- GPIO enable signals set-up for the DC-DC Buck/Boost converter of section 3.1.2.5;
- EP System configuration file loading of section 3.1.7.1;
- initial value set-up for the DC-DC Buck/Boost converter of section 3.1.6;
- time acquisition function for loop timing of section 3.1.8.1;
- reception of packets from CDH of section 3.1.4.3;
- communication with EP System of section 3.1.7.2;

- acquisition data unit of section 3.1.5;
- data storage of section 3.1.3.
- transmission of packets to CDH of section 3.1.4.2.

In order to enable or disable some of these software units simple boolean flags have been inserted in the source code in order to have one single firmware structure, which can be configured for different tests with minimum efforts: these flags are listed in Table 3.14. The source code needs to be eventually re-compiled and the resulting binary file can be renamed to clearly identify what test is going to be performed.

Flag name	**Value**	**Description**
ENABLE_CDH_COM	0	the firmware does not read-/write on UART2
	1	the firmware attempts to read and write data on UART2
DEV_BOARD	0	the firmware attempts to send commands and read response on SPI0 to TrimDAC
	1	the firmware does not send commands on SPI0 to TrimDAC
DEV_ADC	0	the firmware attempts to send commands and read response on SPI0 to ADCs
	1	the firmware does not send commands on SPI0 to ADCs
DEV_PROP	0	the firmware attempts to send commands and read response on I2C Multi-Master from EP System
	1	the firmware does not send EP System commands on I2C Multi-Master

Table 3.14: EPIS-DL firmware flags

3.2.2 EPIS Electrical Integration and Test

Here there are described the steps that led to the partial functional test of the EPIS system, which include the electrical and mechanical integration of the two EPIS boards, as well as the connection with the external sensors and GSE. The firmware flags for this test were set as explained:

- ENABLE_CDH_COM = 0;
- DEV_BOARD = 0;
- DEV_ADC = 0;
- DEV_PROP = 1;

so the communication software unit with CDH and with EP System are disabled.

Test Success Criteria

- EPIS-Power ability to power the EPIS-Data Logger;
- EPIS-Power ability to regulate voltage towards PS.
- EPIS-DataLogger ability to measure on board sensors;
- EPIS-DataLogger ability to measure external sensors;
- EPIS-DataLogger ability to save data packets.

The measure of the data means to have at least the 99% of the data gathered and stored or transmitted. The output voltage towards the PS shall be compliant with the set value with a tolerance of 1%. In the memory, the 99% of the expected data shall be stored.

List of Items

- Personal Computer with Linux system;
- EPIS-Power board;
- EPIS-DataLogger board with CM4 CPU installed;
- Flashed SD Card inserted in EPIS-DataLogger board;
- Thermistors (NTCs);
- RF sensing boards;
- LISN board;
- Multimeter;
- Power bench supply;

These elements shall be connected as shown in the block scheme in Figure 3.13.

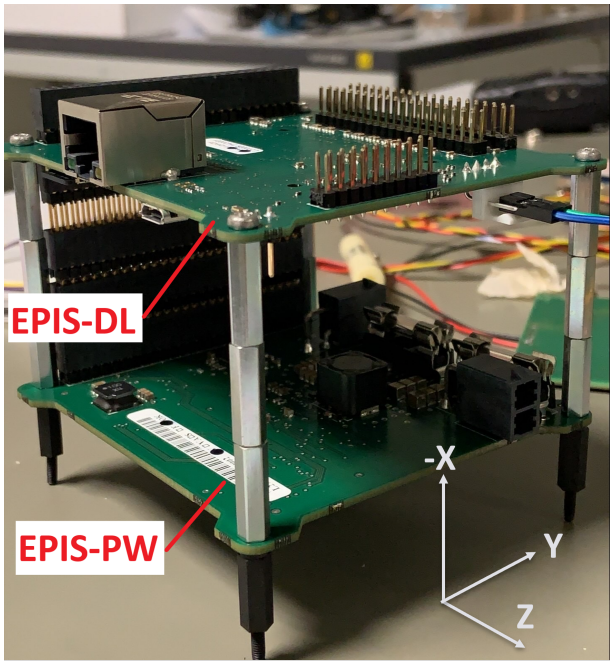


Figure 3.13: EPIS boards integrated (Thermistors, LISN and RF boards missing)

Test Procedure After the integration of the LISN board, RF boards, and temperature sensors with EPIS-DL, the test started with the power up of EPIS-Power with an external power supply. Almost every EPIS-Power functionality was now validated since also EPIS-DL powered up. Then, the EPIS firmware was let run: data acquisition from the two EPIS-DL ADCs was confirmed, as well as the Data Storage unit, by the analysis of the stored telemetry packets. EPIS-DL set the output voltage towards the EP System, measured on the P2 connector of the EPIS-Power and acquired correctly data from EPIS-Power ADC. The complete test procedure was composed of around 50 different steps, each one with its pass/fail criteria and expected results. Here a reduced version is proposed:

1. Set bench power supply to 16V and maximum 10A;
2. Connect power supply to EPIS-Power P3 connector;
3. Connect LISN board to EPIS-Power P2 connector;
4. Connect NTCs array to EPIS-DL P4 connector;
5. Connect RF boards to EPIS-DL P3 connector;
6. Connect LISN board input pins to EPIS-DL P3 connector;
7. Connect the multimeter terminals to LISN board output pins
8. Connect EPIS-DL to Host PC as described in Interface set-up section;
9. Enable the power supply output and wait 30 s;

10. Type "sudo ./CTP-EQM-EPIS-01" to execute the firmware;
11. Verify the correct data collection from voltage and current on board sensors;
12. Verify the correct data collection from temperature sensors;
13. Verify the correct data collection from RF boards;
14. Verify the correct data collection from LISN board;
15. Verify the correct EP System supply voltage of 12.0V measured by the multimeter;
16. Use CTRL+C to exit the software.

Results All the expected results were satisfied and the functional requirement have been verified. In particular:

- EPIS-Power correctly powered up EPIS-DataLogger and its components;
- EPIS-Power correctly provided a voltage of 12.0V on its P2 connector;
- EPIS-DataLogger correctly measured on board voltage and current sensors;
- EPIS-DataLogger correctly measured external temperature sensors;
- EPIS-DataLogger correctly saved data packets in "log.txt";
- EPIS-DataLogger correctly saved time words in "time.txt" files.

However, it was not possible to test correctly the RF boards due to lack of testing equipments able to generate such high frequency signals.

3.2.3 Avionics Electrical Integration and Test

After the functionalities of the EPIS system were verified, a further test was conducted before the final mechanical integration into the structure to verify that EPIS would behave as expected also when connected to the Basic Avionics of the CTP. For this reason the firmware flags for this test were set as explained:

- ENABLE_CDH_COM = 1;
- DEV_BOARD = 0;
- DEV_ADC = 0;
- DEV_PROP = 1;

meaning that the only software unit that was disabled was the communication with the EP System. This test was conducted not only to verify the correct behavior between Basic Avionics and EPIS but also to verify:

- the two communication line from CTP to GSS: HL and RF link;
- Basic Avionics storage capability;

- Basic Avionics data and telemetry acquisition;
- Basic Avionics and EPIS operative modes management and command execution;
- batteries discharging and recharging.

Test Success Criteria

- Data collection, packet preparation and transmission from EPIS-DL, to CDH and finally to GSS;
- Commands reception, validation, interpretation and execution from GSS, to CDH and finally EPIS-DL;
- Correct power distribution from EPS to EPS-Power;
- Correct batteries discharge and charge rates;
- Correct transitions between operative modes.

List of Items

- Personal Computer with Linux system;
- EPIS-Power board;
- EPIS-DataLogger board with CM4 CPU installed;
- Flashed SD Card inserted in EPIS-DataLogger board;
- Thermistors (NTCs);
- RF sensing boards;
- LISN board;
- Multimeter;
- Power bench supply;
- CDH board;
- COMSYS board;
- EPS board;
- FLATSAT board;
- Formatted SD Card inserted in CDH board;
- Two batteries;

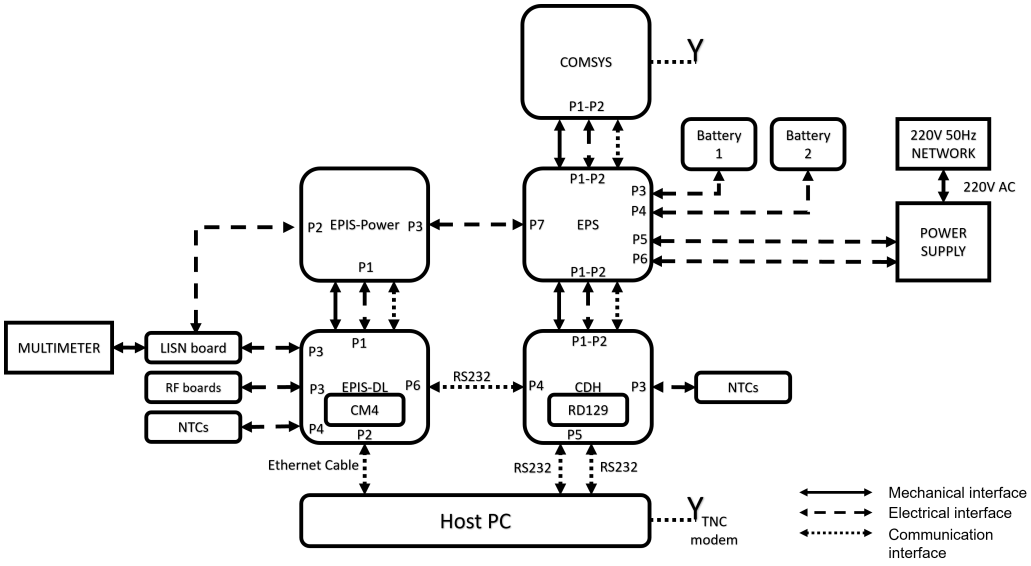


Figure 3.14: Avionics Electrical Integration and Test block scheme

Test Procedure After the setup, the test started with the CTP avionics moving from Dormant mode to Basic mode: both CDH and EPIS-DL firmwares were let run and their execution confirmed on the Host PC dataflows. After the acquisition of consistent telemetry, the commands exchange and execution were verified observing that the complete communication chain worked perfectly. Then the test continued to acquire telemetry, exchange commands and changing the operative modes for at least 4 hours in order to validate CTP's long endurance. The test finally ended with the complete battery recharging. The complete test procedure was composed of around 230 different steps, each one with its pass/fail criteria and expected results. Here a reduced version that covers only the software testing is proposed, where the starting condition is that all mechanical and electrical interconnections are established.

1. Connect the multimeter terminals to LISN board output pins;
2. Launch the EPIS-DL firmware;
3. Launch the CDH firmware;
4. Launch the GSS software;
5. Verify that complete telemetry packets are sent via HL connection of CTP;
6. Verify that complete telemetry packets are sent via RF connection of CTP;
7. Send C06 command - Change Time "C0600000000" to reset the MET;
8. Verify the command execution on GSS GUI;
9. Send C07 command - Change Log "C07log_ctp_eqm_fft_01.txt";

10. Verify the command execution on GSS GUI;
11. Send D03 command - Change Propulsor Supply Voltage "D0312.0" to set 12.0V;
12. Verify the command execution on GSS GUI;
13. Send C02 command - Change Opmode to PS On;
14. Verify the command execution on GSS GUI and by analyzing the measured voltage by the multimeter which should be 12.0V;
15. Send C03 command - Change Opmode to Burst;
16. Verify the command execution on GSS GUI;
17. Wait for 3.5 hours;
18. Send C01 command - Change Opmode to Basic;
19. Verify the command execution on GSS GUI and by analyzing the measured voltage by the multimeter which should be 0V;
20. Set two channels of the bench power supply to 18V and 1.5A;
21. Enable the power supply outputs to start recharging the batteries;
22. Wait until the batteries are fully charged;
23. Disable the power supply outputs;
24. Wait for 10 hours;
25. Send C04 command - Poweroff;
26. Verify the command execution on GSS GUI, the two software stop execution.

Results All the functionalities were verified: telemetry and data were in the nominal ranges, 99% of the packets were stored on the test points (i.e. onboard memories and GSS memory), and commands were received and executed by the CTP that reacted as expected in each operative mode. The GSS correctly visualized on the GUI the telemetries and enabled the operators to send commands via HL. Moreover, the battery provided the power for the expected duration and are recharged within the expected time.

3.2.4 CTP Integration and Full Functional Test

Here the CubeSat Test Platform is finally fully integrated and a complete Functional Test can be performed. The last two functionalities that shall be tested are the EPIS-DL data communication with an EP Systems, and the EPIS-Power supply capabilities when a real load is applied to its output. For the first purpose a **EP System Communication Mock-up** for the T4I's REGULUS Propulsion System was used, which as previously cited communicates using I2C Multi-Master, while for the second purpose a resistive load of around 3 Ohm was connected to the LISN output (and so EPIS-Power); this module not only helped to verify the EPIS-Power

functionalities, but also to confirm the correct behavior of the temperature sensors, since due to Joule's first Law the resistive load would convert electric power to heat. For this reason the resistive load has been installed inside the Propulsor Module for this test. For this test all the software units have been enabled, so the firmware flags have been set as follows:

- ENABLE_CDH_COM = 1;
- DEV_BOARD = 0;
- DEV_ADC = 0;
- DEV_PROP = 0;

enabling all the software components.

3.2.4.1 Test Success Criteria

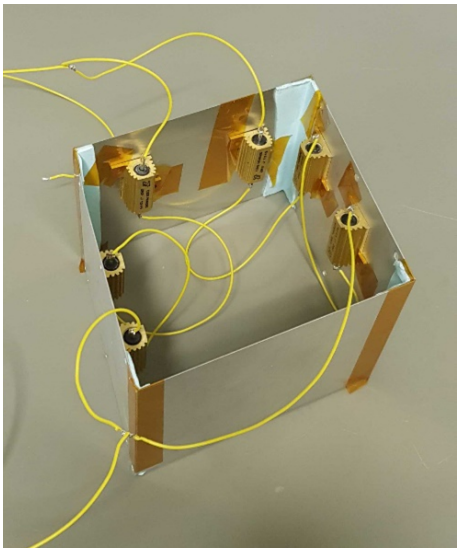
- Data collection, packet preparation and transmission from EPIS-DL, to CDH and finally to GSS;
- Commands reception, validation, interpretation and execution from GSS, to CDH and finally EPIS-DL;
- Correct commands transmission and telemetry reception from the REGULUS Communication Mock-up;
- Correct power distribution from EPS to EPS-Power;
- Correct batteries discharge and charge rates;
- Correct transitions between operative modes.

3.2.4.2 List of Items

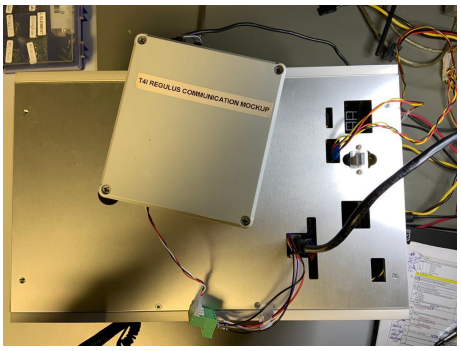
- CTP fully integrated;
- REGULUS Communication Mock-up;
- Resistive load installed inside the CTP;
- Power bench supply;
- Personal Computer with Linux system (GSS).

3.2.4.3 Test Procedure

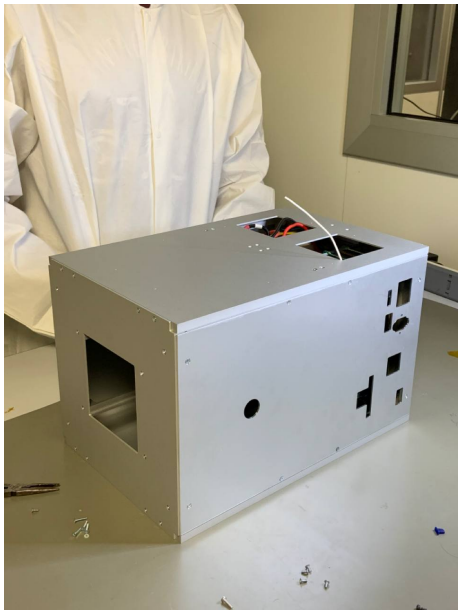
1. Connect the resistive load to the output of LISN board;
2. Connect the REGULUS Communication Mock-up I2C to the EPIS-DL P5 connector;
3. Set-up GSE and GSS;
4. Launch the EPIS-DL firmware;



(a) 3 Ohm Resistive Load



(b) REGULUS Communication Mock-up



(c) CTP fully integrated

Figure 3.15: CTP ready for FFT

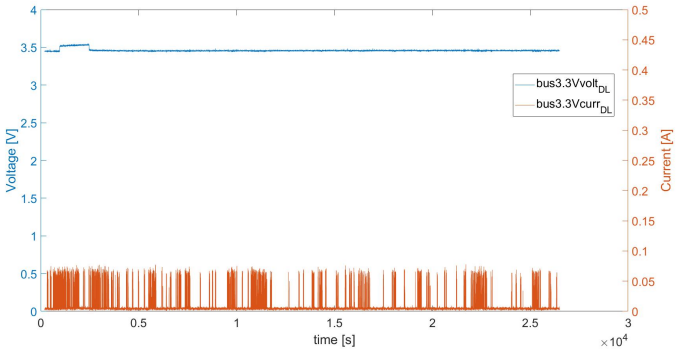
5. Launch the CDH firmware;
6. Launch the GSS software;
7. Verify that complete telemetry packets are sent via HL connection of CTP on GSS GUI;

8. Verify that complete telemetry packets are sent via RF connection of CTP on GSS GUI;
9. Send C06 command - Change Time "C0600000000" to reset the MET;
10. Verify the command execution on GSS GUI;
11. Send C07 command - Change Log "C07log_ctp_eqm_fft_02.txt";
12. Verify the command execution on GSS GUI;
13. Send C11 command - Disable RF link;
14. Verify the command execution on GSS GUI: COMSYS stops to send packets on RF;
15. Send C10 command - Enable RF link;
16. Verify the command execution on GSS GUI: COMSYS starts to send packets on RF;
17. Send D03 command - Change Propulsor Supply Voltage "D0312.0" to set 12.0V;
18. Verify the command execution on GSS GUI;
19. Send C02 command - Change Opmode to PS On;
20. Verify the command execution on GSS GUI by looking at the EP System supply bus current: due to Ohm's Law it should be $I = \frac{V}{R} = \frac{12V}{3Ohm} = 4A$ with a total power drain of around 48 W;
21. Send C03 command - Change Opmode to Burst;
22. Verify the command execution on GSS GUI;
23. Connect REGULUS Communication Mock-up to the power network;
24. Verify the REGULUS telemetry on GSS GUI: it should be of 70 bytes;
25. Send REGULUS "Set Power" command;
26. Verify the command execution on GSS GUI;
27. Send REGULUS "GOTO Auto" command;
28. Verify the command execution on GSS GUI;
29. Send REGULUS "Start Thrust" command;
30. Verify the command execution on GSS GUI;
31. Wait 3 hours;
32. Send REGULUS "Turn Off" command;
33. Verify the command execution on GSS GUI;
34. Send C01 command - Change Opmode to Basic;
35. Verify the command execution on GSS GUI: EP System supply bus voltage is 0V and current 0A;
36. Send C04 command - Poweroff;
37. Verify the command execution on GSS GUI, the two software stop execution;

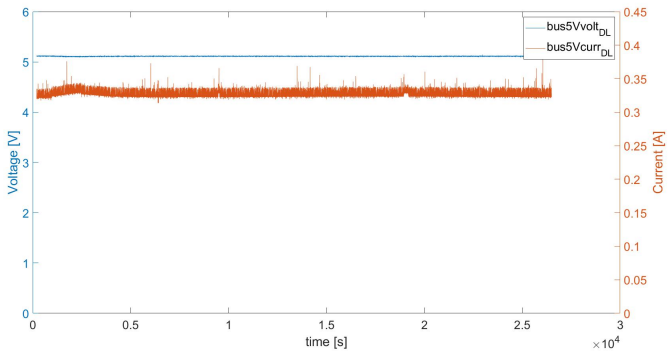
38. Set two channels of the bench power supply to 18V and 1.5A;
39. Enable the power supply outputs to start recharging the batteries;
40. Wait until the batteries are fully charged;
41. Disable the power supply outputs.

3.2.4.4 Results

During the test execution the GSS GUI software helped the test operators to keep under control the critical parameters of the CTP, but in order to have a detailed analysis a further post-processing of the log files has been performed. Here the focus is given only to the data acquired by EPIS system.



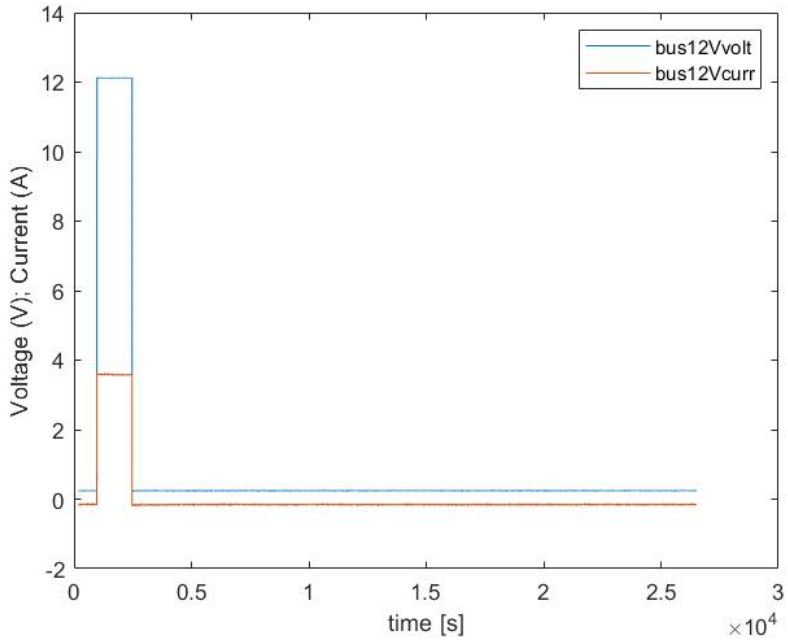
(a) EPIS 3.3V Bus



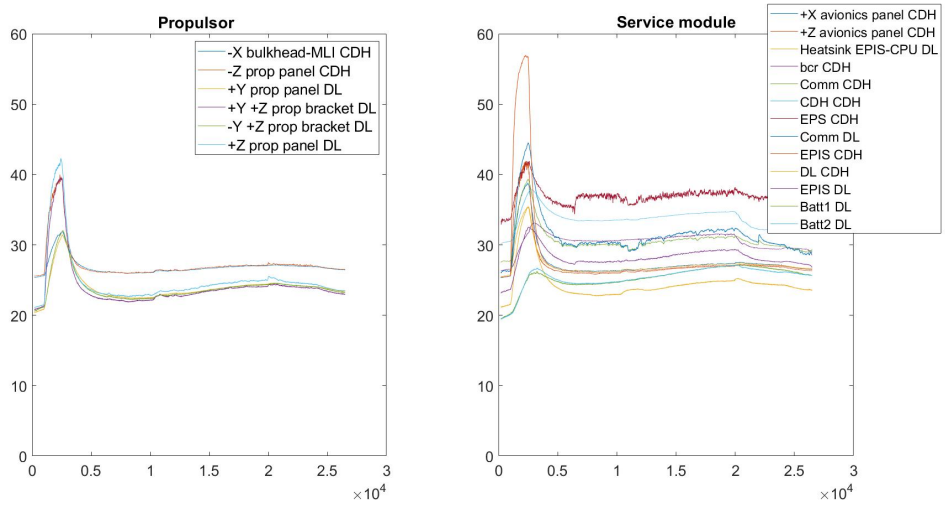
(b) EPIS 5V Bus

Figure 3.16: EPIS post-process log analysis (1)

The on board voltage regulators always kept the power buses at the desired voltages; it's interesting to see that the current flow on these buses changed as expected: on the 3.3V bus the maximum current absorption was around 75 mA which is the sum of all the ICs that are supplied by this line; on the 5V bus



(a) EPIS EP System Power Bus



(b) EPIS Temperatures

Figure 3.17: EPIS post-process log analysis (2)

the maximum current absorption was 350 mA, which is compatible with CM4's power requests. The EP System Power Bus was active only during the PS On and Burst phases: as expected the measured voltage was kept at 12V, with a current

absorption of around $I = \frac{V}{R} = \frac{12V}{3Ohm} = 4A$ for a total power of about 48W.

Moving to the temperature readings, the maximum temperature reached by the boards was about 35°C; a local increment of temperature can be observed on the EPIS-Power board in proximity of the diodes of the regulation circuit (about 55°C). This local increment in temperature is below the maximum rated temperature of the involved components, and the regulation circuit controller reached a maximum temperature of 32.5 °C largely inside the requirements margins.

Data communication with the REGULUS Communication Mock-up was always consistent meaning that all the manual commands issued from the GSS as well as the automatic "Keep-Alive" and "Get Housekeeping telemetry" commands were received by the EP System. This can be said since REGULUS after the reception of a command will respond with a confirmation packet.

Finally, the test can be considered a success since the CTP functions have been fully verified.

3.3 EP System Integration

As already introduced, the selected EP System for the commissioning of the All Electric CTP has been REGULUS Propulsion System [17], a Magnetically Enhanced Plasma Thruster (MEPT) developed by T4i S.p.A.. REGULUS has a size of 1.5U, or 93.8 x 95.0 x 150.0 mm including all its subsystems: the thruster, the Power Processing and Power Control Units, the low pressure fluidic section and a Iodine propellant tank which has not been used for these test campaigns. In this configuration REGULUS provides a total specific impulse up to 3000 Ns. Different total impulses can be provided under specific customer requests, varying the tank size and thus the overall volume of the propulsion unit. The integration of the two systems required to operate at three different levels:

- **Mechanical**: REGULUS was installed inside the Propulsor Module, mechanically latched to the four brackets;
- **Electrical:** the supply line was connected to the LISN output, which is itself connected to EPIS-Power;
- **Communication:** a shielded cable was made to allow the connection between EPIS-DL P5 connector and REGULUS I2C connector (CN1F).

Thrust	0.25 – 0.65 mN (highly modulable, 0.55 mN @ 50 W)
Specific Impulse	Up to 650 s (550 s @ 50 W)
Input power	20 - 60 W (50 W nominal)
Mass flow	0.15-0.20 mg/s with Xenon
Propellant	Iodine (I_2) or Xenon (Xe)
Volume	1.5 U (93.8 x 95.0 x 150.0 mm) referred to a Total Impulse of 3000 Ns
Weight	2.5 kg @ 3000 Ns
Electric Interface	12 V DC regulated
Communications	Can-bus or I2C with CSP protocol

Table 3.15: REGULUS EP System Performance

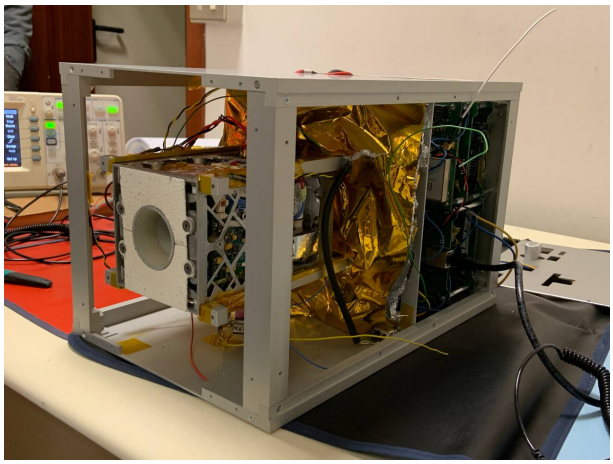


Figure 3.18: REGULUS integrated inside CTP Propulsion Module

Before proceeding with the environmental test, a Reduced Functional Test was carried in order to check that the integration did not interfered with the CTP functionalities. Indeed when the EP System was put in the heating state the communication between REGULUS and EPIS-DL were affected by some packet losses. Some ferrite beads have been installed around data and supply lines to minimize the Electromagnetic Interference (EMI) effects: this solution fixed the communication problem. Thermal straps were added between REGULUS case and the CTP brackets to improve the thermal dissipation, too.

Chapter 4

Environmental Test Campaign and Results

In Chapter 3 all the functionalities of the All Electric CubeSat Test Platform have been verified and validated but only at laboratory conditions. The last two test campaigns have been performed in **environmental conditions** in order to better simulate the real mission scenario of a CubeSat: spacecraft in orbit. For this reason these test campaigns were not carried out at PoliTo but instead at CISAS-UniPD laboratories in Padua, Italy, and finally at the Electric Propulsion Laboratory (EPL) of the European Space Research and Technology Centre (ESTEC) in Noordwijk, Holland. The test plan for these two campaigns is figured in Figure 4.1.

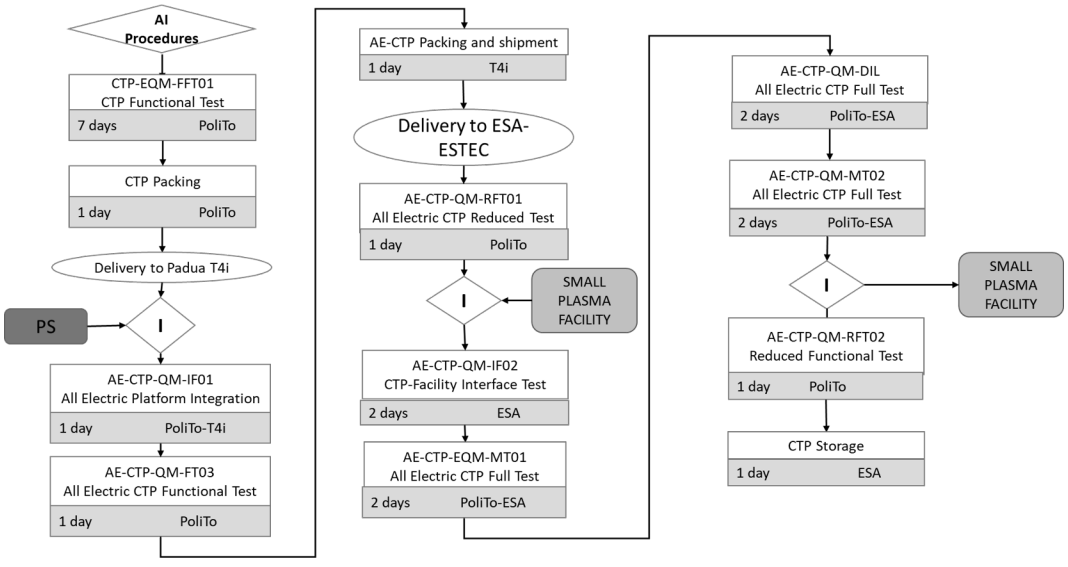


Figure 4.1: CTP commissioning test plan

4.1 Test campaign at CISAS-UniPD

Once REGULUS has been integrated in the CTP, the complete system had to be installed inside the CISAS's vacuum chamber which offered a list of feedthrough interfaces to connect the platform to the GSS equipment. Two interfaces have been used to electrically connect the CTP to GSS: one DB-9 completely dedicated to EPIS-DL, a second DB-9 that hosted:

- two RS232 serial lines;
- antenna wires;
- Load Switch connection.

No recharge cables were installed through the chamber for this test. The data of the propulsion system acquired by EPIS-DL, relayed to CDH and finally sent to ground is forwarded by the GSS to the Propulsion Operation Control Centre (POCC) through a serial line (9600bps, 8N1) automatically by the GSS software. Finally, the Remove Before Test (RBT) key was extracted from CTP.

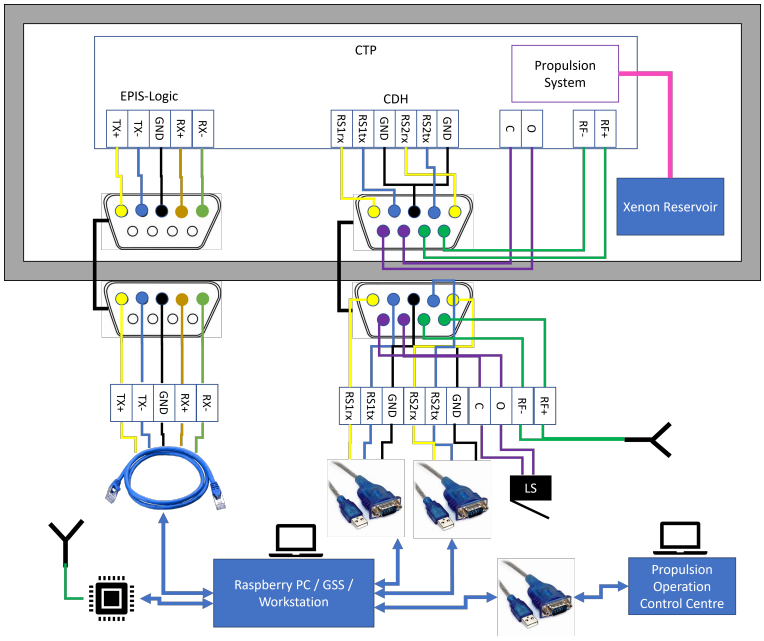


Figure 4.2: CTP interfaces with CISAS's Vacuum Chamber

4.1.1 Test Success Criteria

- Data collection, packet preparation and transmission from EPIS-DL, to CDH and finally to GSS;

- Commands reception, validation, interpretation and execution from GSS, to CDH and finally EPIS-DL;
- Correct power distribution from EPS to EPS-Power;
- Correct batteries discharge and charge rates;
- Correct transitions between operative modes.
- **Correct commands transmission and telemetry reception from REGULUS EP System**;
- **Telemetry reception from POCC**;
- **Correct CTP behavior in vacuum conditions**.

4.1.2 List of Items

- CTP fully integrated with REGULUS installed;
- Personal Computer with Linux system (GSS);
- Payload Operator Control Centre (POCC).

4.1.3 Test Procedure

The test was divided in different phases, and to have a better comprehension of the tasks executed only a brief description will be given.

1. **IF Set-up:** the CTP was mechanically installed inside the chamber as well as the electrical interconnections between GSS and CTP;
2. **Vacuum:** the chamber is closed and the vacuum operations started. To reach an acceptable level of vacuum ($P<10^{-5}$Pa) two hours are needed;
3. **Basic Ops and HL commands:** the same operations performed in the other FFTs are executed to test the correct functionalities of the CTP: communication links, operative mode transitions and data acquisition;
4. **PS activation**: REGULUS was now switched on and entirely supplied by the CTP. Before the thrust phase a controlled heating phase is required;
5. **Burst 20W:** a 15 minutes thrust phase is performed with a REGULUS internal power limit of 20 W;
6. **Burst 30W:** a 20 minutes thrust phase is performed with a REGULUS internal power limit of 30 W;
7. **PS Deactivation and Basic Mode:** REGULUS is switched off and the structure let cool down;
8. **Deactivation:** shutdown command is sent to the CTP and the software stops the execution.

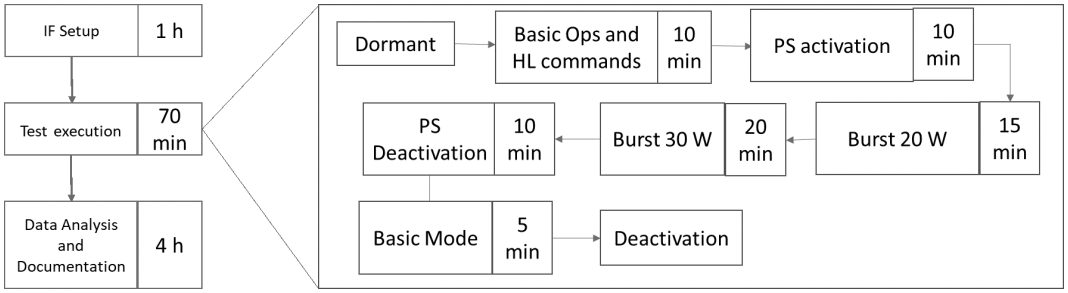


Figure 4.3: Full Functional Test flow activity

4.1.4 Results

The test lasted 4095s. Considering the system features and the test duration 4095 telemetry packets were expected to be saved, and 809 telemetry packets were expected to be received on the GSS. 4095 telemetry packets were actually saved, meaning that 100% of the expected telemetry was correctly saved; while 800 packets were actually received, meaning that 99% of the expected telemetry was correctly received.

Batteries behaviour adapted correctly to the different load conditions met during the test and there were not unexpected anomalies nor off-nominal temperatures (max 32°C).

The EP System Supply Bus was active only during the Propulsion On phase (from second 245 to 3472) to power the EP System: as expected it has been measured a voltage of 12.0 V with a current absorption variable and phase dependent.

Temperatures in the Propulsor Module were fairly low during the whole tests: the highest temperatures were about 45°C at the end of the second burst recorder on the four supporting brackets. Avionics components temperatures remained in the operative ranges during the whole test. The EPS board reached the highest temperature of about 50°C, while EPIS-Power remained always under 35°C.

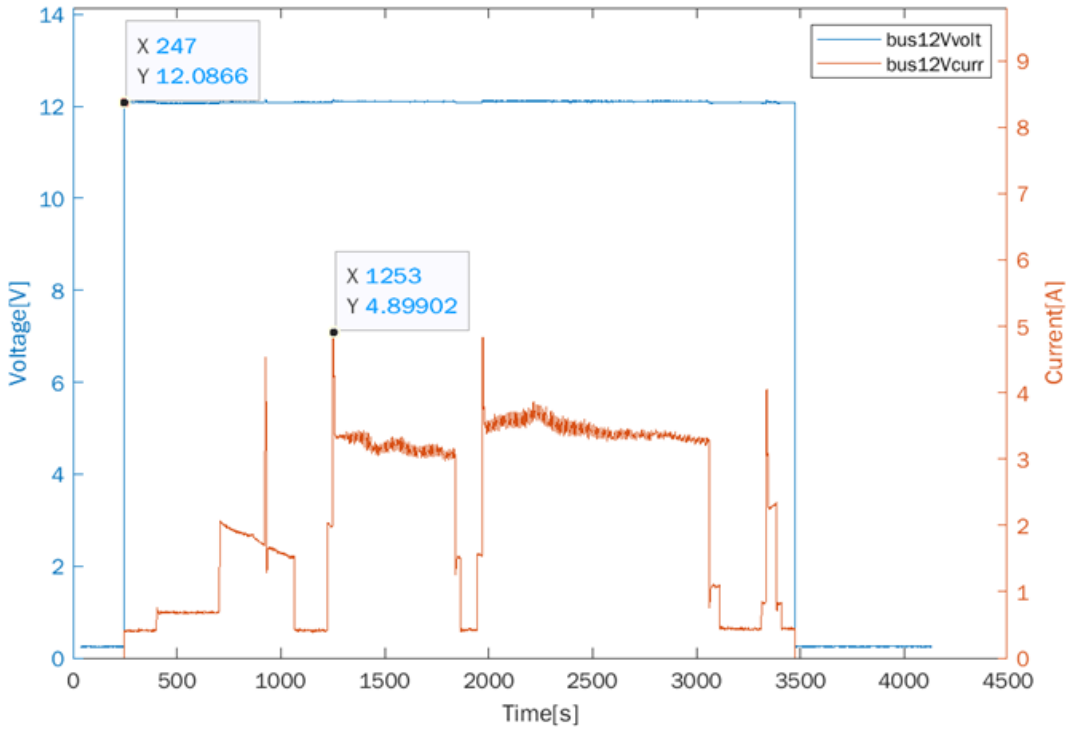
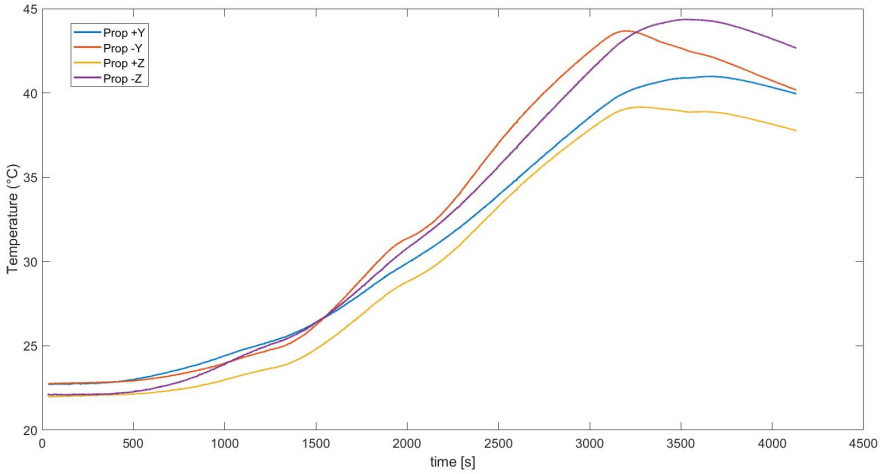
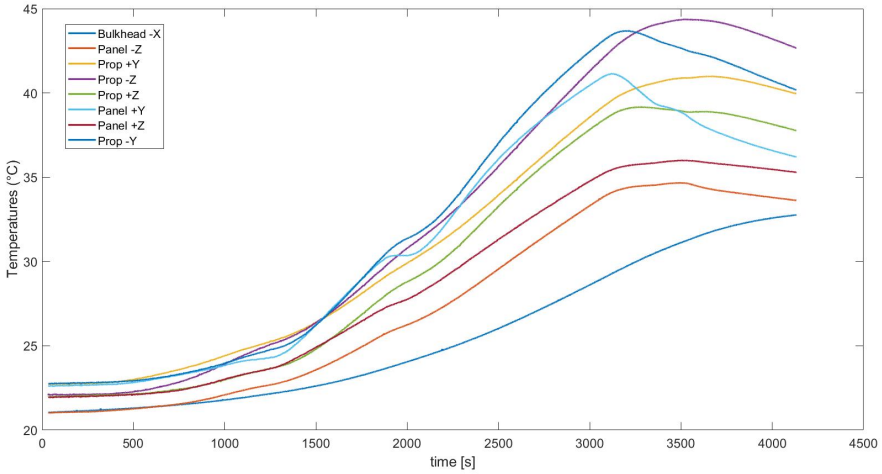


Figure 4.4: EP System Supply Bus

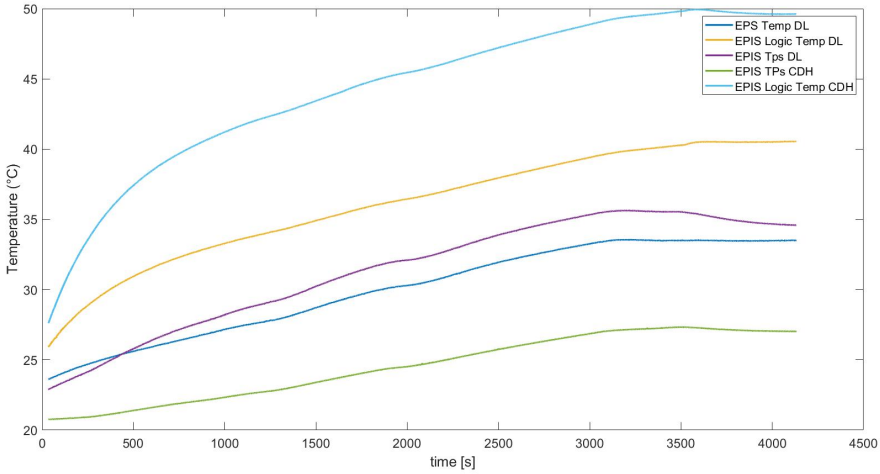


(a) Brackets

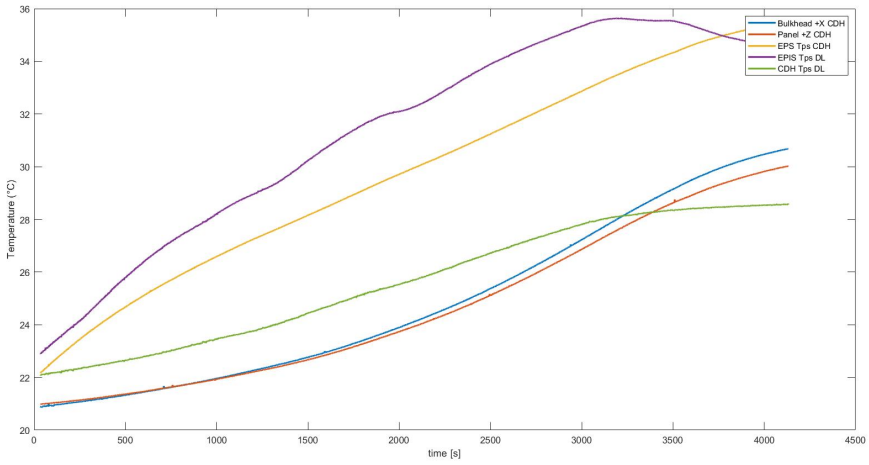


(b) Panels

Figure 4.5: EPIS-DL Propulsion Module temperatures



(a)



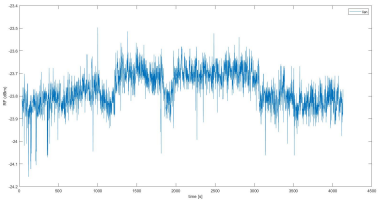
(b)

Figure 4.6: EPIS-DL Avionics Module temperatures

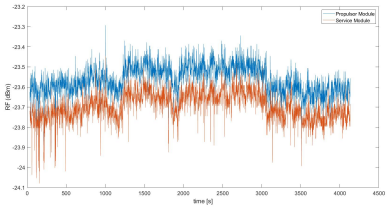
4.1.5 Anomalies

The LISN board circuit sensed some electromagnetic interference, but their magnitude never exceeded 0.5 dBm. From the Figures two thrust phases can be detected, and the greatest perturbations sensed by the RF sensors were in the frequency band 20-50 MHz, but always under 0.5 dBm. This results can be explained by three possible reasons:

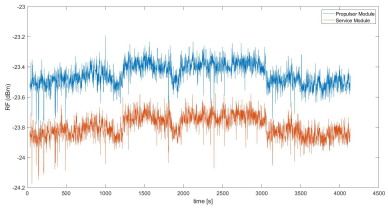
- the electromagnetic interference generated by the integrated system take place outside the recorded frequency bands (<1MHz or >500MHz);
- the acquisition system dedicated to RF boards did not work as expected, and some faults are present either in the software unit or at board level;
- conversion errors from the raw acquired data to the converted ones.



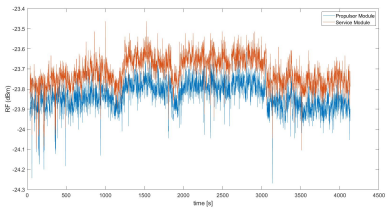
(a) LISN



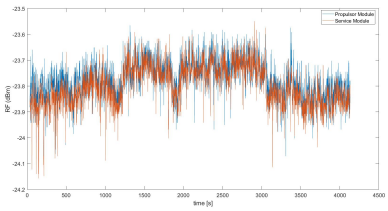
(b) 1-10MHz



(c) 20-50MHz



(d) 50-120MHz



(e) 400-500MHz

Figure 4.7: LISN and RF units

Communication with REGULUS was also affected by some problems. It was switched on at 233s and turned off at 3429s. In this interval, 3196 packets were

expected, while 3091 were received, with a yield of 96.7% overall. Most of the packets were lost during the first phase of the thrust (66.7%). The first thrust (Start Thrust # 1) at 20W was automatically interrupted by REGULUS due to an erroneous configuration parameter, which was then fixed with a the upload of a new parameter. The second thrust (Start Thrust # 2) attempt at 20W lasted almost 15 minutes, then the "turn_off" (Turn Off # 2) command was issued to stop REGULUS. The third thrust (Start Thrust # 3) this time at 30W lasted 20 minutes after which REGULUS automatically switched off due to an off-nominal temperature. A "soft_reset" (Soft Reset #3) command was issued. A fourth thrust (Start Thrust # 4) at 30W of about 1 minute was issued to check REGULUS's health after the off-nominal event, then the "turn_off" (Turn Off # 4) command was issued. The communication with REGULUS was stable the whole time after this fourth burst.

At the beginning of the first two burst phases, the communication from REGULUS to CTP was very unstable and most of the packets were lost. Communication became more stable only after few minutes the beginning of the bursts. This behaviour did not repeat in further bursts, where the communication was stable the whole time. This issue regards only the communication between CTP and REGULUS, as the CTP telemetry was always complete, and commands were correctly sent by the CTP, received by REGULUS and executed.

The complete telemetry from REGULUS was correctly stored in the log files, and it can be analyzed to have a better understanding of the telemetry losses. In Figure 4.8 is shown the first byte of the telemetry response from REGULUS:

- a value of 130 (0x82h) corresponds to the Housekeeping Telemetry;
- a value higher than 130 corresponds to a different command response;
- a value lower than 130 corresponds to **a partial or complete loss of the telemetry packet**.

4.1.6 Conclusion

There were some communication issues between REGULUS and CTP, however they were quite limited in time. CTP functions were all verified: communication between CTP and GSS was always stable and telemetry complete; CTP received and executed command properly; on board power management was correct; thermal environment was within expected ranges; all expected data were correctly measured, collected and stored; CTP correctly sent commands to REGULUS which were always received and executed. No significant differences between the CTP behaviour in lab condition and in vacuum were detected.

CTP was then uninstalled from the vacuum chamber, RBT key re-installed and the two batteries' connectors to the EPS board un-plugged.

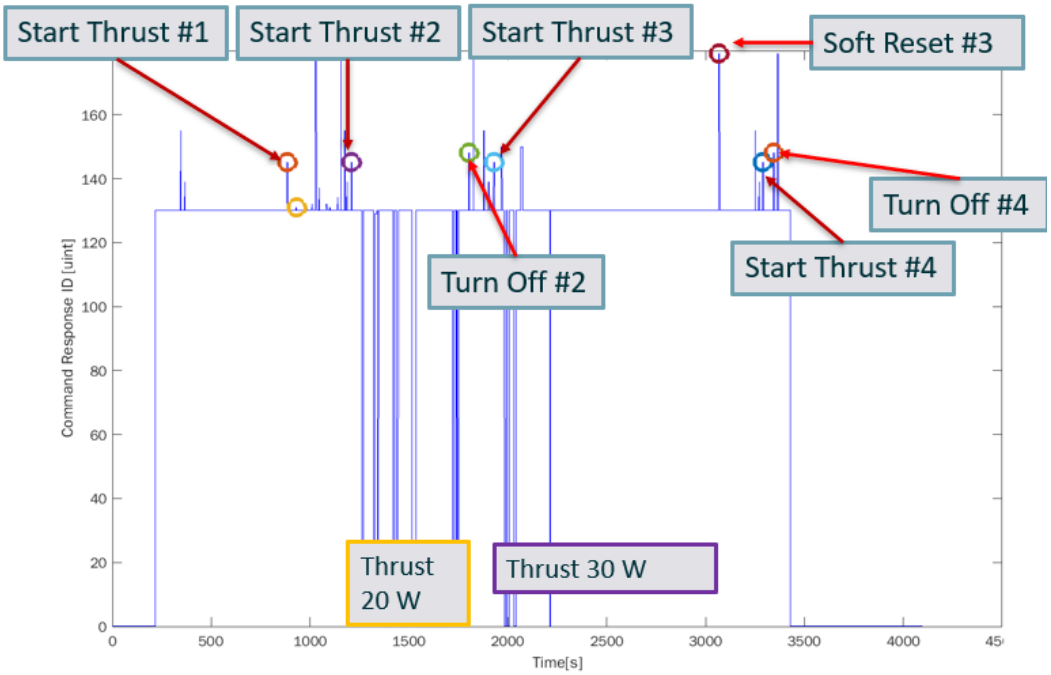


Figure 4.8: REGULUS Command Response

T4I felt confident to proceed to the next test campaign in EPL-ESTEC, which required the shipment of CTP via air: for this reason the batteries were not recharged after the test.

4.2 Test campaign at EPL-ESTEC

The first activity required in EPL-ESTEC was a Visual Inspection and a Reduced Functional Test at laboratory conditions which had the objectives to assert the main functional, operative, interface and physical requirements at system level. This test demonstrated that CTP can perform the main intended functions after the shipment.

The visual inspection was required to verify that no temperature sensors nor communication cables un-plugged during shipment. Indeed, the REGULUS-CTP communication cable detached from REGULUS's connector, and a single thermistor needed to be reattached to its destination panel. Finally, the two batteries' connectors were plugged to the EPS board.

The test completed and CTP functionalities were validated after the shipment. The next task was now to integrate the CTP inside the Small Plasma Facility

chamber (SPF). For the vacuum chamber tests, the platform was mounted on the thrust balance, and the thrust balance mounted inside the chamber. Chambers' thermo-couples were mounted on the CTP to validate internal sensors measurements. A set of dedicated feedthrough allowed the connection to the Ground Support System (GSS), Workstation, Load switch, RF antenna, and power supply from outside the chamber:

- one DB-25 that hosted two RS232 serial lines, antenna wires, Load Switch connection and Ethernet connection;
- two high-voltage feedthrough for battery recharging.

For this session the recharge cables were installed through the chamber. The data of the propulsion system acquired by EPIS-DL, relayed to CDH and finally sent to ground is again forwarded by the GSS to the Propulsion Operation Control Centre (POCC) through a serial line (9600bps, 8N1). The external Xenon reservoir for REGULUS was connected to the CTP and the configuration was now complete.

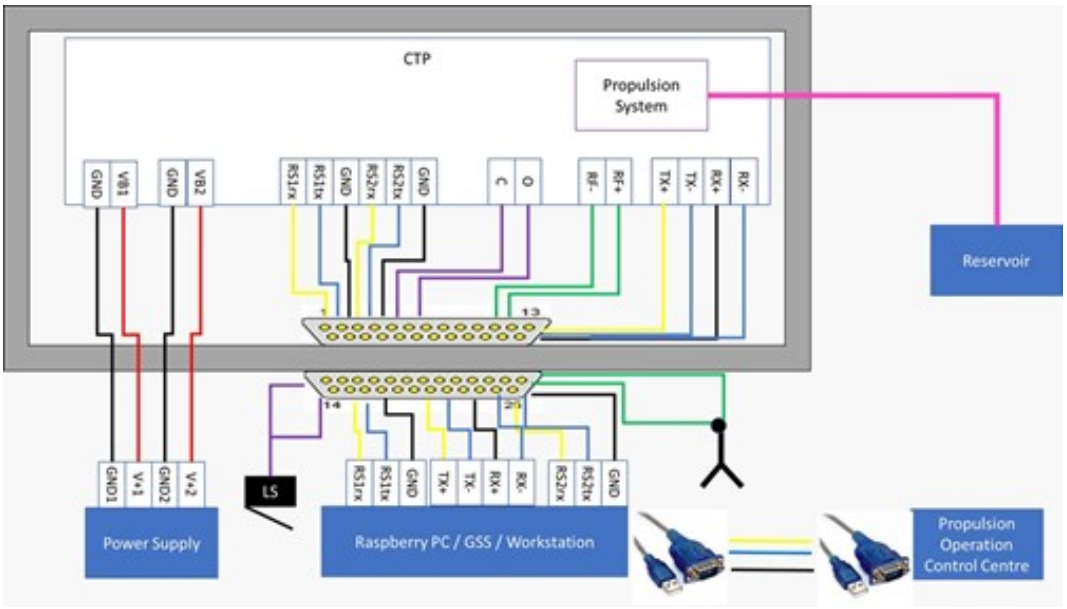


Figure 4.9: CTP interfaces with EPL-SPF

Before proceeding to the vacuum operations, a brief Reduced Functional Test was again performed to be sure that the electrical interconnections were functional: the test went fine and, after a Test Readiness Review (TRR) between PoliTo, ESA and T4I personnel the operations to bring the chamber in vacuum begun.

4.2.1 Test Success Criteria

- Data collection, packet preparation and transmission from EPIS-DL, to CDH and finally to GSS;
- Commands reception, validation, interpretation and execution from GSS, to CDH and finally EPIS-DL;
- Correct power distribution from EPS to EPS-Power;
- Correct batteries discharge and charge rates;
- Correct transitions between operative modes.
- **Correct commands transmission and telemetry reception from the REGULUS EP System**;
- **Telemetry reception from POCC**;
- **Correct CTP behavior in vacuum conditions**.

4.2.2 List of Items

- CTP fully integrated with REGULUS installed;
- Personal Computer with Linux system (GSS);
- Payload Operator Control Centre (POCC).

4.2.3 Test Procedure

The test was divided in different phases, and to have a better comprehension of the tasks executed only a brief description will be given.

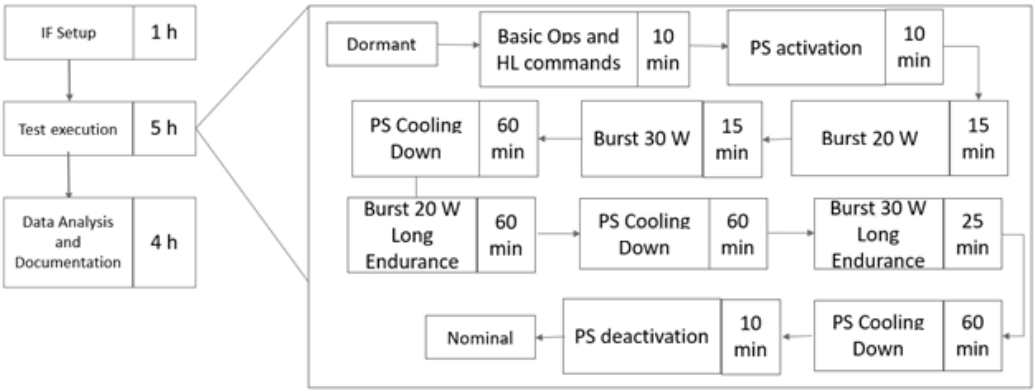


Figure 4.10: Thermal test flow activity

1. **IF Set-up:** the CTP was installed mechanically inside the chamber as well as the electrical interconnections between GSS and CTP;

2. **Fit check:** RFT performed to be sure that the electrical interconnections were functional;
3. **Vacuum:** the chamber was closed and the vacuum operations start. To reach an acceptable level of vacuum (P<10^{-5}Pa) eight hours were needed;
4. **Batteries Recharge**: in the meanwhile the two batteries were recharged by two Power Supplies controlled by EPL-GSE;
5. **Basic Ops and HL commands:** the same operations performed in the other FFTs were executed to test the correct functionalities of the CTP: communication links, operative mode transitions and data acquisition;
6. **PS activation:** REGULUS was now switched on and entirely supplied by the CTP. Before the thrust phase a controlled heating phase was required;
7. **Burst 20W:** a 15 minutes thrust phase was performed with REGULUS internal power limit set at 20W;
8. **Burst 30W:** a 15 minutes thrust phase was performed with REGULUS internal power limit set at 30W;
9. **Cool down:** the EP System was let to cool for 60 minutes;
10. **Long endurance Burst 20W:** a 60 minutes thrust phase was performed with REGULUS internal power limit set at 20W;
11. **Cool down:** the EP System was let to cool for 60 minutes;
12. **Long endurance Burst 30W:** a 25 minutes thrust phase was performed with REGULUS internal power limit set at 30W;
13. **Cool down:** the EP System and CTP were let to cool for 60 minutes;
14. **PS deactivation and Basic Mode:** REGULUS was switched off and CTP put into Basic Mode;
15. **Deactivation:** shutdown command is sent to the CTP and the software stops the execution.

The test results are divided in this format:

- a first section called “Thermal Short”, where the two bursts at 20 W and 30 W had been maintained for 15 minutes;
- a second section called “Long Endurance @ 20W”, where the burst at 20 W had been maintained until the "turn_off" command was issued, after about 60 minutes;
- a second section called “Long Endurance @ 30W”, where the burst at 30 W had been maintained until the EP System shut itself down (that happened after about 25 minutes).

Therefore, the results data will be presented following this categorization.

4.2.4 Thermal Short Results

This part of the test lasted 12605s (3.5 hours). Considering the system features and the test duration, 12605 telemetry packets were expected to be saved, and 2521 telemetry packets were expected to be received: 12580 telemetry packets were actually saved, meaning that 99% of the expected telemetry was correctly saved; while 2436 packets were actually received, meaning that 96.6% of the expected telemetry packets were correctly received.

Batteries behaviour adapted correctly to the different load conditions met during the test and there were not unexpected anomalies nor off-nominal temperatures(max 35°C).

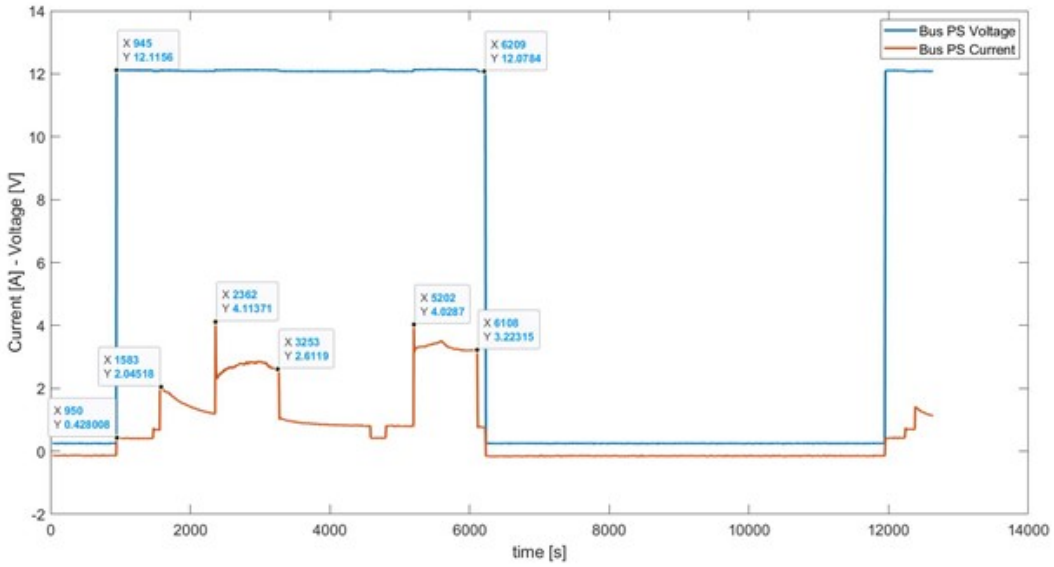


Figure 4.11: EP System Supply Bus - Thermal Short

The EP System Supply Bus was active only during the Propulsion On phase (from second 945 to 6209) to power the EP System: as expected it has been measured a voltage of 12.0 V with a current absorption variable and phase dependent. At first, when only the EP System avionics were powered on, the absorbed current was around 0.4 A, while during the heating phase the absorbed current raised at 2 A. During the 20 W burst the absorbed current was about 2.5 A with a peak of 4.1 A, while during the 30 W burst it was about 3.3 A with a peak of 4.0 A.

Temperatures in the Propulsor Module increased during the two bursts, notably during the 30 W thrust, when they peaked but always remaining inside the operative ranges. The highest temperatures were reached by the brackets that hold the propulsor at about 45°C. In general, the variation of temperature was about +15°C (+20°C for the brackets). During this test the vacuum chamber's

thermocouples were mounted on the CTP external panels, validating the accuracy of the CTP temperature sensors' measurements. Temperatures in the Service Module remained inside the operative ranges and the average variation was about +10°C. Both EPS and EPIS-Power boards reached at maximum 35°C, while the two microprocessors reached about 50°C.

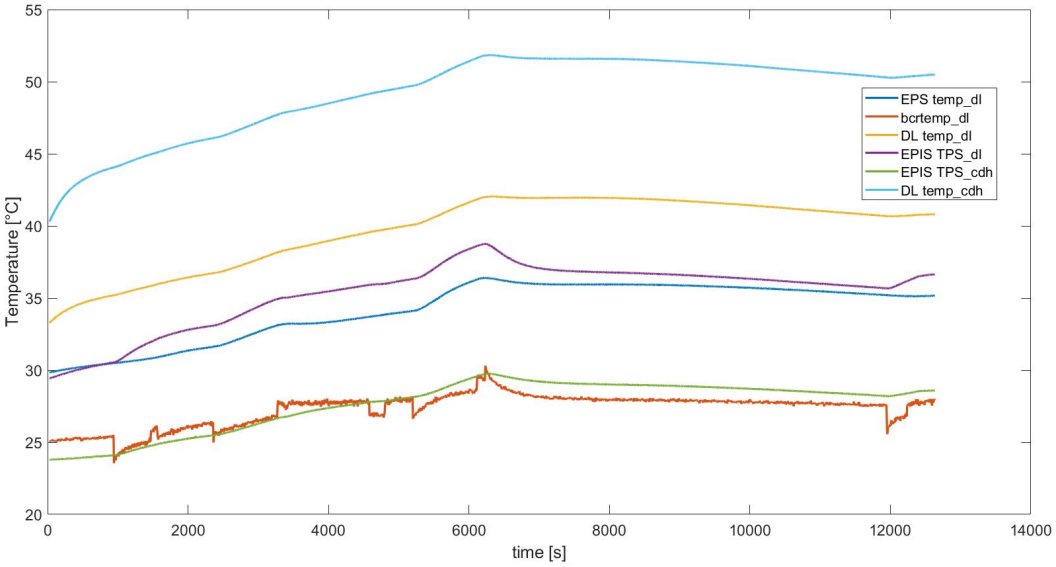
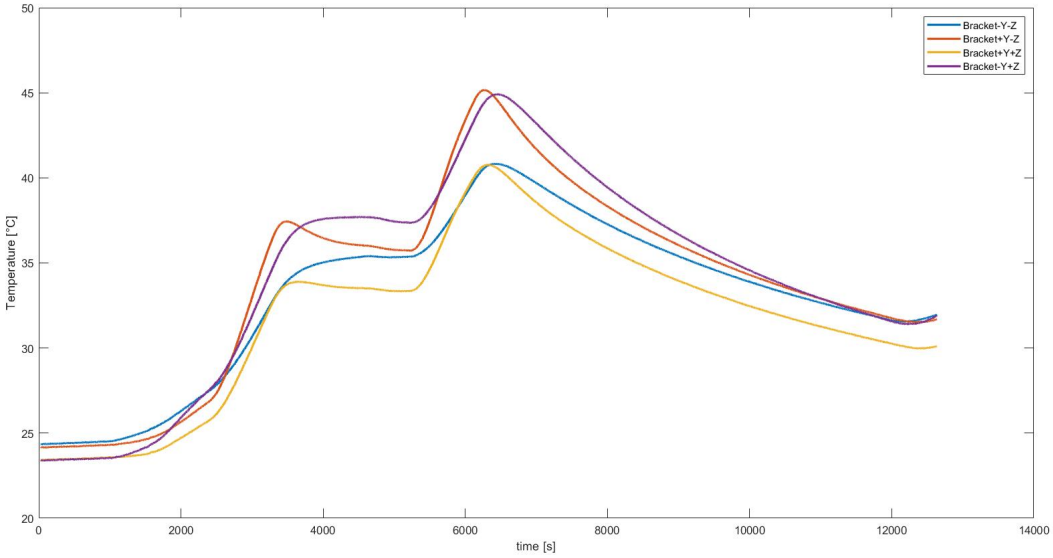
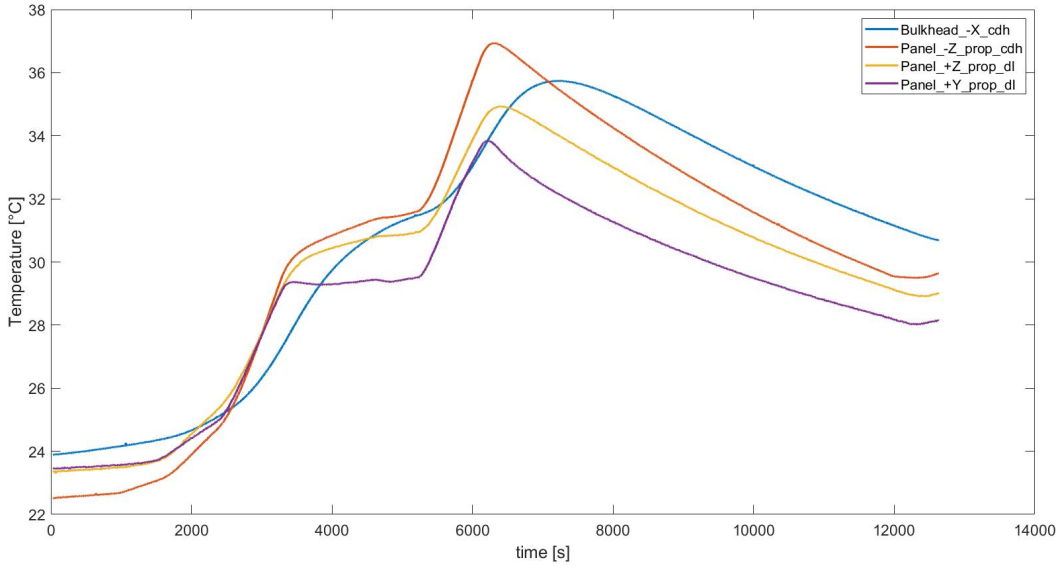


Figure 4.12: EPIS-DL Recorded temperatures - Thermal Short (1)

The LISN circuit sensed electromagnetic interference during the two bursts. It seems that EMI during the first burst at 20 W was bigger than that sensed at 30 W. However, these perturbations never reached a magnitude of 0.5 dBm. The RF sensors followed the same behaviour of the LISN; in general no big perturbations were detected.

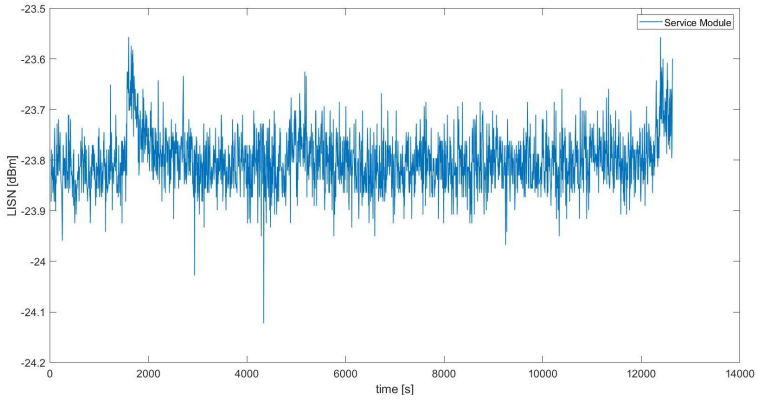


(a) Brackets

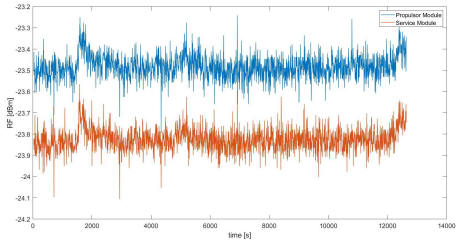


(b) Panels

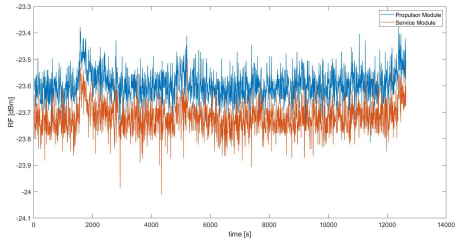
Figure 4.13: EPIS-DL Recorded temperatures - Thermal Short (2)



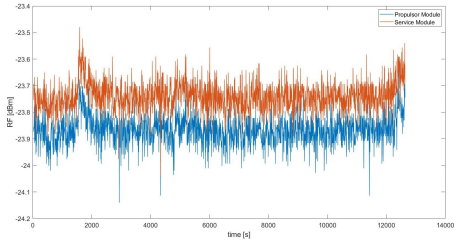
(a) LISN



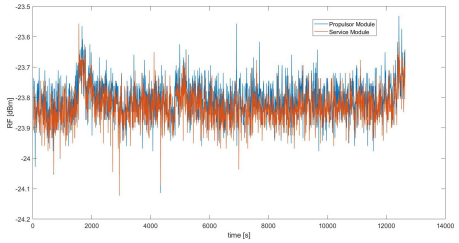
(b) 1-10MHz



(c) 20-50MHz



(d) 50-120MHz



(e) 400-500MHz

Figure 4.14: LISN and RF units - Short Thermal

4.2.5 Long Endurance @ 20W Results

This part of the test lasted 6785s (almost 2 hours). Considering the system features and the test duration, 6785 telemetry packets were expected to be saved, and 1357 telemetry packets were expected to be received: 6717 telemetry packets were actually saved, meaning that 99% of the expected telemetry was correctly saved; while 1297 packets were actually received, meaning that 95.6% of the expected telemetry packets were correctly received.

Batteries behaviour adapted correctly to the different load conditions met during the test and there were not unexpected anomalies nor off-nominal temperatures(max 33°C).

The EP System Supply Bus was active only during the Propulsion On phase (until second 17255) to power the EP System: as expected it has been measured a voltage of 12.0 V with a current absorption variable and phase dependent. This Bus was already activated at the end of the first part of the test (from second 11954). During the heating phase, the absorbed current raised at 1.1 A while during the 20 W thrust phase the absorbed current was about 2.5 A, with a peak absorption of 3.3 A.

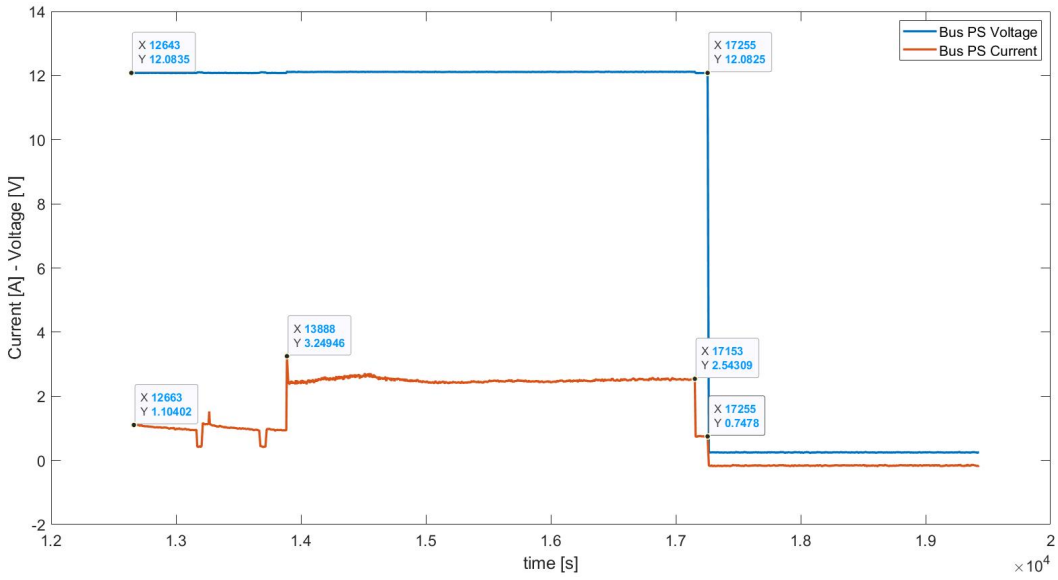


Figure 4.15: EP System Supply Bus - Long Endurance @ 20W

Temperatures in the Propulsor Module increased linearly during the thrust phase but always remaining inside the operative ranges. The highest temperatures were reached by the brackets that hold the propulsor at about 50°C. In general, the variation of temperature was about +15°C (+20°C for the brackets). This behaviour was the same recorded during the Short Thrust, confirming the thermal

behavior of REGULUS. Temperatures in the Service Module remained fairly steady and always inside the operative ranges, the greatest variation was experienced by the Bulkhead (+8°C) while others were under +5°C. Both EPS and EPIS-Power boards reached at maximum 40°C, while the two microprocessors reached about 55°C.

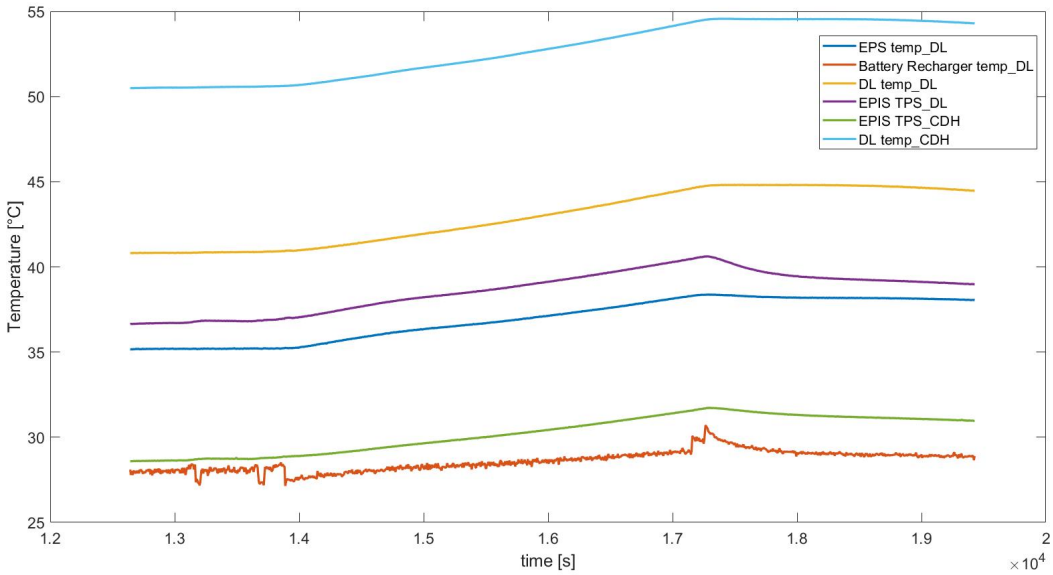
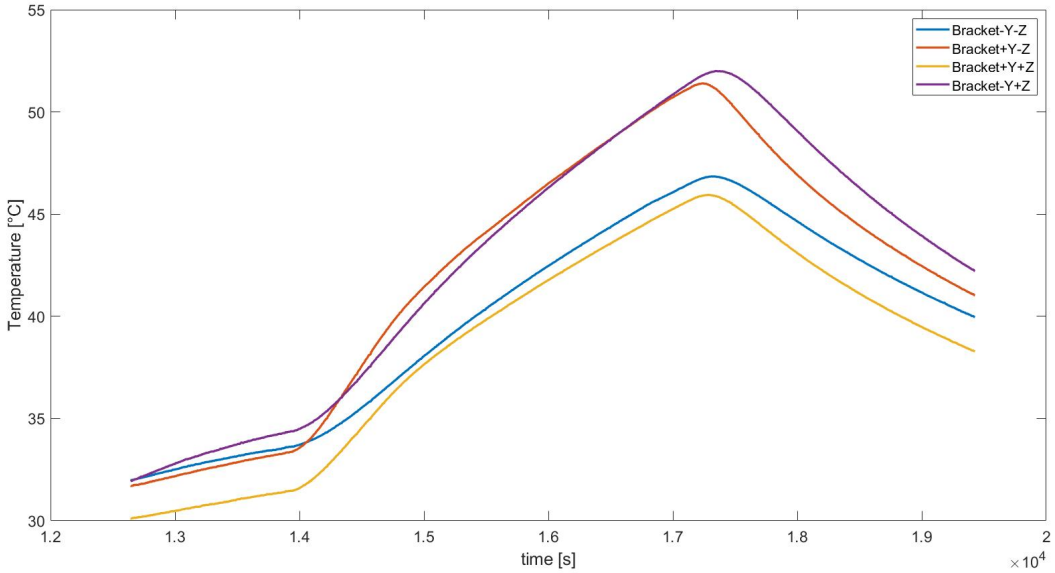
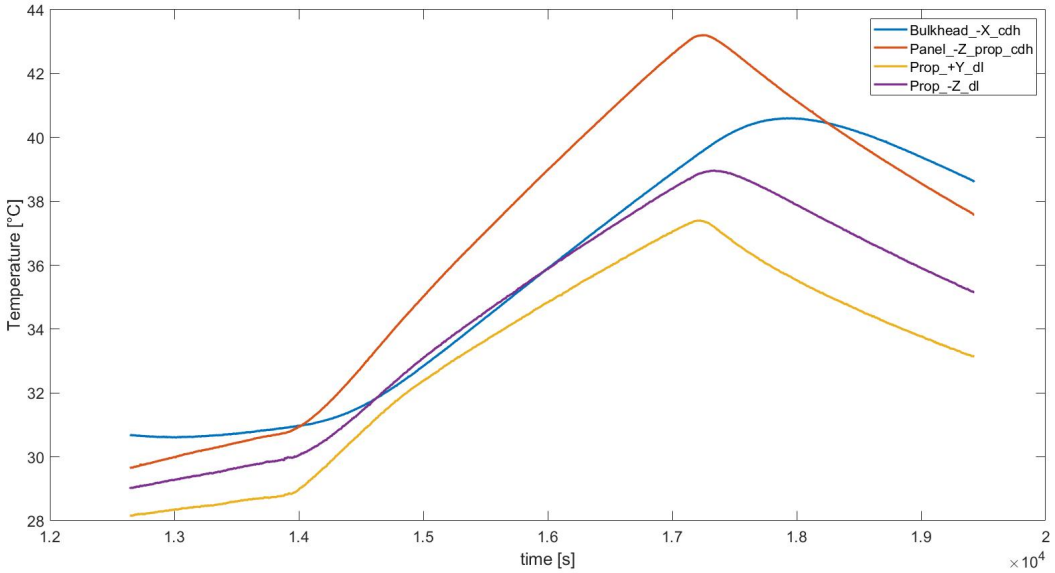


Figure 4.16: EPIS-DL Recorded temperatures - Long Endurance @ 20W (1)

The LISN circuit did not sense EMI during the burst, aside for an initial spike at the very beginning of the firing. The RF sensors followed the same behaviour of the LISN, meaning that no perturbations were sensed.

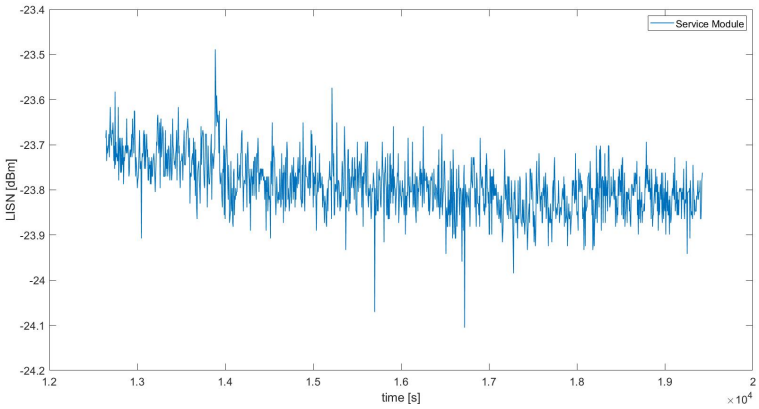


(a) Brackets

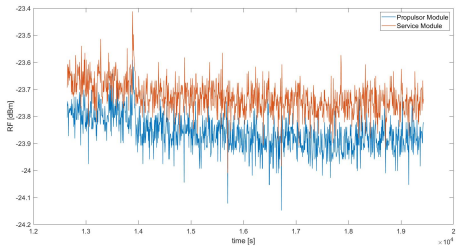


(b) Panels

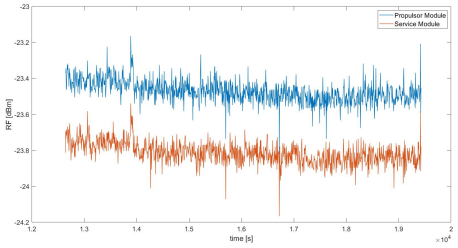
Figure 4.17: EPIS-DL Recorded temperatures - Long Endurance @ 20W (2)



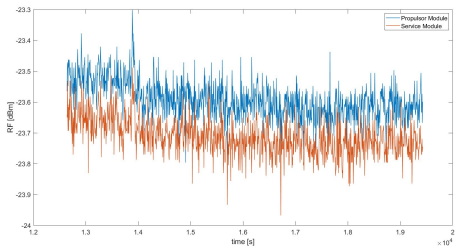
(a) LISN



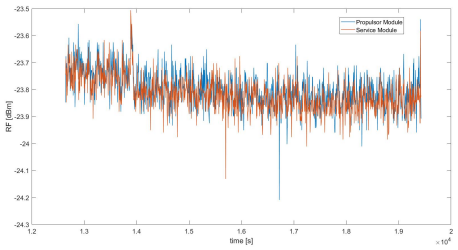
(b) 1-10MHz



(c) 20-50MHz



(d) 50-120MHz



(e) 400-500MHz

Figure 4.18: LISN and RF units - Long Endurance @ 20W

4.2.6 Long Endurance @ 30W Results

This part of the test lasted 7925s (2 hours) and is composesd by a cooling phase and a thrust phase. Considering the system features and the test duration, 7925 telemetry packets were expected to be saved, and 1585 telemetry packets were expected to be received: 7825 telemetry packets were actually saved, meaning that 99% of the expected telemetry was correctly saved; while 1525 packets were actually received, meaning that 96.2% of the expected telemetry packets were correctly received.

Batteries behaviour adapted correctly to the different load conditions met during the test and there were not unexpected anomalies nor off-nominal temperatures(max 33°C).

The EP System Supply Bus was active only during the Propulsion On phase (from second 22429 to 24772) to power the EP System: as expected it has been measured a voltage of 12.0 V with a current absorption variable and phase dependent. REGULUS's heathers were turned on at second 22758, from that moment the propulsor absorbed about 1.15 A. During the 30 W burst the absorbed current was about 3.1 A with a peak consumption of 4.1 A. After the forced turn off the EP System absorbed 0.8 A.

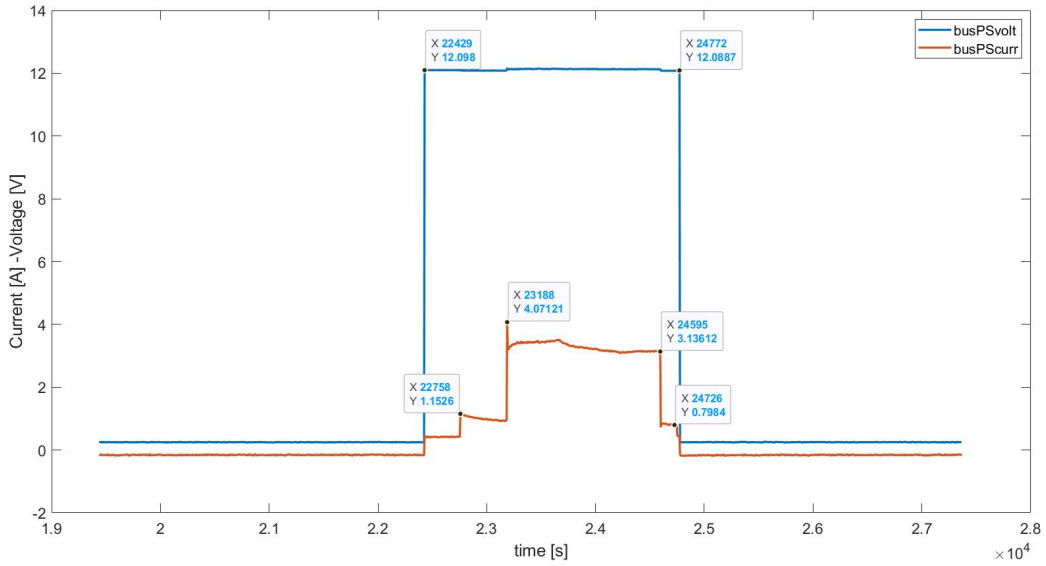


Figure 4.19: EP System Supply Bus - Long Endurance @ 30W

Temperatures in the Propulsor Module increased linearly during the thrust phase but always remaining inside the operative ranges. The highest temperatures were reached by the brackets that hold the propulsor at about 50°C. In general, the variation of temperature was about +15°C (+20°C for the brackets). This

behaviour was the same recorded during the Short Thrust, confirming the thermal behavior of REGULUS. Temperatures in the Service module remained inside the operative ranges and quite constant during the cooling phase (the biggest decrease was about 2°C). During burst the temperatures increased but the variations were quite small (about 3°C for most of the components). Both EPS and EPIS-Power boards reached at maximum 42°C, while the two microprocessors reached about 55°C.

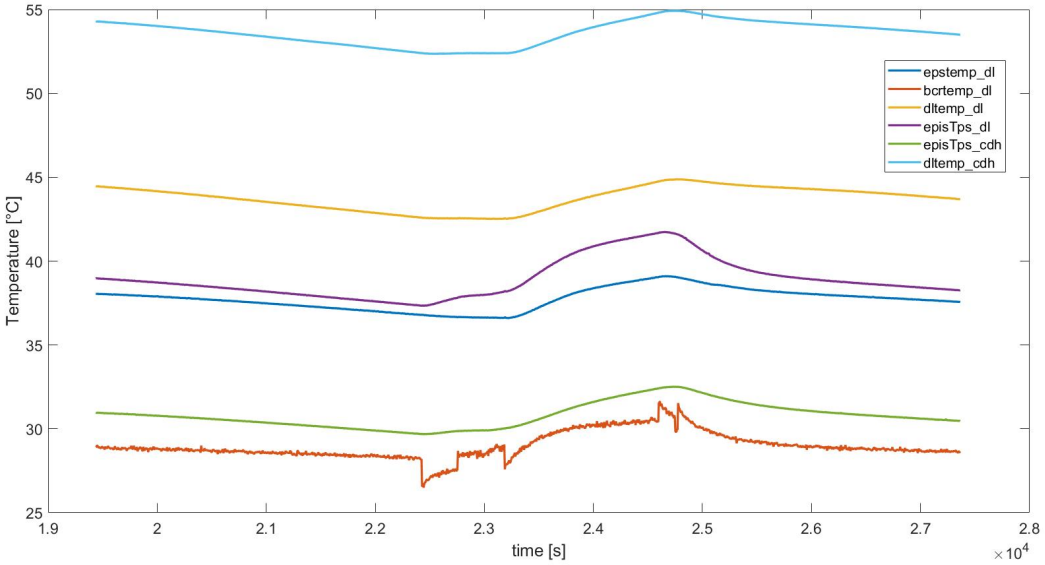
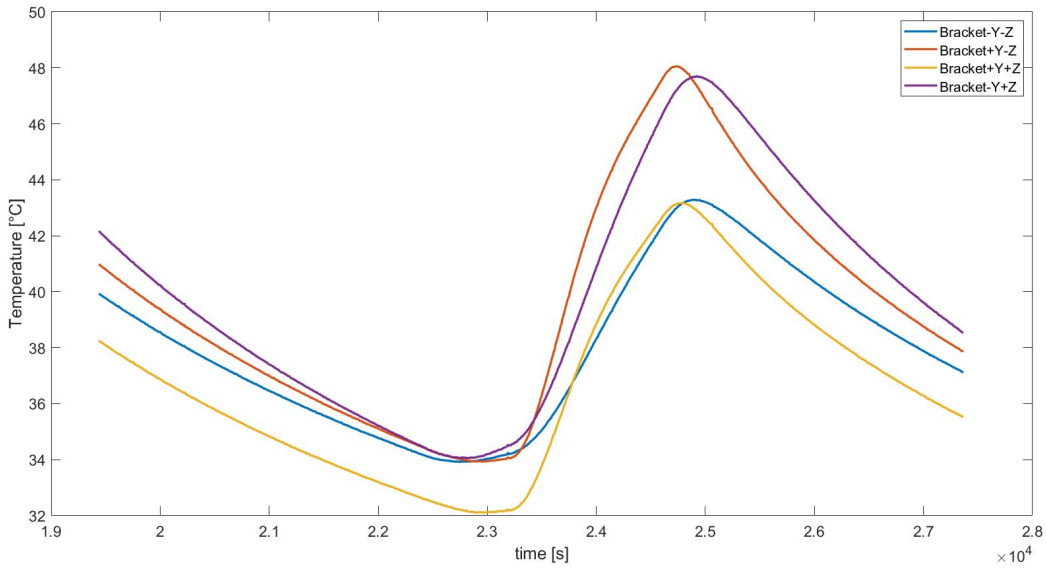
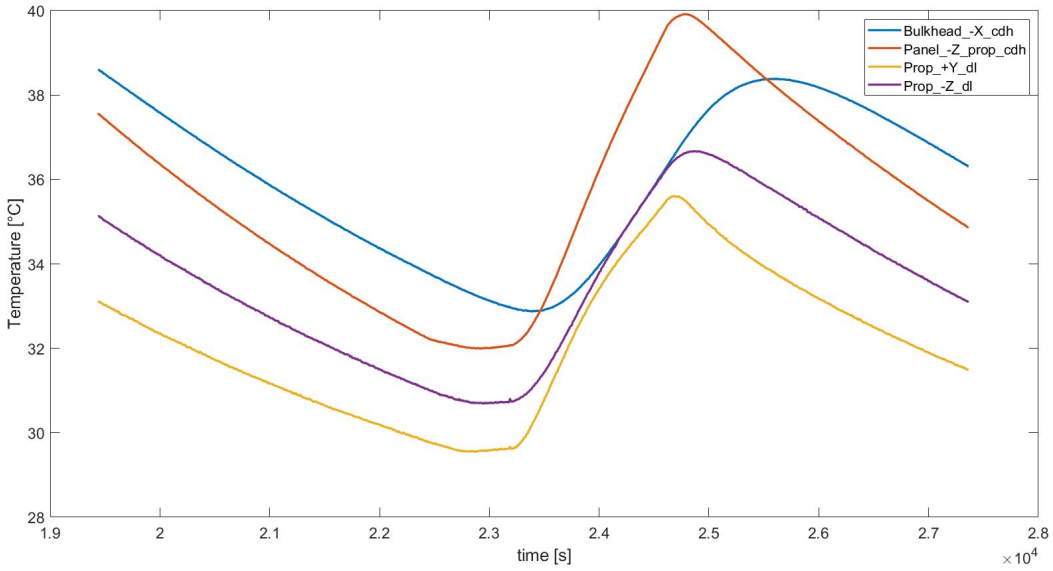


Figure 4.20: EPIS-DL Recorded temperatures - Long Endurance @ 30W (1)

The LISN circuit sensed little electromagnetic interference during the burst, but their magnitude is less than 0.2 dBm. The RF sensors followed the same behaviour of the LISN.

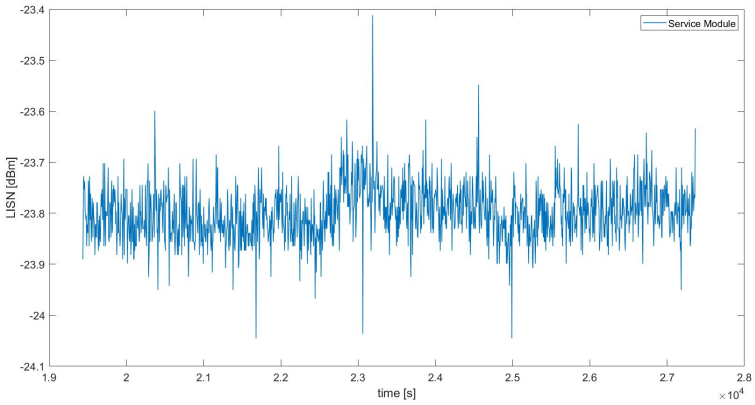


(a) Brackets

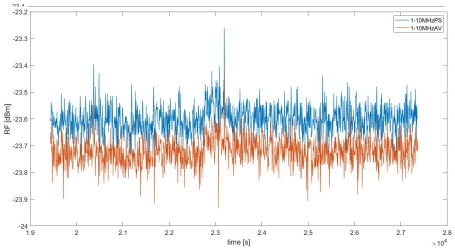


(b) Panels

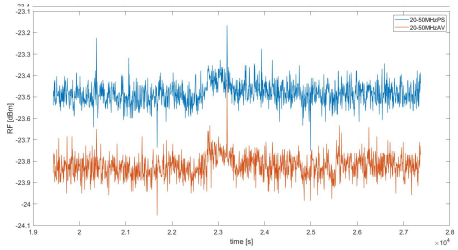
Figure 4.21: EPIS-DL Recorded temperatures - Long Endurance @ 30W (2)



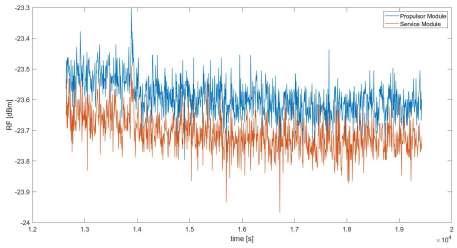
(a) LISN



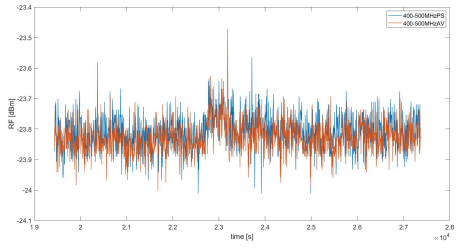
(b) 1-10MHz



(c) 20-50MHz



(d) 50-120MHz



(e) 400-500MHz

Figure 4.22: LISN and RF units - Long Endurance @ 30W

4.2.7 Anomalies and discussions

As observed in Padua communication between REGULUS and CTP became unstable during the initial phases of a burst. This phenomena also occurred during these tests, in particular the complete loss of telemetry lasted:

- 21 seconds for Thermal Short @ 20 W;
- 126 seconds for Thermal Short @ 30 W;
- 81 seconds for Long Endurance @ 20 W;
- 150 seconds for Long Endurance @ 30 W.

The instability affected only the reception of telemetry from REGULUS since every command issued from GSS to EP System was always received and correctly executed.

The problem was analyzed and three possible root causes have been found:

- **Wiring:** the cables used to connect EPIS-DL to REGULUS were shielded but not entirely, indeed some ends of the shielded cable (about 3-4 cm) in the Propulsor Module were exposed;
- **Communication protocol:** I2C is a serial synchronous communication protocol typically used for on-board application. Due to its synchronous nature the communication can be affected by skew problems (transmission delays are characterized by uncertainties) and the interconnection length results limited to few centimeters. For this reason in aerospace applications differential signaling is recommended [18].
- **Physical Layer of I2C:** REGULUS required the I2C to work at 3.3V with external pull-up resistors. On the EPIS-DL two Pull-Up resistors of 10kOhm have been installed on the SDA and SCL lines. This phyisical implementation of the I2C protocol could be not strong enough for such a "harsh" environment in terms of EMI. An higher voltage for the lines (+5V) could have improved the fault tolerance of the serial line to Single Event Upset (SEU).

Electromagnetic Interference sensors never registered a variation of more than 0.5 dBm both on the EP System Supply Line nor in the analyzed frequency bands. This results are in contradiction with the previously cited problems. Further analysis on the PCB design of EPIS-DL need to be conducted in order to establish the root causes of this problem.

Chapter 5

Conclusions

The Commissioning Test Campaign demonstrated that CTP integrated with an EP System, and installed in the SPF-EPL vacuum chamber operates in compliance with the functional, operational and interface requirements. Major EP System telemetry losses happened at the beginning of each firing sessions, underlining the presence of a fault in the communication interface, but not specifying at which end it is located. CTP telemetry was complete and stable, and the system worked for 7 consecutive hours in its longest test.

CTP gathered all the data, executed all the commands, and maintain the communication with the GSS. CTP was able to send commands to EP System, which correctly executed them and switched among its operative modes accordingly. The telemetry demonstrated that the CTP worked properly without off-nominal behaviours and the critical parameters remained within the expected ranges.

The power consumption was compliant with the expected values: the avionics systems guaranteed low consumption which enabled long test's duration. The CTP delivered the required power to the loads, the voltage level of the step-up circuit remained in the required ranges $12V \pm 0.1V$ also during the thrust phases (when the requested power was at its maximum). The CTP batteries discharge/charge rates were verified, as well as the capability of the power system to provide the intended power levels to both avionics and EP System.

Operators could control the test activities through the GSS without any major anomaly: the GSS received and displayed all the telemetries in Real Time while the Graphical User Interface allowed an easy way to check and control the operations.

5.1 Interfaces

- **CTP vs REGULUS**
  - Mechanical Interface: flawless matching of Regulus in the PS box;

  - Electrical Interface: flawless connection both on Supply and Communication lines;
  - Data Interface: all the commands were accepted and executed by REGULUS, but telemetry responses were not always successful during the firing sessions, especially those at higher power.

- **CTP vs GSS**
  - Electrical Interface: flawless connection between the platform and the Ground Support System through the SPF interfaces;
  - Data Interface: most of the packets during the different test sessions were successfully exchanged, with a throughput always greater than 96%.

In order to correctly address the communication problem, **further tests using different communication protocols should be considered** to complete the verification process.

5.2 Electrical power consumption

The avionics systems had a very low consumption (less than 5Wh) as expected, enabling multiple tests to be performed with only one battery recharge in five days. At the end of the test campaign the batteries were still quite charged, with a State of Charge greater than 50%, demonstrating that the platform is able support the test campaign of a miniaturized propulsion system without impacting on the test activities. CTP delivered the desired electrical power to EP System, meaning that EPIS-Power was able to supply the expected power to the EP System in any phase of test campaign up to the maximum possible request of 120W. However, REGULUS was able to fire only at 20W and 30W, which is a lot below the maximum power the CTP can provide. As a result, the CTP behaviour was not explored in its full operative range, and EPIS-Power board never reached a temperature greater than 42°C (38°C below the maximum allowed value).

5.3 EMC/EMI

At platform level, no major impact was observed on the on-board avionics system for any operative mode of the propulsion system. CTP worked properly storing, transmitting data and receiving command. COMSYS emissions did not generate any problem to the other on-board systems during the transmission phase. No impact due to conducted emissions (LISN measurements) which indicates that REGULUS does not generate noise on the power supply line. The magnetic

field emission (which results have been omitted in the previous tests results) was negligible/comparable with respect to the Earth Magnetic Field, meaning that the generated magnetic dipole can be compensated by the actuators of the small satellites. Again, communication anomalies occurred on the line between REGULUS and EPIS-DL that caused loss of telemetry at the beginning of the firing sessions; on-board RF sensors did not measured electromagnetic interference although it is likely they were the reason for the communication issue occurred.

5.4 Thermal environment

The heating was reasonable both in the Service Module and in the Propulsion Module: the highest increase of temperature was about 20°C on the brackets that held the propulsor, since no Thermal Protection System was included in the integration to better spread the heat to the structure.

REGULUS was not able to dissipate heat properly since its internal temperature was the limiting factor of the firing sessions duration, since the EP System reached its upper operative limits after about 30 minutes of burst at 30W.

The major impact on the avionics was due to the high demand in terms of electrical power from the EP system and the subsequently heat distribution through the primary structure in the vacuum chamber.

5.5 Conclusions

This final test campaign allowed the assessment of the mutual impact of an EP System and the CubeSat technology at system and subsystem levels.

From the platform point of view, CTP demonstrated the capability to manage on-board operations with traditional communication protocols, the capability to provide high power (up to 120 W, a very relevant value for a 12U CubeSat) from CTP to REGULUS in any operative mode and for long time. Variations of the RF noise emission have not been observed during the test according to the operative mode of REGULUS, but no major impact was observed on the CTP operativity and availability. Further investigations are desirable to achieve a more accurate noise measure. The generated thermal environment by the EP System during long firings and by the battery recharging at high rate should be considered in detail and a TPS/TCS system (at least passive) would ensure longer operation duration.

From the EP System point of view, it was demonstrated the possibility for an EP System to be integrated in a CubeSat platform without major impact on the basic avionics systems. Future works are driven by the improvement of the data exchange between REGULUS and the platform in terms of communication reliability (e.g. implementing better physical layer), the improvement of the thermal protection of

the EP System's avionics, and the improvement of the sensors and sensing circuits calibration. Moreover, mission tests, including different mission profiles for a 12U CubeSat with propulsion system, would allow to assess the impact of the propulsion system at mission level.

The Electric Propulsion Interface System installed inside the CubeSat Test Platform allowed to better understand the behaviour of an Electric Propulsion System in a real mission scenario, exposing all the design flaws and faults that led to failures that could not be addressed when testing the unit by itself. Vice versa, the problems affecting the CTP were also underlined and will allow to improve future designs of the platform.

Bibliography

[1] Kristina Lemmer. «Propulsion for CubeSats». In: *Acta Astronautica* 134 (2017), pp. 231–243. ISSN: 0094-5765. DOI: https://doi.org/10.1016/j.actaastro.2017.01.048. URL: https://www.sciencedirect.com/science/article/pii/S0094576516308840 (cit. on p. 3).

[2] Fabrizio Stesina. «Validation of a Test Platform to Qualify Miniaturized Electric Propulsion Systems». In: *Aerospace* 6.9 (2019). ISSN: 2226-4310. DOI: 10.3390/aerospace6090099. URL: https://www.mdpi.com/2226-4310/6/9/99 (cit. on pp. 3, 6, 12).

[3] Fabrizio Stesina, Sabrina Corpino, and Daniele Calvi. «A Test Platform to Assess the Impact of Miniaturized Propulsion Systems». In: *Aerospace* 7.11 (2020). ISSN: 2226-4310. DOI: 10.3390/aerospace7110163. URL: https://www.mdpi.com/2226-4310/7/11/163 (cit. on p. 4).

[4] *LTC2449 ADC Datasheet*. Linear Technology. 2017. URL: https://www.analog.com/media/en/technical-documentation/data-sheets/2444589fc.pdf (cit. on pp. 8, 67).

[5] *BM1422AGMV 3-Axis Digital Magnetometer*. ROHM Semiconductor. URL: https://fscdn.rohm.com/en/products/databook/datasheet/ic/sensor/geomagnetic/bm1422agmv-e.pdf (cit. on p. 8).

[6] Jacopo Garrone. *Design and verification of the Thermal Control System for a Cubesat equipped with a miniaturized electric propulsion system*. Master Degree Thesis. 2021. URL: https://webthesis.biblio.polito.it/18389/.

[7] *BHX2 Multi.channel UHF Transceiver*. Radiometrix. URL: http://www.radiometrix.com/files/additional/bhx2.pdf (cit. on p. 28).

[8] *Compute Module 4 SoM*. Raspberry Pi. 2020. URL: https://datasheets.raspberrypi.com/cm4/cm4-datasheet.pdf (cit. on p. 44).

[9] *Compute Module 4 IO Board*. Raspberry Pi. 2020. URL: https://datasheets.raspberrypi.com/cm4io/cm4io-datasheet.pdf (cit. on pp. 44, 56).

[10] *BCM2711 ARM Peripherals*. Raspberry Pi. 2020. URL: https://datasheets.raspberrypi.com/bcm2711/bcm2711-peripherals.pdf (cit. on p. 44).

[11] *minicom serial communication program*. minicom-team. URL: https://salsa.debian.org/minicom-team/minicom (cit. on p. 56).

[12] *ssh secure shell*. URL: https://www.openssh.com/ (cit. on p. 56).

[13] *Raspberry Pi OS*. Raspberry Pi Foundation. URL: https://www.raspberrypi.com/software/operating-systems/ (cit. on p. 57).

[14] *scp secure copy*. URL: https://www.openssh.com/ (cit. on p. 57).

[15] *Visual Studio Code Integrated Development Environment*. Microsoft. URL: https://code.visualstudio.com/ (cit. on p. 58).

[16] *DHCP network management protocol*. URL: https://en.wikipedia.org/wiki/Dynamic_Host_Configuration_Protocol (cit. on p. 58).

[17] Marco Manente et al. «REGULUS: Iodine Fed Plasma Propulsion System for Small Satellites». In: Sept. 2019 (cit. on p. 94).

[18] David A. Gwaltney and Jeri Briscoe. «Comparison of Communication Architectures for Spacecraft Modular Avionics Systems». In: (2006). URL: https://ntrs.nasa.gov/api/citations/20060050129/downloads/20060050129.pdf (cit. on p. 121).

[19] *ESA-μProp 3 Technical Note 1: Test Platform Interface Report*. CubeSat PoliTo Team.

[20] *ESA-μProp 3 Technical Note 2: Test Platform Design Report*. CubeSat PoliTo Team.

[21] *ESA-μProp 3 Technical Note 3: Test Platform AIV Plan*. CubeSat PoliTo Team.

[22] *ESA-μProp 3 Technical Note 4: User Manual*. CubeSat PoliTo Team.

[23] *ESA-μProp 3 Technical Note 7: Mission Test Plan*. CubeSat PoliTo Team.

[24] *ESA-μProp 3 Technical Note 8: Mission Test Report*. CubeSat PoliTo Team.