Samuele Gelmi

Fault Detection Isolation and Recovery for LUMIO mission

Algorithm and methodology

TU Delft

Fault Detection Isolation and Recovery for LUMIO mission
Algorithm and methodology

Master of Science Thesis

By

Samuele Gelmi

in partial fulfilment of the requirements for the degree of

Master of Science
in Space Engineering

at the Delft University of Technology,
to be defended publicly on Wednesday September 25, 2019 at 14:00

Student number:	4729218	
Supervisor:	S.Radu	TU Delft - LR-SSE
Thesis committee:	Dr. A. Cervone	TU Delft - LR-SSE
	Ir. K.J. Cowan	TU Delft - LR-ASM

TU Delft

Preface

This Thesis represents the final step of my Master of Science studies at the Aerospace Engineering faculty in TU Delft. Despite the many challenges that I faced, the overall experience was great, and I can call me proud of the results and honoured to have given my small contribution to a satellite mission as fascinating as LUMIO.

Being the FDIR a new area for me, I would like to express my gratitude to my daily supervisor, Silvana Radu, for introducing me to the topic and supporting me through the whole project. I believe that working on spacecraft safety allowed me to gain a different perspective on satellite missions and operations, which would definitely help me in my future as a professional in the space sector.

Apart from the professional area, during my time at TU Delft I had the pleasure to meet a lot of incredible people, whose company and support was fundamental. Therefore, I would like to thank my fellow space students (Antonio, Tiberiu, Giacomo, André, Andrea and Riccardo in particular) for the nice moments in and out of the classrooms, and the guys from “Rimasterrimi” (Isa, Anna, Emma, Irene, Benedetto, Matteo, Leonardo, Federico) for making me feel Italian vibes abroad. Many thanks also to Maria, Dodo, Simone and Federico who visited me and brought a bit of home in The Netherlands, and to Baldassa for the support given from afar. My gratitude goes also to my family, who encouraged me from the beginning to the very end, with love and sincere enthusiasm for each accomplishment. Finally, thanks to Alice for sharing with me the last two years (and helping me making nicer diagrams for the report).

Samuele Gelmi

Delft, September 2019

Abstract

In recent years, the CubeSat industry has seen increasing development. Despite the main purpose of these platforms is to be utilized in low-earth-orbit (LEO), the use of CubeSats for deep-space missions is deemed as very promising. To achieve this further step effectively, a concrete effort in the improvement of the on-board autonomy and overall reliability of these standardized platforms is required. In this context, the on-board Fault Detection Isolation and Recovery (FDIR) plays a major role, as it allows the satellite to respond autonomously to eventual failures, minimizing the risks associated with more ambitious mission designs.

This Thesis presents the design and implementation of a preliminary FDIR architecture for LUMIO, a CubeSat mission to the Moon that was proposed to ESA by TU Delft and other European and American universities and was declared as a winner within the SysNova contest. The project is still in an early design phase (Phase A) and therefore the dependability and safety-related engineering are rather preliminary. The methodology used for the design of the FDIR relies on the inputs received from the Failure Modes Effects (and Criticality) Analysis (FMEA/FMECA), contingency and FDIR analysis. Based on the information received after Phase-0 was conducted, the mission and the spacecraft (S/C) characteristics were thoroughly studied in an extensive literature review, where a set of potential failure scenarios for all the components was identified and classified. For each failure, its effects upon the spacecraft and mission objectives were analysed, as well as the potential compensating provisions. On top of this, a preliminary integration of the FDIR within the operational modes of the satellite was proposed, with the addition of a Safe Mode to be activated in case a failure is detected.

The design of the FDIR was divided between Failure Detection and Isolation (FDI) and Failure Recovery (FR). A simplistic high-level architecture for the FDI was proposed, based on the division of the S/C into modules; for each module, a detection logic was developed, based on the use of several checks. In particular, a methodology for the design of cross-checks between units was developed and applied to the ADCS sensors. The FR, on the other hand, is also divided into modules, each based on the application of a proper recovery sequence, organized into sequential levels (from L0 to L4). Finally, the FDIR architecture was implemented in MATLAB/Simulink, using the Stateflow environment, which allows incorporating the detection and decision logic into transition diagrams, easy to develop and visualize. On top of this, a simulation model was created in Simulink, to test the FDIR system. The model reproduces the data packages coming from the satellite during the nominal scenario, but failures can also be injected and simulated. To ease the verification of the FDI, a user interface was developed as well. Several tests were performed. The main objective is the verification of the preliminary design: the failure scenarios of the FMECA were injected in the simulation model and sent to the FDIR, to verify the correct detection, isolation and recovery. Moreover, the cross-check methodology developed in this Thesis was tested as well, in order to verify the rate of detection of false negatives and false positives.

The results of this Thesis are manifold. The FMECA analysis performed on the preliminary spacecraft design paved the way for the advancement of the project, since the critical items were identified, and compensating provisions were proposed. The addition of a redundant propulsion system (or the complete change of the current one) and the implementation of a back-up navigation method based on ground radiometric tracking are the main recommendations that resulted from this study. Besides, the addition of another IMU and another reaction wheel were also proposed. The FDIR design that was developed in this Thesis was verified with multiple simulations, which proved the possibility of detecting and recovering the preliminary FMECA failures with the logic proposed. Hence, the on-board FDIR algorithm will increase the autonomy of LUMIO and the overall reliability and availability of the mission. Nevertheless, some criticalities were found, to be solved in the future design: the use of propellant budget measurement, the isolation of star-tracker failures and the detection of IMU failures necessitate further researches. Moreover, considering the satellite dynamics will be paramount in the future works, to broaden the number of failure scenarios included in the FDIR and to achieve a more accurate detection and isolation of those already studied.

Table of Content

List of Tables

List of Figures

Nomenclature

List of acronyms

ADCS: Attitude Determination and Control System
AOCS: Attitude and Orbit Control System
BT: Battery
CAM: Collision Avoidance Manoeuvre
CDF: Concurrent Design Facility
CDHS: Command and Data Handling System
CN: Criticality Number
COTS: Commercial-Off-The-Shelf
CRC: Cyclic Redundancy Check
DOD: Depth-of-Discharge
ECSS: European Cooperation for Space Standardization
EPS: Electrical Power System
ESA: European Space Agency
ESTRACK: ESA's Tracking Network
FDI: Failure Detection and Isolation
FDIR: Fault Detection Isolation and Recovery
FMEA: Failure Modes and Effects Analysis
FMECA: Failure Modes Effects and Criticality Analysis
FR: Failure Recovery
FOV: Field of View
GNC: Guidance Navigation and Control
GT: Gas Thruster
HIM: Halo Injection Manoeuvre
IMU: Inertial Measurements Unit
LEO: Low-Earth-Orbit
LUMIO: Lunar Meteoroid Impact Observer
MT: Main Thruster
NASA: National Space Administration
Nav&Eng: Navigation and Enginnering
NEO: Near-Earth-Orbit
OBC: On-Board Computer
PN: Probability Number
RW: Reaction Wheel
S/C: Spacecraft
S/K: Station Keeping
SADA: Solar Array Drive Assembly
SMIM: Stable Manifold Injection Manoeuvre
SN: Severity Number
SP: Solar Panel
SPF: Single-Point-Failure
SS: Sun Sensor
ST: Star-Tracker
TRL: Technology Readiness Level
TTC: Telemetry Tracking and Command
UHF: Ultra High Frequency

List of symbols

Symbol	Description	SI units
	Latin	
bias	Bias of a measurement	Depends on the cross-check
d	Degradation of the accuracy of a measurement	[-]
DOD	Battery depth-of-discharge	[-]
err	Error function used to cross-check two measurements	Depends on the cross-check
q_i	i-th component of the attitude quaternions	[-]
$R^{N/B}$	Rotational matrix from reference frame B to reference frame N.	[-]
$r_{sun}\|_B$	Normalized Sun vector in the body reference frame	[-]
$r_{sun}\|_N$	Normalized Sun vector in the inertial reference frame	[-]
trs	Threshold used to detect failures in a cross-check	Depends on the cross-check
V	Battery voltage	[V]
	Greek	
α	Azimuth angle of the Sun vector, measured by the Sun sensor	[°]
β	Elevation angle of the Sun vector, measured by the Sun sensor	[°]
Δt	Integration time	[s]
θ_i	i-th Euler angle, describing the S/C attitude with respect to the inertial reference frame	[°]
σ_{err}	Standard deviation of the error function	Depends on the cross-check
$\sigma_{err,f}$	Standard deviation of the error function after a failure occurs to one measurement	Depends on the cross-check
σ_i	Standard deviation of i-th measurement	Depends on the measurement
$(\sigma_i)_{degraded}$	Standard deviation of i-th measurement after a loss of accuracy occurs	Depends on the measurement
ω_i	i-th component of the S/C angular speed, measured by the afferent gyroscope	[°/s]

1 Introduction

In recent years, the CubeSat industry has seen increasing development. Despite the main purpose of these platforms is to be utilized in low-earth-orbit (LEO), the use of CubeSats for deep-space missions is deemed as very promising. To achieve this further step effectively, a concrete effort in the improvement of the on-board autonomy and overall reliability of these standardized platforms is required. In this context, the on-board Fault Detection Isolation and Recovery (FDIR) plays a major role, as it allows the satellite to respond autonomously to eventual failures, minimizing the risks associated with more ambitious mission designs.

The work performed in this report is the result of the MSc Thesis done in the context of the MSc in Space Flight at Delft University of Technology (TU Delft). The topic is the development of an FDIR algorithm for a CubeSat mission to the Moon (LUMIO mission). The purpose of this Chapter is to introduce the project, by providing an overview of the context and the enunciation of the Research Objectives and Question (Section 1.1 and 1.2). Later, the results of the Literature Study on FDIR and LUMIO are discussed in Section 1.3, and the structure of the Thesis is presented (Section 1.4).

1.1 Context

The Lunar Meteoroid Impact Observer (LUMIO) is one of the four projects selected within the SysNova competition held by the European Space Agency (ESA) to develop a small satellite to be deployed by a Lunar Orbiter (Mothership). LUMIO is a CubeSat mission, a collaboration between TU Delft and other European and American universities, with the scientific purpose of observing, quantifying and characterizing the meteoroid impacts, by detecting the flashes they create on the lunar farside, as explained in *"Lunar Meteoroid Impacts Observer, A CubeSat at Earth-Moon L2, Challenge Analysis"* [1]. The project comprises a 12 U CubeSat equipped with an optical payload, the LUMIO-Cam, on-board data processing and a novel micro-propulsion system. The mission has a sophisticated orbit design: after a parking trajectory around the Moon, the spacecraft will undergo a transfer phase to reach a selected Earth-Moon L2 halo orbit, where it will be operative for one year.

LUMIO was selected as one of the two ex-aequo winners of the challenge by ESA, whose CDF department demonstrated the feasibility of the mission. However, the project still needs to undergo further studies. In particular, in its *"LUMIO Study"* [2] ESA underlined the necessity of a Fault Detection Isolation and Recovery (FDIR) design for the mission, especially indicated because of the zero-redundancy design choices, the novelty of some technologies implemented and the necessity of autonomous navigation, without ground support. The objective of this MSc Thesis is to develop such FDIR algorithm for LUMIO.

1.2 Research Objectives & Questions

The Thesis project hereby proposed is finalized at a main **Research Objective**, with several sub-goals:

1. Designing and developing a highly logical Fault Detection Isolation and Recovery architecture for LUMIO mission with simplistic and coherent MATLAB/Simulink implementation.
    a. Perform the Failure Modes Effects Analysis of LUMIO mission, consistently with the current design of the mission and the CubeSat.
    b. Integrate Fault Detection Isolation and Recovery (FDIR) blocks to the finalized Functional Flow diagrams of the satellite operations.

c. Identifying design recommendations for the next phases of the project, based on fault management results.

In order to pursue the project goals, three main **Research Questions** and several sub-questions need to be answered:

1. What inputs for the Fault Detection Isolation and Recovery development can be obtained through the Failure Modes Effects Analysis of LUMIO mission?
   a. What are the failure scenarios that can be implemented at the current stage?
   b. What is the severity level of each considered scenario?
   c. What are the most relevant criticalities in the current FDIR design, according to the analysis?

2. How can the Fault Detection and Isolation activities be integrated within the satellite operations?
   a. What are the preliminary Functional Flow diagrams of LUMIO operations?
   b. In which points can the Fault Detection Isolation and Recovery be integrated into the Functional Flow diagrams?

3. What is the Fault Detection Isolation and Recovery architecture of LUMIO mission?
   a. What is the Fault Detection and Isolation detailed block diagram?
   b. What is the Failure Recovery detailed block diagram?
   c. How can the Fault Detection Isolation and Recovery architecture be implemented in MATLAB/Simulink?

The MSc Thesis is therefore divided into different phases. The first part is aimed at laying down the basement for the development of the FDIR algorithm, by performing the FMEA, in order to identify and classify the failure scenarios to be treated, and by developing a preliminary version of the Functional Flow diagrams of LUMIO mission, which allow to integrate the FDIR within the operations of the satellite. The second part is dedicated to the design of the FDIR system itself, and it is divided between the design of the FDI, aimed at detecting and isolating eventual failures, and the FR, aimed at recovering the satellite. In the final part of the work, the FDIR model is implemented and tested in MATLAB/Simulink.

1.3 Literature Study overview

The following section is an overview of the literature review that has preceded this Thesis project. The objectives of the literature study were the understanding of LUMIO, in terms of both mission and S/C design, and the study of common FDIR methodologies used in the space industry. The main results of the literature review were the identification of the potential failure scenarios of the mission, which represent the baseline for the Failure Modes and Effects Analysis, and the preliminary draft of the Functional Flow diagram for LUMIO mission. Therefore, it is necessary, before presenting the work performed in this Thesis, to outline the methodology that was chosen for the development of the FDIR and the main aspects of LUMIO that will be used for the FDIR design.

1.3.1 LUMIO mission

An extensive study of LUMIO was carried on during the literature review; the main references were the mission Challenge Analysis [1] and the conference paper "*LUMIO: achieving autonomous operations for Lunar exploration with a CubeSat*" [3]. However, in the following section, only the main features of the mission will be summarized, to allow for a better comprehension of the FDIR design that will be developed. Therefore, after a description of the main objectives of LUMIO, two main aspects will be described: the mission scenario and the spacecraft design.

LUMIO is one of the four projects selected within ESA's SysNova competition to develop a small satellite to be deployed by a Lunar Orbiter (Mothership). LUMIO is a CubeSat mission, a

collaboration between TU Delft and other European and American universities, which address the following issues, taken from [1]:

- **Science Question**: What are the spatial and temporal characteristics of meteoroids impacting the lunar surface?
- **Science Goal**: Advance the understanding of how meteoroids evolve in the cislunar space by observing the flashes produced by their impacts with the lunar surface.
- **Science Objective**: Characterise the flux of meteoroids impacting the lunar (farside) surface.

Apart from the scientific objectives, related to the observation of meteoroid impacts on the lunar farside, a fundamental technological objective of the mission is the demonstration of autonomous operations of a CubeSat in the lunar space. Hence, a peculiarity of the mission is the implementation of an autonomous navigation system, based on a full-disk optical navigation technique. In fact, the standard navigation technique for deep space or Earth-orbiting satellites is Earth-based radiometric tracking, which requires direct communication with the ground station, thus increases significantly the cost of a mission of several M€ [1]. Demonstrating autonomous navigation would represent a milestone in the context of the growing CubeSat industry. Since direct communication with ground is not foreseen in the actual LUMIO design, radiometric navigation is not possible; therefore, the payload will be fundamental not only for science but also for navigation.

The mission scenario of LUMIO derives directly from the scientific objectives: after a thorough trade-off between different operational orbits, the baseline for LUMIO mission is the Earth-Moon L2 halo orbit family. This choice was made because a body that orbits the Earth-Moon L2 point always faces the lunar farside, allowing to perform a continuous observation. The reason is that the Langrage point L2 “is at rest with respect to a frame co-rotating with the smaller and larger primaries” [1]. In addition to this, the distance between the S/C and the Moon would permit full-disk observation. Finally, the selected orbit is at 2:1 resonance with the synodic period, which is equal to 29.4873 days. This feature allows LUMIO operation to be “steady, repetitive and regular” [1] and complementary to the ground-based observations of the lunar surface. The state-of-the-art mission scenario for LUMIO is divided into 4 phases, as it can be seen from Figure 1-1; however, for the purpose of the development of the FDIR, a fifth phase is added in this study: Phase 0, the deployment, which is considered part of Phase 1 in the reference. A brief description of the phases is fundamental to complete the overview of the mission:

- **Phase 0 - Deployment**: it starts when the lunar orbiter releases the spacecraft in a selenocentric orbit; after being deployed, the main tasks of the S/C will be the deployment

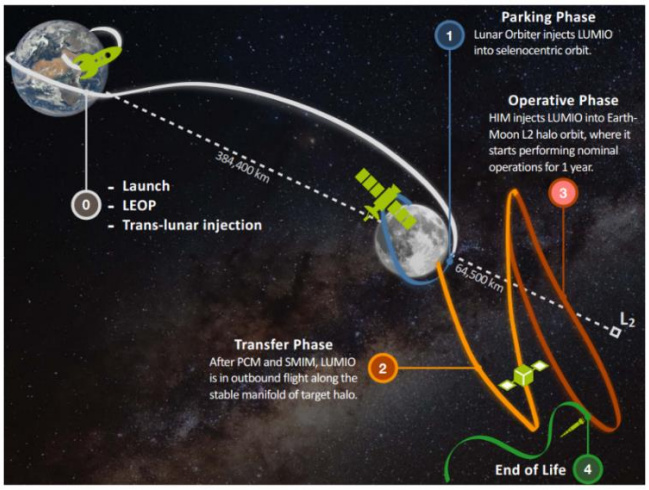

Figure 1-1: LUMIO mission phases. Credits: [1]

of the solar panels and the antennas and the de-tumbling, if needed.

- **Phase 1** - **Parking Orbit Phase**: it starts after the deployment of the appendages is successfully completed. During this phase, the S/C will orbit around the Moon for about 14 days: orbit and attitude maintenance and communication with the Mothership will be the core tasks. In this phase, LUMIO will navigate with the aid of the optical navigation, using the Earth as a reference. Each selenocentric orbit, it will be necessary to execute a Station Keeping (S/K) manoeuvre, i.e. a correction manoeuvre to maintain the S/C within the desired region and compensate the accumulation of errors during the autonomous navigation.
-
- **Phase 2 - Transfer Phase**: the spacecraft transfers from the parking orbit to the halo orbit; it lasts 14 days. The transfer begins with the Stable Manifold Injection Manoeuvre (SMIM) and ends with the Halo Injection Manoeuvre (HIM); moreover, two trajectory correction manoeuvres are also expected. In the time between these manoeuvres, the core tasks will be the autonomous navigation and the communication with the Mothership.

- **Phase 3 - Operational Phase**: the main mission phase, that lasts approximatively 1 year. During Phase 3, LUMIO operations will be divided in a *Science Phase* and a *Navigation and Engineering* (Nav&Eng) *Phase*, each lasting one orbit, i.e. about 14.765 days. In fact, during Science orbit the lunar farside has the right conditions to perform flash observations, as less than half of the surface is illuminated by the Sun, while during the Engineering orbit the conditions are perfect for optical navigation purposes. Figure 1-2 illustrates the concept of LUMIO operations while in the halo orbit.
  The mission scenario for these two phases is partially detailed in [1]:
  - During **Science Phase**, the payload will observe the lunar surface at a high frequency (15 fps), in order to detect flashes caused by meteoroid impacts.
  - During **Nav&Eng Phase**, the S/C shall perform operations related to navigation and orbit maintenance. Continuous images of the full lunar disk shall be taken by the payload, to estimate the position and velocity of the CubeSat. Moreover, the S/C shall execute some S/K manoeuvres to reduce the navigation errors, which cumulate during the autonomous navigation. Finally, during Nav&Eng Phase the S/C will communicate with the Mothership for one hour per day, in case LUMIO-Moon distance is less than 75000 km [1]. This condition will be met for 60% of the operational orbit, i.e. for 16 days out of 29 [1].

- **End-of-Life Phase**: the final phase of the mission, aimed to a safe disposal of the spacecraft. This phase has not been properly designed yet; however, it will mainly consist in the de-commissioning of the systems and in a final manoeuvre, the End-of-Life manoeuvre, aimed at a safe disposal of the system. Two options have been selected: to crush the S/C on the Lunar surface or to propel it away from the Earth-Moon system; nevertheless, no precise planning has been done so far.

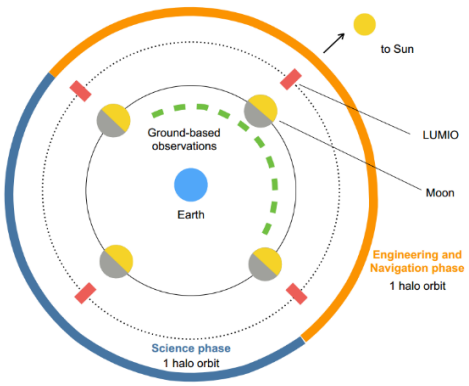

Figure 1-2: LUMIO operational concept. Credits: [1]

Once the mission scenario has bees summarised, it is paramount to describe the design of the spacecraft. In the literature review, the system was broken down into separated subsystems and units. This step is fundamental for the FMEA, as the analysis is executed for every component. The division is summarised in Table 1-1. As it can be seen, not all the units will be included in the FDIR, but only those with an active role and that can be controlled by the on-board processors. Despite most of the units are mentioned in the reference document, the list was not complete, due to the earliness of the project; hence, the deployment system for the antennas and the temperature sensors were added to the original list.

Table 1-1: list of components per subsystem

Subsystem	Component	Nr	Type	Included in FDIR
1. Payload	LUMIO-Cam	x1	Custom design	Yes
2. Payload processor	Payload processor	x1	COTS - Gomspace Nanomind Z7000	Yes
3. CDHS	On board computer	x1	COTS – AAC Microtec Sirius	Yes
4. ADCS	Reaction wheels	x3	COTS - Blue Canyon Technology - RWP-100	Yes
	Cold gas RCS thrusters	x4	COTS modified - VACCO Hybrid ADN MiPS	Yes
	Sun sensors	x2	COTS - Solar MEMS Technology nanoSSOC-D60	Yes
	Star trackers	x2	COTS - Hyperion Technology - ST-400	Yes
	Inertial Measurement Unit[1]	x1	COTS - Sensonor-STIM-300	Yes
	AOCS processor	x1	COTS - GomSpace Nanomind Z7000	Yes
5. Propulsion	Main monopropellant thruster	x1	COTS modified - VACCO Hybrid ADN MiPS	Yes
6. Power System	Solar panels	x2	COTS - AzurSpace 3G30C Solar cells COTS - GomSpace Nanopower MSP Solar arrays	Yes
	EPS	x1	COTS - GomSpace Nanopower P60	Yes
	Battery	x2	COTS - GomSpace Nanopower BPX	Yes
7. Communication	UHF Transponder	x1	COTS - LDSRSP UHF card	Yes
	RF Power Amplifier	x1	Custom design	Yes
	UHF Antenna	x2	Custom design	Yes
8. Structure	Structure	x1	COTS – ISIS B.V. 12U structure	No
	Exterior panels	x6	Custom design	No
	Internal radiation shielding	x1	Custom design	No
	Antenna deployment system[2]	x1	Not specified	Yes
	Solar panels deployment yoke	x2	Custom design	Yes
	SADA	x2	Custom design	Yes
	S/C ejection system	x1	COTS – ISIS Quadpack Deployer	No
9. Thermal	Outer S/C coating	x1	Custom design	No
	Outer solar panels coatings	x2	Custom design	No
	Heaters	x3	COTS – Telpod S.A. GBR-612 series resistor	Yes
	Temperature sensors[3]	x14	Not specified	Yes

[1] The IMU is a COTS component that comprises 3 highly accurate MEMS gyros, 3 high stability accelerometers and 3 inclinometers. The inclinometers will not be considered, as they are meant for terrestrial applications, while all the other sensors will be treated as a single unit.
[2] Addition to the original list from [1]
[3] Addition to the original list from [1]

1.3.2 FDIR methodology

In the literature review, various documents regarding FDIR were analysed. Firstly, since the design of the FDIR is an activity within the framework of Fault Management, one of the main references of the Literature Review was the *"Fault Management Handbook - Draft 2"* by NASA [4]. This document provides extensive guidelines for Fault Management of flight systems developed by NASA and presents the methodologies used in the space agency across all the project life cycle. Other relevant references were a series of lectures held by several companies and institutions during an FDIR Workshop at ESA-ESTEC [5; 6; 7], the TU Delft lecture, held in the AE4S10 MicroSat Engineering course, *"FDIR Development and Lessons Learned on Various Missions"* [8], the journal paper *"FDI(R) for satellites: How to deal with high availability and robustness in the space domain?"* [9] and the conference paper *"Innovative Fault Detection, Isolation and Recovery Strategies On-board Spacecraft: State of the Art and Research Challenges"* [10], in which the current context of FDIR techniques in the space sector is described.

Fault Detection Isolation and Recovery (FDIR) is a core system engineering activity, which starts from the beginning of the mission and ends at the decommissioning of the systems. This system activity is fundamental for the autonomous operations of a spacecraft in a safe environment. In Fault Management, FDIR is the main activity finalized to Failure Tolerance, i.e. to the mitigation of the consequences of on-board failures [4]. In fact, FDIR is dedicated to the continuous control of the spacecraft (S/C) activity, with the purpose of detecting eventual failures when they occur, isolating them, by identifying their location and recovering the satellite operations after the failure. The FDIR activity is executed by the On-Board Computer (OBC) of the satellite, or by a second processor installed on the S/C, which operates with the OBC in a master-slave relation. Hence, the work of the Thesis, which is aimed at the design of the FDIR system for LUMIO, will form part of the overall Fault Management framework for the mission.

The standard approach for FDIR is to implement an on-board algorithm, which executes a predetermined set of actions to detect, isolate and recover eventual failures; the failures to be addressed are assessed prior to the mission [10]. Apart from this standard methodology, novel techniques are described in the references, such as analytical redundancy [10] and soft-computing methods, based on the use of neural networks [10; 8]. Analytical redundancy allows identifying faults without the need for redundant hardware since it uses dynamic models of the system to identify anomalies. Despite the convenience of this method, it will not be considered for LUMIO mission at the current stage, since the earliness of the project does not allow to include the satellite dynamics in the FDIR analysis. The use of neural networks, on the other hand, would present several advantages, since it can guarantee a higher identification precision and detect faults before they happen [8]. However, the novelty and complexity of the method do not make it suited for the implementation on LUMIO CubeSat. In fact, these techniques still need to undergo further in-flight testing [10]. Therefore, the standard methodology was chosen since it has proven robustness and fulfils the common requirements of availability and autonomy [9]. Moreover, it is more suited for a CubeSat mission, in which it is paramount to strive for simplicity. As a result of this preliminary FDIR study, on the one hand, it was assessed the necessity of understanding the failure scenarios and the recovery actions for LUMIO mission. On the other hand, it was established the need for integrating the FDIR within the satellite operations, by representing the Functional Flow diagrams of the mission.

Since the design of the FDIR requires the identification of the failure scenarios of the mission, to be handled by the algorithm, another fundamental reference that was studied is the *"Space Product Assurance - Failure modes effects (and criticality) analysis (FMEA/FMECA)"* by the European Cooperation for Space Standardization (ECSS) [11]. In fact, two standard approaches to assess the potential failure scenarios are commonly used: the Failure Modes and Effect Analysis (FMEA) and the Fault Tree Analysis (FTA). The methodology of the work was based on the FMEA, described in [11], while the FTA, detailed in [12], was left as a recommendation for future phases. In fact, the FMEA allows to identify the potential failure scenarios of the mission and the consequent recovery actions to be executed by the system; thus, it is a fundamental prerequisite for the design of the FDIR and it is sufficient for the current phase of the project. Furthermore, the classification of the scenarios, based on their severity level, is fundamental for fault management

purposes. It should be noticed that the FMEA shall not be executed only once during a project, but it shall be updated and detailed across the different project phases, to reflect and support the design changes. Thus, being LUMIO project in an early stage, the level of detail of the FMEA will be coherent with the current state-of-the-art of the design. During the Literature Study, the first steps of the FMEA were performed: for each component of LUMIO, a set of potential failure scenarios was derived.

1.3.3 Failure scenarios

Following the methodology to perform the FMEA, a significant number of potential failure scenarios for LUMIO were identified in the literature review, for each of the components listed in Table 1-1. The total amount of scenarios is approximatively 100; thus, they will not be documented here, as they will be discussed in Chapter 2. However, it is worth mentioning the methodology that was used to determine the possible failures.

Firstly, in the references [13; 14], some common failure scenarios for sensors and actuators are described and have been used in the present analysis. Regarding the actuators, common failure scenarios include locked output, hard-over (upper/lower saturation level reached) and loss of effectiveness. Figure 1-3, shows the effects of these failures.

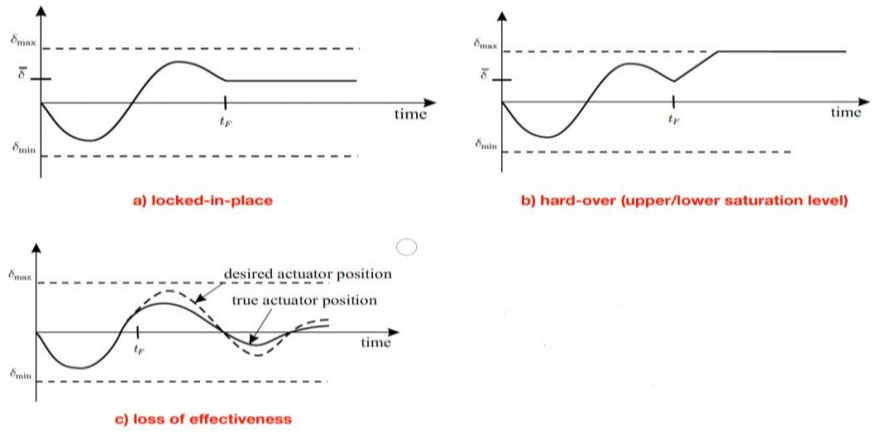

Figure 1-3: common actuator faults. Credits: [13]

Secondly, common sensor failures are sensor bias (a constant error in the measurement), loss of accuracy, sensor drift (growing error) and frozen sensor. These sensor faults are illustrated in Figure 1-4. These examples have been used to model the failure scenarios of all the sensors in LUMIO CubeSat. Apart from these common failures, other scenarios were derived from examples taken from reference documents [8; 11] or reasonable assumptions. For each scenario, a set of possible recovery actions was also designed.

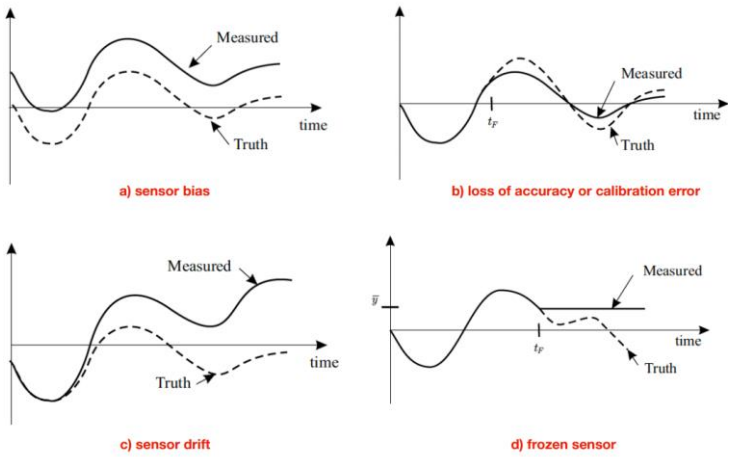

Figure 1-4: common sensors faults. Credits: [13]

1.3.4 Safe Configuration and Collision Avoidance Manoeuvre

In [1], the Mission Phases that were described in Section 1.3.2 are broken down into Mission Modes that refer to a specific task executed by the S/C. Since this division is not relevant for this Thesis, it will not be treated in detail. However, in the literature review, two additional modes were added, due to their importance for the FDIR: Safe Configuration and Collision Avoidance Manoeuvre (CAM). Hence, it is fundamental to describe how these modes were designed.

As described in NASA: Fault Management Handbook, **Safe Configuration** is an important mission mode, that is not part of the nominal behaviour of the system, but it is triggered by FDIR to guarantee the safety of the satellite and the completion of the mission, when a detected failure is not directly resolved by the application of other recovery levels [4]. In case of satellites that operate in an Earth orbit and with a good ground coverage, Safe Configuration shall ensure for a long period of time (up to 1 or 2 months) that the S/C can orbit without any collision risk and without needing any correction manoeuvre, and at the same time the communication with ground shall be maximized. Most of the subsystems will be on idle, apart from those with critical functions, like communication, thermal control and on-board computer. A similar description of Safe Configuration is presented in [15], with more specific focus on OBC functionalities.

The previous characterization of Safe Configuration must be modified for LUMIO mission, since the CubeSat does not operate in an Earth orbit, but in a Lunar-orbit, and the direct communication with ground is not foreseen. The first expected action during Safe Configuration shall be to put the CubeSat in a *Safe Orbit*, which means both that it will not deviate excessively from the nominal mission and it will be safe from collisions with other Near-Earth Objects (NEO). While in Safe Orbit, the satellite will not need to perform any correction manoeuvre for a long period, although inferior to an Earth-orbit mission, e.g. 2 weeks: in fact, from the mission implementation analysis done in [1], a S/K manoeuvre is foreseen at least each 14 days, during the Halo Orbit. Placing the S/C in a Safe Orbit allows the ground control to have the time required to formulate and send a response to the failure. The Safe Orbit for Phase 1 and 3 of LUMIO mission has not been designed yet, but this must be done in the next phases of the project. During Phase 2 (Transfer Phase), instead, there will be no Safe Orbit: achieving safe configuration during transfer will not be possible. Hence, in case of a failure during this phase, the S/C must perform additional correction manoeuvres, which must be pre-calculated by the orbit design team. Finally, the communication with the Mothership shall be maximized during Safe Configuration: the S/C shall continuously check for newly received packets that might contain failure responses from ground and transmit packets with the unsolved failure to the Mothership.

Another mode which is fundamental for FDIR purposes, and hence needs to be defined, is the **Collision Avoidance Manoeuvre**. When LUMIO CubeSat drifts from its nominal path, due e.g. to internal failure, there might be the risk of collision with another NEO. If this were to happen, the consequences would be fatal for both the objects; hence, there is the necessity to constantly check the risk of collision, and to perform a manoeuvre if needed. This is the Collision Avoidance

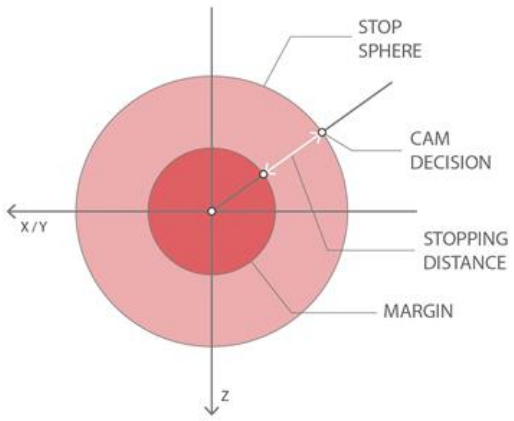

Figure 1-5: CAM and Stop-Sphere

Manoeuvre, which has the purpose to minimize both the collision risk and the ΔV required to change trajectory to a safe configuration [16]. The main risk in LUMIO mission is the collision with the Mothership, due to the proximity between them during the initial phases. During the Operational Phase, on the other hand, the distance between the two bodies will decrease the risk. A common approach is to define a *Stop-Sphere* as the minimum allowed distance between LUMIO CubeSat and the Lunar Orbiter: this is necessary due to the uncertainties in the estimation of the position of the two bodies. Therefore, to avoid a collision, the S/C must not cross the stop-sphere: this sphere represents the last possible position in which the CAM can be performed, in order to avoid a collision. Figure 1-5 represents the execution of CAM and the concept of Stop-Sphere.

Hence, it will be fundamental to know, for each mission step, the precise position of LUMIO CubeSat with respect to the Lunar Orbiter, so the most critical mission steps can be identified and, if a failure happens, collision avoidance measures can be applied. However, at this stage of the project, there is no sufficient information to do that: such a task will be carried out with the development of an orbital propagator, that will calculate the location of the S/C and all the other NEOs, based on the step-by-step mission description [8]. At this stage it is possible to recognise that the Mission Phases, described in Section 2.4, are characterised by different levels of collision risks:

- During Phase 1 there is a high risk of collision in case of anomalies, especially with the Mothership.
- During Phase 2, in case of wrong thrusting direction during the transfer manoeuvres, that requires a high level of total impulse, there is the risk of collision with the Lunar Orbiter or with other NEOs, especially in case the thrust direction is pointing towards the Earth, instead of pointing in the right direction.
- During Phase 3 the S/C is in a Halo Orbit, which has a low collision risk, as it is far from the mothership and most of the NEOs.
- Phase 4 is not sufficiently detailed yet.

The risk of collisions with other NEOs in not only checked using the orbital propagator but also by monitoring continuously the S/C, with the aid of the on-board autonomous navigation tools. These tools comprise the optical navigation, with the use of LUMIO-Cam, and the inter-satellite link, between LUMIO and the Lunar Orbiter. A cross-check between all these data is necessary, to avoid the detection of false risks. The CAM shall be performed the earliest possible when the risk of collision is detected, as the stop sphere is the last chance to perform this manoeuvre. The manoeuvre itself cannot be designed in advance, because the decision is based on the calculated probability of collision and the available ΔV capability of the spacecraft [16]. What needs to be implemented is the on-board algorithm, which must check the risk of collision and compute the manoeuvre, when necessary. This will be a task of the OBC, which will detect eventual risks.

1.4 Thesis outline

This report is divided into multiple chapters, which follow the methodology chosen for the development of the FDIR algorithm. Chapter 2 lays down the theoretical background: the first step of the FDIR design is the identification of the failure scenarios to tackle, which are categorized according to their criticality in the FMECA table. Moreover, the preliminary Functional Flow diagrams for LUMIO mission are drawn, with the objective of identifying the interconnections of the FDIR with the operational modes of the satellite. Finally, the high-level architecture of the FDIR is presented.

Chapter 3 is dedicated to the design of the FDI, the part of the algorithm aimed at detecting and isolating eventual failures. The FDI high-level architecture is described, with the division of the S/C into different modules. On top of this, the methodologies that are used for the detection and isolation are discussed. Later, the FDI logic for each module is explained in detail, as well as the implementation on Simulink.

Chapter 4 deals with the second part of the algorithm, the FR, which is aimed at producing the necessary recovery actions in case a failure has been detected. Firstly, the general strategy used for failure recovery is presented. Later, the logic of each module is explained, along with its implementation on Simulink.

After the model for the FDIR has been developed, the final part of this Thesis is dedicated to testing the Simulink model, to verify its correct functioning and find eventual flaws. Therefore, Chapter 5 is dedicated to the description of the model that was developed to simulate the failure scenarios of the satellite, while the results of the simulations are presented in Chapter 6.

Finally, in Chapter 7 conclusions and recommendations are made for this project.

2 Theoretical background

In this chapter, the preliminary work necessary for the design of the FDIR is presented. Firstly, the Failure Modes Effects (and Criticality) Analysis is performed, leading to the identification of the failure scenarios to be handled by the FDIR. Later, the Functional Flow diagrams of LUMIO operations are drawn, to perform a preliminary integration of the FDIR within the satellite operations. Finally, the high-level architecture of the FDIR is presented, paving the way for the design of the FDI and the FR architectures in the next chapters.

2.1 Failure Modes Effects (and Criticality) Analysis – Summary

The first step for the design of the FDIR is the identification of the possible failure scenarios, which are listed and categorized through two main types of analysis: the Failure Modes and Effect Analysis (FMEA) and the Failure Modes, Effect and Criticality Analysis (FMECA).

2.1.1 Failure scenarios selection

As it was mentioned in Section 1.3.3, in the literature review, several potential failure scenarios for LUMIO were identified, for all the components listed in Table 1-1. These scenarios represent the baseline for the FMEA and the FMECA; however, it was necessary to perform a thorough review of the list, in order to distinguish between the failures that can be treated at the current stage, and those to discard. A discrete number of scenarios was discarded at this stage; they are documented in Table A-1 in Appendix A since they might represent a useful input for the future development of the FDIR. The remaining failures will be used as a baseline for the FMEA and the FMECA.

Due to the vast number of scenarios, it is unreasonable to discuss the rationale behind the removal of each of them; however, some general guidelines can be identified. Firstly, the scenarios regarding software issues were not considered, since the design of the on-board software is not detailed enough yet; thus, there was no sufficient information to design a detection and isolation methodology for these failures.

Another important factor was already mentioned in Section 1.3.2: at the current stage, it was decided to ignore the satellite dynamics, which include all the different orbits and manoeuvres of the S/C during the mission. The choice to exclude the dynamics from the analysis is mainly due to the complexity of the topic since the orbit designed is based on advanced astrodynamics calculations. Moreover, the project is still in an early-stage, therefore the S/C composition is not finalised yet and there is not a precise orbit design that can be used as a baseline for the FDIR analysis. Eventual changes regarding the selected orbit might intervene, in the future. The dynamics and the orbit design are tasks of the Politecnico di Milano, which will work on them between Phase A and Phase B, from October 2019 [1]. Hence, all the scenarios related to navigation and/or dynamics were not considered but are expected to be added in the future phases of the project. This includes also those scenarios of actuators, e.g. reaction wheels, which require the dynamic for correct detection.

Finally, another category of failures that was changed is sensors failures. As it is shown in Figure 1-4, several types of scenarios were considered for each sensor, during the literature review. However, most of the failures were discarded at this stage, due to the difficulty in correctly distinguish them; thus, the scenarios of loss of accuracy, sensor bias and sensor drift are not considered. However, they are grouped under the name of “general failures”, allowing their detection but not their classification. The advantage of this choice is twofold. On the one hand, the FDIR algorithm is simplified; on the other hand, these failures can be detected and recovered even in those cases in which it would be difficult to correctly identify them. The only scenario that was kept from the list of Figure 1-4 is the frozen sensor, whose identification is considered feasible.

2.1.2 FMEA

The FMEA, according to the ECSS standards, is an integral part of the design process, and as such it "shall be initiated for each design phase and updated to reflect design changes along the project life cycle" [11]. Thus, being LUMIO project in an early stage, the level of detail of the FMEA will be coherent with the current state-of-the-art of the design. The results of this analysis are not only useful to design the FDIR algorithm but also to develop the product architecture, the test and operations procedures and the maintenance actions, as it allows to identify the critical items and functionalities of the system.

The reference [11] indicates the steps required to perform the FMEA, which were followed in the literature review: the detailed examination of the system (functional descriptions, operational modes, mission phases), the identification of possible design corrections and, finally, the identification of potential failures, along with the detection method and the corrective actions. The final steps of the analysis are the classification of the failures according to their severity, and the documentation of the results in a final report, in conformity with the standards.

The severity of a failure shall be applied following the levels indicated in Table 2-1, taken from the reference document [11]. The criteria for loss of mission and mission degradation must be defined by the customer; however, in the case of this study, this is not possible. Therefore, the classification will be arbitrary; the proposed severity levels are the followings:

- Severity level **1 - Catastrophic** shall be applied in case of Loss of System.
- Severity level **2 - Critical** shall be applied in case there is Loss of Mission, i.e. the system is safe but cannot pursue the scientific objectives (perform flash detection on the lunar farside from the operational orbit).
- Severity level **3 - Major** shall be applied in case of major mission degradation, e.g. the satellite is able to pursue the scientific objectives partially. An example is a failure that causes the mission lifetime to be significantly reduced, e.g. reduction of 30-90% of overall science returns [2].
- Severity level **4 – Minor** or **Negligible** shall be applied in case of minor mission degradation, e.g. some days without performing observations.

Hence, after selecting which failure scenarios to neglect in this phase, see Section 2.1.1, the remaining ones are collected in the FMEA table. For each scenario, a brief description of the following is provided in the table: the interested item, the failure cause, the failure effects, the

Table 2-1: severity of consequences. Credits: [11]

Severity category	Severity level	Description of consequences (failure effects)	
		Dependability effects (as specified in ECSS-Q-ST-30)	Safety effects (as specified in ECSS-Q-ST-40)
Catastrophic	1	Failure propagation (refer to 4.2c)	Loss of life, life-threatening or permanently disabling injury or occupational illness.
			Loss of an interfacing manned flight system.
			Severe detrimental environmental effects.
			Loss of launch site facilities.
			Loss of system.
Critical	2	Loss of mission	Temporarily disabling but not life-threatening injury, or temporary occupational illness.
			Major detrimental environmental effects.
			Major damage to public or private properties.
			Major damage to interfacing flight systems.
			Major damage to ground facilities.
Major	3	Major mission degradation	
Minor or Negligible	4	Minor mission degradation or any other effect	

failure severity, the detection methods and the compensating provisions. These latter points are particularly relevant in the context of the FDIR design: the detection method allows to define an FDI architecture, while the compensating provision is the base for the design of the FR. Thus, a detailed description of the detection methodologies and the recovery strategies is provided in Chapter 3 and in Chapter 4, respectively.

The Failure Modes and Effects Analysis for LUMIO has been performed; however, since this analysis is the baseline for the Failure Modes, Effects and Criticality Analysis, this latter one shall be discussed, before presenting the final results.

2.1.3 FMECA

The FMECA is another fundamental analysis recommended by ECSS in [11]. The FMECA requires, in addition to the severity level, the evaluation of the criticality of the failures, which shall be assessed from the combination of the Severity Number (SN) and the Probability Number (PN). The Severity Number is opposite of the severity level, as indicated in Table 2-2. The same table shows how to compute the Probability Number from the probability of the failure, but the selection of PNs can be tailored to each specific mission: in [11], a PN of 4 is recommended for a "probable" failure, PN 3 for "occasional" failures, PN 2 for "remote" and PN 1 for "extremely remote". From the SN and PN, the criticality matrix of Table 2-2 can be used to derive the Criticality Number (CN).

Table 2-2: Criticality matrix. The Criticality Number (CN) is derived from the Severity Number (SN) and the Probability Number (PN). Credits: [11]

Severity category	SNs	Probability level			
		10^{-5}	10^{-3}	10^{-1}	1
		PNs			
		1	2	3	4
catastrophic	4	4	8	12	16
critical	3	3	6	9	12
major	2	2	4	6	8
negligible	1	1	2	3	4

The detailed knowledge of the probability of a failure is a demanding task, which requires thorough testing, especially in the case of custom design. If Commercial-Off-The-Shelf (COTS) components are used, on the other hand, it is likely that information on typical failure rates is provided directly by the supplier. Since LUMIO project is in an early stage (Phase A), it is not possible to determine the probability level of the selected failure scenarios. However, a preliminary criticality analysis can be performed, through the definition of PNs based on statistical considerations. The baseline for this project is the study "*A study of on-orbit spacecraft failures*" by S. Tafazoli. [17], in which a vast number of satellite failures occurred between 1985 and 2005 was analysed and categorized.

In the reference, several results are documented. Firstly, the failures are categorized per subsystem; later, per type. Finally, a more detailed analysis is executed on the most critical sub-systems. The categorization per type will not be considered in this study: almost all the failure scenarios of LUMIO considered at this stage are either mechanical or electrical and according to [17] their probabilities are similar, despite a slight prevalence of electrical causes (electrical 45% and mechanical 32%). Thus, the result that is deemed relevant for this project is the division of the failure rates into subsystems and components; Figure 2-1 summarises the findings in [17]. In the diagrams, only the failure rates of components relevant for this Thesis are included; the remaining units are grouped as "Other".

From diagram A in Figure 2-1, it can be seen that the subsystems with the highest failure rates are the Power system and the Attitude and Orbit Control System (AOCS), which comprises the Attitude Determination and Control System (ADCS) and the Guidance Navigation and Control (GNC). This result could be expected since the Power system is particularly critical and subjected to harm, especially the solar panels, while the AOCS includes a vast number of components. The Command and Data Handling (CDH) and the Telemetry, Tracking and Command (TTC) are also

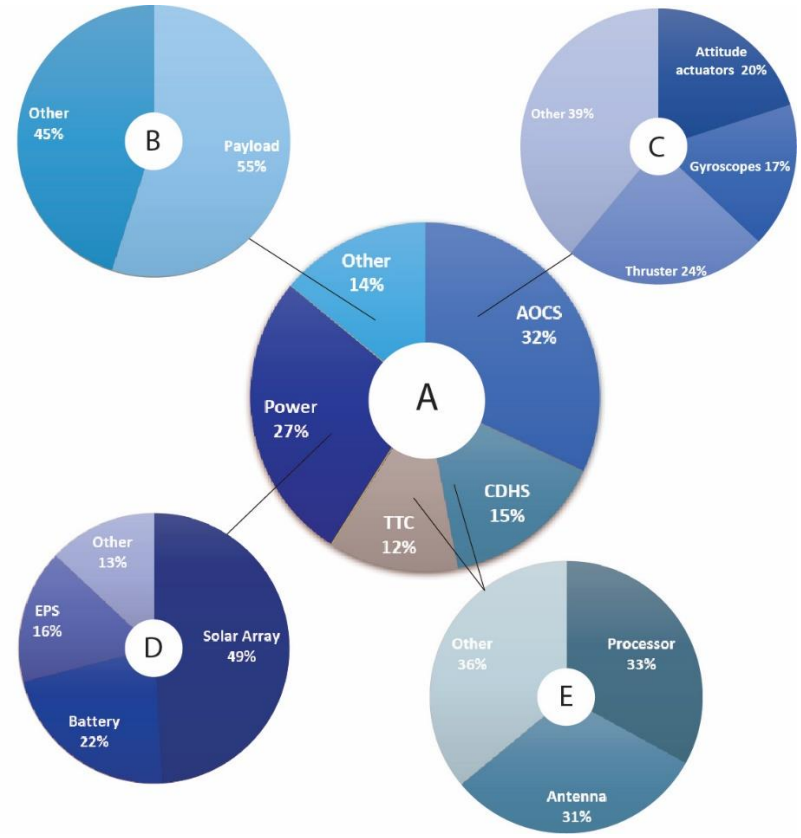

Figure 2-1: statistics of satellite failure rates. In diagram A the failure rates are divided per subsystems, in diagrams B to E the subsystems are broken down into components

responsible for a consistent percentage of failures. In this study, these latter systems were called CDHS and Communication, see Table 1-1. Finally, the Structures and the Payload are grouped inside "Other".

A piece of additional information from [17] was used: only 22% of the AOCS failures occur within the first year of operations, while this percentage for the other subsystems is higher, around 50%. This information is important since LUMIO lifetime is expected to be around 1 year.

Therefore, from the diagrams of Figure 2-1, it is possible to draw some preliminary conclusions about the PNs for the different components. The solar panels are considered the components with the highest probability of failure; thus, they are assigned a PN of 4, the highest. It should be noticed that in [17], the failures of solar arrays deployment and of the SADA were included in this category. Most of the other components have similar failure rates and therefore are assigned PNs of 3 or 2. The selection at this phase is made arbitrarily and it is based also on the technology readiness level (TRL). Units that are (partially) custom, as the propulsion system, the antennas, the payload are associated to a higher risk, due to the lower TRL; hence, they are given a PN of 3. On the other hand, a PN of 2 is assigned to COTS units, which have higher TRL. Some remarks should be made on those units that were not mentioned in [17]. Since no relevant failure rates for the attitude determination sensors (Sun sensors and star-trackers) was found in the reference [17], the lowest probability number is assigned to these components. The heaters and temperature sensors are not mentioned either; however, the thermal failures were considered as mechanical failures in [17] and therefore have a minimal failure rate. Thus, an arbitrary PN of 1 has been assigned to them. In Table 2-3, the selection of PNs is summarized.

The analysis did so far allows to define a preliminary PN value for the components of LUMIO; thus, allowing to expand the FMEA into an FMECA, by calculating the criticality number as shown in Table 2-2. However, some limitations of the methodology shall be addressed. Firstly, the PN that was chosen does not depend on the probability of a failure, but on the failure rate of a component;

Table 2-3: preliminary selection of Probability Numbers used in the FMECA

Probability Number (PN)	Components
1	Sun sensors, star-trackers, EPS, heaters, temperature sensors
2	Processors (all), antennas, batteries, RWs, IMU
3	Payload, antennas, antenna deployment system, thrusters (all)
4	Solar panels, SADA, Solar panels deployment yoke

hence, despite different types of failures might have different probabilities to occur, they will be treated the same. Some arbitrary exceptions were made; for instance, a lower PN was assigned to underheating failures, since the preliminary thermal simulations done in [1] did not predict particular underheating risks. Moreover, the survey in the reference document was based on relatively old satellite missions; the technology advancements might have a significant impact on the results. Despite these limitations, the analysis performed is considered sufficiently accurate for the current stage of the project, and a good starting point for a more detailed FMECA in the next phases.

2.1.4 Results

Since the FMECA worksheet is an expansion of the FMEA, with the inclusion of the PN and the calculation of the CN, only the FMECA table will be documented in this project. Due to the length of the document, it will be included in Appendix B (Table B-1) and summarized in this section.

The use of the FMECA is manifold. In the context of FDIR, it represents a classification of the failure scenarios to be treated and an analysis of the effects and the potential compensatory provisions. Hence, it supports the design of the FDIR and provides inputs for the implementation of changes in future versions. From the point of view of fault management, the FMECA allows to identify the criticalities of the S/C design and contributes to design changes in the next phase. Moreover, the FMECA provides input for the definition of tests, in the more advanced phases of the project. It should be remembered that the FMEA and the FMECA shall be updated at every phase of the project, allowing to keep track of the impact of design changes on system safety and to update the critical items list (CIL), the fault tree analysis and the FDIR system consequently [11].

At the current phase, a preliminary FMECA was performed; nevertheless, some interesting results emerge. In particular, a first version of the critical items list can be done, from the criticality matrix shown in Table 2-4, where the scenarios ID indicated in the FMECA are used.

The scenarios with a CN higher than 9 (circled numbers in Table 2-4) are considered the most critical. As it could be expected from Figure 2-1, the failures of solar panels, SADA and solar panels deployment system are in this group, due to their high probability. On top this, failures of the camera, the propulsion system and the antennas are also highly critical, due to their high severity: a failure of the camera or the propulsion system would irremediably affect the navigation of the S/C, while the failure of the uplink antenna would cause the definitive loss of contact with the satellite. Hence, these units are single-point failures (SPF). The failure of a unit is considered an SPF if it causes the whole system to fail [4]; thus, an SPF has the highest severity number.

Other critical design points are the OBPDP and the IMU, which have at least one scenario with CN equal to 8; the IMU can be considered SPFs as well, due to the absence of available redundancies, while the payload processor can be potentially substituted by the other on-board

Table 2-4: criticality matrix resulting from the current version of the FMECA for LUMIO mission

	PNs			
SNs	1	2	3	4
4	EPS.1 ④	OBPDP.2, OBPDP.3, IMU.1, IMU.2, IMU.3 ⑧	CAM.1, CAM.2, CAM.3, CAM.4, CAM.7, GT.1, MT.2, MT.3, MT.4, MT.5, MT.8 ⑫	⑯
3	DEP.1, EPS.3 ③	BT.3, BT.4, BT.5 ⑥	CAM.5, CAM.6, DEP.2, DEP.3, COMM.1 ⑨	SADA.2, SP.2, DEP.4, DEP.5 ⑫
2	ST.1, ST.2, ST.3, ST.4, ST.5, ST.6, MT.7 ②	OBC.2, OBC.5, RW.1, RW.2, RW.3, RW.4, RW.5, RW.6, RW.7, AOCS.2, AOCS.3, BT.1, BT.2 ④	GT.3, GT.4, GT.5, GT.6, MT.6, COMM.2 ⑥	⑧
1	OBC.4, OBC.6, OBC.7, SS.1, SS.2, SS.3, SS.4, EPS.2, HEAT.1, HEAT.2, TSENS.1, TSENS.2 ①	OBPDP.1, OBC.1, OBC.3, AOCS.1 ②	GT.2, MT.1 ③	SP.1, SADA.1 ④

processors, e.g. the OBC. Hence, the mitigation of the risks created by these scenarios represents an additional starting point for future design changes.

In conclusion, the preliminary FMECA produced significant results that can be the baseline for the future phases of the project. It will be crucial to reduce the criticality of the mission; hence, attention should be given in the design and testing of the most critical units. Different types of interventions can be made. In order to reduce the PN of a failure, the reliability of the unit shall be increased, through the addition of design margins and a thorough proper testing. To reduce the severity of a failure, the best approach is to add a redundancy; this can be achieved with the addition of a redundant unit, which is recommended for the RWs, the IMU and the propulsion system, or a redundant function, e.g. a back-up navigation system to ensure navigation in case of Camera failure. The most suitable option for the navigation is the implementation of ground-based radiometric tracking, which is a commonly used satellite navigation technique [1]. For example, it is suggested to consider the addition of direct communication with ESA's Tracking Network (ESTRACK), which relies on Doppler, ranging and Delta-Differential One-Way Ranging (ΔDOR) techniques to achieve high tracking accuracy [18]. Another possible option is the addition of laser corner reflectors to enable ground-based laser tracking, as it was proposed in [19] for CUTIE, a deep space CubeSat mission to the Moon. The design changes, however, will need to be traded off against the constraints of the project (time, budget, mass and volume) and certain risks might be accepted by the team, as it is intrinsic in a CubeSat mission.

2.2 LUMIO mission Functional Flow

It is fundamental to understand how the FDIR can be integrated within the operations of LUMIO. In Section 1.3.1, the mission breakdown in phases is described; however, in reference [1] a detailed presentation of the operations is missing. Therefore, in the Literature Study, a preliminary version of the Functional Flow diagrams of LUMIO mission was proposed. In this section, an updated version of the Flow Diagrams will be presented, with the integration of the FDIR, where possible. The focus of the analysis is not the definition of a precise sequence of operations, which is not feasible due to the earliness of the project, but the study of when and how the FDIR can be efficiently integrated within the operational modes.

Each Mission Phase was divided into different Mission Modes. Therefore, diagrams for Phase 0+1, 2 and 3 were produced. The first two phases were incorporated, while the End-of-Mission is excluded, for the little relevance in the context of FDIR and for the uncertainties in its design. For each Phase of the mission, two kinds of Functional Flow diagrams were produced:

- One **high-level** diagram, which represents the sequence of the Operational Modes of the satellite during that phase;
- Several **detailed level** diagrams, which describe with a simple flow what happens during each Operational Mode.

2.2.1 Phase 0+1

The high-level diagram for Phase 0 and 1 is represented in Figure 2-2. It is noticeable that the Operational Modes used in the Functional Flows slightly differ from the Mission Modes used in LUMIO mission report [1] since the former are modes of the satellite, while the latter are modes of the mission; however, the operations performed by the system are the same. The satellite Operational Modes used in the Functional Flow diagrams are the following:

- **Activation Mode**: the main purpose of this mode is to activate the satellite when ejected from the Mothership or after a reset. A health-check is performed at the following subsystems: EPS, OBC, and Communication. Transmission of the antenna is off until

deployment is validated as successful. The deployment mode shall be activated only after a fixed amount of time, e.g. 30 minutes.

- **Deployment Mode**: the first goal of this mode is to determine if the deployment of appendages occurred or not. If not, solar array and antennas are deployed, with a max number of attempts.

  Safe Mode: The purpose of this mode is to have limited subsystems on (OBC and EPS), which consume little power. The satellite enters in this mode in the first activation and after a reset (boot loop) to check the status of all subsystems. If everything works correctly, the system just passes through this mode and gets out immediately. The satellite goes through this mode also in case:
  - a failure occurs;
  - the battery voltage levels are above/below the given threshold.

  In order to get out of this mode, the satellite needs to satisfy the 2 conditions (no failure, expected voltage parameters within range). This mode should also be fully reprogrammable from ground so that certain conditions can be bypassed in order to increase the chances of mission success.

- **ADCS Mode**: the purpose of this mode is the acquisition of the S/C orientation and rotational speed, and execution of de-tumbling or de-saturation manoeuvres if necessary (rotational speed higher than de-tumble parameter or reaction wheels rotational speed higher than de-saturation parameter).

- **Parking Cruise Mode**: the objective is to perform optical navigation during the parking orbit, which includes actuating Station Keeping manoeuvres and sending data to the Mothership during the communication windows. The parking cruise mode lasts until the mission proceeds with Phase 2 (Transfer Mode), except for periods in which the de-saturation manoeuvres will be needed.

- **OFF**: it not a proper Operational Mode. The satellite is turned off until the mechanical switch (kill switch) gets un-pressed and the satellite gets powered. Moreover, the analogue protection system will turn OFF the satellite if the voltage drops below a certain threshold.

The boot sequence represented in Figure 2-2 explains what happens to the satellite during its first activation, but the same sequence of operation is performed when a system reset occurs. In the following pages, the detailed level flow diagrams for the Operational Modes of Phase 0 and 1 are represented. In Figure 2-3 the diagram for Activation Mode is represented. Figure 2-4 is dedicated to the diagram of Deployment Mode, while Figure 2-5 and Figure 2-6 show the Safe Mode and a particular FDIR operation that is triggered when the battery level in Safe Mode does not rise for too long. Finally, Figure 2-7, Figure 2-8 and Figure 2-9 describe the flow of ADCS Mode and Parking Cruise Mode.

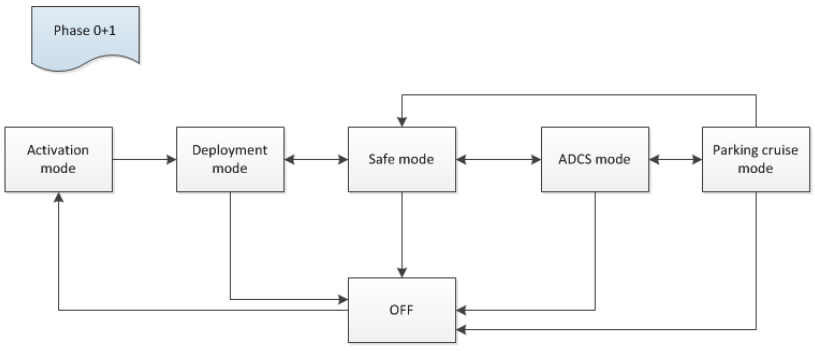

Figure 2-2: high-level Functional Flow diagram for Phase 0 and Phase 1

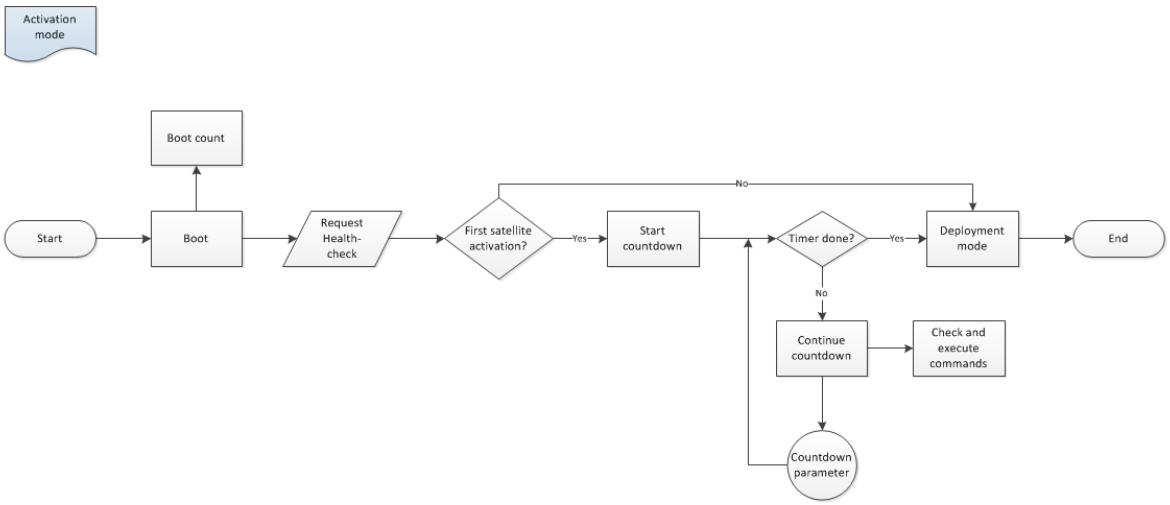

Figure 2-3: detailed level Functional flow diagram of Activation Mode

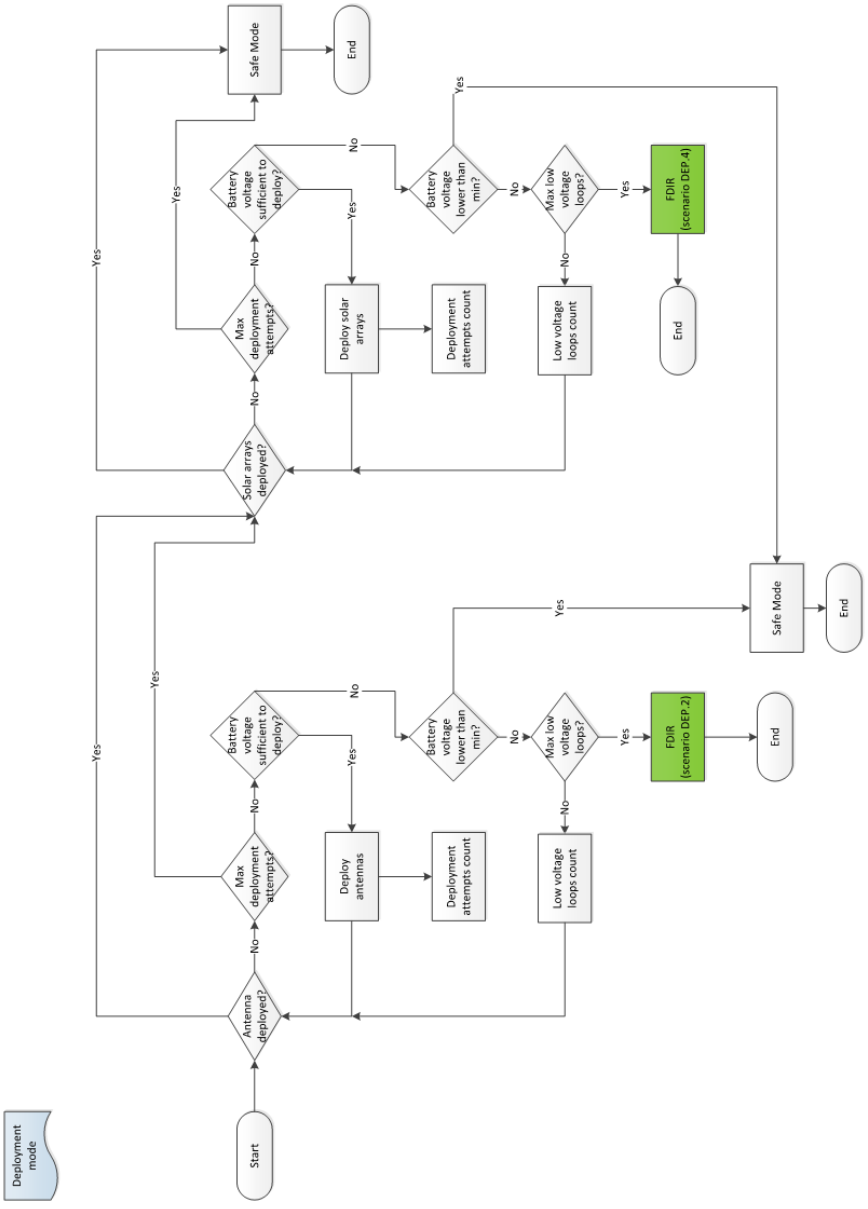

Figure 2-4: detailed level Functional flow diagram of Deployment Mode

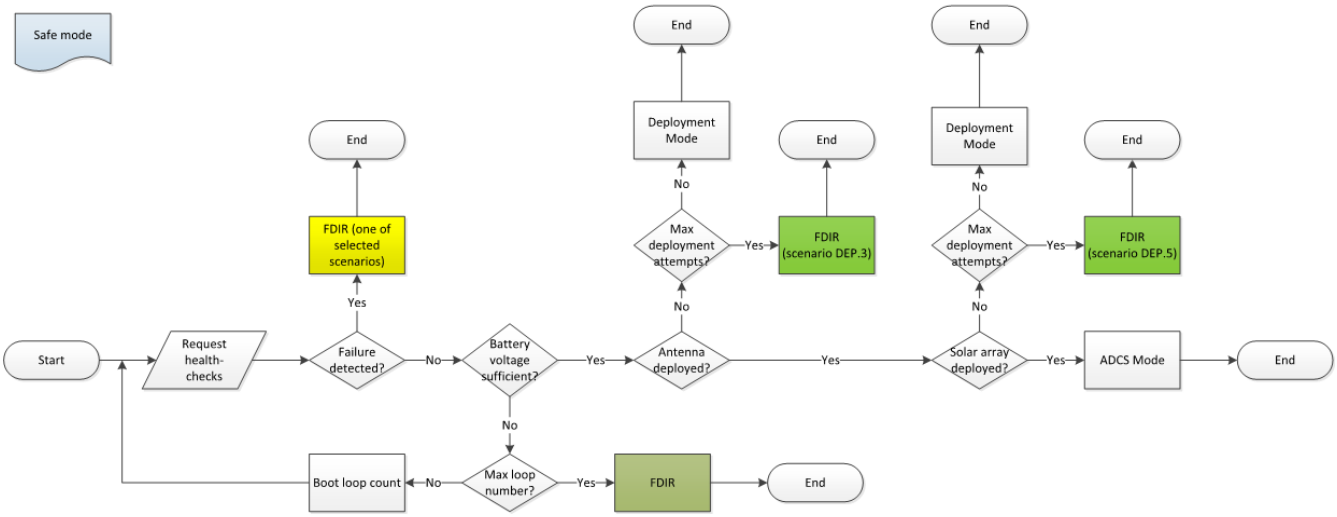

Figure 2-5: detailed level Functional flow diagram of Safe Mode

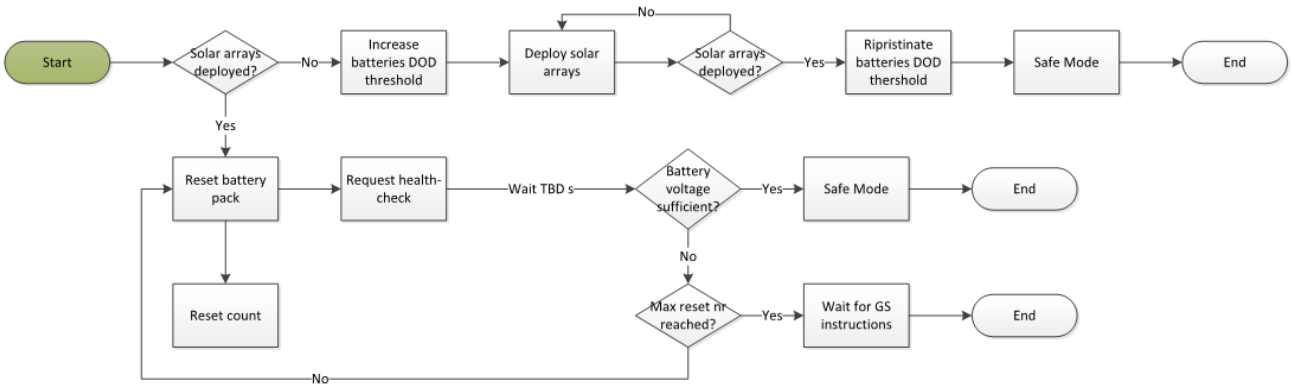

Figure 2-6: detailed level Functional flow diagram of FDIR when battery level in Safe Mode does not rise

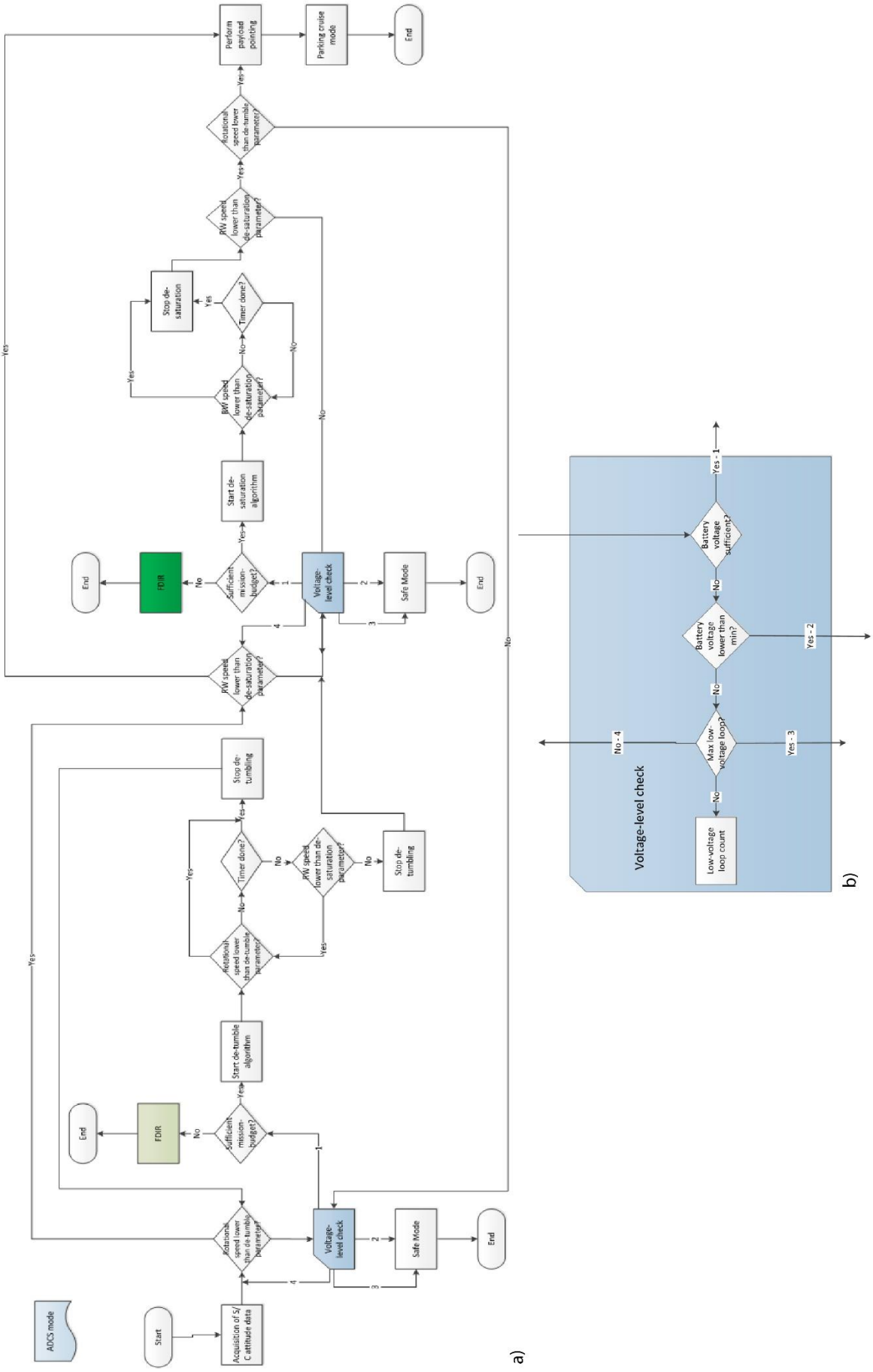

Figure 2-7: detailed level Functional flow diagram of ADCS mode

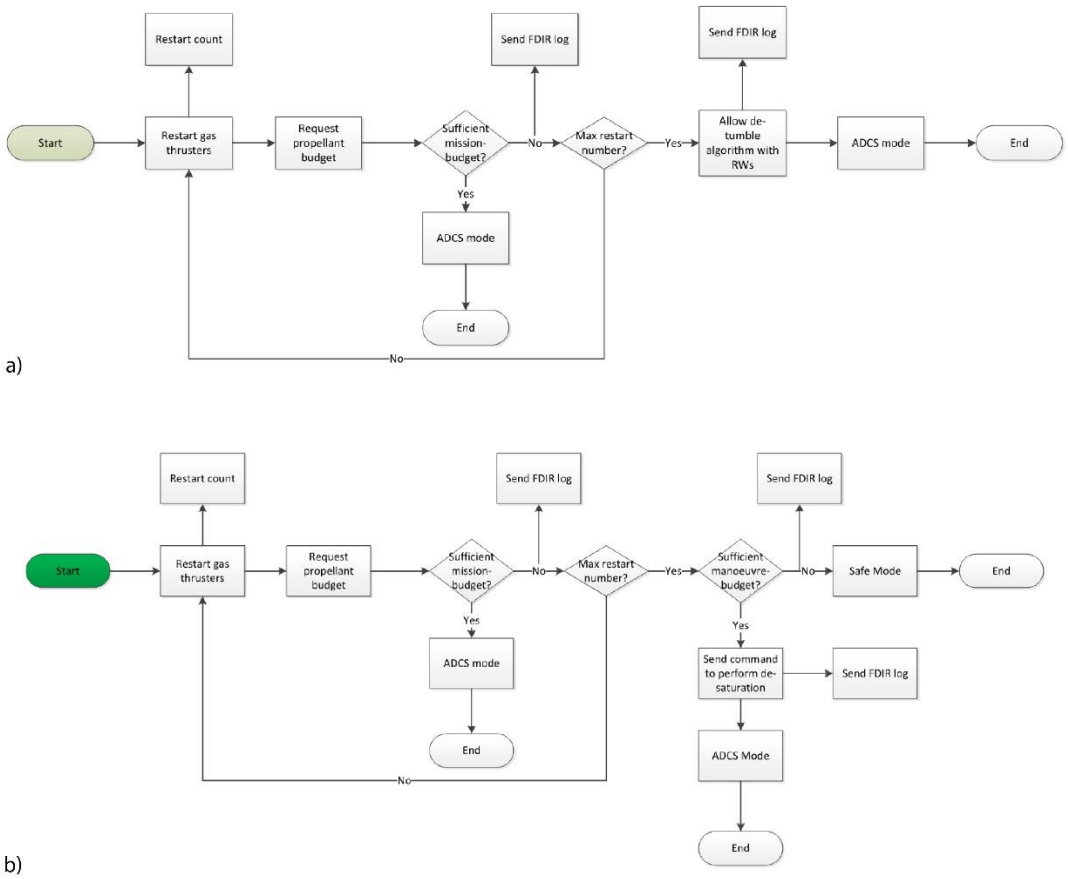

Figure 2-8: flow diagram of the FDIR operations when the propellant budget is not sufficient

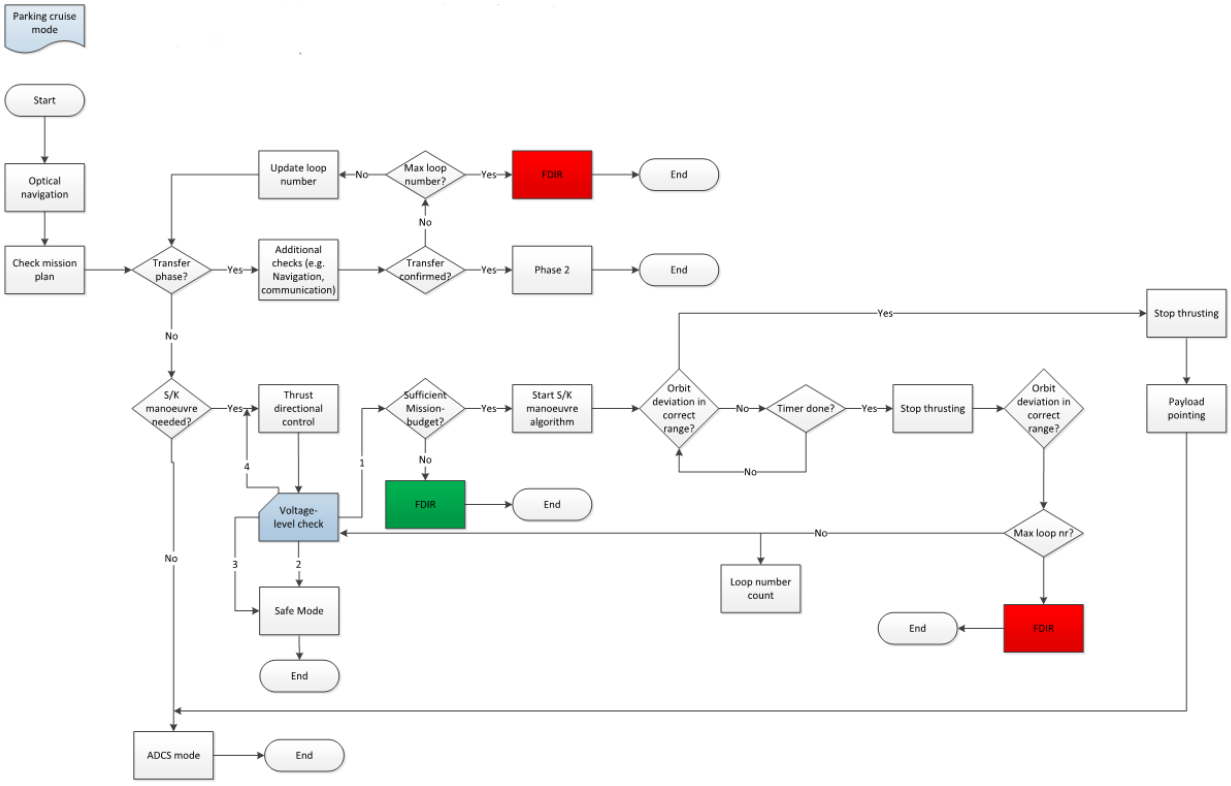

Figure 2-9: detailed Function Flow of Cruise mode

In the detailed flow diagrams, it can be noticed that the FDIR has been integrated into the sequence of operations. However, two different FDIR approaches are used, which are distinguished using different colours.

The yellow blocks are used to indicate when a scenario included in the FMECA is detected by the FDIR. The approach used is the one followed by this Thesis: the failures are checked continuously by the FDIR and can be flagged at any time. The detection and recovery methods for these failures are the object of this work and are outlined in Chapters 3 and 4. The main approach that was used to integrate the FDIR for these scenarios within the operational modes of LUMIO is the design of the Safe Mode, not to be confused with the Safe Configuration described in Section 1.3.1. When a failure occurs and is detected, Safe Mode is triggered to avoid the failure to affect the satellite operations and propagate; later, the recovery actions can be executed.

The green and red FDIR blocks, on the other hand, indicate those situations in which a failure is not detected by the FDIR directly, but it is flagged by the OBC, as part of the operation sequence. The FDIR approach in these cases is different to the one used in this Thesis: these failures cannot happen continuously, but only during specific phases; thus, the detection is made through a failure flag sent by the OBC and a peculiar recovery sequence is designed for each scenario. Hence, the definition of the operational modes of LUMIO allowed to identify new scenarios that could not be found from the analysis of the potential failures for each component of the S/C.

Amongst the new scenarios, some distinctions can be made. Many failures that were found regard the manoeuvres, both during ADCS mode (de-saturation and de-tumbling) and Cruise mode (S/K manoeuvre). Before a manoeuvre, the propellant budget is checked; a distinction is made between mission-budget, which is the expected amount of propellant needed for the rest of the mission, and manoeuvre-budget, which is the expected budget needed for the next manoeuvre. In case the propellant budget is below one or both these values, different recovery actions must be taken, shown in Figure 2-8. Despite the new scenarios mentioned above were found and a recommended FDIR strategy has been proposed, they will not be considered in this study, as they require more detailed knowledge of the S/C operation, the GNC or the S/C dynamics. Hence, they will be left as a recommendation for future phases of the FDIR development.

Another type of scenario that was identified is related to conflicting signals regarding the decision of beginning/ending a manoeuvre; however, the corresponding FDIR blocks are coloured in red, see Figure 2-9, as the knowledge of the S/C dynamics and the navigation are essential to tackle them and they cannot be treated in this stage.

Finally, the scenarios related to deployment failures are treated in this Thesis and therefore are listed in the FMECA table (scenarios DEP.2, DEP.3, DEP.4, DEP.5). As a consequence, the detection and recovery strategies for these failures are different from the rest of work performed in this project, as it will be described in Chapters 3 and 4; thus, their inclusion in the Thesis can be regarded as a baseline for the future implementation of the other scenarios in the FDIR design.

2.2.2 Phase 2

The high-level Functional Flow diagram of the Transfer Phase is represented in Figure 2-10. The Operational Modes of Phase 2 are similar to those of Phase 0 and 1:

- **Activation Mode Transfer**: the main purpose of this mode is to activate the satellite after a reset. A health-check is performed at the following subsystems: EPS, OBC, Communication, and Navigation.
- **Safe Mode Transfer:** same purpose of Safe Mode for Phase 0 and 1.
- **ADCS Mode**: same purpose of ADCS Mode for Phase 0 and 1.
- **Transfer Cruise Mode**: the objective is to perform optical navigation during the transfer, which includes actuating the 4 required manoeuvres and sending data to the Mothership

during the communication windows. The Stable Manifold Injection Manoeuvre (SMIM) and the Halo Injection Manoeuvre (HIM) are deterministic, i.e. will be executed based on the mission plan, with fixed ΔV. Additional checks based on navigation and communication were also included in the diagram. The two Trajectory Correction Manoeuvres (TCM-1 and TCM-2) will be computed by the OBC, based on the orbit deviation accumulated. The parking cruise mode lasts until the last manoeuvre, the HIM, is done, and thus Phase 3 begins. Other tasks include performing communication and RW de-saturation.

Since the Operational Modes of Phase 2 are similar to those of Phase 0+1, apart from small differences that are not relevant for the work of this Thesis, they will not be reported. The strategy for the integration of the FDIR is the same as explained above: to deal with the scenarios investigated in the FMECA, the satellite goes to Safe Mode. Extra-scenarios, related to navigation, are triggered as part of the operation sequence, but will not be covered in this study.

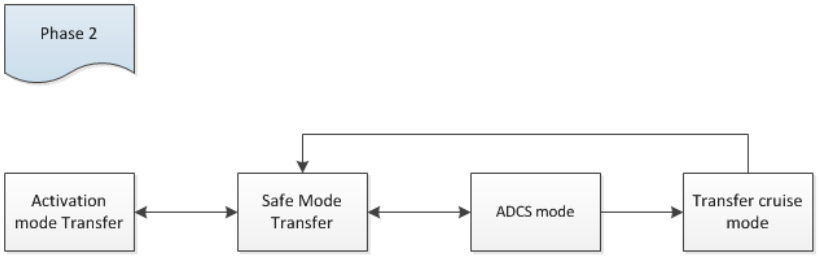

Figure 2-10: high-level Functional Flow diagram for Phase 2

2.2.3 Phase 3

The high-level Functional Flow diagram for Phase 3 (Operational Phase) is documented below, in Figure 2-11. The Operational Modes of Phase 3 are:

- **Activation Mode Operational**: same as Activation Mode of Phase 2.
- **Safe Mode**: same as Safe Mode of Phase 2.
- **ADCS Mode**: same as ADCS Mode of Phase 2.
- **Science Mode**: the purpose of this mode is to pursue the scientific goal of the mission, i.e. to perform impact observation and data processing (IODP). In the meanwhile, data is transmitted during specific communication windows, optical navigation is performed in specific periods and de-saturation manoeuvres are executed when required. This phase is activated when the Moon farside is less than 50% illuminated.
- **Navigation and Engineering Mode**: the purpose of this mode is to perform optical navigation, communication, RW de-saturation and three S/K manoeuvres. This mode is activated when the Moon farside is more than 50% illuminated.

Since the Operational Modes of Phase 3 are similar to those of Phase 0+1, apart from small differences that are not relevant for the work of this Thesis, they will not be reported. The strategy for the integration of the FDIR is the same as explained above: to deal with the scenarios investigated in the FMECA, the satellite goes to Safe Mode. Extra-scenarios, related to navigation, are triggered as part of the operation sequence, but will not be covered in this study.

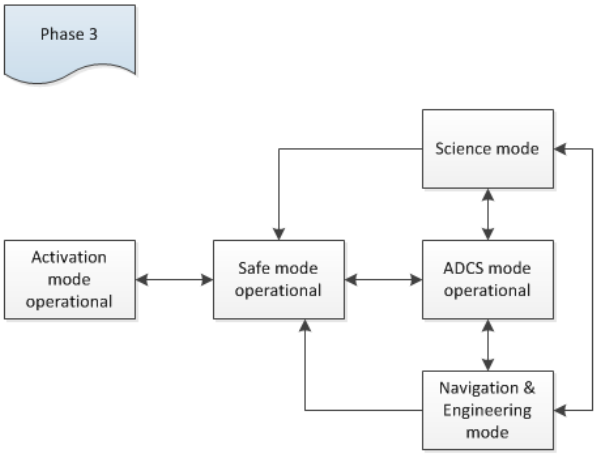

Figure 2-11: high level Functional Flown diagram for Phase 3

2.3 FDIR architecture

After the preliminary integration of the FDIR with the satellite operations was proposed, it is paramount to define also the role of the system in the architecture of the satellite. Figure 2-12 represent the high-level block diagram of LUMIO and how the FDIR is integrated within the system.

The FDIR algorithm will be run by one on-board processor; at the current stage, it is not known if the OBC will execute the FDIR or another smaller processor will be added for this purpose. In any case, the OBC and the FDIR will probably operate in a master-salve relation during the nominal mission, since the commands from the FDIR will be read by the OBC, which will make the final decision. Despite that, the FDIR will also be able to communicate directly with the other processors, in order to avoid an OBC failure to compromise its functionalities and to allow the FDIR

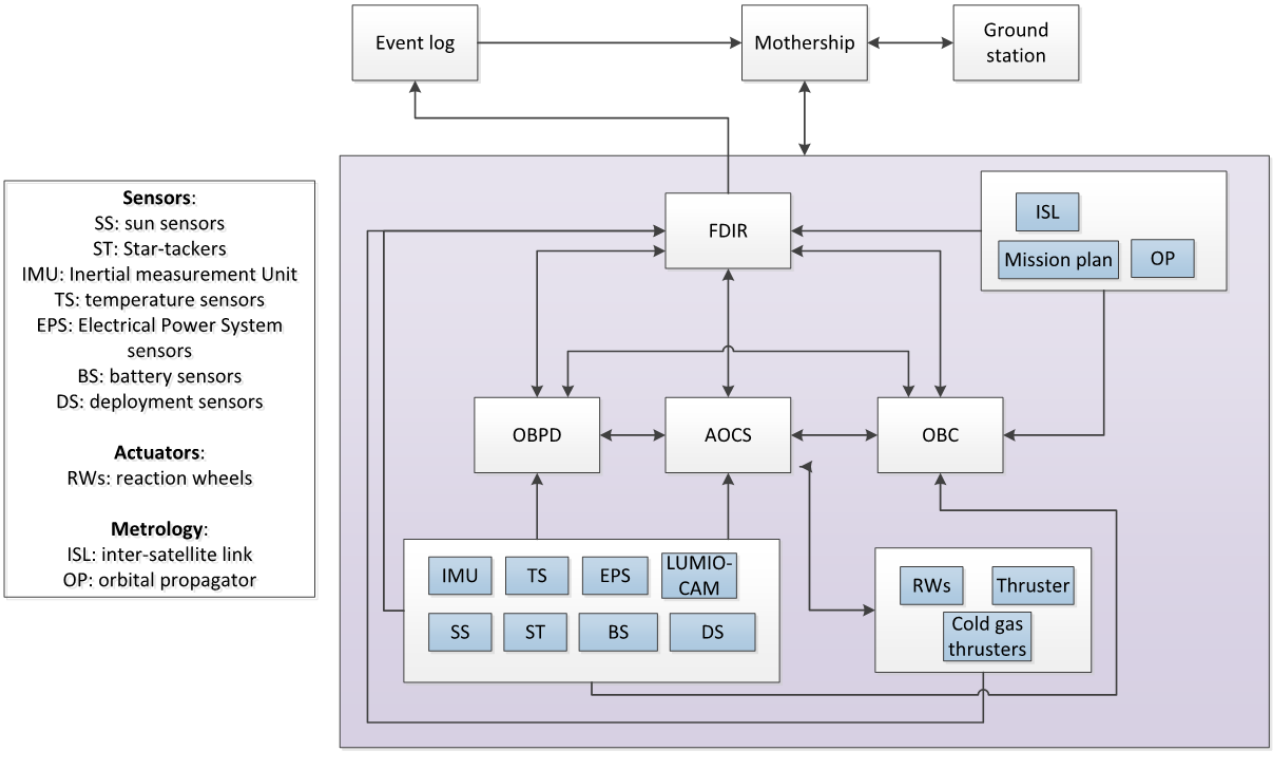

Figure 2-12: FDIR role in the S/C high-level architecture

to send a command to the AOCS processor and the payload processor in case of unrecoverable OBC failure.

Regarding the other units of the satellite, it is not known if direct communication between them and the processor which will run the FDIR will be included. In this study, it will be assumed that the FDIR will communicate directly with all the components; hence, it will receive their data packages directly, without the intermediation of other processors. This assumption simplifies the detection of eventual failures and the application of recovery actions, and is therefore recommended; however, in case a direct communication with a specific unit will not be possible, the FDIR on that component will still be feasible.

When a failure occurs, the FDIR will produce an event log, which will include the alarm flag, the inputs that triggered it, the current timestamp and other relevant parameters; the event log will be sent to the ground station through the Mothership, and eventual commands from ground will be read both by the OBC and the FDIR.

In conclusion, from Figure 2-12 the working principle of the FDIR is evident: the system will take in the data packages coming from all the subsystems and check them to detect and isolate eventual failures. In case of failure, the FDIR will also decide the recovery strategy, and communicate it to the processor in charge of the S/C Housekeeping (nominally, the OBC). The way in which the FDIR communicates with the other systems is through the FDIR log, which contains information about the detection of eventual failures and the recovery action to execute. Therefore, in this study, it was decided to design the high-level architecture of the FDIR algorithm as shown in Figure 2-13.

The algorithm is divided into two main parts: the FDI, in charge of detection and isolation, and the FR, in charge of deciding the recovery action to apply. Each part will produce its own log, and the union of the two will form the FDIR log. The main advantage of this design choice is the possibility to divide the work efficiently; in fact, the tasks of detection and recovery are based on different logics, and the first is independent from the second. Moreover, this choice eases the process of simulation and test of the algorithm, allowing to separate the detection from the recovery. Finally, the division makes it also easier to update the design in the next stages and to keep track of eventual changes.

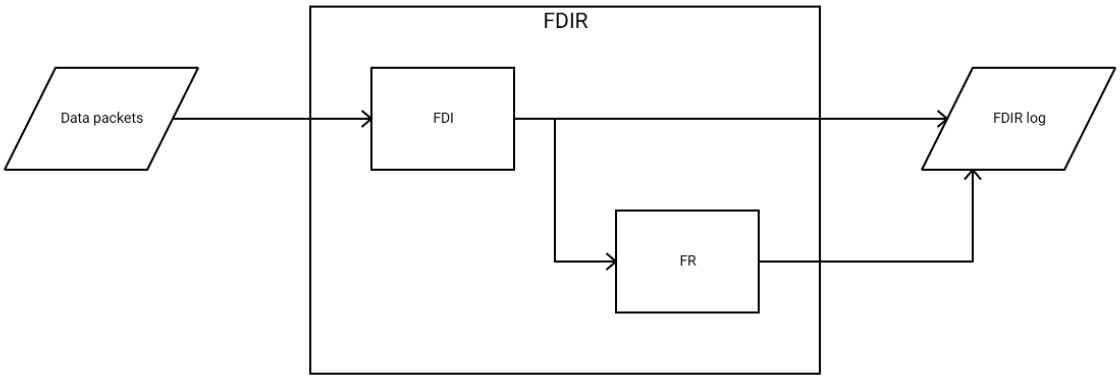

Figure 2-13: high-level architecture of the FDIR algorithm

3 FDI Design

In this chapter, the design of the first part of the FDIR algorithm, the FDI, will be presented. The FDI is aimed at detecting and isolating eventual failures of the satellite and producing a log to indicate the location and type of failure. The chapter is divided into multiple sections. In Section 3.1, the methodology is explained: the high-level architecture of the FDI model is shown, and the checks that are used within the algorithm are present. Later, the detailed architecture of each module in which the algorithm is divided is presented, along with a description of the implementation in MATLAB/Simulink.

3.1 Methodology

In this, section, the methodology that was used to design the FDI algorithm is presented. Firstly, the high-level architecture is described in Section 3.1.1. Later, the techniques used to detect and isolate the failures are illustrated in Section 3.1.2, while Section 3.1.3 will be focused on a more detailed description of the cross-check, which is the most used. Finally, the practical implementation of the FDI will be introduced in Section 3.1.4

3.1.1 High-level FDI architecture

The design of the FDI for LUMIO mission is based on the results of the FMECA, which is a list of all the failure scenarios that must be detected and isolated by the algorithm. As it is possible to see from the table, each component of the S/C is associated to several scenarios; hence, the first logical step to design the system is to group the failures related to the same unit/subsystem and treat them separately from the others. The result is the high-level architecture that is shown in Figure 3-1, where the FDI is divided into different *modules*; each module creates its own FDI log. The following modules were conceived:

- **Main thruster**: comprises all the scenarios related to the main thruster and its thermal control system.
- **Camera**: includes all the scenarios of LUMIO-Cam, related to science and navigation.
- **Processors**: includes failures of the OBC, the AOCS processor and the payload processor.
- **ADCS**: comprises the scenarios of all the ADCS units and their thermal control systems, namely the RWs, the gas thrusters, the star-trackers, the Sun sensors and the IMU.
- **Power**: comprises all the failures related to the power system, i.e. EPS, solar panels, SADA, batteries.
- **Deployment**: detects failures occurred during the deployment of the antennas and the solar panels.
- **Communication**: comprises the scenarios of the communication system and failures that can be detected from the Mothership package.

The decision on which failures to group in a specific module depends on several considerations. Firstly, most of the failure scenarios related to the same unit are mutually exclusive; hence, it is reasonable to group them. The knowledge of the S/C architecture is another fundamental information; in particular, what are data packets that are received by the FDIR and which data they contain. At this stage, due to the earliness of the design, this information is not available. Thus, some assumptions were made: as it was mentioned in Section 2.3, it is assumed that each unit will send its packet to the FDIR, without intermediation. Moreover, since the content of each package is not known either, a preliminary version of each data packet is conceived, containing the data that are deemed reasonable and relevant for this Thesis. Hence, it is logic to group in the same FDI module failures that can be detected through the data contained in the same data package.

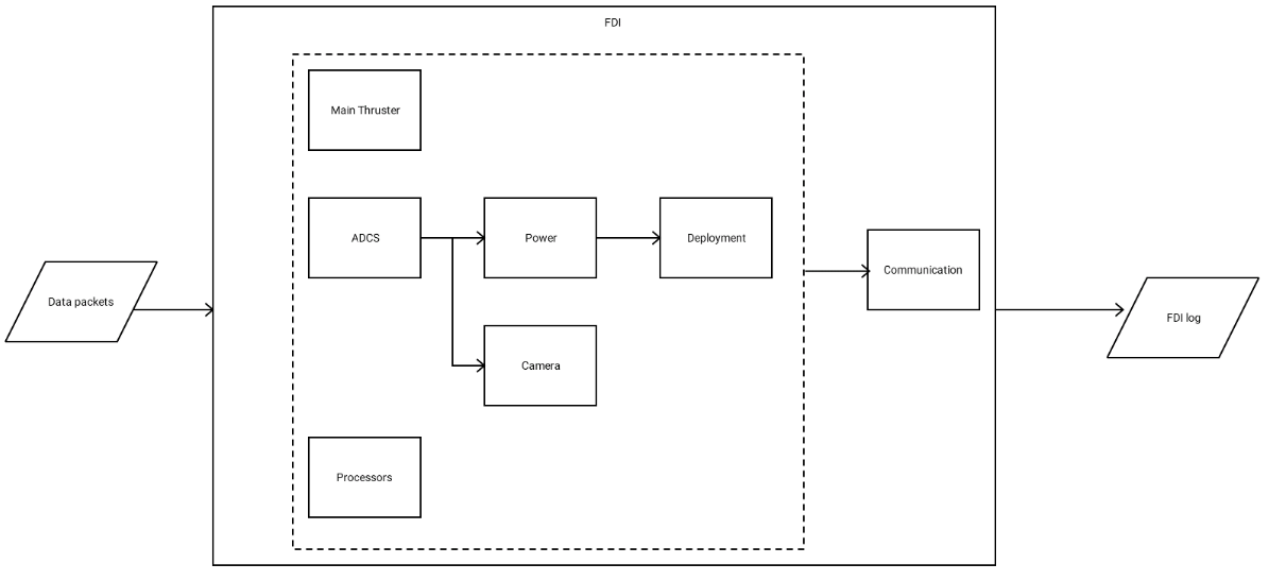

Figure 3-1: high-level FDI architecture

An additional consideration can be made, about how the FDI modules interact with each other. As it can be seen from Figure 3-1, some modules require the knowledge of the log of other modules as an input; therefore, it is reasonable to group failures that are needed by other modules: as a consequence, the whole ADCS is comprised in a single module since the ADCS log is an input for the Power FDI and the Camera FDI. Grouping the ADCS failures has also an additional advantage: in the future phases, new failure scenarios will be added, based on the S/C dynamics, which was not considered in this Thesis, and it is foreseen that for this purpose the data of the all ADCS units will be used together.

Finally, it should be noticed that it was decided not to create a specific module for the thermal control system, but to consider the thermal failures in the same modules of the afferent units. In fact, it is assumed that there is not a central processor dedicated to thermal data, but this task is distributed in all the systems. On top of this, the readability of the model improves.

The division into modules is not the only possible architecture for the FDI module; in fact, other alternatives are the creation of a unique model for all the failures, or the partition into different modules, e.g. one per unit. The architecture proposed in Figure 3-1, however, presents several advantages. Firstly, the division into modules allows to create a more readable model, which is easier to interpret and update in the future phases. Moreover, the simulation and test of each module is eased, as it is independent from the other parts of the model. The use of a different partition is possible, and more advantageous alternatives will likely be conceived in the next phases of the project; however, at the moment, the proposed architecture is deemed reasonable.

3.1.2 Detection and Isolation methodology

Once the division of the FDI into modules has been outlined, it is fundamental to describe how the failure detection and isolation are achieved, within a module. The logic is strongly dependent on the failure scenario that is considered and on the available data; however, it is possible to identify some general strategies. The failure of a unit can be detected internally by the unit itself, but additional checks are performed to permanently monitor the spacecraft [8]. The purpose of the additional checks is to determine if the output of a unit is consistent over a certain time and to determine the location of certain failure, in case there were doubts. The fundamental assumption is that only one failure happens at a time, since the probability of multiple failures is minimal, and it will not be considered. There are various kinds of checks:

- **Health checks**: acknowledgment every time a unit/subsystem/system gets out of idle or is started (this is not part of the FDIR, but of the Housekeeping and Processing tasks of the OBC).
- **Validity flags**: provided periodically by unit/subsystem/system, e.g. one flag every orbit.
- **Range checks**: check if the data is in a certain range, e.g. checking a distance using different measurements (also called **limit-checking**).
- **Plausibility check**: check if data is realistic given the operating conditions.
- **Continuity checks**: a model is used to predict the new measurements, and the result is compared with the actual measurement.
- **Cross-checks**: comparing the measurements from a unit with the values provided by similar units or with a value stored in memory.

It is noticeable that the health-check and the validity flag are similar, as they are both flags sent by the unit to communicate the presence/absence of a failure. Since the actual design of the data packages is not available, only a preliminary implementation of these checks is done; hence, they were grouped in a single flag, the validity flag, which is sent by every component that has a processor or a microcontroller. The validity flag notifies if the unit is healthy or not; however, it does not specify which failure occurred. In the future phases of the project, it is expected that the validity flag and health checks will be implemented more accurately, based on the actual design of each unit.

In case a failure occurs but it is not detected by the validity flag, different strategies are used. As it can be seen, all the check methodologies mentioned above are rather simple; however, more advanced techniques are commonly used for fault detection. Most of them rely on the creation of 'residuals', which are variables that quantify the inconsistency between a process and its ideal mathematical model [20]. Ideally, the residuals are close to 0 in absence of failures, while their value grows in case a fault occurs. In case of dynamic processes, different methodologies can be applied for residual generation, which are extensively described in literature: for instance, parity checking, state estimation, e.g. Kalman filter, and parameter estimation are illustrated by R. Isermann in "*Fault-diagnosis system - An introduction from Fault Detection to Fault Tolerance*" [21]. However, these techniques are aimed at detecting failures of the S/C dynamics, which at this phase of the project is not considered. Hence, researching how these techniques can be applied for the FDIR of LUMIO is recommended for the follow-up projects.

In the FDI framework, the cross-check between two units can be regarded as the simplest residual generator method. In this Thesis the cross-check was widely used: since two units are unlikely to fail at the same time, comparing their output allows to detect eventual failures. Hence, the next section is dedicated to the detailed description of the cross-check methodology used in this study. In case cross-checks options are not available, other methods, as limit-checking and plausibility checks are used. These methods are described in [21] but are deemed weaker, as they are based on checking only one directly measured variable.

3.1.3 Cross-check methodology

Cross-checking two units means to compare their outputs, in order to detect eventual failures. In case the outputs are logical values, e.g. a valve can be open or close (1 or 0), the cross-check can be achieved through if-if not logic, for instance through a truth table. This type of cross-check was used for most of the failure scenarios considered in this study, e.g. the failures of the main thruster. However, if the outputs to compare are actual measurements, e.g. angles measurements, the cross-check involves evaluating if an *error function* is lower than a threshold, as shown in Equation (1). This situation applies mostly to failures of the ADCS sensors. It should be noticed that, in some cases, the error function is a vector; thus, the condition expressed in Equation (1) should hold for all its components.

$$err < trs \tag{1}$$

Therefore, to perform a cross-check it is fundamental to define two elements: the error function and the threshold.

The k-th element of the error is computed as a function of measurement 1 and measurement 2. Despite the measurements are two, each of them might include more than one variable, e.g. a star-tracker measures 4 attitude quaternions; thus, the error function depends on *n* variables x_i, as it can be seen from Equation (2). Moreover, as mentioned above, the error function itself could be a vector with more than one value. The error function respects the definition of residual that is given in [20]: ideally, each component of the error should be 0 (since the two measurements should be consistent); however, the error is a statistical quantity, thus it will have a mean of 0 and an accuracy error, which depends on the accuracy of the two initial measurements. The assumption of normal distribution will be used, as it is the most common and easy to model. Thus, the accuracy error can be described with multiples of the standard deviation σ_{err}, which depends on the standard deviations of the initial measurements. For instance, using a 1-σ approach allows covering roughly 68 % of the of cases.

$$err_k = f_k(x_1, x_2, \dots, x_n) = 0 \pm \sigma_{err} \tag{2}$$

The definition of the error function depends on the variables to compare; thus, it shall be defined case by case. As it was mentioned above, the error functions to be used for the cross-checks in the ADCS system were defined in this Thesis. With the cross-check, the correct functioning of two units, e.g. two star-trackers, can be verified. However, no information is obtained about the process that is measured by the sensors, e.g. the S/C dynamics. This represents a limitation that should be addressed in the next stages of the project, when the dynamics will be considered.

While the error function depends on the units that are cross-checked, the threshold can be defined arbitrarily. In the framework of residual checking, several researches were carried out to select the best threshold. A valuable overview is provided in [22]. The main distinction is made between fixed and adaptive threshold, which are discussed in [21] as well. Following the need for simplicity, the easiest approach will be applied in this Thesis: a fixed threshold will be defined for each cross-check. The selection of the appropriate threshold is a critical task, dictated by the trade-off between two different aspects, derived from the FDIR requirements. On the one hand, the FDIR shall avoid false positives, i.e. triggering the recovery when no failure occurred, in order to minimize the time spent on recovery and avoid chain failures. Many false positives in the ADCS are given by external perturbations, e.g. radiations. This is especially applicable in LEO since the aerodynamic drag effect can affect the dynamics. Thus, the thresholds for the cross-checks in the ADCS need to be selected carefully. On the other hand, it is fundamental to avoid false negatives since the FDIR aims at promptly detecting eventual failures and recovering them. Selecting a “weak” threshold, i.e. a large value, would minimize false positives, but would not allow to detect a significant number of failures. Instead, selecting a “strict” threshold, i.e. a little value, would lead to improved fault detection and fewer false negatives, but it would increase the number of false positives.

Most of the approaches for the definition of the thresholds for residual checking are heuristic, based on experimental data [22]. Due to the earliness of LUMIO project this methodology cannot be applied. Further approaches have been investigated, to select a threshold analytically, starting from requirements on the detection of false positives and/or false negatives [22]. However, the methodologies investigated in literature are used to check the residuals in dynamical processes and are therefore rather complex and not applicable for the cross-checks that will be treated in this Thesis. Thus, in this study a new methodology for the definition of the threshold to check the error function will be proposed. The baseline is the quantitative estimation of false positives and false negatives detected by the cross-check. To do so, the accuracy of the error function, i.e. its standard deviation (or an approximation of it), should be calculated first.

To compute the accuracy of the error function, the starting point are the specifications of the components to check, namely their accuracy. Then, the theory of propagation of uncertainty shall be used [23]. Assuming to have a set of *m* functions f_k dependent on *n* variables x_i, in the form of

$\{f_k(x_1, x_2, \dots, x_n)\}$, the variance-covariance matrix of the variables can be defined as written in Equation (3). Each diagonal term of the matrix is the variances of one variable, i.e. its standard deviation squared. The off-diagonal elements are the covariances between two variables, which are equal to 0 when the errors are symmetrical and not correlated.

$$\Sigma^x = \begin{bmatrix} \sigma_1^2 & \dots & \sigma_{1n} \\ \dots & \dots & \dots \\ \sigma_{1n} & \dots & \sigma_n^2 \end{bmatrix} \tag{3}$$

In case f is a non-linear combination of the initial variables, the variance-covariance matrix of f can be approximated with a first-order Taylor expansion; this would lead to the result of Equation (4).

$$\Sigma^f = J\Sigma^x J^T \tag{4}$$

The matrix J mentioned in Equation (4) is the Jacobian matrix, whose element in the k-th row and i-th column is $\frac{\partial f_k}{\partial x_i}$. Each diagonal elements of matrix Σ^f is the variances of one element of function f. Hence, using this equation it is possible to compute the standard deviation of each function f_k (in our case, the standard deviation of each element err_k of the error vector), starting from the accuracy of the initial measurements that are being cross-checked.

Once the standard deviation of the error function is defined, it allows making some considerations on the choice of the threshold. As mentioned above, the two aspects to consider are the probability of detecting false positives or false negatives. A false negative occurs when a failure in one of the instruments that are cross-checked is not detected. A false positive occurs when a failure is detected, despite no failure happened. However, the application of recovery levels after a failure is detected, treated in Chapter 4, should be taken into consideration. The first recovery level to be activated is L0, during which no action is taken for a given amount of time, typically around 10/20 [s]. Therefore, if a failure occurs, it is not only fundamental that the FDIR detects it, but also that the detection is continuous during L0, otherwise other recovery levels will not be triggered. On top of this, in order to increase the robustness of the recovery action and to avoid a continuous jumping in and out of L0, it was decided to introduce a delay once recovery is triggered: in order to exit L0, the absence of failure should be confirmed for 3 [s] in a row. Therefore, a better definition of false negative is: the probability of <u>not</u> detecting a failure for 3 [s] in a row, when a failure occurred. Similarly, it is possible to give a better definition of false positive: the probability of detecting <u>no</u> failure for 3 [s] in a row, when no failure occurred.

Therefore, it is possible to define the requirements regarding false negative and false positive, in terms of probability:

- The selected failures shall be detectable without false negatives, i.e. if the probability of detecting false negatives for three times in a row shall be less than 5%.
- The FDIR shall have a probability of detecting false positives for three times in a row less than 5%.

The value of 5% was chosen in absence of similar requirements in the reference and it was deemed reasonable. In fact, the definitions of false negative and false positive that were used are conservative; therefore, the actual percentage will be inferior to 5%. In the next stages of the project, this probability is expected to further decrease.

Firstly, the rate of **false negatives** should be analysed. Several types of failures can occur to an instrument, e.g. frozen sensor or sensor drift; however, it is possible to study the rate of false negatives by considering only two types of scenarios: *loss of accuracy* (change of standard deviation) and *bias* (change of mean value). The error distributions resulting from these types of failures are shown in Figure 3-2. In fact, due to how the error function is computed, a loss of accuracy of one measurement would lead to a loss of accuracy of the error function. On the other

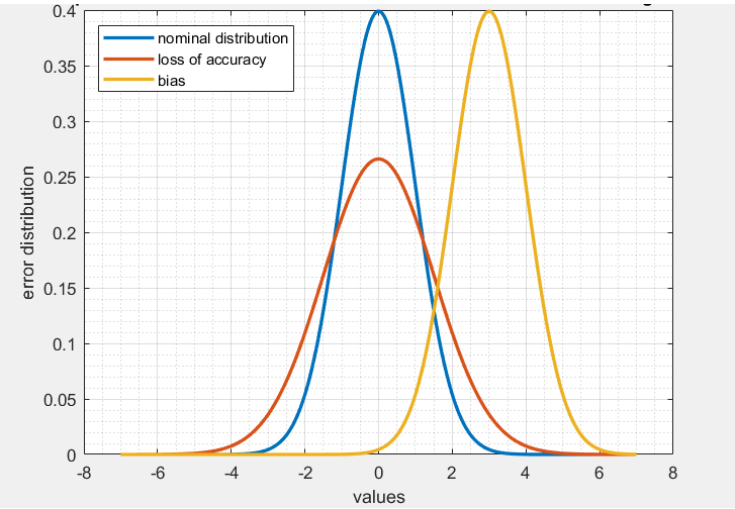

Figure 3-2: comparison between the error distribution in nominal case and during failures

hand, any other failure of a sensor results in a change of the mean value that it is measured: a sensor bias results in a constant bias, a frozen sensor or a sensor drift result in an increasing bias. A change in the mean value of one measurement causes a change in the mean value of the error function. Thus, it is possible to completely characterize the detection rate of false negatives by focusing on these two types of failures. A similar classification of failures was proposed in [24], but applied for the FDI of a process, rather than a single sensor.

Firstly, the change of accuracy of the error function will be considered. In the nominal case, each element of the error function has a standard deviation σ_{err} that can be calculated as shown in Equation (4). In case a loss of accuracy occurs, the standard deviation of the error function changes to a new value $\sigma_{err,f}$, which can be expressed as a function of the initial σ_{err}, as shown in Equation (5).

$$\sigma_{err,f} = y\sigma_{err} \tag{5}$$

The objective of this analysis is to research the minimum value of $\sigma_{err,f}$ that can be detected without false negatives, given a certain threshold. According to the definition given above, a false negative occurs when a failure is not detected for 3 [s] in a row. Hence, in order to have a probability of false negative of 5%, the probability of not detecting the failure at a specific time should be calculated as in Equation (6).

$$prob = \sqrt[3]{0.05} \sim 37\% \tag{6}$$

According to the probabilities of normal distributions, this probability is comparable to the probability of obtaining a measurement within the interval $[-0.5\ \sigma_{err,f},\ 0.5\ \sigma_{err,f}]$, which is around 38%. This concept is illustrated in picture a) of Figure 3-3. Therefore, it can be concluded that, chosen a threshold, the minimum loss of accuracy that would be detected without false negative would be as shown in Equation (7).

$$\sigma_{err,f} = 2\ trs \tag{7}$$

The result of Equation (7) shows how the minimum loss of accuracy detectable increases when increasing the threshold; thus, it gives an important tool to support the decision of the threshold.

The next type of failure is the bias, i.e. change in the mean value of the error function. Using the same requirement of 5% probability of false negatives, Equation (6) holds also in this case. Assuming a positive bias, it means that there should be about 37% of probability that a value of the error function is lower than the threshold: from the probabilities of normal distributions, it is known that the probability of value lower than $-0.3\ (\sigma)$ is about 38%, in case of a normal distribution with mean equal to 0. Hence, the minimum bias that respects the requirement can be calculated as done in Equation (8). Figure 3-3 b) illustrates the concept, showing the probability distribution for the minimum bias that can be detected, given a certain threshold.

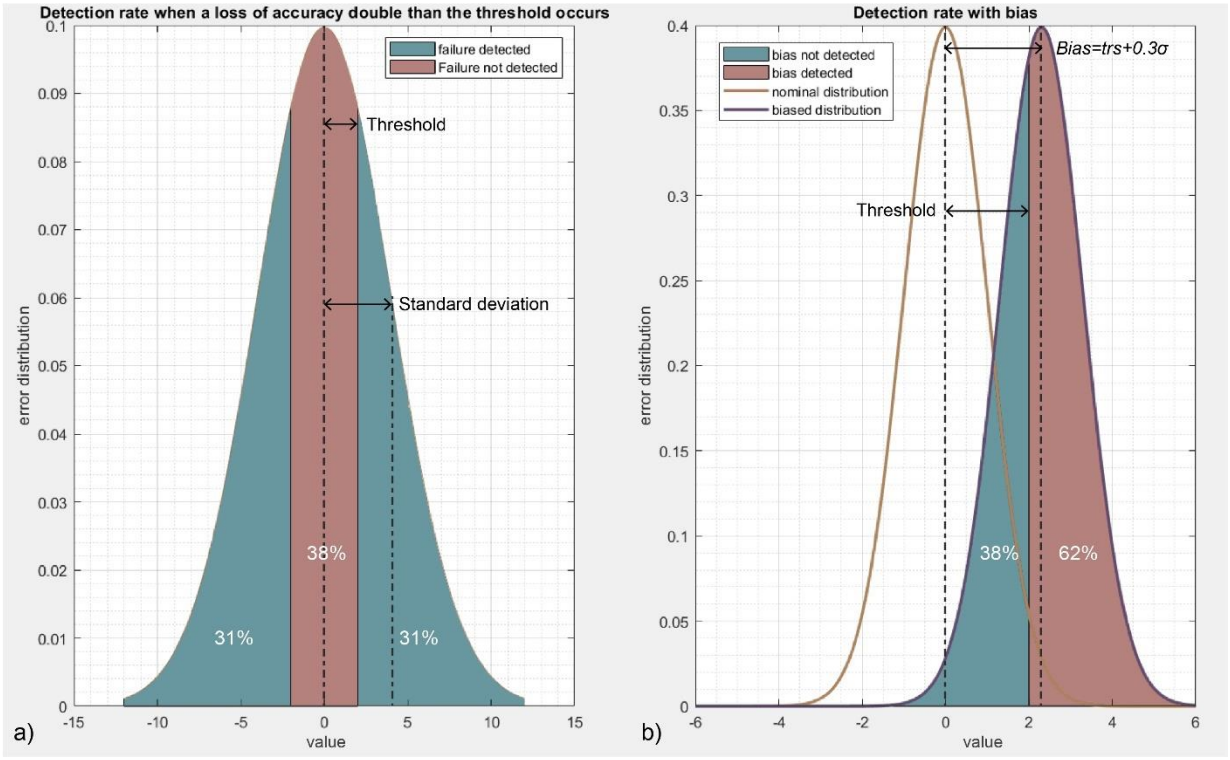

Figure 3-3: failure detection rate in case the minimum loss of accuracy (figure a) and bias (figure b) occur

$$|bias| = trs + 0.3\sigma_{err} \tag{8}$$

Once the rate of false negatives has been studied, the rate of **false positives** should be estimated. The probability of false positives should be less than 5%: this means that the probability of detecting no failure for 3 times in a row (when no failure occurred) should be higher than 95%. Hence, the probability of not detecting a failure should be as calculated in Equation (9).

$$prob = \sqrt[3]{0.95} \sim 98\% \tag{9}$$

Using the probabilities of normal distribution, this probability is close to the probability of detecting a measurement in the range of $[-2.5\sigma_{err}, 2.5\sigma_{err}]$. Hence, a threshold of $2.5\sigma_{err}$ is the minimum threshold that would allow to satisfy the requirement on the detection of false positive.

In conclusion, the analysis done so far allowed to establish a methodology to define the threshold for a cross-check between two sensors. Despite some results were obtained and are listed in Table 3-1, it is not possible to draw general conclusions regarding the best choice of threshold. In fact, other factors play an important role: the accuracy requirements on the measurements that are compared and the expression of the error function. In fact, despite the measurements have a nominal accuracy, a different accuracy is likely to be required by the actual system; hence, a

Table 3-1: guidelines for the decision of the threshold, based on the standard deviation σ of the error function

Requirement	Guideline	Explanation
False positive	$trs > 2.5\ \sigma_{err}$	The threshold shall be higher than 2.5 times the standard deviation of the error function.
False negative (loss of accuracy)	$(\sigma_{err})_f = 2$ trs	The minimum degradation of the standard deviation that can be detected without false negatives depends on the threshold.
False negative (bias)	$bias = trs\ + 0.3\ \sigma_{err}$	The minimum bias that can be detected without false negative depends on the threshold and on the nominal standard deviation.

minimal performance degradation might be accepted. Moreover, the expression of the error function, in particular of its standard deviation, is important to understand how a failure of a measurement propagates in the error. So far, only the loss of accuracy or the bias of the error function were studied, but it is fundamental to link them to the failures of the initial measurements; this must be done case by case.

From the analysis, it is evident that the more similar are the two units that are cross-checked, the easier will be the cross-check. For instance, in case the units were the same, the error function of Equation (2) would be simply the difference between their measurements. On the other hand, the more different are the units, the more complex will be the cross-check. This fact has also another subtle consequence: since the number of operations to perform increases the standard deviation of the error function, error detection through cross-check of different units is less accurate. Therefore, it is always preferable to perform cross-checks between measurements that necessitate little operations to be compared. This is a relevant limitation of the cross-check methodology proposed in this Thesis since in the current design of LUMIO the number of available redundancies is limited. For instance, in this Chapter the cross-check methodology will be applied to units that are relatively different to each other (star-trackers, Sun sensors and IMU) and several challenges will arise. On top of this, another relevant criticality of this methodology can be pointed out: even when a failure is detected, it is not isolated. In order to locate the failure, additional checks must be integrated in the FDI logic.

The methodology proposed so far, which is based on the study of the statistical model of a measurement to detect eventual failures, can be compared to the methodology of binary thresholds proposed in [21], as they both rely on statistical considerations. The approach in [21] is used to analyse one variable to detect failures of a process: knowing the expected mean value and the standard deviation, statistical tests are executed to verify a change of one of these parameters. Despite the similarities, the two methodologies are different in purpose: the approach proposed in this section is used to detect failure of a unit and requires the outputs of two units, while the technique described in [21] is aimed at detecting failures of a process, from the analysis of one variable. However, the two methodologies are not antithetical: the cross-check can be used to check a unit and, once the absence of failures is confirmed, the binary threshold method can be applied to its measurement to detect failures of the process. Moreover, it is also possible to apply the binary threshold method to the error function since the expected mean value and standard deviation are known: continual statistical test can be performed to verify that the mean value of the error function is 0, and the standard deviation is as expected; however, in this Thesis this approach was not carried on, due to the augmented complexity of the calculations, in contrast with the requirement of simplicity. The selection of a fixed threshold for a cross-check represents a faster and simple option, more suitable for this stage of the project.

3.1.4 Algorithm implementation

The model for the FDI algorithm that was described so far has been implemented in the MATLAB/Simulink programming environment.

The choice of implementing the FDIR algorithm in Simulink, which is a software for modelling and simulating dynamical systems, is due to the importance that the satellite dynamics has for the FDIR algorithm. An example of ADCS simulator built in Simulink is documented in [25]. Thus, despite the dynamics will not be considered in this Thesis, using Simulink is necessary to advantage its implementation in the future phases of LUMIO project. Moreover, the software interface, which is based on the creation of complex structures made by logical relations between blocks, is particularly suited for visualizing the FDIR architecture and identify eventual criticalities. On top of this, the process of updating the model in the further phases of the project is eased.

In this Thesis, the Stateflow environment inside Simulink was widely used. This tool allows to reproduce the detection logic of each FDI module in a chart, made of transitions between states. Thus, a complex decision logic can be implemented, using truth table, graphical functions or costumed MATLAB functions. The readability and modifiability of the charts are the main advantages of this tool.

3.2 Main thruster module

In this section, the logic of the FDI for the main thruster will be explained. Firstly, the FDI architecture is presented, along with the checks that are used to perform failure detection. Later, the Simulink model will be described.

3.2.1 FDI logic

The Main Thruster FDI module is aimed at detecting the failures of the main thruster that are listed in the FMECA. Moreover, it is assumed that three temperature sensors are installed on the unit; hence, their failure scenarios are also part of this module.

The first step which is necessary to design the FDI logic is the knowledge of the data contained in the packet coming from the thruster, to develop a check methodology. Due to the earliness of the project, this information is not known; thus, some reasonable assumptions will be made, in order to allow for the detection of the selected scenarios. Firstly, it is assumed the all the data of the packet will be contained in bits, with binary values (0/1), and they will be followed by control bits, used to detect the "malformed data packet" failure. Several methodologies are used for the selection of the redundant bits; the simplest is the parity bit, one bit that is added to a string to ensure the presence of an odd or even number of 1s [26]. However, in this Thesis, it was decided to use another technique, the cyclic redundancy check (CRC), which is more complex to implement but can detect multiple errors in communication [27]. Moreover, some Simulink pre-defined blocks already implement the code to generate and check CRC bits, thus simplifying the practical implementation. The commonly used code CRC-16-CCITT will be used, which is associated with the polynomial $x^{16} + x^{12} + x^5 + 1 = 0$.

The data contained in the main thruster packet are listed in Table 3-2; as it can be seen, this is a preliminary version of the packet, as it contains only the information that is essential for the FDI. The baseline for the definition of the packet is the schematic model of a mono-propellant thruster, shown in Figure 3-4, taken from [28]. As it can be seen, the thruster can be modelled as a propellant tank, which receives pressurant gas through a pressure regulator (R) and is connected with the nozzle through a valve (V). The main elements that were considered in this Thesis are the propellant budget and the valve, while the pressure regulator was not included at the current stage. Some remarks must be made. Firstly, it can be seen that the validity flag is included in the packet: as it was mentioned in 3.1.2, when the flag is negative it means that a generic failure occurred to the unit and was detected internally.

Secondly, the thrust command is worth discussing. In [1], the nominal thrust that can be produced by the system is assumed to be 0.1 [N]. Hence, in this Thesis it is assumed that 0.1 [N] is the maximum command, and any value above it would trigger the detection of the "command out of range" failure scenario. The command is sent in [cN], to be an integer, which can be converted into binary more easily.

Table 3-2: data in the main thruster packet

Data	Value	Bits	Description
Main thruster validity flag	0/1	1	The value 1 indicates that the validity flag is positive, the value 0 that it is negative.
Thrust command	0-15 [cN]	4	The thrust command.
Valve state	0/1	1	0 if the valve is closed, 1 if open.
Propellant budget	0-1950 [g]	11	The current propellant budget, which is used to detect whether the budget is decreasing or not.
T sensor 1 state	0/1	1	0 if the sensor is off, 1 if it is on.
T sensor 1	+/- 100 [°C]	8	The temperature of the main thruster as detected by temperature sensor 1.
T sensor 2 state	0/1	1	0 if the sensor is off, 1 if it is on.
T sensor 2	+/- 100 [°C]	8	The temperature of the main thruster as detected by temperature sensor 2.
T sensor 3 state	0/1	1	0 if the sensor is off, 1 if it is on.
T sensor 3	+/- 100 [°C]	8	The temperature of the main thruster as detected by temperature sensor 3.

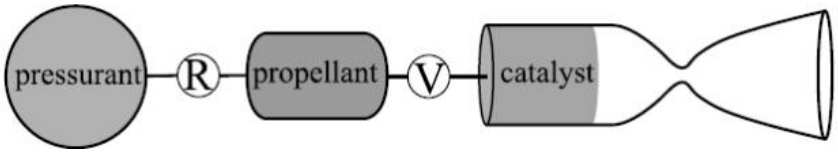

Figure 3-4: main elements of a mono-propellant thruster. Credits: [28]

Another relevant consideration regards the propellant budget. The unit is a COTS mono-propellant thruster and the propellant (ADN) is stored in a liquid state. At the current stage, it is not known how the estimation of the propellant budget will be included in the final design; in fact, the direct measurement of liquid fuel level in a tank is challenging in absence of gravity. An overview of the most popular methods for the estimation of the propellant budget is presented in the paper *"Review of Propellant Gauging Methods"* [29] and in *"Comparative Assessment of Gauging Systems and Description of a Liquid Level Gauging Concept for a Spin Stabilized Spacecraft"* [30]. Amongst the various approaches, the most used is the bookkeeping method, which is based on the manoeuvre data to estimate the propellant consumed. Despite this approach can be used during thruster's operations and requires no additional sensors [30], it cannot be employed in the context of FDIR, as a measurement independent of the valve state is needed to properly detect a failure. Another common approach is the pVT method, in which the estimation of the tank ullage volume through measurements of the temperature and pressure is used to quantify the propellant volume [30]. This method can be used in the context of FDIR since it is independent from the valve state; however, the main disadvantage is the low measurement accuracy when conventional pressure transducers are used, and the decrease of the accuracy during the mission [30]. Moreover, it is not known at the current stage if the necessary sensors (tank pressure and temperature) will be included in the design of the propulsion system of LUMIO. Alternative methods as thermal propellant gauging and gas injections has been analysed in [29; 30], but are not suitable for measurements during thruster's operations. Finally, the measurement of the propellant mass flow with a mass flow meter is a feasible option that can grant a high accuracy during firing. Despite in [30] the absence of available mass flow meters for space applications at the time was underlined, advancements in this field were made; for instance, a COTS unit is produced by Bradford [31]. However, the supplier specifies the suitability for GEO satellites but does not mention deep space applications; moreover, it is not known if it will be possible to include this type of sensor in the final design of LUMIO propulsion system. Despite all the aforementioned limitations, the propellant budget is included in the FDI design at this phase, since its trend is fundamental to detect some failure scenarios. It should be noticed that the only relevant information that is used in the check is the trend (constant or decreasing), not the absolute value. In case in the information will not be available in the final design, it will still be possible to detect the failures of the thruster with the S/C dynamics by checking the S/C position and velocity, but this check is not developed at the current stage. An example of FDI logic for a thruster based on the dynamics is provided in [32].

The FDI logic of the module is shown in Figure 3-5. The FDI log is divided into three: the first part is dedicated to the detection of proper thruster failures, the second is dedicated to the failures of the temperature sensors, the third to failures of the thruster's temperature (over/underheating). The logic used to perform the three checks showed in yellow in Figure 3-5 is worth discussing.

The first check is aimed at detecting failure scenarios MT.2 to MT.5 in the FMECA table. The principle is the following: based on the command, it can be understood if the unit is in thrusting mode (command greater than 0) or in idle (command equal to 0). Based on the mode, the expected values of the valve state and the propellant trend can be found: in thrusting mode, the valve should be open and the propellant decreasing, in idle the valve is closed and the propellant constant. In case there is no accordance between the parameters, a failure occurred, according to the logic of the truth table shown in Table 3-3.

The second check of Figure 3-5 is a cross-check between the three temperature sensors. As it was mentioned above, it is assumed that three temperature sensors will be installed on the main thruster. However, in case one of them irremediably fails, it will be turned off; hence, three different detection logics were developed.

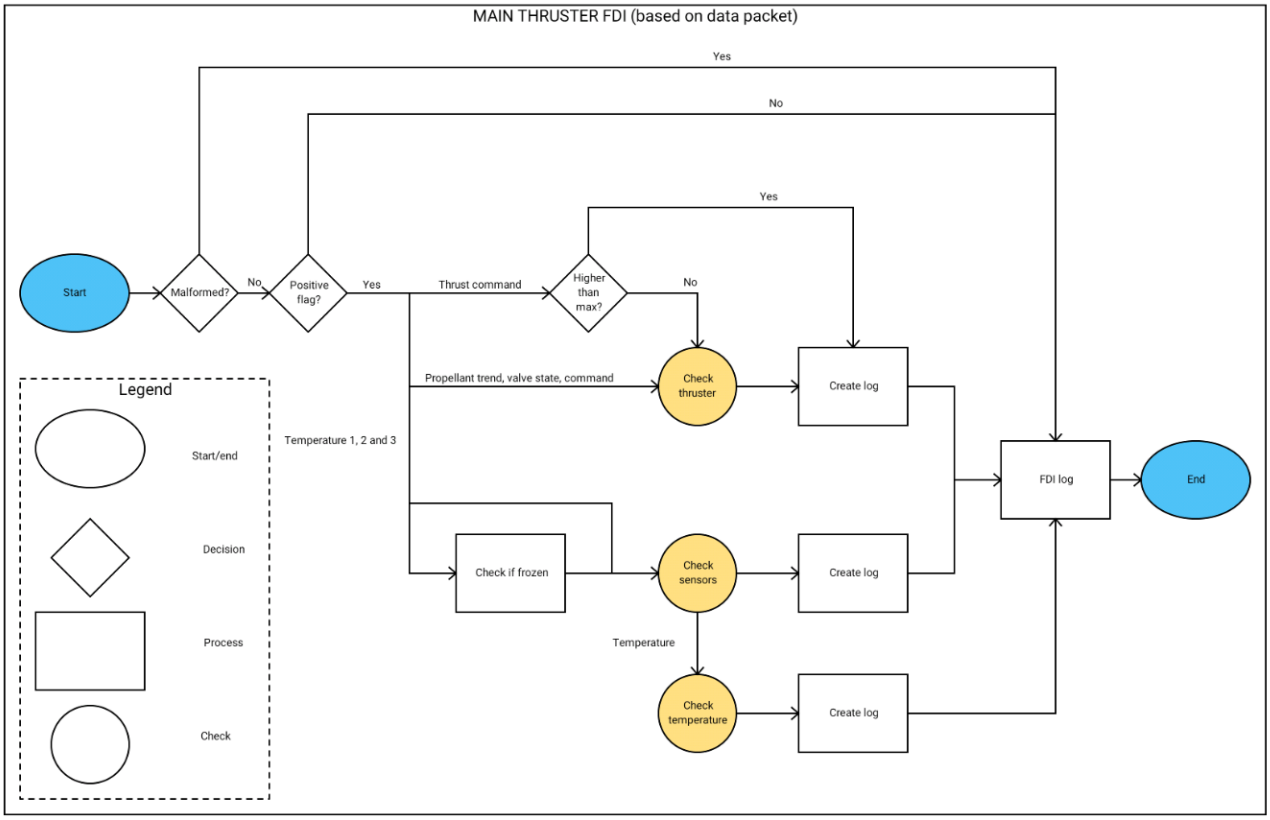

Figure 3-5: Main thruster FDI architecture

The first logic is applied when three temperature sensors are available. Two factors limit the application of the cross-check methodology described in Section 3.1.3. Firstly, the model of temperature sensors is not known; hence, the accuracy is unknown. Secondly, the sensors will probably be installed in different parts of the thruster; thus, they will measure different temperatures, due to the gradients of the unit. At this stage, the location of the sensors and the expected temperature gradients are unknown. Therefore, a preliminary cross-check is designed. The output of each sensor is compared to the other two and the threshold used is 5 [°C], arbitrarily chosen. In case the measurement of one sensor differs from both the other two of more than the threshold, a failure flag is created. In case of failure, a distinction between “general failure” and “frozen sensor” is made.

The second logic is applied when only two sensors are available. In that case, a simple cross-check with a threshold of 5 [°C] is used. In case the threshold is surpassed, a failure is detected; however, it is not isolated, as it is unknown which sensor failed. Hence, an additional check is performed: a limit check. Since the temperature of the thruster is expected to be within a certain range, if a sensor’s output exceeds that range, it is assumed that the unit is faulty. As it can be seen, this method is less accurate, as the temperature increase might be caused by a thruster’s failure; however, it is assumed that only one failure can happen at a time. In case both the sensors

Table 3-3: truth table used for the detection of failures of the main thruster

Condition	**Value (true/false)**							
Idle mode	T	F	T	T	T	F	F	F
Valve closed	T	F	T	F	F	F	T	T
Propellant constant	T	F	F	T	F	T	F	T
Scenario	No failure	No failure	Propellant sensor failure	Valve sensor failure	Locked output	Propellant sensor failure	Valve sensor failure	No thrust
FMECA ID	/	/	MT.4	MT.5	MT.3	MT.4	MT.5	MT.2

measure a temperature above the threshold, or none of them do, the failure cannot be isolated. Again, in case of isolated failure, a distinction between “general failure” and “frozen sensor” is made.

Finally, the third logic is applied when only one sensor is available. In this case, the limit checking is used solely. Hence, the risk is to confuse an overheating/underheating failure for a failure of the sensor; however, this situation is rare, as it occurs only after two of the temperature sensors already failed, which has little probability. Again, in the case of failure, a distinction between “general failure” and “frozen sensor” is made.

The final check showed in Figure 3-5 takes the temperature, measured by the set of sensors, as an input to detect eventual overheating or underheating failures (MT.6 and MT.7 in the FMECA). The temperature is calculated as an average of the measurements from the healthy temperature sensors. In this case, a limit check is sufficient for the purpose: if the temperature exceeds the minimum/maximum threshold, the failure flag is created. The operative temperature range is from 0 to 50 [°C], while the range during idle is -10 to 60 [°C], according to [1].

3.2.2 Simulink model

The main thruster module takes the packet coming from the main thruster as input and returns the FDI log as an output. The log is an array of three elements [a, b, c], where a, b and c are integers. The first element of the log, **a**, is referred to main thruster functioning, whereas **b** refers to the temperature sensors of the thruster and **c** refers to the thruster’s temperature. The meaning of the possible values of a, b and c are explained in Table 3-4. It can be noticed that, in case of malformed data package or negative validity flag, the three elements have the same value.

Table 3-4: FDI log of the main thruster module

Element	Value	Meaning
All	0	No failure
	1	Malformed data package
	2	Negative validity flag
a	3	Locked output
	4	No thrust
	5	Propellant sensor failure
	6	Valve sensor failure
	7	Command out of range
b	3	Sensor 1 - general failure
	4	Sensor 1 -frozen
	5	Sensor 2 - general failure
	6	Sensor 2 -frozen
	7	Sensor 3 - general failure
	8	Sensor 3 -frozen
	9	Unidentified failure
c	3	Overheating
	4	Underheating

The Simulink model is shown in a series of pictures, in Figure 3-6. After checking if the package is malformed, in figure a), the data package is divided, and the validity flag is checked, as shown in figure b). Later, in case of positive validity flag, the inputs enter in two Stateflow charts, as it can be seen in figure c).

The first chart in Figure 3-6 c) replicates the simple logic of the main thruster FDI, with the aid of a truth table equal to Table 3-3, and it is reported in Figure C-1, in Appendix C. The second Stateflow chart is dedicated to thermal control. There are two main sub-modules: one for the temperature sensors, one for the temperature of the thruster.

The chart of the temperature sensors is divided into three sub-states, which implement the three different logics, based on the number of available sensors. The cross-checks are executed with the aid of custom MATLAB functions. In case the failure of one sensor is triggered, a distinction is

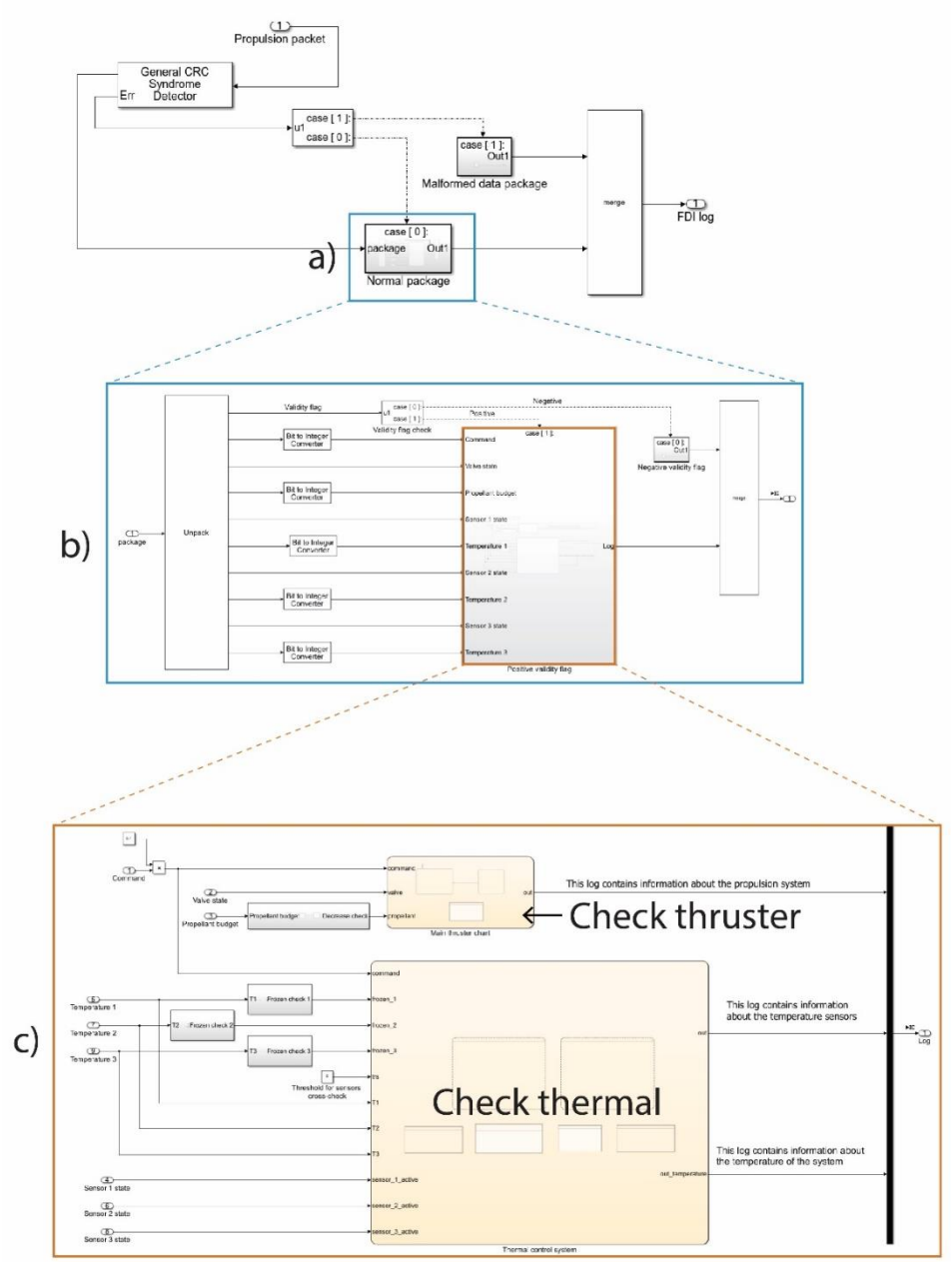

Figure 3-6: Simulink model for the Main Thruster FDI. In a) the data package is checked, in b) the validity flag is controlled. Finally, in c) the data enter inside two Stateflow charts, where the other checks are performed

made between general failure and frozen sensor, but a waiting time of 10 [s] is included before triggering the frozen sensor failure since, during that time, the measurement shall be checked. The Stateflow chart for cross-checking three sensors is shown in Figure C-2, in Appendix C, while the cross-check of two sensors is shown in Figure C-3. It can be seen that, in the latter case, the "unidentified failure" is implemented to deal with a failure which was not isolated. Finally, the simple limit-check used to detect failures when only one temperature sensor is available is shown in Figure C-4.

The chart for the temperature check implements the simple limit-check described in Section 3.2.1, and it is shown in Figure C-5, in Appendix C.

3.3 ADCS module – Reaction wheels

The ADCS module comprises all the units involved in the ADCS; hence, it is divided into 4 modules: Reaction Wheels, Gas Thrusters, Attitude Determination Sensors and IMU. Therefore, these modules will be treated separately. In this section, the logic of the FDI for the reaction wheels will be explained. Firstly, the FDI architecture is presented, along with the checks that are used to perform failure detection. Later, the Simulink model will be described.

3.3.1 FDI logic

The Reaction Wheels FDI module is aimed at detecting the failures of the wheels that are listed in the FMECA. Moreover, it is assumed that three temperature sensors and one heater are installed on the unit; hence, their failure scenarios are also part of this module.

The first step which is necessary to design the FDI logic is the knowledge of the data contained in the packet coming from the RWs, in order to develop a check methodology. Due to the earliness of the project, this information is not known; thus, some reasonable assumptions will be made, in order to allow for the detection of the selected scenarios. As it was done for the main thruster, it assumed the all the data of the packet will be contained in bits, with binary values (0/1), and they will be followed by control bits, generated using the CRC method, see Section 3.2.1. The preliminary version of the data package from the RWs is shown in Table 3-5.

As it can be seen, certain information in the packet is referred to the whole set of RWs (validity flag, response time, mode); other data are specific to each wheel (voltage, commanded speed, speed). The FDI logic of the module is shown in Figure 3-7. The validity flag and the response time are used to detect a speed controller failure (scenario RW.1 in the FMECA). The FDI logic following these checks is divided into several parts: the first part is dedicated to the detection of proper wheels failures, the second is dedicated to the failures of the temperature sensors, the third to failures of the RWs temperature or the heaters. The logic used to perform the three checks showed in yellow in Figure 3-7 is worth discussing.

Table 3-5: data in the Reaction Wheels packet

Data	Value	Bits	Description
RW validity flag	0/1	1	The value 1 indicates that the validity flag is positive, the value 0 that it is negative.
RW response time	0-10 [s]	4	The response time of the RW speed controller, it must be an integer.
RW mode	0/1	1	0 if RWs are in idle, 1 if they are active.
RW1 voltage	0-20 [V]	5	The voltage of RW1.
RW1 commanded speed	0-2000 [rpm]	12	The speed command received by RW1.
RW1 speed	0-2000 [rpm]	12	The current speed of RW1
RW2 voltage	0-20 [V]	5	The voltage of RW2.
RW2 commanded speed	0-2000 [rpm]	12	The speed command received by RW2.
RW2 speed	0-2000 [rpm]	12	The current speed of RW2.
RW3 voltage	0-20 [V]	5	The voltage of RW3.
RW3 commanded speed	0-2000 [rpm]	12	The speed command received by RW3.
RW3 speed	0-2000 [rpm]	12	The current speed of RW3.
Sensor 1 state	0/1	1	0 if the sensor is off, 1 if it is on.
T sensor 1	+/- 100 [°C]	8	The temperature of the RWs as detected by temperature sensor 1.
Sensor 2 state	0/1	1	0 if the sensor is off, 1 if it is on.
T sensor 2	+/- 100 [°C]	8	The temperature of the RWs as detected by temperature sensor 2.
Sensor 3 state	0/1	1	0 if the sensor is off, 1 if it is on.
T sensor 3	+/- 100 [°C]	8	The temperature of the RWs as detected by temperature sensor 3.
Heater voltage	0-9 [V]	4	The voltage given to the heater installed on the RWs.

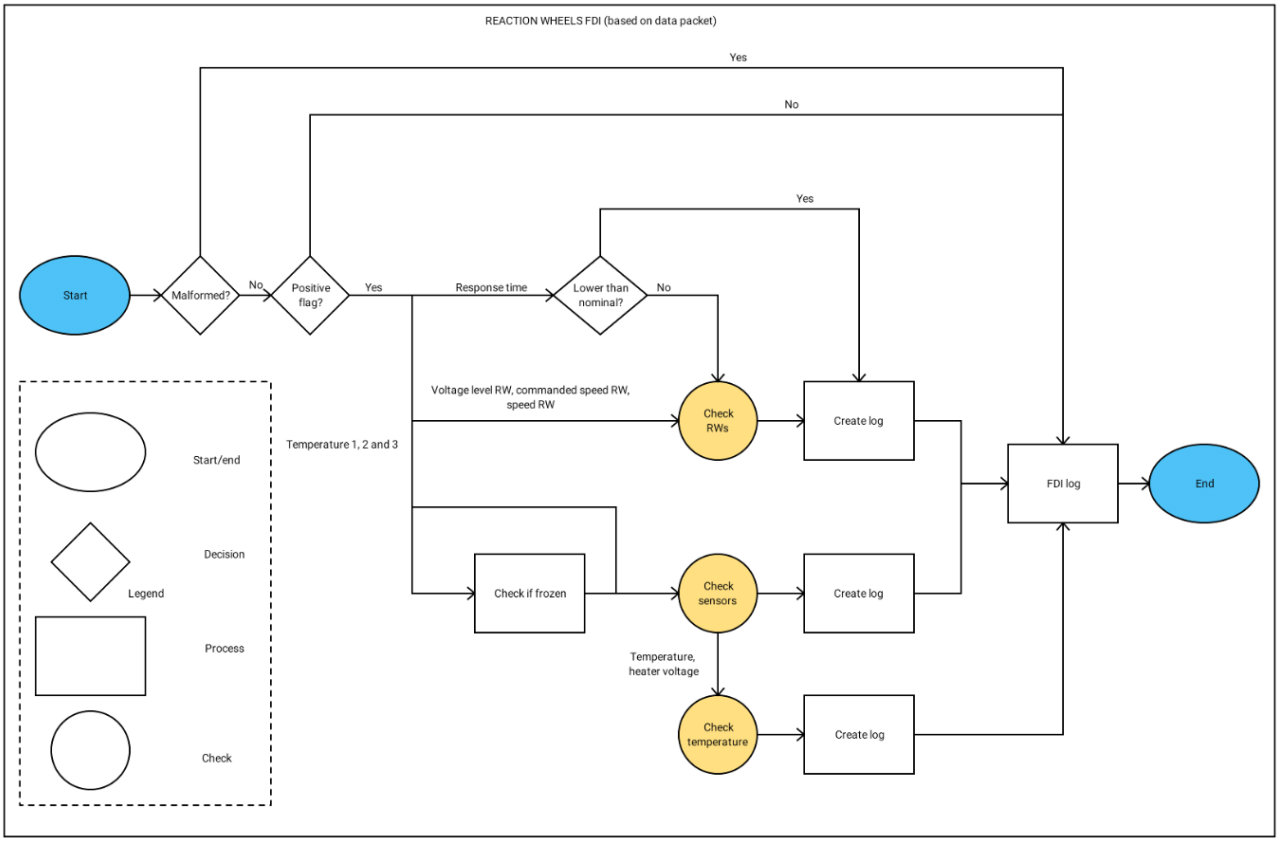

Figure 3-7: RW FDI architecture

During the FMECA, only the simplest failure scenarios of the RWs were selected for this stage; these scenarios can be detected with simple checks, e.g. voltage below a threshold. Hence, the logic is simple and can be realized with a truth table, shown in Table 3-6. In the future stages, the possibility of detecting other failures will be included, using the approximation of the actuator's dynamics through transfer functions [33].

The second check is used to detect failures of the temperature sensors; since the same logic described for the Main Thruster module in Section 3.2.1 is used, it will not be described. Finally, in the last check, the temperature of the RWs and the voltage of the heater are cross-checked, in order to detect eventual over/underheating scenarios or failures of the heater. A simple logic has been used and it is shown in Figure 3-8. If the temperature is above the maximum allowed value, the heater voltage is checked: if the heater is on, it's a heater failure (locked output, scenario HEAT.2 in the FMECA); otherwise, it is overheating (scenario RW.5 in the FMECA). Vice versa, in case the temperature is below the limit, if the heater is off the scenario HEAT.1 is flagged; otherwise, the underheating failure (RW.6) is detected. The allowed temperature range for the RWs are from -20 to 70 [°C] when active, from -40 to 80 [°C] when in idle, according to [1].

Table 3-6: truth table used to detect failures of the RWs

Condition	**Value (true/false)**			
Voltage below threshold	F	T	F	F
Actual speed differs from commanded speed	F	-	T	F
Actual speed is constant	-	-	T	T
Scenario	No failure	Under-voltage	Locked speed	General failure
FMECA ID	/	RW.2	RW.3	RW.4

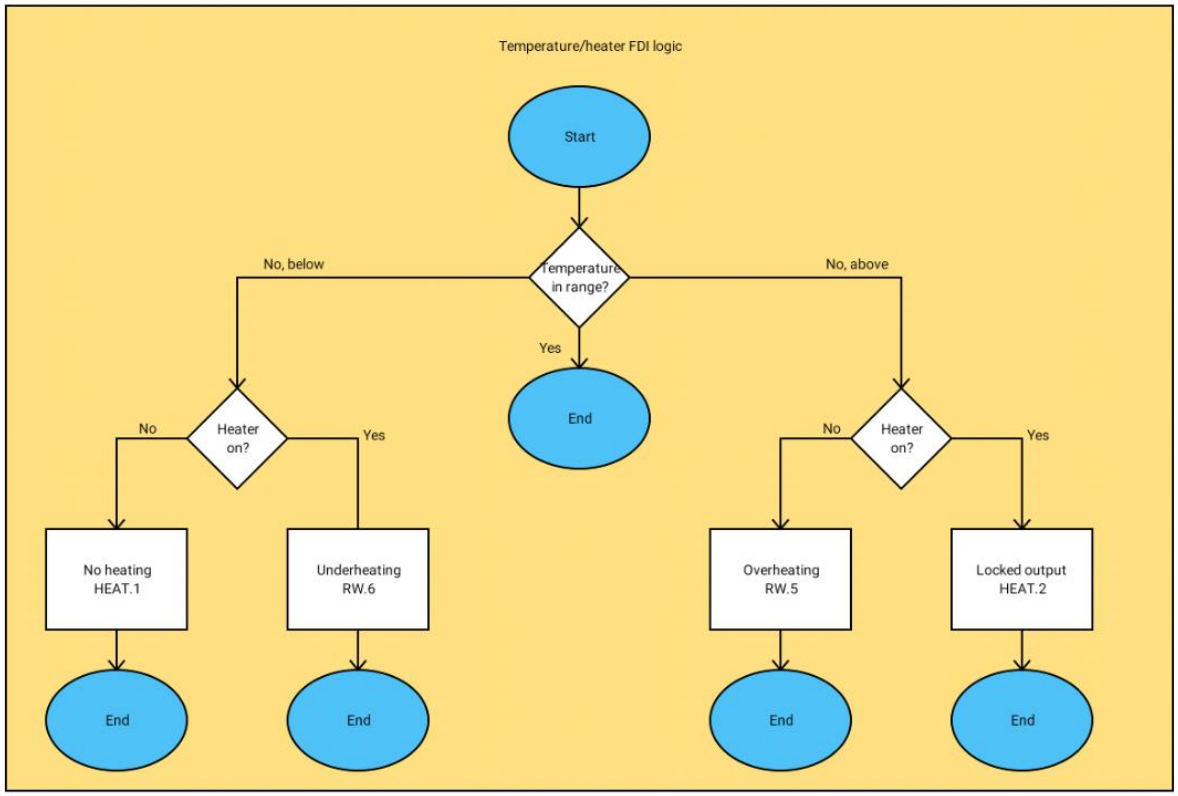

Figure 3-8: logic used in the temperature and heater check

3.3.2 Simulink model

The reaction wheels module takes the packet coming from the RWs as input and returns the FDI log as an output. The log is an array of five elements [a, b, c, d, e], all integers. The first three

elements of the log, **a**, **b** and **c**, are referred to the RW1, RW2 and RW3, respectively, whereas **d** refers to the temperature sensors of the RWs and **e** refers to the temperature of the system. The meaning of the possible values of a, b, c, d and e are explained in Table 3-7. It can be noticed that, in case of malformed data package, negative validity flag or speed controller fault, the elements have the same value, as these scenarios are considered as failures of the whole system.

The module reproduces the logic described in this section. The detection of malformed data

Table 3-7: Reaction Wheels FDI log

Element	Value	Meaning
All	0	No failure
	1	Malformed data package
	2	Negative validity flag
	3	Speed controller fault
a/b/c	4	RW: Under-voltage
	5	RW: Locked output
	6	RW: General failure
d	4	Sensor 1 - general failure
	5	Sensor 1 -frozen
	6	Sensor 2 - general failure
	7	Sensor 2 -frozen
	8	Sensor 3 - general failure
	9	Sensor 3 -frozen
	10	Unidentified failure
e	4	Overheating
	5	Heater – locked output
	6	Underheating
	7	Heater – no heating

package and validity flag is built as in the Main Thruster module, shown in Section 3.2.2; hence, they will not be described again. The inputs enter in several Stateflow charts, as shown in Figure C-6, in Appendix C. Three Stateflow charts are dedicated to the RWs, while the final chart implements the checks related to the thermal control.

The charts dedicated to the RWs implement the decision logic described in this section, through a truth table equal to Table 3-6. The Stateflow model is documented in Figure C-7.

The fourth Stateflow chart is dedicated to thermal control. There are two main sub-modules: one for the temperature sensors, one for the temperature of the RWs and the heaters. The part related to the temperature sensors is the same as the one for the main thruster; thus, it will not be described. The part regarding the temperatures follows the logic described in Figure 3-8, to distinguish between temperature failures and heater failures. The Stateflow chart is documented in Figure C-8, in Appendix C.

3.4 ADCS module – Gas Thrusters

The second module internal to the ADCS module is dedicated to the gas thrusters. In this section, the logic of this module is explained. Firstly, the FDI architecture is presented, along with the checks that are used to perform failure detection. Later, the Simulink model will be described.

3.4.1 FDI logic

The Gas Thrusters FDI module is aimed at detecting the failures of the gas thrusters that are listed in the FMECA. As it can be seen from the FMECA table, the thermal failures for this system are not considered, as the gas thrusters belong to the same unit of the main thruster, and the thermal control part has already been treated as part of the Main Thruster FDI module.

The data packet coming from the gas thruster system is conceived similar to the one from the main thruster and can be seen in Table 3-8. As for the main thruster, some remarks can be made.

Firstly, the thrust command is worth discussing. In [1], the nominal thrust that can be produced by each thruster is assumed to be 10 [mN]. Hence, in this Thesis it is assumed that 10 [mN] is the maximum command, and any value above it would trigger the detection of the “command out of range” failure scenario (GT.2 in the FMECA).

Another uncertain point in the data package is the information about the propellant budget, which is essential for the detection of eventual failures at this stage, in order to cross-check with the valve state and the thrust command; in particular, the crucial information is the propellant trend (constant or decreasing). At the current stage, it is not known if the propellant will be stored in the liquid state

Table 3-8: data contained in the packet coming from the gas thrusters

Data	Value	Bits	Description
GT validity flag	0/1	1	The value 1 indicates that the validity flag is positive, the value 0 that it is negative.
GT1 command	0-15 [mN]	4	The thrust command received by GT1.
GT1 valve state	0/1	1	Output of the valve sensor of GT1: 1 if valve is open, 0 if close.
GT2 command	0-15 [mN]	4	The thrust command received by GT2.
GT2 valve state	0/1	1	Output of the valve sensor of GT2: 1 if valve is open, 0 if close.
GT3 command	0-15 [mN]	4	The thrust command received by GT3.
GT3 valve state	0/1	1	Output of the valve sensor of GT3: 1 if valve is open, 0 if close.
GT4 command	0-15 [mN]	4	The thrust command received by GT4.
GT4 valve state	0/1	1	Output of the valve sensor of GT4: 1 if valve is open, 0 if close.
Propellant budget	0-210 [g]	8	The propellant budget of the gas thrusters.

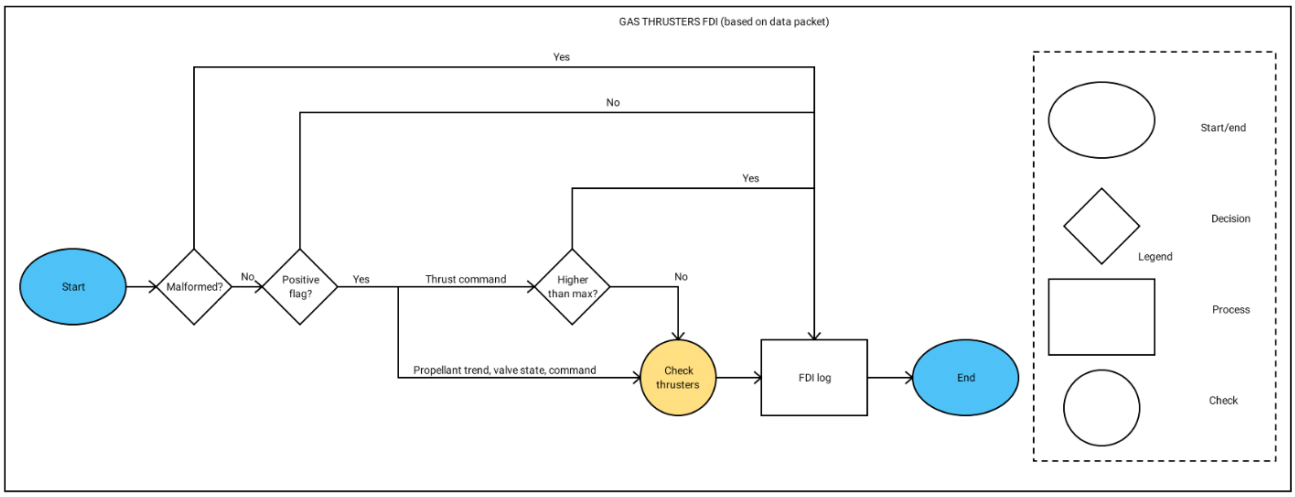

Figure 3-9: Gas Thrusters FDI architecture

or gaseous state, despite in [1] it is assumed that liquefied butane will be used. In case the propellant will be stored in the liquid state, the considerations made for the main thruster's propellant trend in Section 3.2 are valid. In case of storage in the gaseous state, the gas law can be used for the estimation, in a similar fashion to the pVT method. However, in case the possibility of measuring the propellant budget will not be included in the final design, failure detection can still be achieved with the analysis of the changes of the S/C dynamics while firing the thrusters. This possibility will not be explored in this Thesis but left as a recommendation for the future stages.

The FDI block diagram is shown in Figure 3-9; it is the same as the one for the Main Thruster FDI, shown in Figure 3-5, without the checks relative to the temperature sensors and the temperature of the unit. Hence, only one check is executed, to detect any failure of the gas thrusters. A similar logic is used for this check, based on the comparison between the thrust command, the valve state and the propellant trend. However, the application of this methodology has some limitations in the case of the gas thrusters, as the four thrusters share the same propellant tank. Therefore, two different checks can be made: in case the system is in "firing" mode, i.e. at least one gas thruster is active, or when the system is in idle (the command is 0 for all the thrusters).

In the first case, when the command of at least one thruster is higher than 0, there are situations in which a failure is detected, but not isolated, as can be seen from the summary made in Table 3-9.

Table 3-9: cases that are used to detect eventual failures of the gas thrusters, when the whole system is in "firing" mode

Propellant trend	Gas thruster command	Gas thruster valve state	Comment
Decreasing (propellant is consumed)	Active (command>1)	Open	The thruster is ok.
		Close	Unidentified failure. It could be a "No thrust failure" if the valve is actually closed, or a "Faulty sensor" failure if the valve is actually open, but there is no way to cross-check.
	Idle (command=0)	Open	Unidentified failure. It could be a "Locked output failure" if the valve is actually open, or a "Faulty sensor" failure if the valve is actually closed, but there is no way to cross-check.
		Close	The thruster is ok.
Constant (no propellant is consumed)	Active (command>1)	Open	Propellant sensor failure, since there is contradiction between the propellant sensor and the valve sensor.
		Close	"No thrust" failure.
	Idle (command=0)	Open	Valve sensor failure, because there is contradiction between the propellant sensor and the valve sensor.
		Close	The thruster is ok.

The cases shown in the table are the baseline of a truth table, which is used to check the behaviour of the thrusters. It should be noticed that there are several ways to achieve isolation of the failure, in the case in which is not identified. One option is to add a redundant sensor for the propellant level or for the valve state; however, this option is deemed unlikely. The study of the S/C dynamics, on the other hand, would make it easy to distinguish between the different failure scenarios, by looking at the effects of the failure on the dynamics.

The check during “idle” mode, when the commands of all the thrusters are 0, is executed following the logic shown in Figure 3-10.

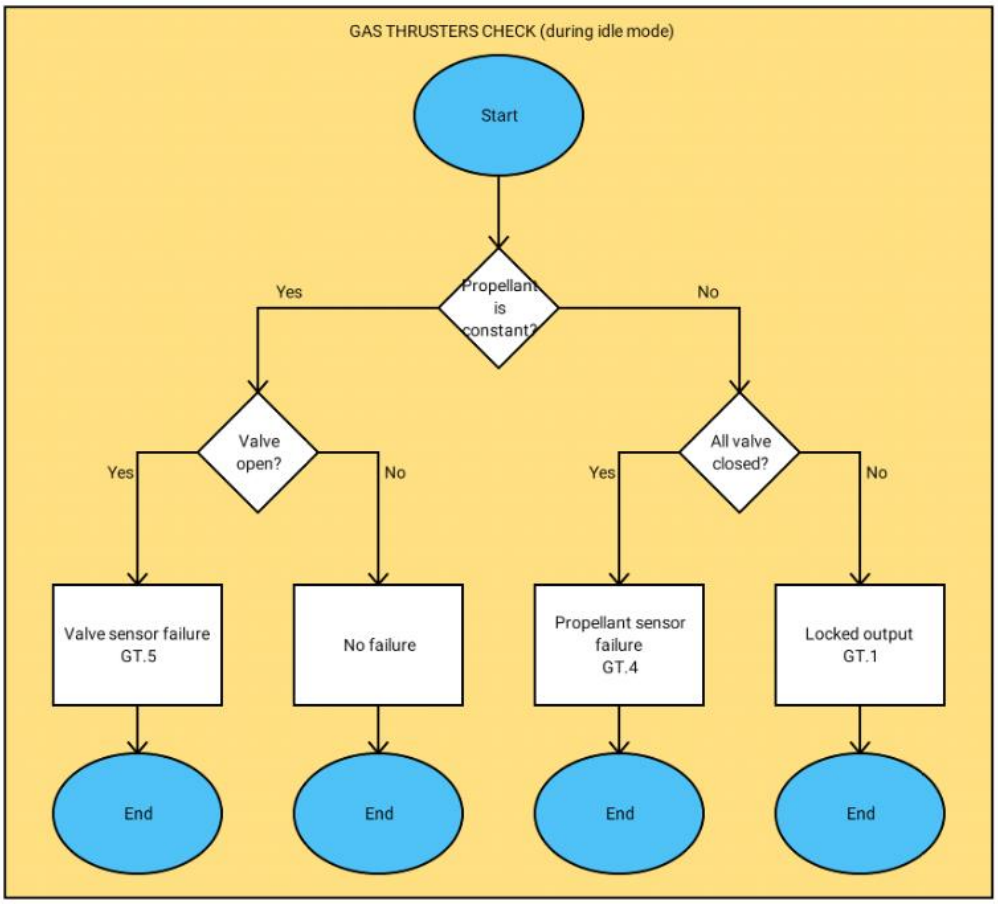

Figure 3-10: logic used to check the gas thrusters during "idle" mode

3.4.2 Simulink model

The gas thrusters module takes the packet coming from the gas thrusters as input and returns the FDI log as an output. The log is an array of four elements [a, b, c, d], all integers. Each element corresponds to one of the four thrusters. The meanings of the possible values of a, b, c and d are shown in Table 3-10.

Table 3-10: Gas thrusters FDI log

Element	Value	Meaning
All	0	No failure
	1	Malformed data package
	2	Negative validity flag
	3	GT: command out of range
	4	Propellant sensor failure
	5	Valve sensor failure
	6	No thrust
	7	Locked output
	8	Unidentified failure

The Simulink module reproduces the logic described in this section. The detection of malformed data package and validity flag is built as in the Main Thruster module, shown in Section 3.2.2; hence, they will not be described again. The Stateflow model is documented in Figure C-9, in Appendix C. From the figure, it can be seen the distinction between “firing mode”, in which the truth table shown in Table 3-9 is used, and “idle”, in which the logic in Figure 3-10 is used.

3.5 ADCS module – Attitude determination

The third module internal to the ADCS module is dedicated to the attitude determination sensors, namely the star-trackers (STs) and the Sun sensors (SSs). The failures related to these units have been grouped since the cross-check is necessary for their detection. In this section, the logic of this module is explained. Firstly, the FDI architecture is presented, along with the checks that are used to perform failure detection. Later, the Simulink model will be described.

3.5.1 FDI logic

The Attitude determination FDI module is aimed at detecting the failures of the star-trackers and Sun sensors that are listed in the FMECA. On top of this, the failures related to the thermal control system of the star-trackers (temperature sensors and heaters) are also considered. However, since the FDI for the thermal failures is the same as the ones described for the reaction wheels in Section 3.3, it will not be explained again.

The data packages conceived for the attitude determination sensors are shown in Table 3-11. Some remarks are necessary. Firstly, despite the Sun sensors on board of LUMIO are two, in this Thesis it will be assumed that only one of them will be working at a time, despite it is not known if this will be the case in the final design. The choice is made as the presence of two SSs is the only full redundancy in the current design of LUMIO; hence, it is possible to experiment the implementation of a cold redundancy in the FDIR system, which might be useful for the future phases, in which more redundant units will be added. In a cold redundancy, the redundant unit is off during the nominal mission, and it is activated only as a recovery action after a failure of the first unit [34].

An additional remark regards the inclusion of the packet coming from the orbital propagator, which will calculate the location of the S/C and all the other NEOs, based on the step-by-step mission description [8]. At the moment, the orbital propagator has not been designed yet; in this Thesis, it is assumed that the propagator will contain data on the expected mission scenario. For instance, data about the expected attitude of the S/C are included, to be used in case an extra cross-check for the attitude sensors is needed. Moreover, the information regarding the expected presence of the Sun in the field of view of the Sun sensor and the star-trackers is contained. This information will be used to cross-check the flag “Sun in field of view”, contained in the packet from each sensor, to verify the presence of a failure. In fact, it is expected that during certain phases of the mission, the Sun will not be in the field of view of the Sun sensor, e.g. during eclipse, or it will enter in the field of view of one star-tracker, e.g. during a manoeuvre; thus, it is important to avoid triggering failures in those cases.

The FDI logic for the Attitude sensor FDI module is shown in Figure 3-11. As can be seen, multiple checks need to be explained. Before performing the main check, used to detect eventual failures is the cross-check, the availability of the units is assessed, in order to understand which units can be cross-checked. A unit is available only if no failure is detected (the data package is not malformed, the validity flag is positive) and if the attitude with respect of the Sun is correct, e.g. the Sun is in the field of view of the Sun sensor.

After the availability checks, the available units are cross-checked. Several situations are possible:

- Cross-check between two star-trackers and one Sun sensor
- Cross-check between two star-trackers
- Cross-check between one star-tracker and one Sun sensor

Table 3-11: content of the packets from the attitude determination sensors. It should be noticed that the "orbital propagator" is not treated as a proper packet since it is assumed that it runs in the same processor of the FDIR. Therefore, the data of the orbital propagator are not converted into bits.

Packet	Data	Value	Bits	Description
Sun Sensor	SS validity flag	0/1	1	The value 1 indicates that the validity flag is positive, the value 0 that it is negative.
	SS sun in FOV	0/1	1	The value 1 indicates that the Sun is in the FoV, the value 0 that the Sun is out of the FoV.
	alpha	0.00-360.00 [°]	17	Azimuth angle of the Sun-vector in the Body reference frame measured by the SS.
	beta	0.00-360.00 [°]	17	Elevation angle of the Sun-vector in the Body reference frame measured by the SS.
Star-tracker 1	ST1 validity flag	0/1	1	The value 1 indicates that the validity flag is positive, the value 0 that it is negative.
	ST1 sun in FOV	0/1	1	The value 1 indicates that the Sun is in the FoV, the value 0 that the Sun is out of the FoV.
	q1	+/-1.0000000	25	The first quaternion measured by ST1, describing the attitude of the body reference frame with respect to the inertial frame.
	q2	+/-1.0000000	25	The second quaternion measured by ST1, describing the attitude of the body reference frame with respect to the inertial frame.
	q3	+/-1.0000000	25	The third quaternion measured by ST1, describing the attitude of the body reference frame with respect to the inertial frame.
	q4	+/-1.0000000	25	The fourth quaternion measured by ST1, describing the attitude of the body reference frame with respect to the inertial frame.
	Sensor 1 state	0/1	1	0 if the sensor is off, 1 if it is on.
	T sensor 1	+/- 100 [°C]	8	The temperature of ST1 as detected by temperature sensor 1.
	Sensor 2 state	0/1	1	0 if the sensor is off, 1 if it is on.
	T sensor 2	+/- 100 [°C]	8	The temperature of ST1 as detected by temperature sensor 2.
	Sensor 3 state	0/1	1	0 if the sensor is off, 1 if it is on.
	T sensor 3	+/- 100 [°C]	8	The temperature of ST1 as detected by temperature sensor 3.
	Heater voltage	0-9 [V]	4	The voltage given to the heater installed on ST1.
Star-tracker 2	Same as packet from ST1			
Orbital propagator	q	Any	\	The vector of quaternions describing the real attitude of the body reference frame with respect to the inertial frame. They are used for cross-check in extreme cases.
	$r_{sun}\|N$	Any	\	The normalized sun vector in the inertial reference frame. It is used to cross-check the STs and the SS.
	SS: Sun in FOV nominal	0/1	\	The value 0 indicates that it is expected that the Sun is out of the SS FoV, the value 1 indicates that the Sun is expected to be in the FoV.
	ST1: Sun in FOV nominal	0/1	\	The value 0 indicates that it is expected that the Sun is out of the ST1 FoV, the value 1 indicates that the Sun is expected to be in the FoV.
	ST2: Sun in FOV nominal	0/1	\	The value 0 indicates that it is expected that the Sun is out of the ST2 FoV, the value 1 indicates that the Sun is expected to be in the FoV.

- Check of one star-tracker only
- Check of one Sun sensor only
- No available units

If no units are available for the cross-check, it means that other failures, e.g. malformed data package, were detected before, in the available checks. However, the probabilities of this scenario

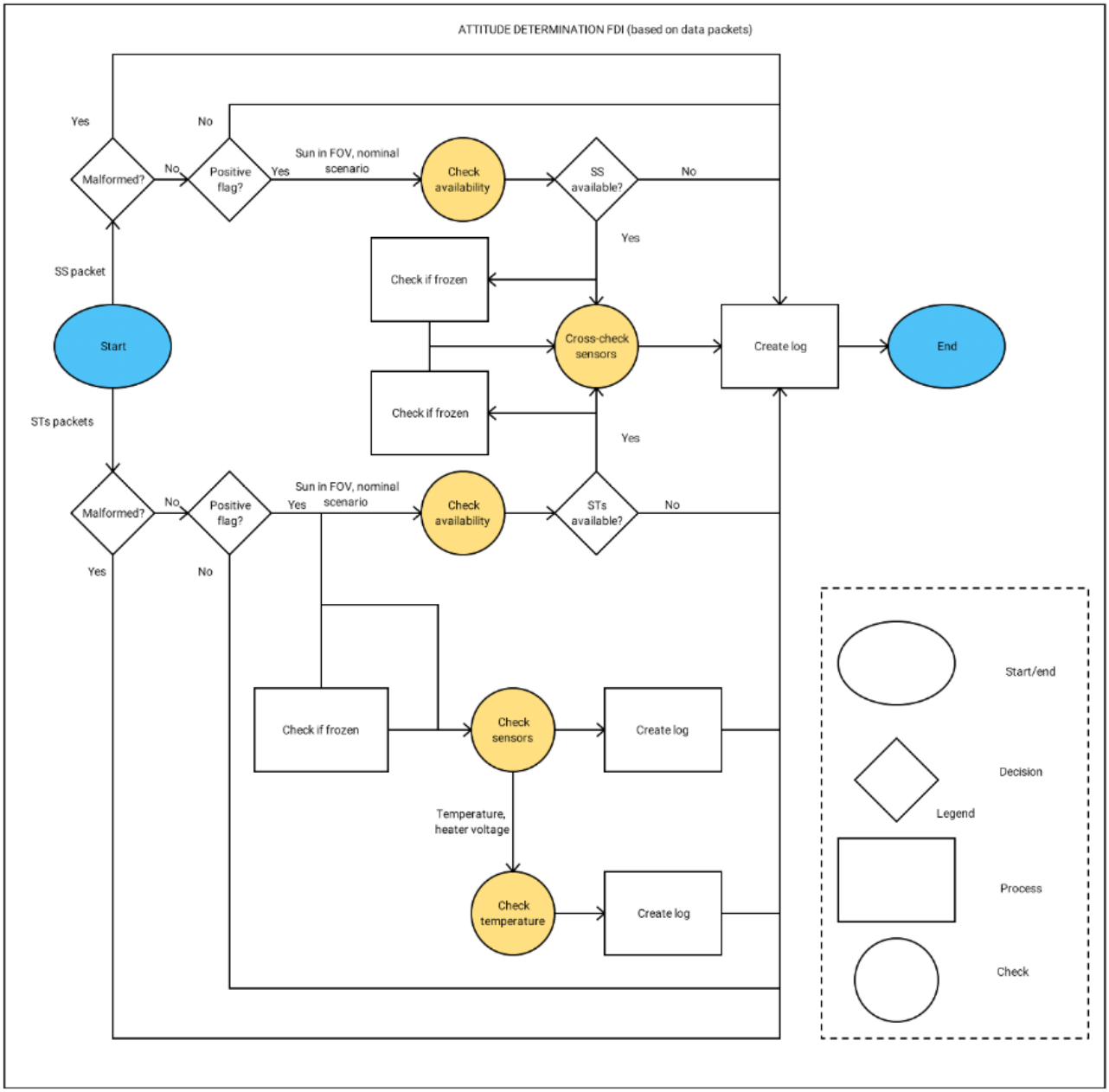

Figure 3-11: Attitude Determination FDI architecture

to occur is considered low, as it would imply the presence of multiple failures.

The scenarios in which only one sensor is available are also considered unlikely but might occur in particular situations, e.g. one star-tracker fails, and the eclipse does not allow to use the Sun sensor. At this stage of the FDIR design, the possibilities are two: either the available sensor is not checked, or the sensor is cross-checked with the orbital propagator, which is considered infallible at this phase of the design. In this Thesis, it was decided to follow two different strategies: it the available sensor is a star-tracker, it is cross-checked with the orbital propagator; otherwise, no extra-check is executed.

If more than one unit is available, it is possible to detect eventual failures in the measurements through a cross-check. If three sensors are available, the most consistent cross-check can be used for detection and the least consistent for isolation, as suggested in [35]; otherwise, the failure can be detected only, and the isolation is performed through cross-checking with the orbital propagator. Therefore, it is necessary to describe the cross-check methodology used to compare two star-trackers and to compare one star-tracker and one Sun sensor.

3.5.1.1 Star-trackers cross-check

Two star-trackers are installed on-board of LUMIO. Despite during some mission phases one of them might be unavailable, due e.g. to the Sun in its field of view, they are expected to work simultaneously for most of the time; hence it can be considered as a hot redundancy [34]. Thus, it is possible to do a cross-check between their measurement, in order to detect eventual failures.

Before describing the sensor's output, the reference frames used in the context of attitude determination must be outlined. Three reference frames are used for LUMIO mission:

- **N**: it is the inertial Earth-centred J2000 reference frame. The x-axis points at the vernal equinox, the z-axis is aligned with the Earth's spin axis and the y-axis completes the orthonormal frame.

- **O**: it is the orbital reference frame, which is centred in the S/C centre of mass. It is defined in [1] as follows: the x-axis (roll axis) is the normalised Moon-pointing vector $r_1 = x_p$, while the z-axis is given by the cross product $r_2 = r_{sun} \times x_p$, where r_{sun} is the Sun-pointing vector. The y-axis (yaw axis) completes the orthonormal frame; therefore, in this reference frame the x-y plane always coincides with the CubeSat-Moon-Sun plane.

- **B**: it is the body reference frame, which is centred in the S/C centre of mass and has fixed axes with respect to the satellite; therefore, it rotates with the S/C.

The star-tracker measure the 3D attitude of the S/C, i.e. it measures the orientation of the body-fixed reference frame **B** with respect to the inertial reference frame **N**. Hence, in this Thesis, the definition of reference frame O will not be used. There are several methods to express the attitude of an object, but the most used are mainly two: Euler angles and attitude quaternions [36]. The main advantage of Euler angles is that they are easy to visualize; hence, they are commonly used to express the nominal attitude. However, Euler angles have a significant drawback, which makes them unsuitable for on-board operations: when considering the S/C kinematic equations, a singularity occurs for specific values of the Euler angles [36]. Therefore, the quaternions are more indicated to be used for the on-board calculations and it is assumed that the output of the star-tracker is given in that form.

The nominal accuracy of the star-tracker quaternion measurement is not known at the moment. It is assumed that this information will be given by the supplier of the unit; however, in the datasheet of the sensors that will be used on-board of LUMIO the only information available about the accuracy is given in terms of the Euler angles [37]. The 3-σ accuracy is given as 10 arcsec for pitch and yaw axis, 120 arcsec for roll axis. One possible approach is to use the relation between Euler angles and quaternions to obtain the accuracy of quaternions, using the same methodology to estimate the propagation of inaccuracy that was explained in Section 3.1.3. However, the transformation from Euler angles to quaternions is complicated, as numerous trigonometric functions are involved. Moreover, the accuracy of the quaternions obtained in this way would not be independent of the attitude information, i.e. it would change continuously. Therefore, the strategy adopted at this stage of the process will be to cross-check the Euler angles, when possible. When it will not be possible, a value of accuracy of the quaternions will be assumed.

In the case of cross-check between two star-trackers, it is possible to compare directly their quaternions measurements. However, since the accuracy available at the moment is given only in terms of the Euler angles, the quaternions coming from the instruments will be converted into Euler angles and the comparison will be done based on them. Thus, the error function as defined in Equation (2) of Section 3.1.3 will simply be made of a vector with three components, which are shown in Equation (10).

$$\begin{cases} err_1 = |(\theta_1)_{ST1} - (\theta_1)_{ST2}| \\ err_2 = |(\theta_2)_{ST1} - (\theta_2)_{ST2}| \\ err_3 = |(\theta_3)_{ST1} - (\theta_3)_{ST2}| \end{cases} \tag{10}$$

Due to the simplicity of the expressions of Equation (10), the computation of the accuracy of the i-th component of the error function, starting from the accuracy of the initial measurements, is straightforward. The result is shown in Equation (11), where the fact that the two star-trackers are the same model was also used.

$$\sigma_{err,i} = \sqrt{(\sigma_i)_{ST1}^2 + (\sigma_i)_{ST2}^2} = \sqrt{2}\sigma_i \text{ for i=1,2,3} \tag{11}$$

Therefore, when cross-checking the two star-trackers, the minimum threshold to compare the i-th component of the error function is $trs_i = 2.5\sqrt{2}\sigma_i$, using the guideline in Table 3-1. From this information, it is possible to derive the minimum loss of accuracy and bias that can be detected without false negatives, as they were defined in Section 3.1.3.

The minimum loss of accuracy of the error function that can be detected is such that $\sigma_{err,f} = 2\ trs$, according to the analysis in Section 3.1.3. Therefore, it is possible to calculate what is the minimum loss of accuracy of one of the initial measurements that can be detected without false negatives. In fact, it is possible to compute *d*, the degradation of the accuracy of ST1, which would generate an accuracy of the error function double to the threshold:

$$\sigma_{err,f,i} = 2trs = 5\sqrt{2}\sigma_i = \sqrt{(d\ \sigma_i)_{ST1}^2 + (\sigma_i)_{ST2}^2}$$

$$5\sqrt{2} = \sqrt{1 + d^2}$$

$$d = 7$$

Hence, the minimum loss of accuracy of the initial measurement that can be detected without false negatives is $(\sigma_\theta)_{degraded} = 7\sigma_\theta$. Hence, given the specifications of the STs of LUMIO, the minimum loss of accuracy that can be estimated is of about 0.006 [deg] for yaw and pitch axes, and 0.08 [deg] for roll axis. This result is line with the required pointing accuracy of the ADCS, which in [1] is reported as 0.1 [deg]. It is easier to compute the bias since the bias of the error function in this simple case is equal to the bias of the initial measurement; therefore, the minimum bias that can be detected with this method is $bias = 2.8\sigma_\theta$, according to the analysis in Section 3.1.3.

3.5.1.2 Star-tracker and Sun sensor cross-check

The Sun sensor gives a 2D attitude information, as it measures the position of the sun-vector, i.e. the vector from the S/C to the Sun.

The cross-check between a Sun sensor and a star-tracker is useful not only to detect eventual failures in the Sun sensor, but also to isolate detected failures of one of the star-trackers. However, comparing the signals from these instruments is more challenging, as their outputs are not directly comparable. The output of the star-tracker are the four attitude quaternions, which represent the orientation of the body-fixed reference frame **B** with respect to the inertial frame **N**. The output of the Sun sensor are two angles, called α (azimuth) and β (elevation), which represent the orientation of the Sun vector in the body reference frame B, $r_{sun}|_B$, in spherical coordinates (the sun vector norm is 1). The convention used to define the spherical coordinates is shown in Figure 3-12. Therefore, it is necessary to develop a methodology to compare these measurements.

It is assumed that the inertial Sun vector $r_{sun}|_N$ is known all along the mission; in fact, the inertial Sun vector from the Earth and the Sun is known at any time. The vector from the S/C to the Sun can be assumed to be the same if the S/C orbit is close to the Earth, or a further transformation can be made to increase the accuracy, based on the position of the S/C with respect to the Earth. As this last operation is dependent on navigation, which is not considered in the present study, it will be assumed that the inertial Sun vector will always be available and correct, i.e. with 100% accuracy.

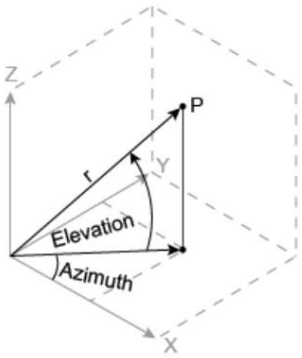

Figure 3-12: convention used for the spherical coordinates. Credits: [38]

Therefore, it is possible, starting from the Star-tracker measurement, to calculate the rotation matrix from the reference frame B to reference frame N [36]:

$$R^{N/B} = \begin{bmatrix} 1-2(q_2^2+q_3^2) & 2(q_1q_2+q_3q_4) & 2(q_1q_3-q_2q_4) \\ 2(q_2q_1-q_3q_4) & 1-2(q_1^2+q_3^2) & 2(q_2q_3+q_1q_4) \\ 2(q_1q_3+q_2q_4) & 2(q_2q_3-q_1q_4) & 1-2(q_1^2+q_2^2) \end{bmatrix} \tag{12}$$

It should be noticed that the matrix $R^{N/B}$ in Equation (12) is written based on the convention of rotation of the reference system; therefore, in order to rotate the coordinates of a vector from reference B to reference N, the transposed matrix should be used. Hence, it is possible to obtain the Cartesian coordinates of the Sun vector in the reference frame B, as they were measured by the star-tracker. The operation is shown in Equation (13).

$$r_{sun}|_B = \left(R^{N/B}\right)' r_{sun}|_N \tag{13}$$

It is therefore possible to compare this vector with the vector measured by the Sun sensor; however, a further step is necessary: the transformation of the vector in spherical coordinates, in order to compare the azimuth and elevation angles with α and β. In order to do so, the equations for the transformation are as shown in Equation (14).

$$\begin{cases} \alpha_{\mathrm{ST}} = atan2(y,x) \\ \beta_{\mathrm{ST}} = atan2\left(z, \sqrt{(x^2+y^2)}\right) \end{cases} \tag{14}$$

It should be noticed that the four-quadrant inverse tangent function (atan2) is used, instead of the arctangent, since the former returns values in the interval [-π, π], while the latter is limited to the interval [-π/2, π/2]. Moreover, a further operation is needed, to convert the angles from radians to degrees. Finally, it is possible to compute the error function, which will have two components:

$$\begin{cases} err_1 = |\alpha_{ST} - \alpha| \\ err_2 = |\beta_{ST} - \beta| \end{cases} \tag{15}$$

Once the error function has been defined, the thresholds must be selected. To do so, the methodology explained in Section 3.1.3 should be applied; however, it is evident that in this case the number and complexity of the operations make it harder to compute the value of the standard deviation of the error function. In order to ease the calculations, the common engineering approximation of Equation (4) can be used, which assumes that the initial variables are symmetrical and not correlated [39].

$$\sigma_f \sim \sqrt{\left(\frac{\partial f}{\partial x_1}\right)^2 (\sigma_{x1})^2 + \cdots + \left(\frac{\partial f}{\partial x_n}\right)^2 (\sigma_{xn})^2} \tag{16}$$

In Equation (16), f is the function, which depends on n parameters x_i that are the initial measurements. In the case studied in this Section, f is the error function and the parameters x_i are 6: the attitude quaternions measured by the star-tracker and the azimuth and elevation angles from the Sun sensor.

It should be noticed that, at the moment, it is not known whether the errors of the quaternion measurements are correlated or not, although it is reasonable to assume that they are. However, the approximation of Equation (16) can be considered valuable since the nominal errors of the star-tracker measurements are small compared to other sensors; thus, the correlation terms will be even smaller and possibly neglectable. Therefore, the expression of Equation (17) can be obtained:

$$\begin{cases} \sigma_{err1} \sim \sqrt{\sum_{i=1}^{4} \left(\frac{\partial err_1}{\partial q_i}\right)^2 (\sigma_{qi})^2 + \left(\frac{\partial err_1}{\partial \alpha}\right)^2 (\sigma_\alpha)^2} \\ \sigma_{err2} \sim \sqrt{\sum_{i=1}^{4} \left(\frac{\partial err_2}{\partial q_i}\right)^2 (\sigma_{qi})^2 + \left(\frac{\partial err_2}{\partial \beta}\right)^2 (\sigma_\beta)^2} \end{cases} \tag{17}$$

Despite the simplification, the computation of the standard deviation remains challenging. In particular, it can be noticed that the partial derivatives with respect to the quaternions will depend on the current values of the attitude quaternions; hence, the standard deviation of the error function is not constant but depends on the current attitude. The most correct approach to calculate the value of σ_i is therefore to continuously compute it, based on the current attitude. However, this approach would lead to increased complexity of the cross-check algorithm.

An alternative approach can be followed, based on a comparison between the star-tracker and the Sun sensor. These instruments have, in fact, different accuracies: in particular, star-trackers are very accurate sensors, with accuracy in the order of 10 arcsec (3-σ), while the accuracy of the Sun sensor on-board of LUMIO mission is of about 0.5 degrees (3-σ) [40]. Hence, an additional approximation can be made: since the contribution of the Sun sensor's measurements are expected to be significantly higher (in nominal conditions), the expressions in Equation (17) can be simplified into those of Equation (18):

$$\begin{cases} \sigma_{err1} \sim \sigma_\alpha \\ \sigma_{err2} \sim \sigma_\beta \end{cases} \tag{18}$$

The result of Equation (18) could be expected since, when comparing two instruments with large differences in accuracy, the more influent on the comparison is the less accurate one. Following the guidelines that were derived in Section 3.1.3, the minimum thresholds to verify the error function are $trs_1 = 2.5\sigma_\alpha$ and $trs_2 = 2.5\sigma_\beta$. However, a lot of simplifying assumptions were made so far; therefore, it is deemed more reasonable to select $trs_1 = 3\sigma_\alpha$ and $trs_2 = 3\sigma_\beta$, in order to prevent an excessive rate of false positives. Nevertheless, it will be necessary to verify, through simulations and tests, that this choice respects the requirements about the rate of false positives.

From the selected threshold, it is possible to estimate the minimum loss of accuracy and bias that can be detected without false negatives, as they were defined in Section 3.1.3. The minimum loss of accuracy of the error function that can be detected is such that $\sigma_{err,f} = 2$ trs. Using Equation (18), it can be concluded that the same loss of accuracy can be detected for the Sun sensor, i.e. an accuracy 6 times larger of the nominal. Instead, a bias of about 3.3 times the nominal standard

deviation can be detected. Regarding eventual failures of the star-tracker, it is evident that, through this cross-check, it is possible to detect only significant degradations of its performance, despite it is difficult to estimate them.

In conclusion, this cross-check is an important instrument to check the behaviour of the Sun sensor, but a weak tool to detect failures in the star-tracker. Therefore, priority should be given to the cross-check between two star-trackers and, in case a failure is detected, the cross-check with the Sun sensor can be used to isolate the failure. In case the failure is too little to be detected, the error functions obtained from the two cross-checks (ST1-SS and ST2-SS) can be compared, and the highest one can be used to locate the faulty unit.

3.5.2 Simulink model

The attitude determination module takes the packet coming from the attitude sensors as input and returns the FDI log as an output. The log is an array of seven elements [a, b, c, d, e, f, g], all integers. The first three elements of the log, **a**, **b** and **c**, are referred to SS, ST1 and ST2, respectively. The remaining elements of the log refer to the thermal control systems of the star-trackers: **d** and **f** refer to the temperature sensors of ST1 and ST2, respectively, while **e** and **g** refer to their temperature (e for ST1, g for ST2). The meanings of the possible values of a, b, c, d, e, f and g are explained in Table 3-12.

The Simulink module reproduces the logic described in this section. The detection of malformed data package and validity flag is built as in the Main Thruster module, shown in Section 3.2.2; hence, they will not be described again. The data from the Sun sensor and the star-trackers enter an “availability check”, as it was described in Section 3.5.1. The implementation in Simulink of the Sun sensor availability check is shown in Figure C-10, in Appendix C, while the star-tracker availability check is documented in Figure C-11. After this check, the data enter in the block shown in Figure C-12, where the other relevant checks are implemented in three Stateflow charts: in the first the cross-check is executed, while the other two are related to the thermal checks for ST1 and ST2. The implementation of the thermal checks in Stateflow follows the same principle described for the Reaction Wheels Module in Section 3.3.

The Stateflow model where the cross-checks are implemented is divided into several states, which are entered based on the availability state of the three sensors, as it is shown in Figure C-13. If two

Table 3-12: Attitude determination FDI log

Element	Value	Meaning
All	0	No failure
	1	Malformed data package
	2	Negative validity flag
a	3	SS: Sun not in FoV failure flag
	4	SS: Sun not in FoV
b/c	3	ST: Sun in FoV failure flag
	4	ST: Sun in FoV
a/b/c	5	SS/ST: general failure
	6	SS/ST: frozen sensor
	7	SS/ST: unidentified failure. It is recommended to use the S/C dynamics.
	8	SS/ST: not enough sensors for cross-check
d/f	3	ST: temperature sensor 1 - general failure
	4	ST: temperature sensor 1 -frozen
	5	ST: temperature sensor 2 - general failure
	6	ST: temperature sensor 2 -frozen
	7	ST: temperature sensor 3 - general failure
	8	ST: temperature sensor 3 -frozen
	9	Unidentified failure
e/g	3	ST: Overheating
	4	ST: Heater – locked output
	5	ST: Underheating
	6	ST: Heater – no heating

star-trackers are available, the logic in Figure C-14 is used: if the Sun sensor is also available, a triple cross-check is executed; otherwise, the two star-trackers are checked with the aid of the orbital propagator. In case only one star-tracker is available, the logic in Figure C-15 is used: the unit is compared with the orbital propagator and the Sun sensor, if possible. Finally, Figure C-16 shows the logic in case no star-tracker is available, and no check is executed. The equations for the cross-checks are implemented in custom MATLAB functions inside the model.

3.6 ADCS module – IMU

The fourth module internal to the ADCS module is dedicated to the IMU. In this section, the logic of this module is explained. Firstly, the FDI architecture is presented, along with the checks that are used to perform failure detection. Later, the Simulink model will be described.

3.6.1 FDI logic

The IMU FDI module is aimed at detecting the failures of the IMU that are listed in the FMECA. Despite the IMU comprises gyroscopes and accelerometers, the failure scenarios of the latter units were not considered at the current stage, due to the absence of cross-check options and to the impossibility of using the S/C dynamics to check eventual failures.

The data package coming from the IMU is shown in Table 3-13. As can be seen, the measurements of the accelerometers are included, despite they will not be checked. The FDI logic is shown in Figure 3-13. Some remarks are necessary. Firstly, since it is assumed that the IMU has only one central micro-controller or processor for all its sensors, in case of negative validity flag the whole unit will be considered faulty.

Another remark regards the detection of the failure scenario DEP.1 in the FMECA, namely “Excessive tumbling rate”. Despite this is a failure of the deployment system of the S/C, it can be detected only through the gyroscopes, which measure the angular speed. Hence, it is included in the IMU FDI.

Finally, from Figure 3-13, it can be noticed that only the gyroscopes are checked. Since no direct cross-check option, i.e. another IMU, is available and the S/C dynamics is not used, the only possibility at the current stage was to introduce a cross-check between the gyroscopes of the IMU and the attitude quaternions measured by the STs. However, this choice has several limitations. The main disadvantage is that the STs can work only at low rotational speed [37], lower than 1 [°/s], while the phases of the mission in which the role of the IMU is crucial are those in which the angular speed is high, e.g. during de-tumbling. Moreover, when the cross-check is possible, a high number of operations is needed to compute the error function, including an integration; thus, the cross-check will not be very accurate. Nevertheless, due to the absence of other options in the design, this cross-check will be implemented, with the recommendation of adding extra check options in the future, e.g. a redundant IMU.

Table 3-13: data contained in the packet coming from the IMU

Data	Value	Bits	Description
IMU validity flag	0/1	1	The value 1 indicates that the validity flag is positive, the value 0 that it is negative.
Accelerometer 1 output	+/- 10 [g]	5	The acceleration measured by accelerometer 1.
Accelerometer 2 output	+/- 10 [g]	5	The acceleration measured by accelerometer 2.
Accelerometer 3 output	+/- 10 [g]	5	The acceleration measured by accelerometer 3.
Gyroscope 1 output	+/- 400.00 [°/s]	17	Output of gyroscope 1, i.e. ω_1.
Gyroscope 2 output	+/- 400.00 [°/s]	17	Output of gyroscope 2, i.e. ω_2.
Gyroscope 3 output	+/- 400.00 [°/s]	17	Output of gyroscope 3, i.e. ω_3.

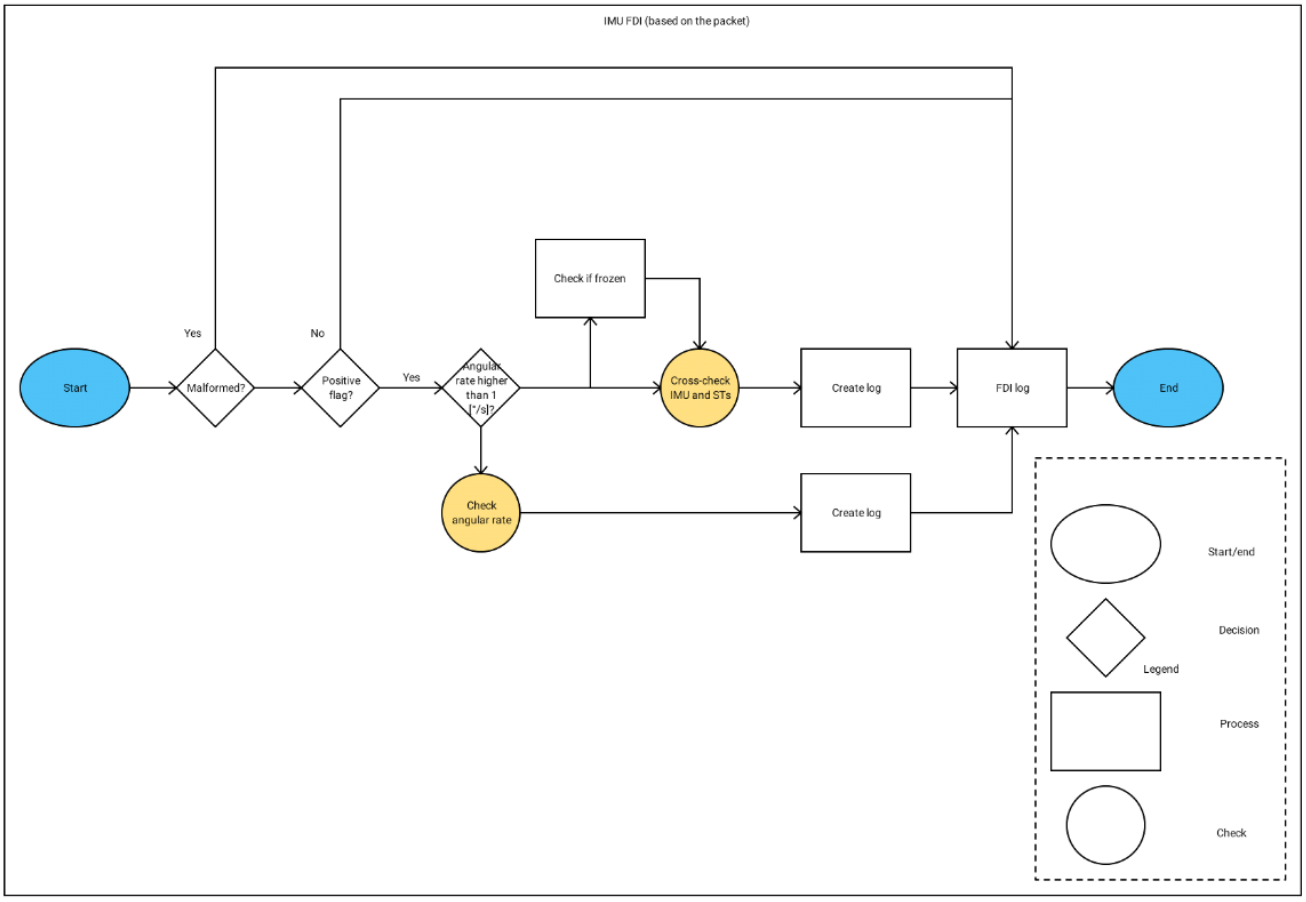

Figure 3-13: IMU FDI architecture

3.6.1.1 IMU and star-tracker cross-check

In this section, the cross-check methodology outlined in Section 3.1.3 will be applied to compare the outputs of the IMU and the star-tracker.

As the cross-check between star-tracker and IMU is used only to check the behaviour of the IMU, not for the star-tracker, it will be executed only when the signal from this latter unit will be available and correct: the FDI log for the star-tracker will be controlled and the cross-check will be executed only if no failure is flagged. In case a cross-check is available, the first operation to be performed is the calculation of the quaternion rates, from the measurements of the quaternions (ST) and angular rates (IMU), as shown in Equation (19), taken from [36].

$$\begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \\ \dot{q}_3 \\ \dot{q}_4 \end{bmatrix} = \frac{1}{2} \begin{bmatrix} 0 & \omega_3 & -\omega_2 & \omega_1 \\ -\omega_3 & 0 & \omega_1 & \omega_2 \\ \omega_2 & -\omega_1 & 0 & \omega_3 \\ -\omega_1 & -\omega_2 & -\omega_3 & 0 \end{bmatrix} \begin{bmatrix} q_1 \\ q_2 \\ q_3 \\ q_4 \end{bmatrix} \tag{19}$$

The operation of Equation (19) is not sufficient to define an error function for the cross-check since the time derivative of the quaternions is not known. Therefore, a further step is needed: the time derivative of the quaternions will be integrated, to obtain the expected values of the quaternions in the next measurement. Later, the expected values can be compared with the ones measured by the star-tracker. However, the integration of the time derivatives of the quaternions is a delicate operation, as it will introduce a further error, due to the difference between the real system and the discrete-time integration. Hence, the appropriate methodology shall be selected. The easiest option is to use the forward Euler's integration formula, as shown in Equation (20), or the backward Euler's formula of Equation (21). However, in order to decrease the integration error, another method can be chosen, while keeping the calculations simple, i.e. the Crank-Nicholson method, shown in Equation (22). More complex formulas, e.g. the Simpson method, cannot be applied, as

they require the knowledge of the quaternion derivatives at a higher number of time samples inside the interval [t, t+Δt].

$$((q_i)_{t+\Delta t})_{exp} = (q_i)_t + \Delta t\,(\dot{q_i})_t \tag{20}$$

$$((q_i)_{t+\Delta t})_{exp} = (q_i)_t + \Delta t(\dot{q_i})_{t+\Delta t} \tag{21}$$

$$((q_i)_{t+\Delta t})_{exp} = (q_i)_t + \frac{\Delta t}{2}[(\dot{q_i})_t + (\dot{q_i})_{t+\Delta t}] \tag{22}$$

Once the expected value of the quaternions at the next time step is computed, it can be compared with the value measured by the ST, to define the error function of Equation (23).

$$err_i = |(q_i)_{t+\Delta t} - ((q_i)_{t+\Delta t})_{exp}| \text{ for i=1, ..., 4} \tag{23}$$

Once the error function is defined, the threshold must be selected. A procedure will be outlined for the selection of trs_1, i.e. the threshold for the error function err_1, and later it will be extended to obtain the other three thresholds.

In order to define the threshold, the standard deviation of the error function should be calculated first. Since the angular velocity's measurements are obtained from three different gyroscopes, it can be assumed that their errors will not be correlated. This is a weak assumption, which requires an experimental proof since the three sensors are installed on the same unit, but it is useful because it allows using the approximated formula of Equation (16) to compute the standard deviation of the error function. The result is shown in Equation (24), where it can be seen that two distinct parts contribute to the final error: the error propagated from the initial measurements and the integration error, here called σ_{int}.

$$\sigma_{err1} \sim \sqrt{2(\sigma_{q1})^2 + \Delta t \sum_{i=1}^{4}\left(\frac{\partial(\dot{q_1})}{\partial q_i}\right)^2 (\sigma_{qi})^2 + \Delta t \sum_{i=1}^{3}\left(\frac{\partial(\dot{q_1})}{\partial \omega_i}\right)^2 (\sigma_{\omega i})^2 + (\sigma_{int})_1} \tag{24}$$

It can be easily verified that Equation (24) holds for all the three different integration methods that were shown above. Therefore, the choice of the integration method affects only the integration error, not the propagation of inaccuracy from the initial measurements; thus, it is more convenient to choose the Crank-Nicholson method shown in Equation (22), as it yields a lower error. In fact, the integration error caused by this method can be approximated as in Equation (25), taken from [41],where *c* is a value of time in the interval [t, t+Δt]:

$$\sigma_{int} = \frac{\ddot{q}_1(c)}{12}(\Delta t)^3 \tag{25}$$

Regarding the first part of Equation (24), despite the simplifying assumptions, the computation of the standard deviation remains hard. In particular, it can be noticed that the partial derivatives with respect to the quaternions will depend on the current values of the angular velocity and vice versa; hence, the standard deviation of the error function is not constant but depends on the current attitude and angular rate. The most correct approach to calculate the value of σ_1 is, therefore, to continuously compute it, based on the current data. However, this approach would lead to increased complexity of the cross-check algorithm. In order to define a fixed value for the threshold, some additional considerations can be made.

Firstly, despite there is not a definitive value for the accuracy of the gyroscopes, it can be assumed that it will be significantly higher than the accuracy of the star-tracker. The calculation of the accuracy of the IMU gyroscopes is made challenging by the vast number of possible errors that should be considered, e.g. random walk error, noise, etc. Despite the required accuracy of the gyroscopes is not known, it is possible to estimate their accuracy from the datasheet. In fact, the two most relevant contributes to the error are the scale-factor error and the bias. From the datasheet of the IMU [42], the scale-factor error is in the order of 500 [ppm], while the maximum nominal bias is 250 [°/h]. Hence, at angular rates around 1 [°/s], i.e. at angular rates that allow to cross-check the IMU and the ST, a good approximation of the accuracy is 0.15 [°/s] (3-σ). From the comparison with the star-tracker accuracy, it can be concluded that the ST is significantly more accurate. Hence, it is considered possible to simplify the expression in Equation (24), by neglecting the contributes of the accuracy of the quaternion's measurements. Nevertheless, the resulting expression would still depend on the current attitude, as it can be seen from Equation (26).

$$\sigma_{err1} \sim \sqrt{\frac{\Delta t}{4}\left(q_4{}^2(\sigma_{\omega 1})^2 + q_3{}^2(\sigma_{\omega 2})^2 + q_2{}^2(\sigma_{\omega 3})^2\right)} + (\sigma_{int})_1 \tag{26}$$

In order to obtain a fixed value for the threshold, the first step is the neglection of the integration error, due to the impossibility to estimate its value; this will lead to an underestimation of σ_{err1}. Secondly, the dependence of the value of σ from the quaternions should be removed. Two options are possible: the first is to approximate the quaternion weights in Equation (26) to 1, i.e. the maximum possible. Since the norm of the quaternion vector is always 1, this operation would lead to overestimate σ_1 excessively. The second option, which is deemed better since it does not involve any approximation, is to study the norm of the error vector, instead of all the components separately. Therefore, the norm of the error vector becomes the new error function: this is possible because the norm is 0 in nominal conditions, see Equation (27).

$$err = \sqrt{err_1^2 + err_2^2 + err_3^2 + err_4^3} \tag{27}$$

The resulting standard deviation would be as in Equation (28): the result is not dependent of the quaternions, as the norm of a quaternion is equal to 1 [36]. Moreover, it was assumed that all the gyroscopes have the same accuracy σ_ω.

$$\sigma_{err} = \sqrt{\sigma_{err1}^2 + \sigma_{err2}^2 + \sigma_{err3}^2 + \sigma_{err4}^2} = \tag{28}$$

$$= \sqrt{\frac{\Delta t}{4}\left(q_1{}^2 + q_2{}^2 + q_3{}^2 + q_4{}^2\right)\left((\sigma_{\omega 1})^2 + (\sigma_{\omega 2})^2 + (\sigma_{\omega 3})^2\right)} = \frac{\sqrt{3\Delta t}}{2}\,\sigma_\omega$$

The time interval is 1 [s]. The use of a smaller time interval would decrease the error propagation and the integration error as well; hence, it is recommended for the next phases.

Due to the multiple approximations that were made, it is deemed reasonable to select a threshold of $trs = \frac{3\sqrt{3}}{2}\sigma_\omega$. The various approximations that were made so far do not allow to estimate the minimum loss of accuracy and bias that can be detected with this cross-check. The values of $(\sigma_\omega)_{degraded} = 3\sqrt{3}\sigma_\omega$ and $bias = (\frac{3\sqrt{3}}{2} + 0.3)\sigma_\omega$ can be hypothesised, based on the results in Table 3-1, but a thorough analysis through testing and simulation shall be performed, to verify that the threshold respects the requirements about the rate of false positives and false negatives. However, some conclusions can already be drawn about the cross-check between the star-tracker and the gyroscopes. The high number of operations that are required, including an integration, make it hard to detect with a good accuracy an eventual failure of the IMU. On top of this, the aforementioned limitations about the simultaneous use of the two units pose an additional challenge. Therefore, this cross-check can be implemented as a method to detect significant

degradations of the IMU, but it cannot be the only tool to use. The addition of checks based on the dynamics of the S/C might be beneficial to increase the capabilities of the FDI system, as it was proposed in [43], but the most efficient way to improve the situation would be the addition of a redundant IMU, to allow for a more accurate cross-check. At the same time, this choice would permit to accommodate eventual failures with the addition of an L2 recovery level (see Chapter 4).

3.6.2 Simulink model

The IMU FDI module takes the packet coming from the IMU and the Attitude determination FDI log as input and returns the FDI log as an output. The log is an array of two elements [a, b], all integers. The first element is referred to the accelerometers of the IMU, the second to the gyroscopes. The meaning of the possible values of **a** and **b** is explained in Table 3-14.

Table 3-14: IMU FDI log

Element	Value	Meaning
All	0	No failure
	1	Malformed data package
	2	Negative validity flag
b	3	S/C: excessive rotational speed
	4	Gyroscopes: general failure
	5	Gyroscopes: frozen sensor
	6	Cross-check not available: it is recommended to use the S/C dynamics.

The Simulink module reproduces the logic described in this section. The detection of malformed data package and validity flag is built as in the Main Thruster module, shown in Section 3.2.2; hence, they will not be described again. The Stateflow model is documented in Figure C-17, in Appendix C. Two states can be distinguished: in case of high angular rates, no cross-check is done, while for low rates the IMU is cross-checked with the star-tracker, as shown in Figure C-18.

3.7 Power module

The Power module comprises failure scenarios of the EPS, the solar panels, the SADA and the batterie. Hence, it can be divided into two sub-modules, one dedicated to the solar panels, the EPS, the SADA, one for the batteries. The reason for grouping these modules into one is twofold. Firstly, it is assumed that all the information is contained in the same package, sent by the EPS. Secondly, the FDI log of all these units is needed by the Deployment FDI.

In this section, the logic of this module is explained. Firstly, the FDI architecture is presented, along with the checks that are used to perform failure detection. Later, the Simulink model will be described.

3.7.1 FDI logic

The Power FDI is aimed at detecting failures of the EPS, the solar panels, the SADA and the batteries. On top of this, the failures related to the thermal control system of the battery system (temperature sensors and heater) are also considered. However, since the FDI for the thermal failures is the same as the ones described for the reaction wheels in Section 3.3, it will not be explained again. I should be noticed that, unlike in the case of the star-trackers, only one thermal control system is assumed for both the batteries, since it is expected that the batteries will be installed close to each other, possibly in a single package.

The first step which is necessary to design the FDI logic is the knowledge of the data contained in the packet coming from the EPS, in order to develop a check methodology. The data in the packet are summarised in Table 3-15. As can be seen, the data from all the relevant units are included. Due to the earliness of the design, the only data from the EPS itself is the validity flag; the same applies to the SADA. From the solar panels, the power produced and the temperature of each

Table 3-15: data contained in the packet coming from the EPS and from the nominal scenario

Packet	Data	Value	Bits	Description
EPS	EPS validity flag	0/1	1	The value 1 indicates that the validity flag is positive, the value 0 that it is negative.
	SP1 power	0-35 [W]	6	The power produced by solar panel 1.
	SP2 power	0-35 [W]	6	The power produced by solar panel 2.
	SP1 temperature	+/-100 [°C]	8	The contrast value of the image taken by the camera.
	SP2 temperature	+/-100 [°C]	8	Attitude of the camera: 1 if the Moon is in the FOV, 0 if not.
	SADA validity flag	0/1	1	The value 1 indicates that the validity flag is positive, the value 0 that it is negative.
	Battery mode	0/1	1	0 if constant DOD, 1 if batteries are discharging.
	B1 DOD	0-100 [%]	7	The DOD of battery 1.
	B1 V	0-4200 [cV]	9	The voltage of battery 1. It is written in [cV] in order to have a more accurate measurement.
	B2 DOD	0-100 [%]	7	The DOD of battery 2.
	B2 V	0-4200 [cV]	9	The voltage of battery 2. It is written in [cV] in order to have a more accurate measurement.
	Sensor 1 state	0/1	1	0 if the sensor is off, 1 if it is on.
	T sensor 1	+/- 100 [°C]	8	The temperature of the battery system as detected by temperature sensor 1.
	Sensor 2 state	0/1	1	0 if the sensor is off, 1 if it is on.
	T sensor 2	+/- 100 [°C]	8	The temperature of the battery system as detected by temperature sensor 2.
	Sensor 3 state	0/1	1	0 if the sensor is off, 1 if it is on.
	T sensor 3	+/- 100 [°C]	8	The temperature of the battery system as detected by temperature sensor 3.
	Heater voltage	0-9 [V]	4	The voltage given to the heater installed on the battery system.
Other	SP1 nominal power	0-35 [W]	\	The nominal power that should be produced by solar panel 1.
	SP2 nominal power	0-35 [W]	\	The nominal power that should be produced by solar panel 1.
	DOD nominal	0-100 [%]	\	The nominal DOD of the batteries.

panel are received. The batteries send the voltage level, the depth of discharge (DOD) and the thermal data. The voltage is the difference in potential between the terminals of the battery, while the DOD is the percentage of battery capacity that has been discharged, expressed as a percentage of the maximum capacity [44].

In addition to the packet from the EPS, other data are needed by the FDI system. Firstly, the nominal power that should be produced by each panel is needed, to detect any failure. Similarly, the nominal DOD of the battery can be used for a comparison with the actual DOD.

The FDI logic of the EPS module is shown in Figure 3-14. As it can be seen, in case of a malformed data package, or negative EPS validity flag, no further check is performed, as a failure of the whole system was found. Otherwise, the two main sub-modules that were mentioned above can be distinguished. A check is performed between the data of solar panels and SADA, while two checks are made for the batteries. It is worth mentioning how these checks were made.

The first check that is performed allows detecting failures of the solar panels and the SADA, namely scenarios SP.1, SP.2, SADA.1 and SADA.2 from the FMECA. In fact, these failures have the same symptoms on the S/C: the power produced decreases. However, an ADCS failure might cause a wrong orientation of the satellite, leading to the same effect. Therefore, a proper detection logic must be defined, which is shown in Figure 3-15. A failure is triggered only if the power of one or both the panels is below the nominal. If both the panels have low power production, scenarios SP.1 and SP.2 are excluded since it is assumed that the two panels cannot fail at the same time. In case one panel fails but the failure cannot be identified through the SADA validity flag or the ADCS log, a plausibility check is used: if the panel temperature is high, it is considered plausible

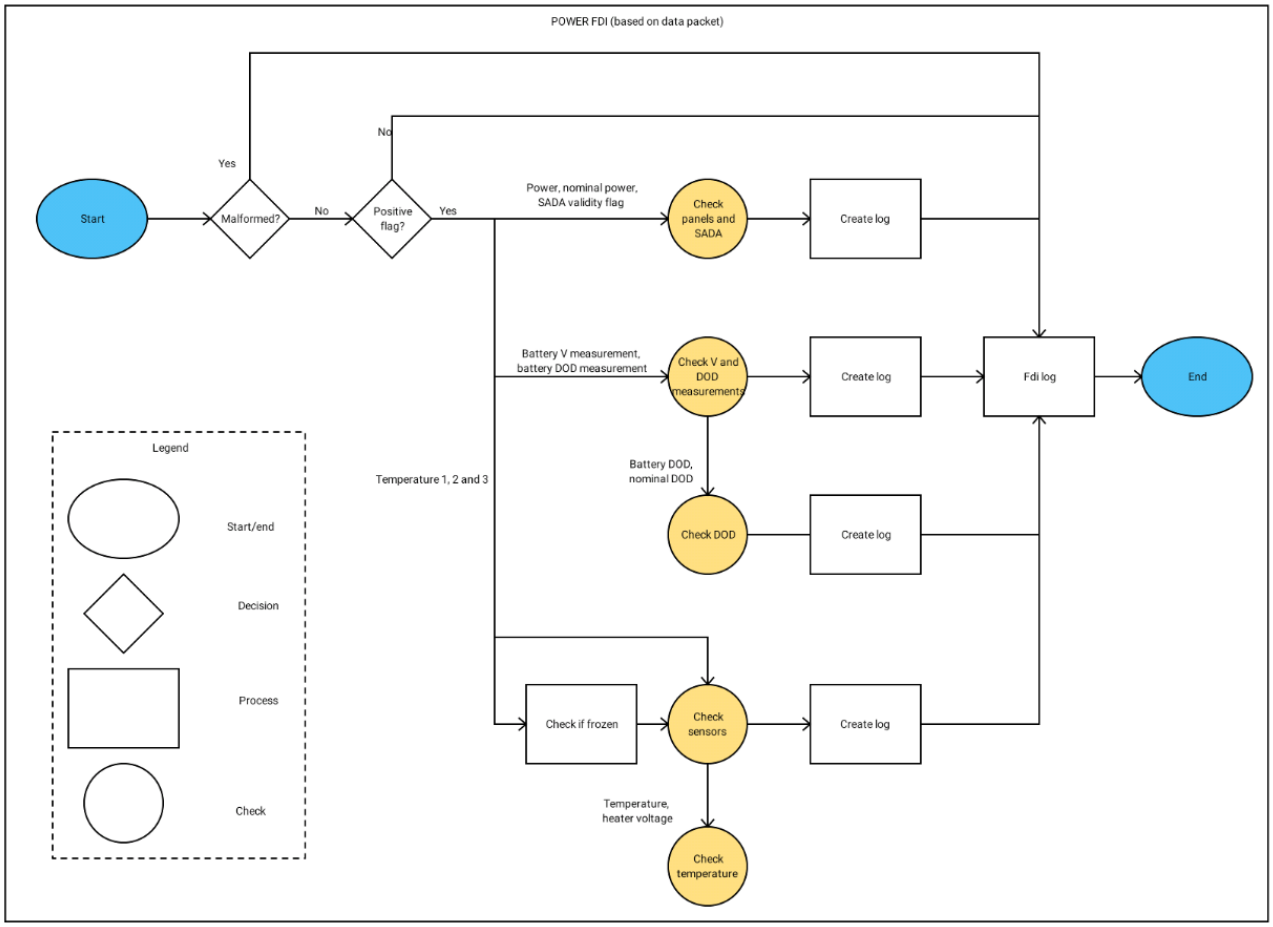

Figure 3-14: Power FDI architecture

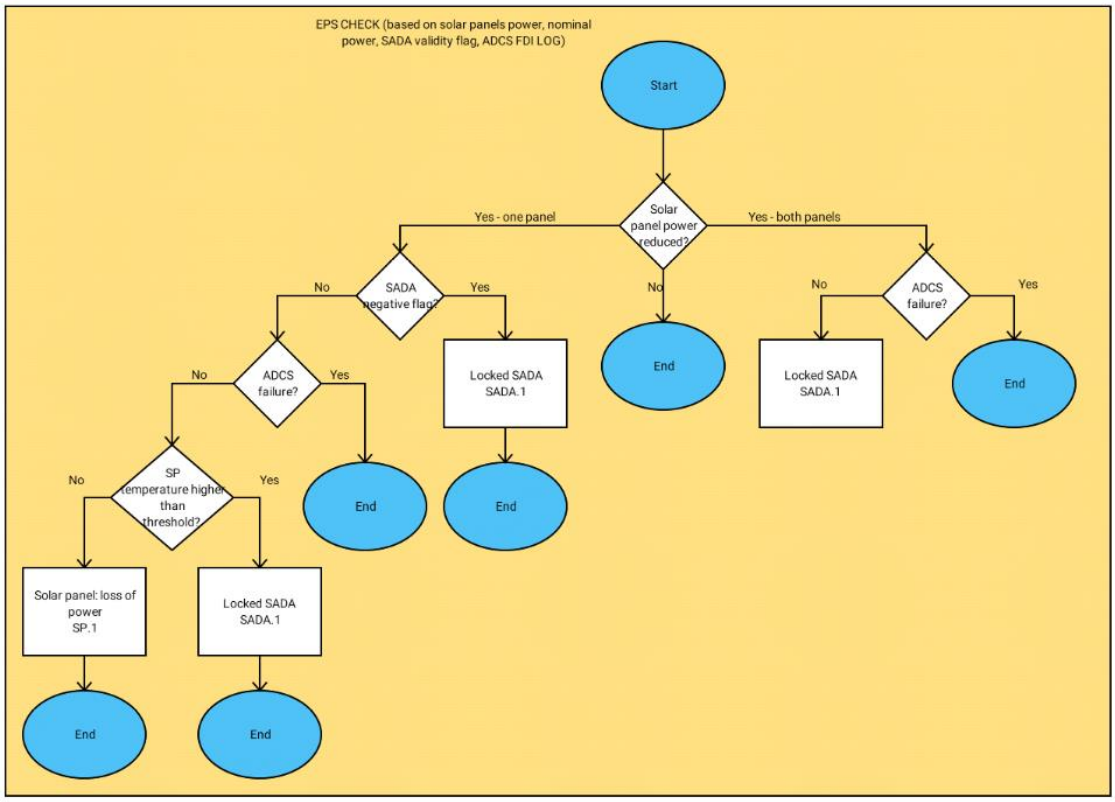

Figure 3-15: logic used in the check of the panels and SADA

that the panel is in the right orientation (facing the Sun); thus, a panel failure is flagged. Otherwise, it is deemed plausible that the panel has a wrong attitude; hence, a SADA failure is detected.

The second module in the EPS FDI is dedicated to battery failures. The relevant checks are two. In the first, the voltage level and DOD of each battery are compared, in order to detect any discrepancy (scenarios BT.4 and BT.5 of the FMECA, namely voltage measurement failure and DOD measurement failure). This check is based on the model of a Li-ion battery. However, modelling a Li-ion battery is a challenging activity, as the properties vary with aging and temperature; moreover, the seemly constant voltage level makes it hard to relate it directly with the DOD. Due to the preliminary nature of this study, the proposed model will be deemed sufficient. The model of the voltage-DOD relation is based on the battery datasheet provided by the battery supplier and documented in Figure 3-16, taken from [45]. From the figure, the approximate relation in Equation (29) is obtained.

$$\begin{cases} V = -\left(\frac{DOD}{100}\right) + 4.2 & if\ DOD \leq 80 \\ V = -3\left(\frac{DOD}{100}\right) + 5.8 & if\ DOD > 80 \end{cases} \tag{29}$$

It can be easily seen that the relation would produce a curve similar to the one in Figure 3-16. Nevertheless, the development of a more accurate model is recommended for the future stages of the project. Based on Equation (29), a check logic is developed: the expected voltage level given the actual DOD is calculated using the equations, and it is compared with the actual voltage measurement. An arbitrary threshold of 10% is selected for the DOD. In case only one battery is available, in case of discrepancy failure BT.4 is triggered. In case two batteries are available, a distinction is made between scenarios BT.4 and BT.5. Since the two batteries are the same and they work in parallel, it is assumed that in the nominal mission their DOD should be the same. Therefore, the DODs and voltage levels of the two batteries are compared, in order to distinguish between the two scenarios. In case only one battery is available (the other had a failure and was switched off), the extra-check is not performed, and the scenario BT.4 is flagged.

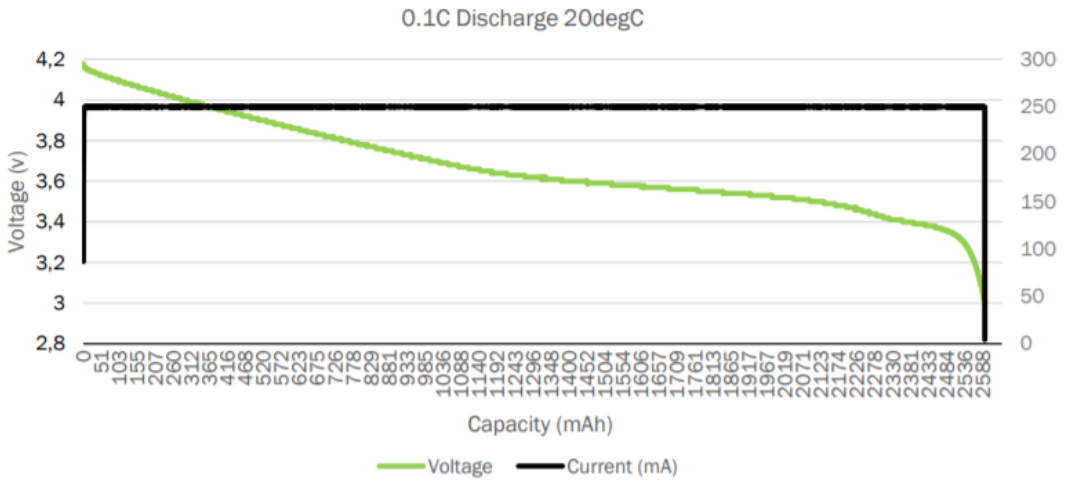

Figure 3-16: discharge curve of the GOMSPACE NanoPower battery, as indicated in the datasheet [45]

The second check is aimed at detecting failure scenario, by checking the DOD of the batteries: in this Thesis, it is assumed that over-discharge occurs for DOD above 80%, as documented in [44]. An additional check is made: the DOD of each battery is compared to the nominal DOD, which is the expected value of DOD. In case the DOD is below nominal, a flag is created. On top of this, a flag is created in case the DOD values of the two batteries differ significantly. These flags are not related to failures in the FMECA, but to failure scenarios that might be implemented in the future. In fact, DOD levels below the expectancies might be caused by a reduced battery capacity; however, the checks to perform in this case are manifold, and the current stage is too early. For instance, data about the power produced and consumed by the S/C must be taken into account; moreover,

more than one charge cycle must be analysed, e.g. more than one orbit, before detecting the failure. Hence, these failures will not be treated in detail in this Thesis, but they are included in this logic as a preliminary step for the future phased of the FDIR design.

3.7.2 Simulink model

The power FDI module takes in the packet coming from the EPS and the log from the ADCS system and creates the FDI log for the EPS. The log is an array of nine elements [a, b, c, d, e, f, g, h, i], all integers. The first two elements, **a** and **b**, refer to the solar panels, which are distinguished between solar panel 1 (SP1) and solar panel 2 (SP2). Elements c, d, e, f and g are related to the battery system: **c** and **d** refer to battery 1 (B1), **e** and **f** to battery 2 (B2), while element **g** is used to indicate if the two batteries differ excessively from each other. Finally, elements h and i refer to the thermal control system of the battery package: **h** indicates eventual failures in the temperature sensors, while **i** indicates failures in the temperature control. The meaning of the FDI log is resumed in Table 3-16.

The Simulink module reproduces the logic described in this section. The detection of malformed data package and validity flag is built as in the Main Thruster module, shown in Section 3.2.2; hence, they will not be described again. The data enter in different Stateflow charts, which implements the various checks, as can be seen from Figure C-19, in Appendix C.

The first module is shown in Figure C-20 and is dedicated to the solar panels and SADA check, following the logic of Figure 3-15. The following module is aimed at the FDI of the batteries and, as can be seen from Figure C-21, is divided between different cases. When two batteries are available, the logic shown in Figure C-22 is used; otherwise, the logic of Figure C-23 is followed. In both cases, it can be seen that two checks are implemented. First, the voltage measurement and the DOD measurement are compared. Second, the DOD is checked.

Table 3-16: Power FDI log

Element	Value	Meaning
All	0	No failure
	1	Malformed data package
	2	Negative validity flag
a, b	3	SP loss of power
	4	SADA failure
	5	Attitude failure due to ADCS
c, e	3	Battery voltage measurement failure
	4	Battery DOD measurement failure
d, f	3	Battery DOD below nominal
	4	Battery over-discharged
g	3	Battery 1 and 2 have different DOD
h	3	Sensor 1 - general failure
	4	Sensor 1 -frozen
	5	Sensor 2 - general failure
	6	Sensor 2 -frozen
	7	Sensor 3 - general failure
	8	Sensor 3 -frozen
	9	Unidentified failure
i	3	Overheating
	4	Heater – locked output
	5	Underheating
	6	Heater – no heating

3.8 Camera module

In this section, the logic of the FDI for the Camera will be explained. Firstly, the FDI architecture is presented, along with the checks that are used to perform failure detection. Later, the Simulink model will be described.

3.8.1 FDI logic

The Camera FDI module is aimed at detecting the failures of the camera that are listed in the FMECA. To define the detection logic, the packet summarised in Table 3-17 was conceived. As can be seen, not only the data coming from the Camera are needed for the detection, but also data related to the nominal scenario. In particular, three data are needed: the nominal attitude with respect to the Moon (in/out the FOV), in order to understand if the absence of the Moon in the field of view was expected or not, and the science and navigation frequencies.

Table 3-17: data contained in the packet coming from the camera and from the nominal scenario

Packet	Data	Value	Bits	Description
Camera	CAM validity flag	0/1	1	The value 1 indicates that the validity flag is positive, the value 0 that it is negative.
	Saturation	0-100 %	7	The saturation value of the image taken by the camera.
	SNR	0-15	4	The SNR of the camera.
	Contrast	0-100 %	7	The contrast value of the image taken by the camera.
	Moon in FOV flag	0/1	1	Attitude of the camera: 1 if the Moon is in the FOV, 0 if not.
	Frequency science	0-20 [Hz]	5	The image acquisition frequency during science.
	Frequency navigation	0-20 [mHz]	5	The image acquisition frequency during navigation.
Other	Moon in FOV nominal	0/1	\	Attitude of the camera according to the nominal scenario: 1 if Moon is in the FOV, 0 if not.
	Frequency science nominal	0/15 [Hz]	\	If the science mode is on, the nominal frequency should be 15 Hz, otherwise 0 Hz.
	Frequency navigation nominal	0/2/17 [mHz]	\	The nominal frequency of image acquisition during navigation. It is 0 when navigation is off, 2 during low-frequency acquisition and 17 during high-frequency acquisition.

The FDI logic of the module is shown in Figure 3-17. It can be seen that, as the detection methods to detect the failures are independent, different parallel checks are made. The first check is aimed at detecting scenario CAM.1 in the FMECA, namely “Camera image processing disturbances”. To do so, three parameters are checked: saturation, signal-to-noise ratio (SNR) and contrast. These parameters were chosen arbitrarily, as at the current stage there is no information regarding the data sent by the unit, and they are general indicators of the performance of a camera. The detection strategy is simple and based on limit checking: if the values of these parameters are above an arbitrary threshold, a failure is triggered. In the future phases of the project, when the design of the camera will be completed, the design of the FDIR shall be updated to incorporate a more accurate check of the image parameters, in accordance with the real system.

The second check is used to detect the “Moon out of FOV” failure scenario, labelled as CAM.2. The failure is triggered only in case the Moon is expected to be in the field of view in the nominal scenario, and no failure of the ADCS occurred. In fact, any failure of the ADCS might lead to a wrong attitude of the S/C, causing a wrong pointing of the payload.

The final checks are aimed at detecting the scenarios related to the image acquisition frequency. There are two possible cases for the nominal science frequency: it can be at 15 [Hz] during scientific observations, or 0 [Hz] the rest of the mission. Therefore, two scenarios were identified during the FMECA: “Low science frequency” (CAM.5) and “High science frequency” (CAM.6). The detection is based on simple limit-checking logic. In case the science frequency is lower than 15 [Hz] during science, failure CAM.5 is detected. In case the science frequency is above 0 [Hz] when the S/C is not in Science mode, failure CAM.6 is detected.

Regarding the image acquisition frequency during navigation, there are three nominal scenarios: high-frequency acquisition at 17 [mHz], low-frequency acquisition at 2 [mHz] and no acquisition, at

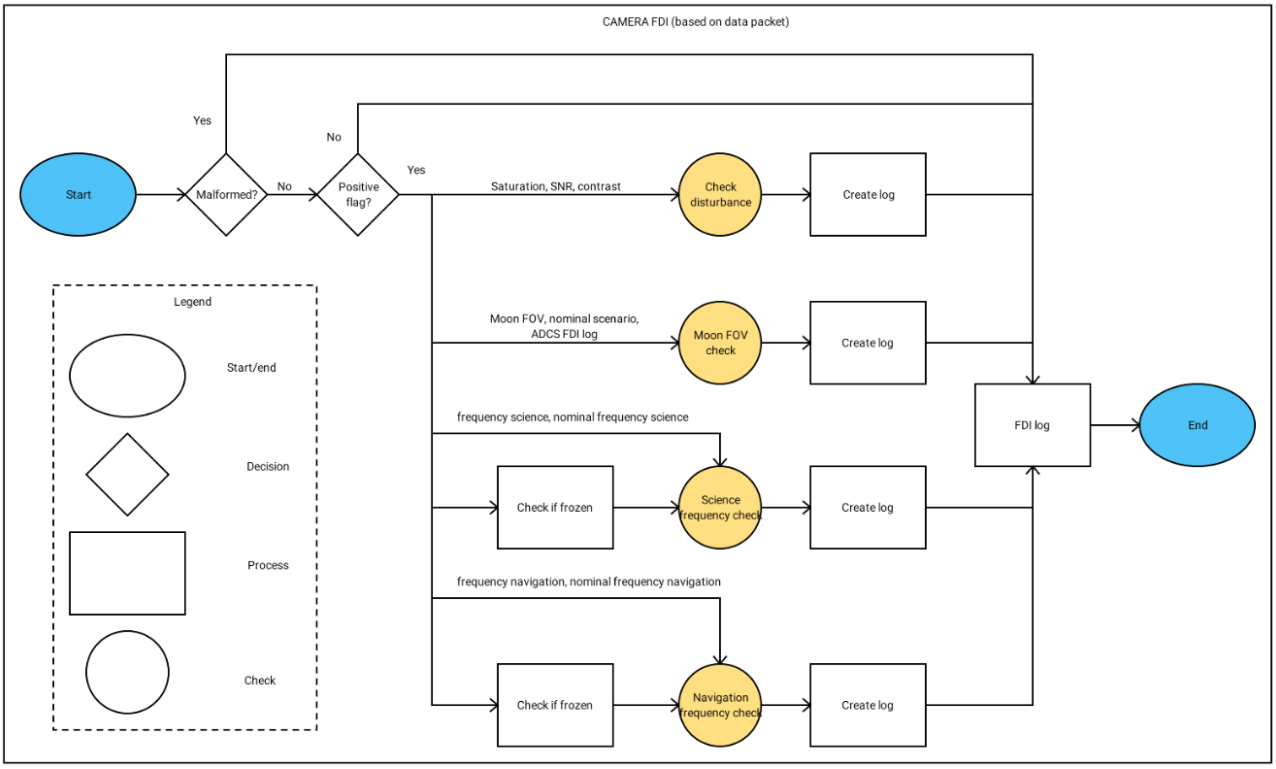

Figure 3-17: Camera FDI architecture

0 [Hz]. Thus, two different scenarios were identified during the FMECA: “Frozen navigation frequency” (CAM.3), when the frequency is frozen at the wrong value, or “General failure – navigation frequency” (CAM.4) when the actual frequency differs from the nominal one, but it is not frozen at another value. Therefore, the detection logic is implemented with the aid of a simple truth table, shown in Table 3-18.

Table 3-18: truth table used to detect failures of the navigation frequency of the Camera

Condition	**Value (true/false)**				
Actual frequency equals the nominal frequency	T	F	F	F	F
Actual frequency is 17 [mHz]	-	T	F	F	F
Actual frequency is 2 [mHz]	-	F	T	F	F
Actual frequency is 0 [mHz]	-	F	F	T	F
Scenario	No failure	Frozen frequency	Frozen frequency	Frozen frequency	General failure
FMECA ID	/	CAM.3	CAM.3	CAM.3	CAM.4

3.8.2 Simulink model

The camera FDI module takes in the packet coming from the camera and creates the corresponding FDI log. The log is an array of four elements [a, b, c, d], all integers. Each element is related to a different type of camera failure: the meaning of the possible values of **a**, **b**, **c** and **d** is explained in Table 3-19.

The Simulink module reproduces the logic described in this section. The detection of malformed data package and validity flag is built as in the Main Thruster module, shown in Section 3.2.2;

Table 3-19: Camera FDI log

Element	Value	Meaning
All	0	No failure
	1	Malformed data package
	2	Negative validity flag
a	3	Camera disturbances
b	3	Camera: Moon out of FOV failure
c	3	Low science frequency
	4	High science frequency
d	3	Navigation frequency locked at 1 image per minute (high-frequency)
	4	Navigation frequency locked at 1 image per 10 minutes (low-frequency)
	5	Navigation frequency locked at 0 Hz
	6	Navigation frequency general failure

.

hence, they will not be described again. The data enter in a single Stateflow chart, which implements the various checks that were described in this section, as can be seen from Figure C-24, in Appendix C.

3.9 Deployment module

In this section, the logic of the FDI for the Deployment module will be explained. Firstly, the FDI architecture is presented, along with the checks that are used to perform failure detection. Later, the Simulink model will be described.

3.9.1 FDI logic

The Deployment FDI module is aimed at detecting the failures of the deployment of the antennas and the solar panels. The deployment design was not discussed in [1]; therefore, in this Thesis, it was assumed that the technique used is the hold-release mechanism, with a burn-wire system, commonly used for small satellites [46]: the appendage is stowed during launch, and it is held on position by a wire. A deployment sensor is also included: a switch, pressed by the antennas when they are stowed and released when they open. When it is time to deploy the antenna, a countdown starts, and the mechanism opens, therefore the wire is cut by heating, created with the passage of a current through a resistance.

The deployment failure scenarios, which are included in the FMECA (DEP.2 to DEP.5) are peculiar, as they can occur only at the beginning of the mission, during Phase 0. On top of this, the detection is not direct, because the failure flag is sent by the OBC after a certain number of trials was attempted, as it was described in Section 2.2. Hence, the packet received by the deployment system is simple, and consists of two types of information, as shown in Table 3-20: the failure flag from the OBC, and the deployment switch state. If the switch is pressed, it means the units were not deployed.

Table 3-20: Deployment system data packet

Input	Value	Description
Antenna deployment error flag	0/1	The value 1 indicates that the OBC sent an error flag related to the antenna deployment, while the value 0 indicates that no error flag was produced.
Antenna switch	0/1	1 if the switch is pressed (antennas not deployed) and 0 if the switch is not pressed.
Solar panels deployment error flag	0/1	The value 1 indicates that the OBC sent an error flag related to the panels' deployment, while the value 0 indicates that no error flag was produced.
Solar panels switch	0/1	1 if the switch is pressed (panels not deployed) and 0 if the switch is not pressed.

From what said so far, the detection logic is straightforward and shown in Figure 3-18. It should be noticed that the malformed data package is checked, despite it does not correspond to any failure in the FMECA: this check has been introduced since it is unknown at the moment which packet will contain this information; however, this check is expected to be updated in the future phases of the project. Later, two checks are executed: one for the deployment of the antennas, one for the deployment of the panels. If the failure flag from the OBC is received, the deployment switch is checked, in order to verify that the flag from the OBC is correct; receiving a wrong failure flag due to an OBC failure is deemed unlikely, but the scenario is taken into consideration anyway (scenario OBC.6 in the FMECA). If the failure flag is correct, the FDI log from the Power module is checked: in case a power failure occurred, the deployment failure is considered as electrical failure, otherwise it is deemed as a mechanical failure. Hence, creating a log for the scenario "DOD below nominal" in the Power FDI (see Section 3.7) is particularly useful during the deployment, because the scenario is not treated as a proper failure, but allows to distinguish the nature of a deployment failure between electrical and mechanical. In fact, the most likely cause of electrical failure during the deployment is the insufficient charge of the batteries.

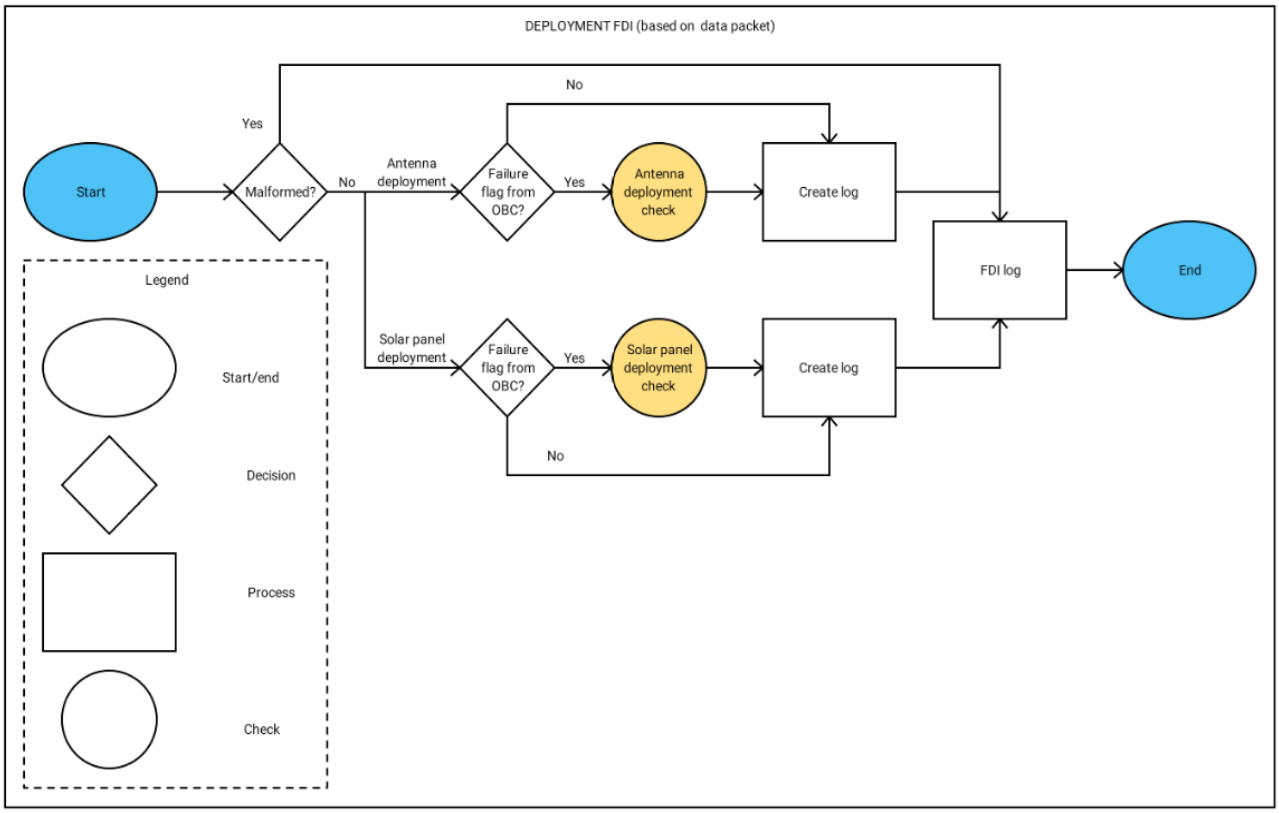

Figure 3-18: Deployment FDI architecture

3.9.2 Simulink model

The deployment FDI module takes in the packet coming from the deployment system and creates the corresponding FDI log. The log is an array of two elements [a, b], where a and b are integers in the interval [0, 4], each associated with a different scenario detected, as indicated in Table 3-21. The first element of the log, **a**, is referred to the antenna deployment system, whereas **b** refers to

Table 3-21: FDI log of deployment module

Value	Meaning
0	No failure
1	Malformed data package
2	Wrong failure triggering (the OBC flags the failure but the switch is not pressed, i.e. deployment happened)
3	Electrical failure
4	Mechanical failure

the solar panels deployment system.

The Simulink module reproduces the logic described in this section. The detection of malformed data package is built as in the Main Thruster module, shown in Section 3.2.2; hence, it will not be described again. The data enter into two Stateflow charts, which implement the checks for the Antenna deployment and the Solar Panels deployment. The Stateflow chart of the Antenna Deployment check is shown in Appendix C, in Figure C-25. As can be seen, three states are present in the chart, associated with the possible scenarios. The transition between the states depends on a graphical function, which implements the check logic described above. The Stateflow chart for the Solar Panels Deployment check is the same; hence, it is not reported.

3.10 Processors module

In this section, the logic of the FDI for the Processors module will be explained. Firstly, the FDI architecture is presented, along with the checks that are used to perform failure detection. Later, the Simulink model will be described.

3.10.1 FDI logic

The Processors module is aimed at detecting failures of the three processors, namely the OBC, the Payload processor, also called on-board-payload-processor (OBPDP) and the AOCS processor.

As it was explained in Section 2.1.1, in this Thesis, the failure scenarios of software were not considered, due to the earliness of the design. Hence, only a limited number of failures were examined for the processors in the FMECA, namely the malformed data package and the hardware failures, which at the current stage can be detected exclusively with the validity flag. Thus, the three packets from these sensors contain only the flag and the processor's state, as shown in Table 3-22. The processor state is used in case one of the processors is switched off, as a recovery action, as it is explained in Chapter 4.

Table 3-22: data contained in the packet coming from the three processors

Packet	Data	Value	Bits	Description
PD processor	State	0/1	1	0 if OFF, 1 if ON.
	Validity flag	0/1	1	0 for negative validity flag, 1 for positive validity flag.
OBC	State	0/1	1	0 if OFF, 1 if ON.
	Validity flag	0/1	1	0 for negative validity flag, 1 for positive validity flag.
AOCS processor	State	0/1	1	0 if OFF, 1 if ON.
	Validity flag	0/1	1	0 for negative validity flag, 1 for positive validity flag.

The inclusion of a processor's state in its data package is a contradiction (no data package is received from a processor that is switched off): in the final design, the state will probably be acknowledged by other means, e.g. by another processor. However, the simplistic design of the data package shown in Table 3-22 is considered sufficient for the current stage. In the next phases of the project, when the software design will be more detailed, the software failures shall be taken into account as well, and the data packets from the processors shall be detailed more.

As the inspection of the control bits and the validity flags are the only means to achieve failure detections, the FDI logic for the Processors module is straightforward, as illustrated in Figure 3-19. The logic is applied to each of the three processors. The check is not executed for a processor which is OFF.

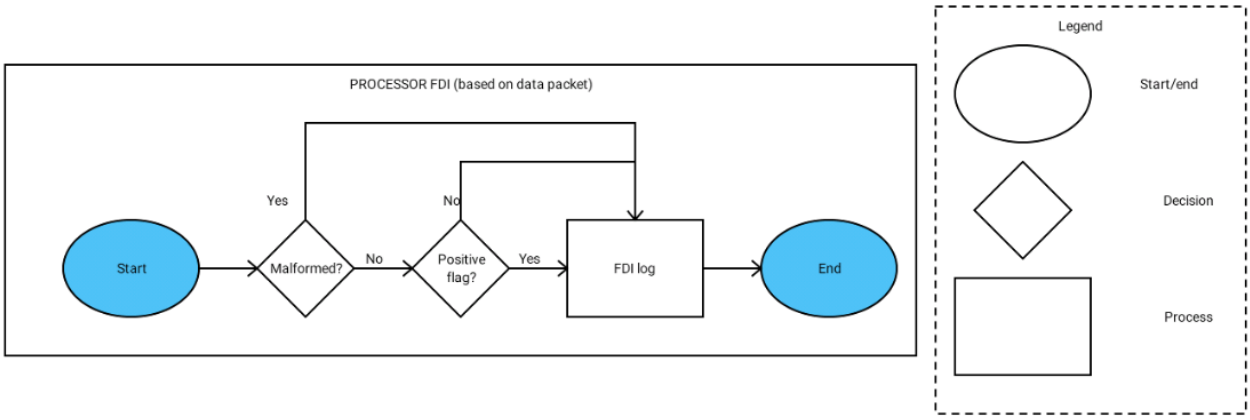

Figure 3-19: Processor FDI architecture

3.10.2 Simulink model

The processors FDI module takes in the packet coming from the processors and creates the corresponding FDI log. The log is an array of three elements [a, b, c], all integers. Each element is related to a processor: **a** is the log for the Payload (PD) processor, **b** the log or the OBC and **c** the log for the AOCS processor. The meaning of the possible values of a, b and c is explained in Table 3-23.

The Simulink module reproduces the logic described in this section. The detection of malformed data package and negative validity flag is built as in the Main Thruster module, shown in Section 3.2.2; hence, they will not be described again.

Table 3-23: processors FDI log

Element	Value	Meaning
All	0	No failure
	1	Malformed data package
	2	Negative validity flag

3.11 Communication module

In this section, the logic of the FDI for the Communication module will be explained. Firstly, the FDI architecture is presented, along with the checks that are used to perform failure detection. Later, the Simulink model will be described.

3.11.1 FDI logic

The FDI module of the Communication comprises all the failure scenarios that can be detected through communication with the Mothership, in particular, the failures of the antennas. In the current stage of the process, since the design of the antennas is at an early stage, there is no detailed information about the possible failures of the system, and about their symptoms. Thus, checking the communication with the Mothership is deemed as the only way to detect eventual failures of the antennas, in the current phase of the project. The logic developed so far is simple and allows the future implementation of more complex and accurate checks, based on the final design of the units. However, not only failures of the antennas are considered in this FDI module, but also some scenarios that were considered in the FMECA as OBC failures, i.e. “human failure” (OBC.5) and “time-keeping bug” (OBC.7).

In specific time-windows, LUMIO S/C will communicate with the Mothership, as it was described in Section 1.3.1. The data contained in the packet from the Mothership are unknown are the moment, due to the earliness of the project; however, in this Thesis a preliminary version of the packet was conceived, with the information that is necessary for the detection of the selected scenarios. The

Table 3-24: data contained in the packet coming from the Mothership and from the nominal scenario

Packet	Data	Value	Bits	Description
Mothership	Command	0-100	7	A hypothetical command from ground, which can go from 0 to 100.
	Mothership time	0-1000 [s]	10	The value of the Mothership time.
	Uplink receipt	0/1	1	1 if the S/C message was received by the Mothership, 0 if it was not.
Other	Communication window	0/1	\	0 if the S/C is not in the communication window, 1 if it is.
	Downlink receipt	0/1	\	0 if the Mothership message was not received, 1 if it was received.
	LUMIO time	0-100	\	The value of LUMIO on-board time.

first data is the command from the ground crew, which is simply modelled as a number from 0 to 50: if the value is higher, it is assumed that the command is faulty. Another useful information is the Mothership time. Finally, the uplink receipt is included, i.e. the confirm from the Mothership that the message from LUMIO was received. The data are collected in Table 3-24, along with other information that the FDIR system necessitates. Firstly, the nominal scenario is needed, to know if the communication window is active or not. Secondly, the S/C needs the downlink receipt from the antenna, i.e. the confirm that the Mothership packet was received. Finally, the knowledge of LUMIO on-board time is necessary for the cross-check with the Mothership time.

The FDI logic is shown in Figure 3-20. During the communication window, the FDI checks if a message from the Mothership was received. In case it was not, and no other failure was detected in the S/C, it is assumed that a failure of the downlink antenna occurred, scenario COMM.2 in the FMECA. If the message is received, multiple checks are executed: the command is compared with the threshold, to detect an eventual human failure, while LUMIO time is compared with the Mothership time. Finally, the message receipt from the Mothership is checked: if the Mothership did not receive the message from LUMIO and no other failure occurred in the S/C, a failure of the uplink antenna is assumed, scenario COMM.1.

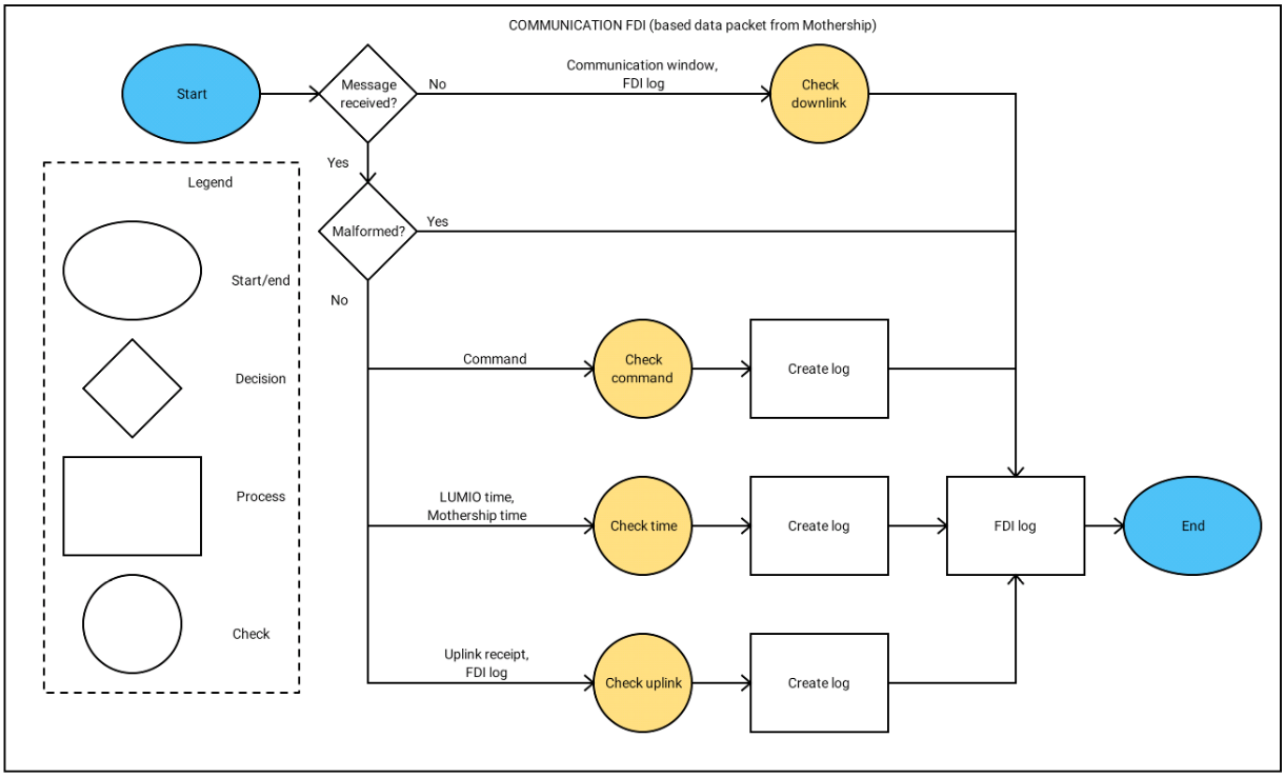

Figure 3-20: Communication FDI architecture

3.11.2 Simulink model

The communication FDI module takes in the packet coming from the Mothership and creates the corresponding FDI log. The log is an array of three elements [a, b, c], all integers. Each element is related to a different type of failure: the meaning of the possible values of **a**, **b** and **c** is explained in Table 3-25.

Table 3-25: communication FDI log

Element	Value	Meaning
All	0	No failure
	1	Malformed data package from Mothership
	2	Downlink failure
a	3	Human failure
b	3	Time-keeping bug
c	3	Uplink failure

The Simulink module reproduces the logic described in this section. First, the downlink receipt is examined, as shown in Figure C-26, documented in Appendix C. In case of negative receipt (no message received) the “Check downlink” is executed, as shown in Figure C-27. In case a message was received, the first step is the examination of the control bits, in order to detect a malformed data package. This check is built as in the Main Thruster module, shown in Section 3.2.2; hence, it will not be described again. If the package is not malformed, its data are divided into three checks, which can be seen in Figure C-28. The first two checks are related to the command from the ground and to the on-board time and they are based on a simple limit-checking; hence, they will not be shown, due to their simplicity. The third check is aimed at detecting uplink failures and is shown in Figure C-29.

4 FR Design

Failure Recovery (FR) is a fundamental task of the FDIR system, which is aimed at performing actions after a failure was detected, in order to resolve the issue while minimizing the impact on the mission. In this chapter, the design of the Failure Recovery system will be presented. Firstly, in Section 4.1 the general methodology will be described; later, the following section will explain how the methodology was applied for each module.

4.1 Methodology

The Failure Recovery system receives the log of the FDI system, in which the information about eventual failures detected is contained. Based on this data, the log of the FR, which indicates which recovery action to perform, is created. The FR is therefore divided into different modules, which replicate the modules of the FDI, as it can be seen from Figure 4-1: within each module, the FDI log is checked and, if a failure is detected, the Recovery starts. The advantage of this architecture is twofold. On the one hand, it is easier to develop and test each module, as it is independent of the others. On the other hand, it makes it possible to organize the recovery actions, by prioritizing those related to the most critical systems. This is particularly important in case of a chain of failures, in which a root failure has different symptoms across the whole S/C. In this case, it is important that the FR system does not trigger multiple recovery actions, but only those related to the root cause.

An additional remark must be made on the functional flow diagram of the FR, shown at the right of Figure 4-1: after a failure is detected and the system enters the “Recovery” state, the absence of failure in the next FDI log is not sufficient to exit the “Recovery” and enter in the “No action” state. In fact, an extra check is added: in order to end the recovery, the FDI should confirm the absence

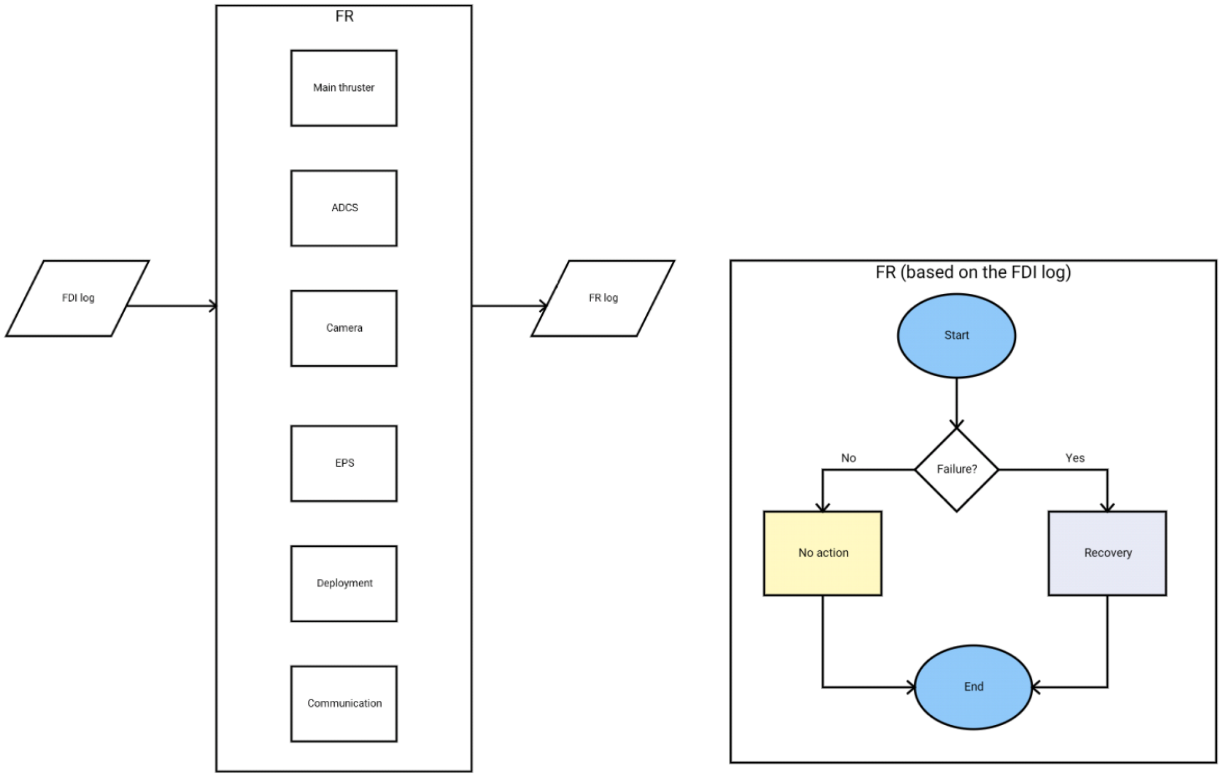

Figure 4-1: high-level architecture of the FR system (left) and functional flow diagram of each FR module (right)

of failure for at least 3 [s] consecutively, which correspond to 3 steps of the algorithm. This additional check is necessary to make the FDIR more robust and to avoid a continuous jumping in and out of L0.

As a result of the Literature Study, in which different recovery strategies were compared, it was decided to organize the recovery actions in a hierarchical structure, which is a common strategy for the FR of space missions [9; 10; 47; 48]. The hierarchy is made of different levels (usually from L0 to L4) which are characterized by a crescent impact on the mission and, ideally, are activated successively, in order of criticality. The lowest level is L0, the highest L4; a higher level is triggered only after lower levels have been activated several times without success, or when the severity of the failure justifies it. The levels were defined as follows:

- **L0** means not to perform any action and wait for a fixed number of seconds. It is used to deal with internal malfunctioning of a unit, that does not affect the satellite subsystem's performance, or that can be recovered by local correction, made internally in the unit involved.

- **L1** consists of a local reconfiguration, i.e. switching the faulty unit off/on for a given number of attempts. On top of this, since the communication between LUMIO and the Mothership is limited, in the present work L1 will also include sending the FDIR log, to ensure that the information is sent as soon as possible.

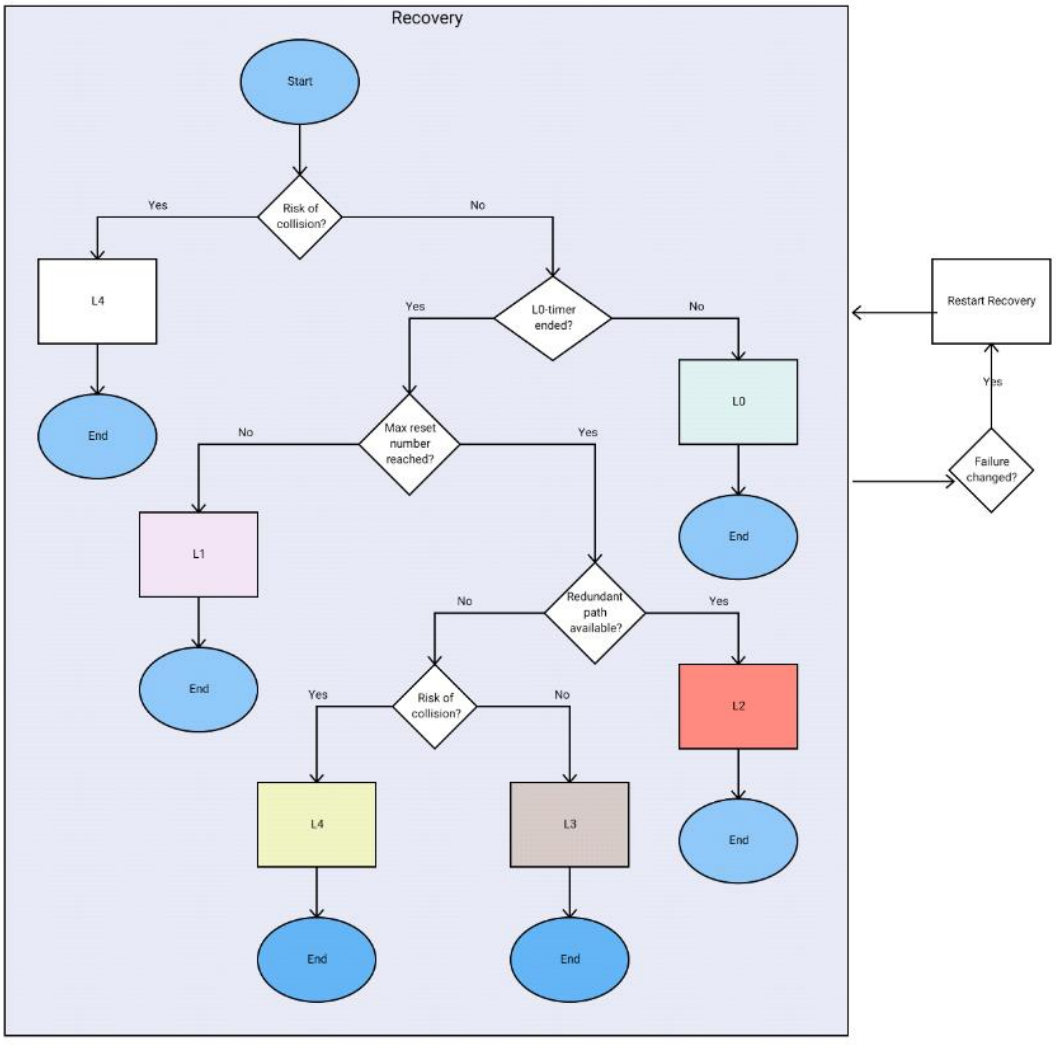

Figure 4-2: flow diagram of the Recovery state, which is activated when a failure is detected by the FDI

- **L2** is triggered by the FDIR when the previous levels did not resolve the failure, or when the fault does not allow the S/C to maintain the current mode. The recovery action is the substitution of the unit or function, by switching to a redundant one.
- **L3** means that the system is switched to Safe Configuration, after placing it in a Safe Orbit, where it will wait for instructions from the ground station.
- **L4** is applied in case a risk of collision is detected; therefore, the recovery action is to compute and execute a Collision Avoidance Manoeuvre, and then activate L3 and place the CubeSat in Safe Orbit.

The strategy described above is the baseline for the development of the "Recovery" state in each FR module, which is triggered when a failure occurs, see the right diagram in Figure 4-1. The logic used in the implementation is described in the flow diagram of Figure 4-2. It can be seen that an extra check is added and performed continuously when the FR is in the "Recovery" state: if the failure detected by the FDI changes, the "Recovery" is restarted from the beginning. The purpose of this check is twofold. Firstly, it ensures that, if a failure is solved and a new one occurs right after, the correct recovery sequence is followed for the new failure; otherwise, there would be the risk to activate extreme recovery levels, e.g. L3, ahead of time. Secondly, the extra check helps to deal with situations in which a wrong failure is detected at first, or two failures of different nature are detected at the same time. In fact, the recovery parameters, e.g. the waiting time during L1, are dependent on the failure to recover; therefore, it is fundamental to set their values in accordance to the actual failure that occurred. Nevertheless, to avoid an infinite loop, a loop counter is added: in case the "Recovery" is restarted for 4 times, the extra-check is disabled to allow the prosecution of the recovery sequence.

Each recovery level of Figure 4-2 is modelled according to the description made above. The flow diagram of level L0 is shown in Figure 4-3. It can be noticed that the only action executed within this recovery level is to advance the L0-timer. When the timer is over, the "Recovery" state will transit to level L1, according to the logic of Figure 4-2. However, an additional check is made within L0: whenever the FDI log shows that the failure has been recovered, a recovery-timer is started. As it was explained above, the timer lasts 3 [s] and it is included to decrease the probability of false negatives and thus make the FR algorithm more robust. If the absence of failure is confirmed for the whole duration of the recovery-timer, the FR exits the "Recovery" state and enters in the "No action" state, as shown in Figure 4-1. Otherwise, the recovery-timer is reset and L0 proceeds.

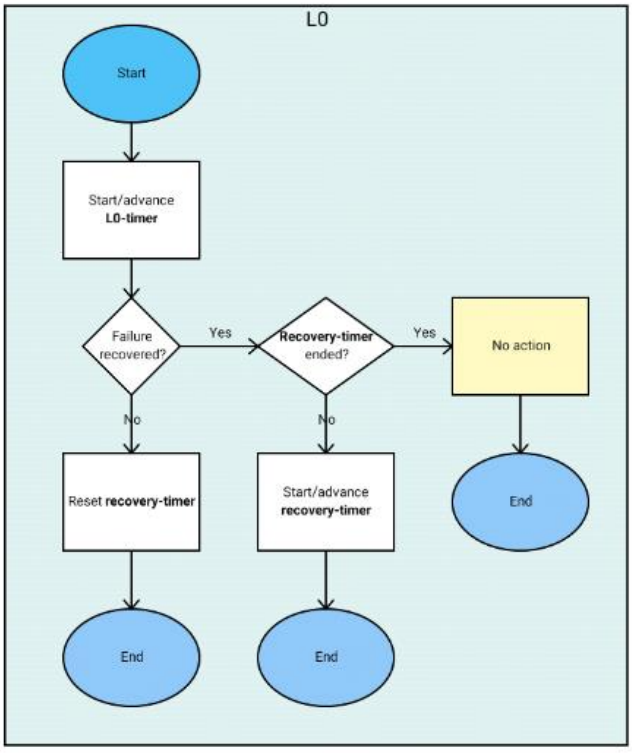

Figure 4-3: L0 flow diagram

The following recovery level that is entered within the "Recovery" state is L1. The functional flow diagram of L1 is shown in Figure 4-4. The main action executed at this level is to send a reset command to the faulty unit; whenever this is done, the reset number is increased by one, and, after the maximum is reached, the "Recovery" state will transit to the next level, according to the logic of Figure 4-2. However, several timers are included in the flow diagram, to incorporate additional checks. Firstly, a wait-timer of 3 [s] is included before sending the reset command. This timer is added to confirm the

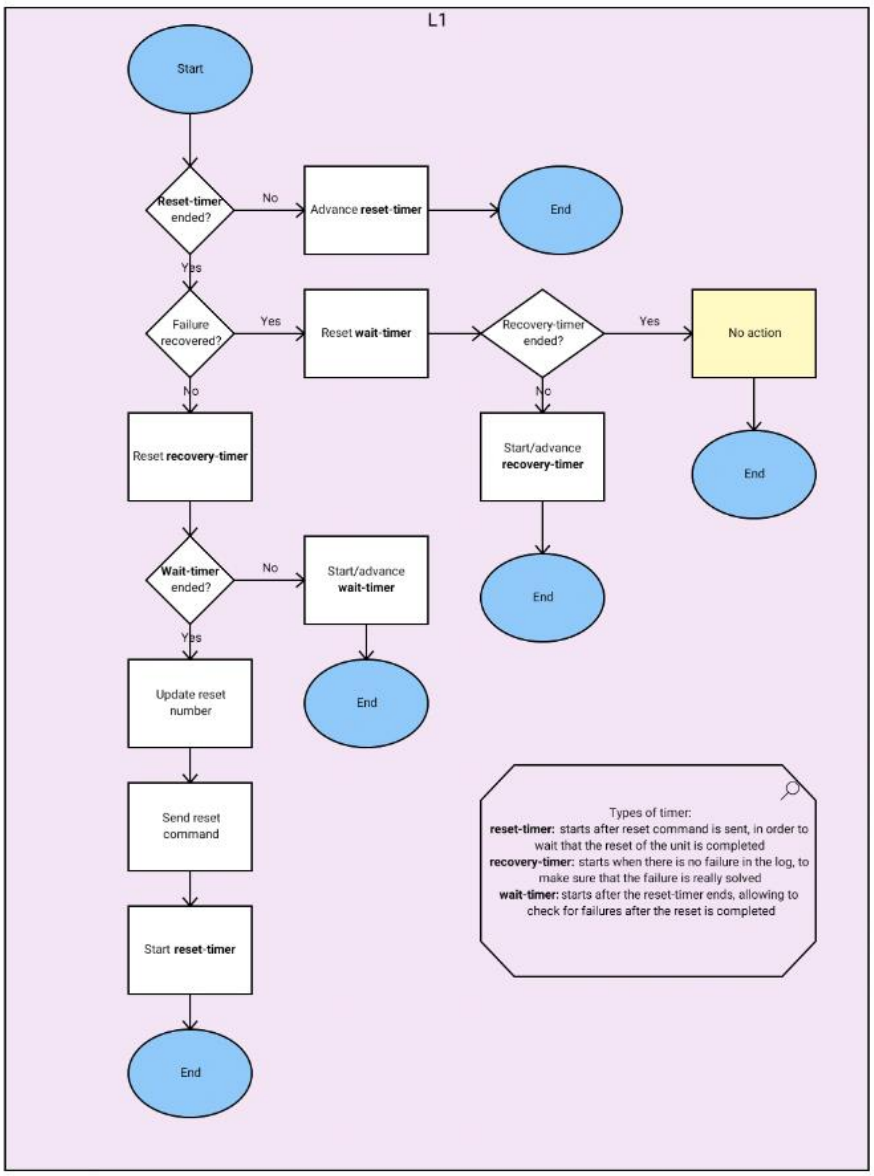

Figure 4-4: L1 flow diagram

presence of a failure in the unit, before executing any action. On top of this, in case the FDI log shows that the failure has been recovered, a recovery-timer is started, similar to the one described in L0. It can be noticed that this architecture poses a risk, i.e. to continuously transit from the wait-timer to the recovery-timer, in case a borderline failure occurs and the FDI is not able to create a congruent log for more than 3 [s] in a row. To avoid an infinite loop, an extra check is added and after a maximum number of 5 loops the system proceeds with sending the reset command; however, this loop-number check is not included in the diagram of Figure 4-4, in order to keep it simple and readable. Finally, a reset-timer is included. This timer is activated after the reset command is sent and it allows the FR to wait for the reset to occur, before proceeding with the recovery decision. During the reset-timer no transition can occur within the FR: the external check showed in Figure 4-2, in which the FDI log is checked to assess a change in the failure, is not executed either. After the timer ends, the FDI log is checked again, i.e. the wait-timer is started.

The next recovery level that is implemented at this stage is L2. Only a few systems allow to include this level, and only for a limited number of failures. The logic of L2 is shown in Figure 4-5: the recovery level is divided into two states. Firstly, the state “L2-activate redundancy” is entered: the faulty unit is switched off and the redundancy is activated. Secondly, the state “L2-wait” is entered,

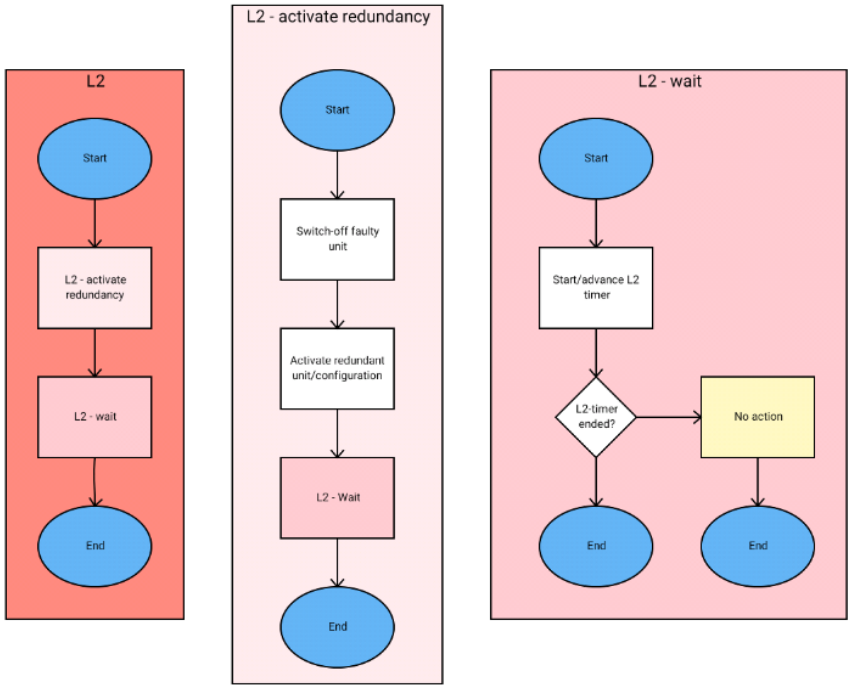

Figure 4-5: L2 flow diagram; this recovery level is divided into two sub-levels, to improve its readability

and it consists of a simple timer, at the end of which the FR exits the “Recovery” state and enters in the “No action” state. The timer is necessary because switching the system to the redundant configuration might take time and it is important to avoid triggering new recovery actions in the meanwhile.

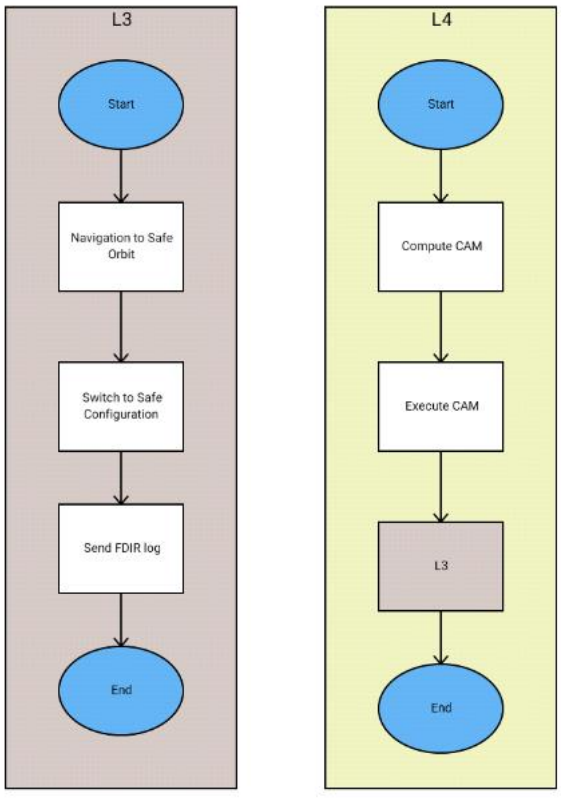

Figure 4-6: L3 and L4 flow diagram

In case L2 is not available, L3 or L4 are activated. The simplified flow diagrams for these recovery levels are shown in Figure 4-6 and follow the definitions of these levels that were given above.

From Figure 4-6, it can be seen that L3 and L4 require the computation of orbital manoeuvres to place the S/C in a Safe Orbit or to avoid collision; being the project in an early stage, the implementation of these procedures is not feasible, and it would be out of the scope of this work since the dynamics is not included at this stage, as explained in Section 2.1.1. Therefore, these recovery actions will not be included at this stage: when the level L3 is reached, the simulation will simply be stopped, and the development of the more advanced recovery levels will be left for the future phases of the project.

A final remark must be made, regarding the selection of the recovery parameters, e.g. the waiting time during L0 or the number of resets during L1. In the final design, each failure scenario will have its parameters, based on the criticality of the failure and on the mission phase in which it will occur. For instance, a Camera failure during science might be less critical than during

navigation, allowing for higher waiting time. In the FMECA, some suggestions were made about the recovery parameters for each failure, but they must be considered as preliminary, due to the lack of information at this phase. Nevertheless, in the implementation of the FR strategy that is described in this Chapter, a more simplistic selection of the recovery parameters was done, to simplify the code and diminish the simulation time. However, the possibility of updating the recovery parameters in the future phases of the project has been included and eased in the code, by creating dedicated functions that can be easily modified independently from the rest of the recovery logic.

Using the methodology described so far, the FR model was implemented in Simulink. In particular, the Stateflow environment was used since it is particularly suited for fault management, as it allows to incorporate the decision logic described so far in transition diagrams, which are easy to develop and visualize. Thus, developing the algorithm using Stateflow has the advantage of making the work easy to share and understand and will ease the prosecution of the project in the next phases. In the following sections, the FR model of each of the modules of Figure 4-1 will be described. For each module, it will be necessary to describe firstly its high-level architecture, i.e. the division of the FR in sub-modules, and the recovery logic that was applied. Finally, the Stateflow model of the module will be shown more in detail.

4.2 Main Thruster module

This section is dedicated to the description of the FR module aimed at the recovery of eventual failures of the main thruster.

4.2.1 Recovery strategy

In the FDI model that was developed for the main thruster, two types of failures were considered: failures of the thrusters, and failures related to its thermal control system. As it was mentioned in Section 3.2, in this study it is assumed that the thruster is equipped with three temperature sensors, but no heater. Therefore, following the FDI structure, the FR model of the Main thruster is divided into two main sub-modules: the first is related to the thruster itself, the second one to the temperature sensors. This division is fundamental, as the temperature sensors are assumed to be independent of the thruster; hence, it must be avoided to trigger a recovery of the thruster in case of a sensor failure and vice versa.

Despite the division between the thruster and the thermal control, it should be noticed that some of the failures that were detected by the thermal control part of the FDI, e.g. overheating, must be treated by the FR as thruster failures since they can be solved only by recovering this unit. Hence, a decision logic must be developed, for those situations in which there is a contemporary detection of two types of failures, e.g. valve failure and overheating. This situation might occur in case two failures happen at the same time or, more importantly, in the case of chain failures. At this stage, priority is given to the proper thruster failures, e.g. valve failures, as they can have a major impact on the mission and require a faster response.

The recovery strategy of each of the two sub-modules follows the general model described in Section 4.1, based on a sequence of recovery levels. In case of thruster failure, level L2 cannot be applied, as no redundancy is present; instead, in case of failure of the temperature sensors, L2 can be applied up to two times, as the units on-board are three. Moreover, it should be noticed that L3 for the temperature sensors might not be the same as the common L3, as it was defined in Section 4.1. In fact, the Main Thruster is a unit that will not be critical in terms of temperatures, according to the thermal simulations performed so far [1]. Therefore, in case L3 was reached, i.e. the control of the unit's temperature was not possible any more, the activation of Safe Configuration would be excessive. Therefore, one possible implementation of L3 for this system is to proceed with the mission, ignoring the thermal behaviour of the thruster. The FDIR log shall be sent to the ground station anyway and, in case the ground crew, from the analysis of the on-board data, assessed that the real temperatures of the thruster were critical, more drastic actions might be commanded

to LUMIO. However, these actions will not be implemented at this stage, as they are expected to be implemented in the next phases of the project.

Finally, the definition of the recovery parameters is worth discussing: it is reasonable that different failures shall be handled differently. For example, a “Locked output” failure requires a fast response than an “Underheating” failure. The recovery parameters used in this phase of the project are mainly arbitrary and will need to be updated based on the behaviour of the real system. For the main thruster, the following strategy has been applied: the failures due to temperature have larger L0 time (40 [s]), due to their lower criticality and to the large time required to change temperature, while the other failures have smaller L0 time (10 [s]). An exception is made for the “Sensor failure” since the FDI requires about 20 [s] to properly distinguish between this failure and another one, e.g. locked output. Hence, the L0 time for this failure is set to 20 [s]. The maximum number of resets in L1 is set arbitrarily to 4 for all the failures, and the reset-timer is set to 20 [s]. For the temperature sensors, instead, due to the limited types of possible failures, the recovery parameters are assumed to be fixed and were selected arbitrarily: the L0 time is set to 20 [s], the maximum number of resets in L1 is set to 4 for all the failures, and the reset-timer is set to 20 [s].

4.2.2 Stateflow model

The module is implemented in a Stateflow chart, which takes the FDI log as input and returns the FR log as an output. The FR log is an array of four elements [A, B, C, D]. The first element, A, specifies the current failure of the Main thruster that is being recovered, while B specifies the recovery action that is being executed. Similarly, C indicates the failure of the temperature sensor that is being recovered and D the recovery action for the sensors. The meaning of the FR log is shown in Table 4-1. Some remarks are necessary. Firstly, it can be noticed that the “Overheating” and “Underheating” failures are treated as failures of the Main Thruster. Secondly, it can be noticed that, despite a distinction has been made for the temperature sensors between “general failure” and “frozen sensor”, this is ignored in the FR, since these units are the least relevant and it is considered more convenient to keep the FR simple. Finally, it is noticeable that the recovery action log indicates not only the current level applied but also the distinction between different actions inside the level.

Table 4-1: FR log of the Main Thruster module

Sub-Module	Element	Value	Meaning
Main thruster	A	0	No failure
		1	Malformed data package
		2	Negative validity flag
		3	Locked output
		4	No thrust
		5	Sensor failure
		6	Command out of range
		7	Overheating
		8	Underheating
	B	0	No action
		1	L0 – wait
		2	L1 - reset
		3	L1 – wait
		4	L3
Temperature sensors – Main thruster	C	0	No failure
		1	Temperature sensor 1 failure
		2	Temperature sensor 2 failure
		3	Temperature sensor 3 failure
		4	Unidentified failure
	D	0	No action
		1	L0 – wait
		2	L1 - reset
		3	L1 – wait
		4	L2 – switch to redundancy
		5	L2 - wait
		6	L3

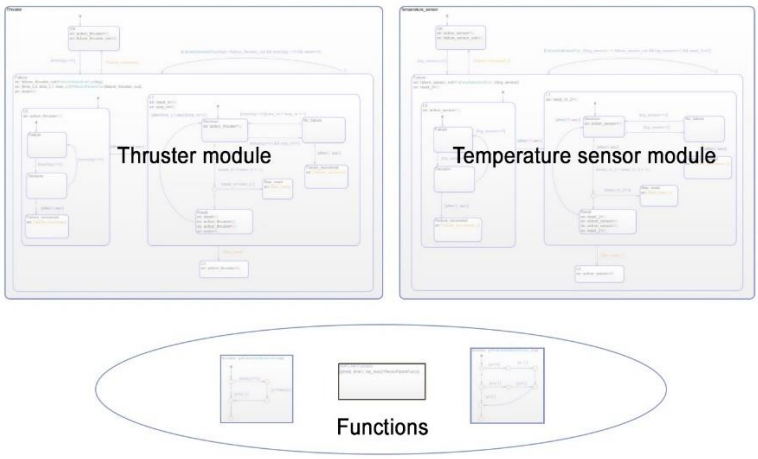

Figure 4-7: high-level architecture of the FR model of the Main Thruster

The “Main Thruster FR” module implemented in Stateflow can be seen in Figure 4-7: it can be seen that the model is divided into two sub-modules and that some functions, which are used within the models, are also included.

The first module, which is aimed at recovering eventual failures of the Main Thruster itself, is shown in detail in Figure 4-8. Recovery level L2 is omitted since no redundancy is available. Moreover, level L4 is not included either, as it will be implemented in the next phases of the project. The transition from the “No action” state to the “Recovery” state is based on the FDI log. The reverse transition is based on the event “Failure recovered”, which is created when no failure is detected for more than 3 [s] in a row.

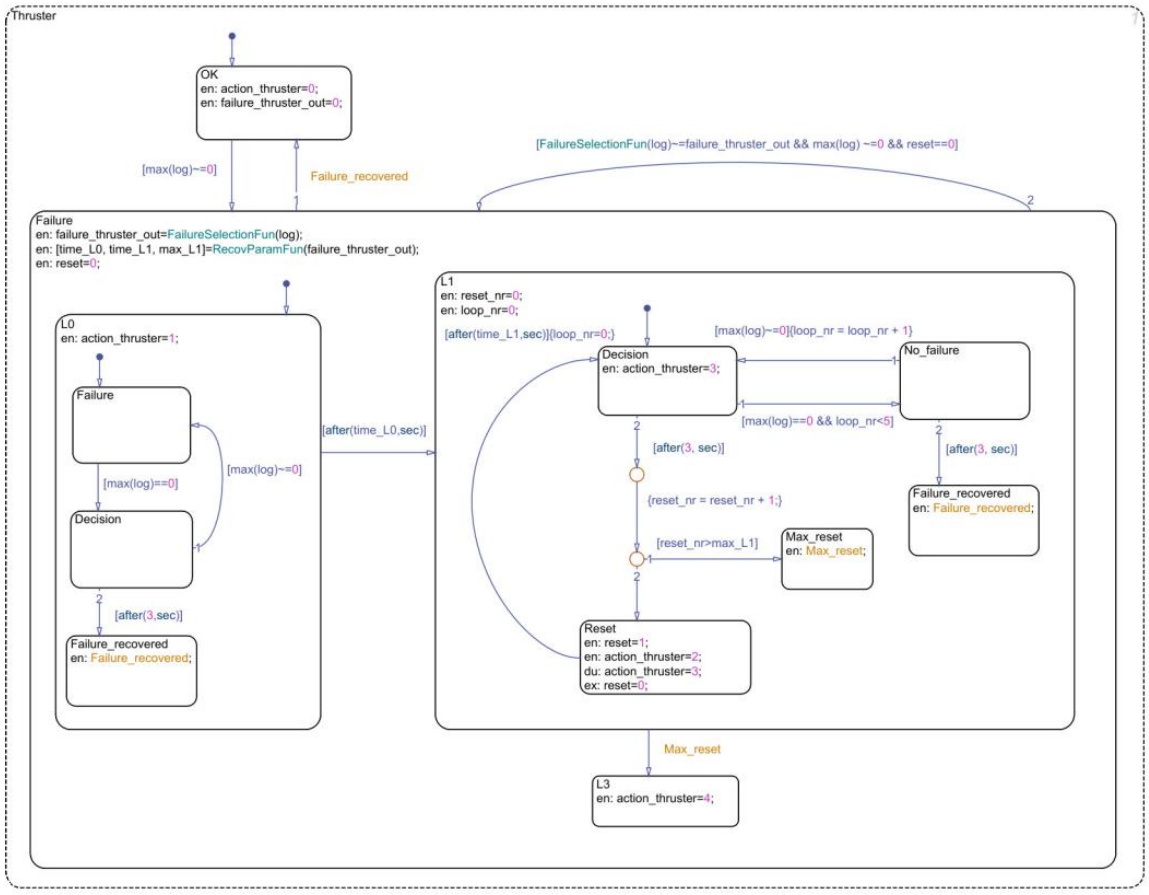

Figure 4-8: Stateflow model of the Main Thruster FR – detail 1

When the “Recovery” state is entered, the failure to be recovered, i.e. the element A of the FR log as shown in Table 4-1 is selected, using the first of the functions mentioned above. The function is fundamental in case two failures happened at the same time; as it was mentioned above, the proper thruster failures are prioritized against temperature failures. Since the selection of the failure strongly depends on the FDI log, it is foreseen that this function will need to be changed and updated in the future; therefore, it is convenient to keep it separated from the rest of the module. Once the failure to be recovered is selected, the recovery parameters e.g. the L0 waiting time, are defined accordingly, using the second of the functions that can be seen in Figure 4-7. The logic that is used to select the parameters was described in Section 4.2.1.

A self-transition is included in the “Recovery” state, to implement the extra check shown in Figure 4-2: if the FDI log changes and the failure detected is different from the previous step, the recovery sequence is restarted. It is noticeable that this happens only if the new failure is entirely different or has a lower priority to the former failure. For instance, if during an “Overheating” failure a “No thrust” failure occurs, the recovery is restarted, as the new failure has higher priority, but if the reverse situation occurs, nothing happens. Moreover, this self-transition is disabled after a reset command is sent, to give the unit time to restart.

Once the general structure has been described, the recovery levels within the “Recovery” state can be analysed. The first recovery level is L0, which is created as it was modelled in Figure 4-3. After L0 timer is over, L1 is entered and it is modelled as illustrated in Figure 4-4. During L1, the main states are two: “Decision”, which implements the wait-timer and it is used to confirm the presence of a failure, and “Reset”. When the “Reset” state is entered, a reset command is sent. Later, the reset-timer starts, and no action is performed until it ends. In addition to the two states described so far, the “No failure” state is used to implement the recovery-timer. There are three ways to exit L1. If the maximum reset number is reached, the level L3 is activated. Alternatively, if the failure is

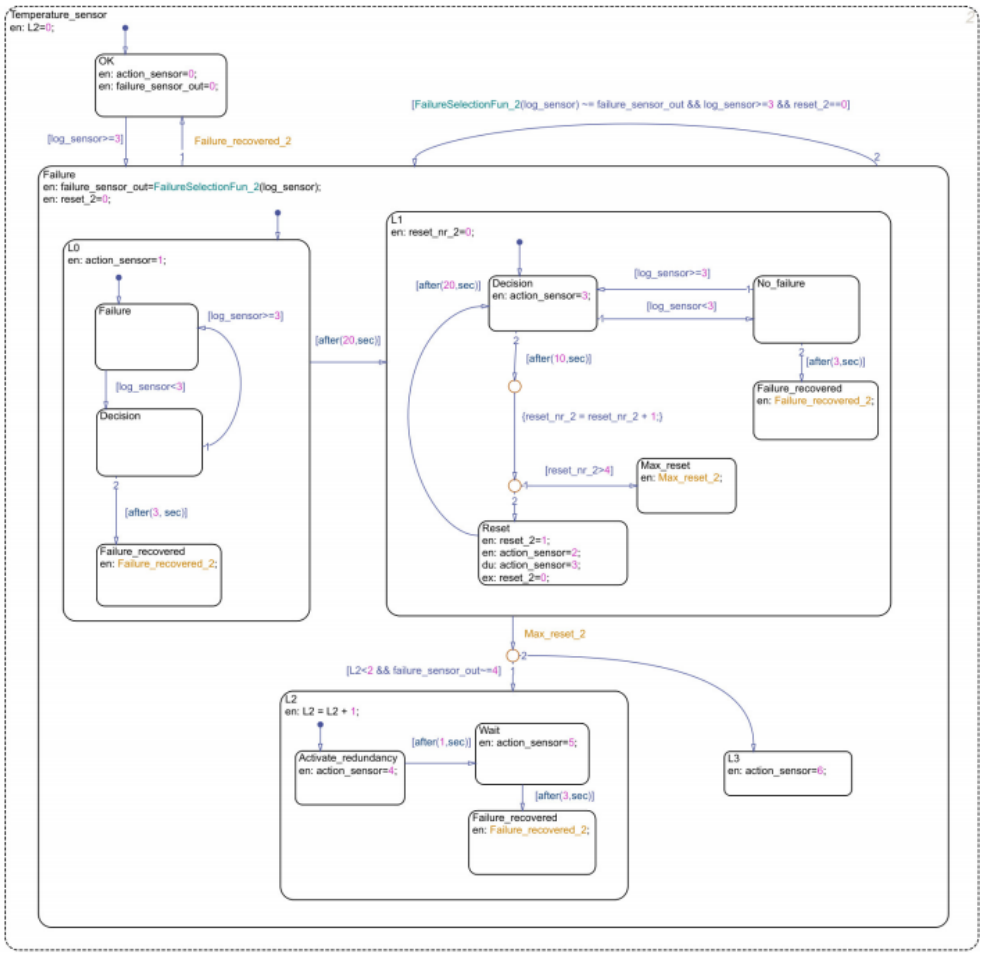

Figure 4-9: Stateflow model of the Main Thruster FR – detail 2

recovered for the whole duration of recovery-timer, the “No action” state is entered. Finally, if after the reset-timer the failure changes, the “Recovery” state is restarted from L0. As it was explained in Section 4.1, the final recovery level, L3, is not detailed: when it is triggered, the simulation stops.

The second FR module is dedicated to the recovery of failures of the temperature sensors installed on the Main Thruster and it is shown in Figure 4-9. The general architecture of this FR module is the same as the Thruster module, described above. The two possible states are “No action” and “Recovery”, as shown in Figure 4-1, and the transition between them is dictated by the detection of a failure (to enter the “Recovery” state) or the broadcast of an event called “Failure Recovered” (to enter the “No action” state), which is created after created when no failure is detected for more than 3 [s] in a row.

When the recovery is started, the third function of Figure 4-7 is used to select the failure of the temperature sensors based on their FDI log. In this way, element C of the FR log is created, see Table 4-1. As it was mentioned above, no distinction was made in the FR between “general failure” and “frozen sensor”, since it is usually difficult to correctly distinguish between the two. Hence, the only distinction that is done is between failures of temperature sensor 1, 2 or 3. A fourth option is included: in case a failure is detected but not isolated, the recovery actions are applied to all the temperature sensors currently available. This situation does not occur when all the sensors are working but can happen after one of them is switched off and a new failure occurs.

The recovery levels within this module are L0, L1, L2 and L3. The structure of L0 and L1 is the same as described above for the Thruster module. When the maximum number of resets in L1 is reached, there are two possible options. If L2 is available, the faulty unit is switched off and the “No action” state is entered again. Being the sensors 3, L2 can be executed two times; therefore, a counter is implemented to check the possibility of activating L2: in the future phases, the counter is expected to be substituted by a check of the availability of the redundant unit, made by the OBC. Moreover, another condition must be met: the failure location shall be isolated, i.e. the failure selected (element C of the FR log) shall not be equal to 4 (see Table 4-1): if the failure is not identified, it is not possible to select the unit to switch off. The other option is to enter L3. As it was mentioned above, L3 is not detailed in this study, and the simulation will be stopped.

4.3 ADCS module – Reaction Wheels

The ADCS module comprises all the units involved in the ADCS; hence, it is divided into 4 modules: Reaction Wheels, Gas Thrusters, Attitude Determination Sensors and IMU. This section is dedicated to the description of the FR module aimed at the recovery of eventual failures of the reaction wheels.

4.3.1 Recovery strategy

In the FDI model that was developed for the reaction wheels, two types of failures were considered: failures of the wheels, and failures related to its thermal control system. The set of wheels is equipped with three temperature sensors and a heater. Therefore, following the FDI structure, the FR model of the reaction wheels is divided into three main sub-modules: the first is related to the reaction wheels, the second one to the temperature sensors and the third to the heater. This division is fundamental, as the temperature sensors are assumed to be independent of the wheels and the heater; hence, it must be avoided to trigger a recovery of the reaction wheels in case of a sensor failure or a heater failure and vice versa. Since the module for the temperature sensors is the same as the one developed for the main thruster and discussed in Section 4.2, it will not be discussed in this section.

The RWs are treated as a single subsystem made of three units; hence, there are two distinct types of failures: failures of a single wheel, e.g. locked speed, and failures of the whole system, e.g. overheating. If a single unit failure occurs, the recovery action is executed on the faulty unit only, while, in the case of system failure, the recovery action is applied on the whole subsystem,

i.e. on all the three wheels. Therefore, in this project, it was decided to prioritize the failures of the whole system, as they affect all the units. It can be observed that, despite in this study the possibility of multiple failures occurring at the same time will be ignored, the distinction between single wheel failures and whole system failures allows to react to the cases in which more than a failure is detected: if it happened, the recovery action would be applied to all the wheels. This choice is particularly important not only for tackling multiple, parallel failures, which are considered unlikely, but also for tackling unforeseen failures that affect more than one unit at a time and were not comprised amongst the failure scenarios analysed in this study.

The recovery strategy of each of the sub-modules follows the general model described in Section 4.1, based on a sequence of recovery levels. In case of failure of a reaction wheel or heater, level L2 cannot be applied, as no redundancy is present. Moreover, additional considerations must be made regarding the recovery of a heater failure. During L1, the reset command is sent. However, the way to execute this reset will depend on the final design of the thermal control system: if the heater is controlled by a micro-controller, this latter unit will be reset, otherwise, in case the heater will switch on/off with a thermal switch, the only option is to cut out its power supply. It can be noticed that, in this latter case, the reset action is likely to succeed in resolving a “Locked heater” scenario, but not a “No heating” scenario. The definition of L3 will depend on the final thermal control design as well, but, likely, the recovery level will not follow the usual definition of L3, as it was given in Section 4.1. A “Locked heater” scenario can be solved simply by switching the unit off, without the need for other actions. On the other hand, the heater is expected to be used only during the first phase of LUMIO mission, for short periods of time; hence, in case of a “No heating” failure, activating the Safe configuration for the whole S/C would be excessive, as it would be enough to stop using the RWs during the short eclipse periods.

Finally, the definition of the recovery parameters is worth discussing. In case of reaction wheel’s failure, the parameters are chosen arbitrarily, according to the following logic: the failures due to temperature have larger L0 time (40 [s]), due to their lower criticality and to the large time required to change temperature, while the other failures have smaller L0 time (20 [s]). The reset-timer is set to 20 [s] and the maximum number of resets in L1 is set to 4. The recovery parameters for heater failures, instead, are: L0 time of 10 [s], reset-timer of 10 [s] and maximum number of L1 resets of 4.

4.3.2 Stateflow model

The module is implemented in a Stateflow chart, which takes the FDI log as input and returns the FR log as an output. The FR log is an array of seven elements [A, B, C, D, E, F, G]. The first three elements are related to the Reaction Wheels sub-module: A indicates the current failure of the RWs that is being recovered, B specifies the RW that is being recovered, C the recovery action that is being executed. Similarly, D refers to the failure of the temperature sensor that is being recovered and E contains the recovery action for the sensors. Finally, F indicates the heater failure that is being recovered and G the recovery action. The meaning of the FR log is shown in Table 4-2.

The high-level architecture of the Stateflow model of the Reaction Wheels FR is shown in Figure 4-10. The three sub-modules that were mentioned above can be seen, along with some functions that are used by the model. The three sub-modules are all contained in a super-state, which is used to perform continuously a crucial operation: re-arrange the FDI log in order to be more easily processed by the FR system. This operation is dependent on how the FDI log is created: since the FDI log is expected to be updated in the future phases, an external function is used, to keep it separated from the rest of the architecture and ease the implementation of future changes.

The first module, which is aimed at recovering eventual failures of the RWs, is shown in detail in Figure D-1 of Appendix D, while the third one, related to heater failures, is shown in Figure D-2. The structure is the same as described in Section 4.1, but it can be noticed that recovery level L2 is omitted since no redundancy is available. Moreover, level L4 is not included either, as it will be implemented in the next phases of the project. Since the architecture of the Stateflow models is similar to the Main Thruster one, described in Section 4.2, it will not be analysed in detail.

Table 4-2: FR log of the Reaction Wheels module

Sub-Module	Element	Value	Meaning
Reaction Wheels	A	0	No failure
		1	Malformed data package
		2	Negative validity flag
		3	Locked output
		4	No thrust
		5	Sensor failure
		6	Command out of range
		7	Overheating
		8	Underheating
		9	Multiple failures
	B	0	No RW
		1	RW1
		2	RW2
		3	RW3
		4	All the RWs
	C	0	No action
		1	L0 – wait
		2	L1 - reset
		3	L1 – wait
		4	L3
Temperature sensors – RWs	D	0	No failure
		1	Temperature sensor 1 failure
		2	Temperature sensor 2 failure
		3	Temperature sensor 3 failure
		4	Unidentified failure
	E	0	No action
		1	L0 – wait
		2	L1 - reset
		3	L1 – wait
		4	L2 – switch to redundancy
		5	L2 - wait
		6	L3
Heater	F	0	No failure
		5	Heater – locked output
		7	Heater – no heating
	G	0	No action
		1	L0 – wait
		2	L1 - reset
		3	L1 – wait
		4	L3

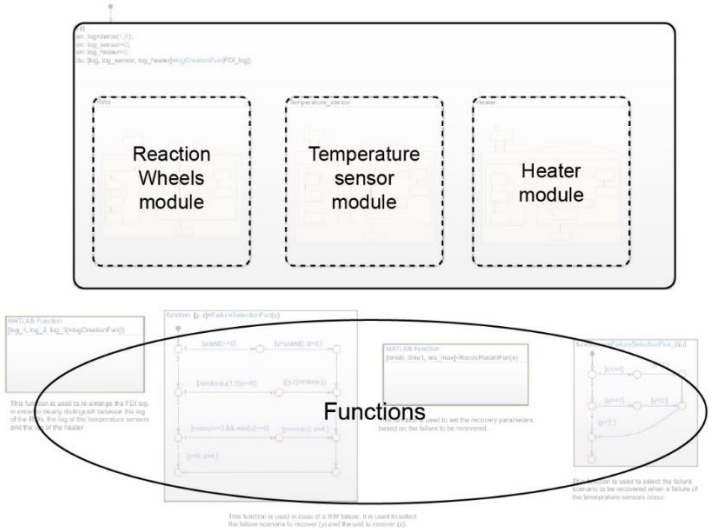

Figure 4-10: high-level architecture of the FR model of the Reaction Wheels

4.4 ADCS module - Gas Thrusters

This section is dedicated to the description of the FR module aimed at the recovery of eventual failures of the gas thrusters.

4.4.1 Recovery strategy

The FR model of the gas thrusters consists of a single module since the thermal control of the unit is included in the thermal control of the main thruster.

The four gas thrusters are different units, which share the propellant tank and the control system, i.e. a microcontroller or a processor; therefore, two types of failures might occur: failures that affect a single thruster and failures that affect the whole system. If a single unit failure occurs, the recovery action is executed on the faulty unit only, while, in the case of system failure, the recovery action is applied on the whole subsystem, i.e. on all the gas thrusters. Therefore, it is fundamental to specify in the FR log which unit is being recovered. Since more than one thruster at a time might be faulty, due to a common failure, e.g. malformed data package, to an unidentified failure or to multiple failures in parallel (which is unlikely and not considered), it is important to prioritize the failures with the highest criticality: for instance, a “No thrust” failure is deemed more critical than a “Malformed data package” failure.

The recovery strategy of this FR module follows the general model described in Section 4.1, based on a sequence of recovery levels; however, additional considerations should be made. Firstly, it is necessary to explain how some recovery actions can be executed. In case, during L1, a reset was needed, it would not be possible to reset a single thruster, but the whole system should be restarted: if the other thrusters kept working, the S/C would drift since the torque produce would not be controlled. Hence, during L1 the operations of the Gas Thruster shall be interrupted and the whole system shall be restarted, even in case of a single thruster failure. In case a failure was not solved with L1, the possibility of L2 is included in the recovery sequence, despite the system is not fully redundant. The application of L2 is subject to the three following conditions: 4 thrusters should still be active, i.e. L2 was not applied before, the current failure should affect only one thruster and, finally, the current failure is not a “Locked output” or “Unidentified failure”, since they do not allow to simply switch off the unit. Applying L2 is possible because the system of 4 gas thrusters in a pyramidal configuration is theoretically redundant, since three thrusters are sufficient to produce a 3-axes control torque, in case of little disturbance torque: the demonstration has been included in Appendix E. It is not known if the disturbance torque is expected to be low enough to allow a 3-thrusters operation during LUMIO mission, since no feasibility study was performed yet. However, the implementation of L2 is assumed to be possible at this stage. In fact, also other solutions can be implemented: in case of a de-tumbling manoeuvre, for example, it is possible to support the action of the three remaining thrusters with the Reaction Wheels. Therefore, it is important to notice that applying L2 does not only mean to switch off one of the thrusters, but it also implies changing the control algorithm that is used to create the torque using the gas thruster systems. Hence, such algorithm should be developed in the next phases and it should be enabled in case the FDIR sent the L2 signal to the system.

Finally, the definition of the recovery parameters is worth discussing. At this stage, the parameters are chosen arbitrarily: the L0-timer is set to 20 [s], due to the large time necessary to properly distinguish between a sensor failure and a valve failure. The reset-timer is set to 20 [s] and the maximum number of resets during L1 is set to 4.

4.4.2 Stateflow model

The module is implemented in a Stateflow chart, which takes the FDI log as input and returns the FR log as an output. The FR log is an array of six elements [A, B, C, D, E, F]. The first element, A, indicates the current failure of the Gas Thrusters that is being recovered. Elements B, C, D, E indicate whether GT1, GT2, GT3 and GT4 are being recovered. In fact, the four units belong to the same system, thus some failure can affect only a single unit, e.g. a valve failure, while others can influence the whole system, e.g. failure of the propellant sensor. Moreover, there might be situations in which it is not possible to isolate a failure to a single thruster. Hence, it is important to

Table 4-3: FR log of the Gas Thrusters module

Element	Value	Meaning
A	0	No failure
	1	Malformed data package
	2	Negative validity flag
	3	GT: command out of range
	4	Propellant sensor failure
	5	Valve sensor failure
	6	No thrust
	7	Locked output
	8	Unidentified failure
B	0	No failure GT1
	1	Failure GT1
C	0	No failure GT2
	1	Failure GT2
D	0	No failure GT3
	1	Failure GT3
E	0	No failure GT4
	1	Failure GT4
F	0	No action
	1	L0 – wait
	2	L1 - reset
	3	L1 – wait
	4	L2 – switch to redundancy
	5	L2 - wait
	6	L3

specify which thruster is recovered: this is also relevant for the application of L2, as it was explained above. Finally, F specifies the recovery action that is being executed. The meaning of the FR log is shown in Table 4-3.

The Stateflow model of the Gas Thrusters FR is shown in Figure D-3 in Appendix D, along with some external functions that are called inside the model. The architecture of the FR is the same as described in Section 4.4.1, and the recovery levels are L0, L1, L2 and L3. Since the architecture is the Stateflow model is similar to the Main Thruster one, described in Section 4.2, it will not be analysed in detail.

4.5 ADCS module - Attitude determination

This section is dedicated to the description of the FR module aimed at the recovery of eventual failures of the attitude determination sensors, i.e. the star-trackers and the Sun sensors.

4.5.1 Recovery strategy

The FR model for the attitude determination system is divided into several sub-modules, to avoid triggering recovery actions for the whole set of units in case of failure of one of them: one sub-module is dedicated to recovering failures of the star-trackers, one for failures of the Sun sensors. On top of this, in this study, it was assumed that each star-tracker is equipped with three temperature sensors and one heater; therefore, other 4 sub-modules are included. Since the modules for the temperature sensors and the modules for the heaters are the same as those developed for the main thruster and reaction wheels, they will not be discussed again in this section.

As the two star-trackers are totally independent units, the case in which they both fail at the same time will not be considered; therefore, if a failure of ST1 is present, that unit will be recovered, otherwise, ST2 will be recovered. Hence, if both fail, only ST1 will be recovered, but this is considered acceptable due to the low probability of this event. The only case in which both the STs are recovered is when the FDI log contains the indication of an unidentified failure for both the

units, which is deemed as an extreme situation. As it was done for the main thruster, temperature failures as “overheating” are treated with lower priority than other failures. In the case of the Sun sensors, the selection of the unit to recover is straightforward, as it is assumed that only one unit is working at a time.

The recovery strategy of each of the two sub-modules follows the general model described in Section 4.1, based on a sequence of recovery levels. L2 is included for both the star-trackers and the Sun sensors. The application of L2 for the Sun sensor is simple: the faulty unit is switched off, and the redundant one is switched on. On the other hand, the application of L2 for the star trackers must be explained. In fact, it is not clear at this stage if the system is fully-redundant or not: despite one star-tracker provides sufficient information for the 3-axis attitude determination, it can work only when the Sun is far from its Field of View. Therefore, it was decided to place two star-trackers on opposite sides of LUMIO CubeSat, so at least one of them could work at a given time. It may happen that during some phases both the units will be able to work, but, from the available information, it is not possible to know when and how often it will occur. Therefore, the application of L2 for this system does not simply imply to switch off the faulty unit, but it might also require a change in the nominal mission scenario, allowing to continuously point the other star-tracker away from the Sun. However, the role of the FDIR will be limited to triggering L2, and the necessary actions will be executed by the on-board processors, e.g. by the ADCS processor, and will be developed in the next phases of the project.

Finally, the definition of the recovery parameters is worth discussing. In this phase, the parameters were chosen arbitrarily; for the star-trackers, the failures related to temperature have longer L0 time of 40 [s], while the others have an L0 time of 20 [s]. For the Sun sensors, L0 time is 20 [s]. The reset-timer is set to 20 [s] and the maximum number of L1 resets is set to 4 for both the modules.

4.5.2 Stateflow model

The module is implemented in a Stateflow chart, which takes the FDI log as input and returns the FR log as an output. The FR log is an array of 13 elements [A, B, C, D, E, F, G, H, I, J, K, L, M]. The first three elements refer to the star-trackers module. Element A indicates which failure is been recovered, B indicates which unit is been recovered and C contains the recovery action. Elements D and E refer to the temperature sensors of ST1, while F and G refer to its heater: D and F indicate the failure that is being recovered, while E and G contain the recovery action. Similarly, elements H and I are referred to the temperature sensors of ST2, while J and K contain the log for the heater of ST2. Finally, element L indicates the failure of the temperature sensor, while M indicates the recovery action for the unit. The meaning of the FR log is shown in Table 4-4.

Table 4-4: FR log of the Attitude determination module

Sub-Module	Element	Value	Meaning
Star trackers	A	0	No failure
		1	Malformed data package
		2	Negative validity flag
		3	ST: Sun in FoV failure flag
		5	ST: general failure
		6	ST: frozen sensor
		7	ST: unidentified failure
		8	Overheating
		9	Underheating
	B	0	No ST
		1	ST1
		2	ST2
		3	ST1 and ST2
	C	0	No action
		1	L0 – wait
		2	L1 - reset
		3	L1 – wait
		4	L2 – switch to redundancy
		5	L2 – wait

Sub-Module	Element	Value	Meaning
		6	L3
Temperature sensors – ST1	D	0	No failure
		1	Temperature sensor 1 failure
		2	Temperature sensor 2 failure
		3	Temperature sensor 3 failure
		4	Unidentified failure
	E	0	No action
		1	L0 – wait
		2	L1 - reset
		3	L1 – wait
		4	L2 – switch to redundancy
		5	L2 - wait
		6	L3
Heater – ST1	F	0	No failure
		5	Heater – locked output
		7	Heater – no heating
	G	0	No action
		1	L0 – wait
		2	L1 - reset
		3	L1 – wait
		4	L3
Temperature sensors – ST2	H	0	No failure
		1	Temperature sensor 1 failure
		2	Temperature sensor 2 failure
		3	Temperature sensor 3 failure
		4	Unidentified failure
	I	0	No action
		1	L0 – wait
		2	L1 - reset
		3	L1 – wait
		4	L2 – switch to redundancy
		5	L2 - wait
		6	L3
Heater – ST2	J	0	No failure
		5	Heater – locked output
		7	Heater – no heating
	K	0	No action
		1	L0 – wait
		2	L1 - reset
		3	L1 – wait
		4	L3
Sun sensors	L	0	No failure
		1	Malformed data package
		2	Negative validity flag
		3	SS: Sun not in FoV failure flag
		5	SS: general failure
		6	SS: frozen sensor
		7	SS: unidentified failure
	M	0	No action
		1	L0 – wait
		2	L1 - reset
		3	L1 – wait
		4	L2 – switch to redundancy
		5	L2 – wait
		6	L3

In Figure 4-11 the high-level architecture of the FR module is shown and the division into six different sub-modules can be seen. The module of the STs is shown in Figure D-4 in Appendix D, while the FR of the Sun sensors is shown in Figure D-5.

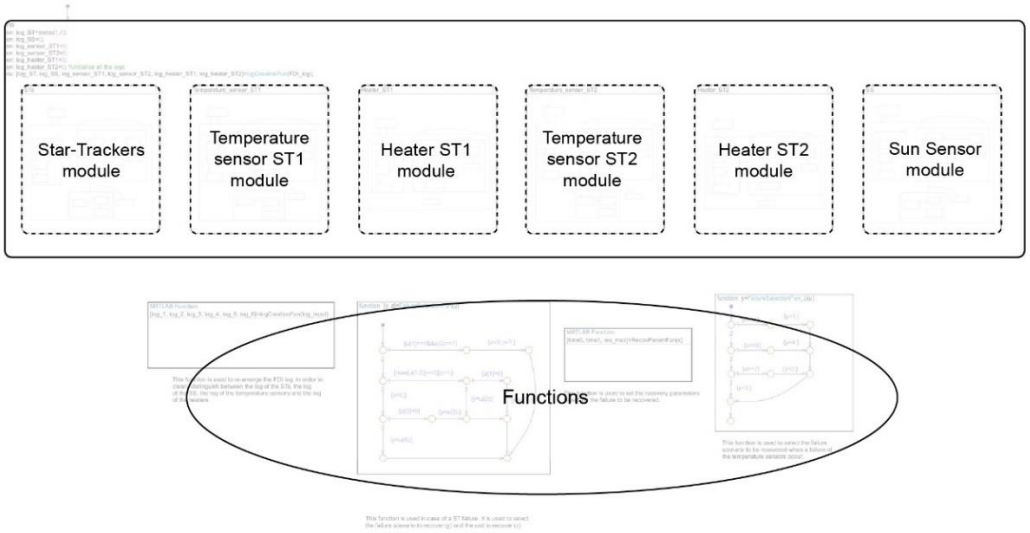

Figure 4-11: high-level architecture of the FR model of the Attitude determination module

4.6 ADCS module - IMU

This section is dedicated to the description of the FR module aimed at the recovery of eventual failures of the Inertial Measurements Unit.

4.6.1 Recovery strategy

The IMU comprises three accelerometers and three gyroscopes; however, it is not known, at the moment, if a dedicated recovery action will be possible for these sensors, i.e. if it will be possible to reset a single sensor or not. Therefore, at this stage, it was decided to develop a unique module for the whole IMU, while at the same time including the possibility to implement actions dedicated to single sensors, in the future phases.

The recovery strategy of this FR module follows the general model described in Section 4.1, based on a sequence of recovery levels. Level L2 is not possible, due to the lack of redundancy. The recovery parameters were chosen arbitrarily: L0 time is 20 [s], the reset-timer is 20 [s] and the maximum number of resets during L1 is set to 4.

Table 4-5: FR log of the IMU module

Element	Value	Meaning
A	0	No failure
	1	Malformed data package
	2	Negative validity flag
	3	S/C: excessive rotational speed
	4	Gyroscopes: general failure
	5	Gyroscopes: frozen sensor
B	0	No sensor selected
	1	Accelerometers
	2	Gyroscopes
	3	Accelerometers and gyroscopes
C	0	No action
	1	L0 – wait
	2	L1 - reset
	3	L1 – wait
	4	L3

4.6.2 Stateflow model

The module is implemented in a Stateflow chart, which takes the FDI log as input and returns the FR log as an output. The FR log is an array of 3 elements [A, B, C]. Element A indicates which failure is being recovered, while element B specifies if the failure occurred to the set of accelerometers or to the gyroscopes (or both). This distinction will open the possibility for the development of dedicated actions in the future. Finally, element C indicates the recovery action that is executed. The meaning of the FR log is shown in Table 4-5. It should be noted that the flag "Cross-check not available", contained in the FDI, is ignored, in order to not consider it as a failure.

The Stateflow model of the IMU FR is not reported, as it is implemented following a similar approach to the other modules, e.g. the main thruster module described in Section 4.2.

4.7 Power module

This section is dedicated to the description of the FR module aimed at the recovery of eventual failures of the Power module, which comprises all the units of the Power subsystem: the EPS, the solar panels, the SADA and the batteries.

4.7.1 Recovery strategy

The Power module is divided into several sub-modules: the first is related to the EPS and is aimed at recovering failures of the EPS, the solar panels and the SADA since it is not known, at the moment, how the recovery actions for these units can be realized, e.g. if it will be possible to actuate a reset of the solar panels or not. Therefore, at this stage, all the failures are grouped in this module, but the possibility to implement actions dedicated to single units in the future is granted by specifying in the FR log which system is recovered.

The second module is dedicated to the batteries, while the third and fourth modules are related to the temperature sensors and the heater of the battery pack. Since the modules for the temperature sensors and the modules for the heaters are the same as those developed for the main thruster and reaction wheels, they will not be discussed again in this section. The two batteries are treated as separated units, but with a shared thermal control system. Therefore, there are two distinct types of failures: failures of a single battery and failures of the whole system, e.g. overheating. If a single unit failure occurs, the recovery action is executed on the faulty unit only, while, in the case of system failure, the recovery action is applied on the whole subsystem, i.e. on all the batteries. Therefore, in this project, it was decided to prioritize the failures of the whole system, as they affect all the units.

The recovery strategy of the FR modules follows the general model described in Section 4.1, based on a sequence of recovery levels. However, some remarks are necessary. Regarding the EPS sub-module, L2 is not included since there is no redundancy for the EPS, the panels or the SADA. In case a battery failure is recovered, instead, L2 is included. In fact, during most of the mission, the battery system will be redundant, and it will be possible to operate will only one battery. However, an extra-check shall be included in the future, since during Phase 2, the Transfer Phase, the actuation of L2 will not be possible. Moreover, another check will be included at this stage: in case of failure of both the batteries, L2 is not possible either. This situation might occur because of a simultaneous failure of the two units, which is considered not likely, or because of a temperature failure, which would affect both the batteries at the same time. This failure scenario is possible, but the presence of a heater installed on the battery package makes it less likely to occur.

Finally, the definition of the recovery parameters is worth mentioning. In the sub-module dedicated to the EPS, the following logic was used: the failures of the EPS have shorter L0 time of 20 [s], due to their criticality, while failures of the panels or the SADA have an L0 time of 30 [s]. For the batteries, the L0 time in case of over-discharge is set to 20 [s] due to the criticality of this failure, while the other scenarios have L0 time of 30 [s]. For all the sub-modules, the reset-timer is set to 20 [s] and the maximum number of resets during L1 is set to 4.

4.7.2 Stateflow model

The module is implemented in a Stateflow chart, which takes the FDI log as input and returns the FR log as an output. The FR log is an array of ten elements [A, B, C, D, E, F, G, H, I, J]. The first three elements are related to the EPS. Element A indicates which failure is being recovered, B specifies the system that is recovered, e.g. SADA. Finally, the recovery action to execute is indicated in C. Similarly, elements D, E and F are related to failures of the batteries: D specifies the failure to recover, E specifies the battery to recover and F the action to execute. G and H are

Table 4-6: FR log of the Power module

Sub-Module	Element	Value	Meaning
EPS	A	0	No failure
		1	Malformed data package
		2	Negative validity flag
		3	SP loss of power
		4	SADA failure
	B	0	No failure
		1	SP1
		2	SP2
		3	All the EPS
	C	0	No action
		1	L0 – wait
		2	L1 - reset
		3	L1 – wait
		4	L3
Batteries	D	0	No failure
		1	Battery voltage measurement failure
		2	Battery DOD measurement failure
		3	Battery over-discharged
		4	Overheating
		5	Underheating
	E	0	No failure
		1	Battery 1
		2	Battery 2
		3	Both batteries
	F	0	No action
		1	L0 – wait
		2	L1 - reset
		3	L1 – wait
		4	L2 – switch to redundancy
		5	L2 - wait
		6	L3
Temperature sensors – battery	D	0	No failure
		1	Temperature sensor 1 failure
		2	Temperature sensor 2 failure
		3	Temperature sensor 3 failure
		4	Unidentified failure
	E	0	No action
		1	L0 – wait
		2	L1 - reset
		3	L1 – wait
		4	L2 – switch to redundancy
		5	L2 - wait
		6	L3
Heater - battery	F	0	No failure
		5	Heater – locked output
		7	Heater – no heating
	G	0	No action
		1	L0 – wait
		2	L1 - reset
		3	L1 – wait
		4	L3

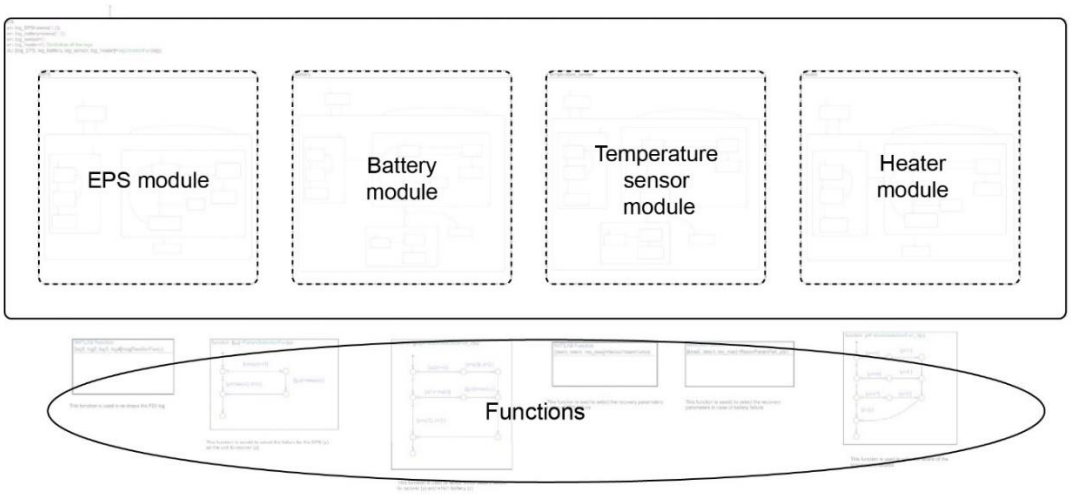

Figure 4-12: high-level view of the Power FR Stateflow model

related to the temperature sensors: G refers to the failure of that is being recovered and H contains the recovery action. Finally, F indicates the heater failure that is being recovered and G the recovery action. The meaning of the FR log is shown in Table 4-6.

The high-level view of the “Power FR” Stateflow model is shown in Figure 4-12. The four aforementioned modules can be clearly distinguished. The detailed Stateflow models of the EPS and Battery are reported in Appendix D, in Figure D-6 and Figure D-7.

4.8 Camera module

This section is dedicated to the description of the FR module aimed at the recovery of eventual failures of the Camera.

4.8.1 Recovery strategy

The Camera module is not divided into different sub-modules, as the unit is treated as a single entity, and no thermal control was considered at this stage. Nevertheless, it is worth discussing how the failure to recover is selected. This is a crucial task, as failures of different nature could occur to the camera and more than one element of the FDI log might be different from 0, which indicates the absence of failures; hence, there is the need to prioritize the most critical scenarios. The most critical failure is considered the presence of Camera disturbances, which might also be a root cause for other failures. For instance, in case of excessive disturbances, the Camera might not be able to detect the Moon in its field of view. Secondly, the failures related to the navigation frequency are considered, as their impact on the mission could be significant. A lower criticality is given to the “Moon out of FOV” scenario since it could be caused by the previous failures. Finally, the scenario that affects science is considered less critical since it does not jeopardise the safety of the satellite.

The recovery strategy of this FR module follows the general model described in Section 4.1, based on a sequence of recovery levels. Level L2 is not possible, due to the lack of redundancy. The recovery parameters were chosen arbitrarily: the failures with higher criticality have shorter L0 time of 20 [s], while the other failures have an L0 time of 60 [s]. The reset-timer is set to 20 [s] and the maximum number of resets during L1 is set to 4.

4.8.2 Stateflow model

The module is implemented in a Stateflow chart, which takes the FDI log as input and returns the FR log as an output. The FR log is an array of 2 elements [A, B]: A indicates which failure is being recovered, and B specifies the recovery action that is executed. The possible values of A and B are shown in Table 4-7.

Table 4-7: FR log of the Camera module

Element	Value	Meaning
A	0	No failure
	1	Malformed data package
	2	Negative validity flag
	3	Camera disturbances
	4	Navigation frequency locked at 1 image per minute (high-frequency)
	5	Navigation frequency locked at 1 image per 10 minutes (low-frequency)
	6	Navigation frequency locked at 0 Hz
	7	Navigation frequency general failure
	8	Camera: Moon out of FOV failure
	9	Low science frequency
	10	High science frequency
B	0	No action
	1	L0 – wait
	2	L1 - reset
	3	L1 – wait
	4	L3

The Stateflow model of the Camera FR is not reported, as it is implemented following a similar approach to the other modules, e.g. the main thruster module described in Section 4.2.

4.9 Deployment module

This section is dedicated to the description of the FR module aimed at the recovery of eventual failures during Deployment (Phase 0 of the mission).

4.9.1 Recovery strategy

The FR for the deployment does not follow the general strategy that was described in Section 4.1 and used for the other subsystems, due to the peculiarity of the module. In fact, the FDIR for the deployment is not expected to be performed continuously during the mission, but only during Phase 0, when the deployment is scheduled to occur. Therefore, not all the subsystems will be active during that Phase, and certain recovery actions might not be possible. Moreover, the strategy for the detection of deployment failures is peculiar, as it is not based on checking directly the signals coming from the units: the failure flag is sent by the OBC after the maximum number of attempts are tried, or in case the battery voltage is not sufficient; hence, it would be unreasonable to use L0 and L1 since these recovery levels are expected to be already implemented in the satellite operations before the FDIR is triggered. It is, therefore, necessary to design a proper recovery strategy in case of deployment failure.

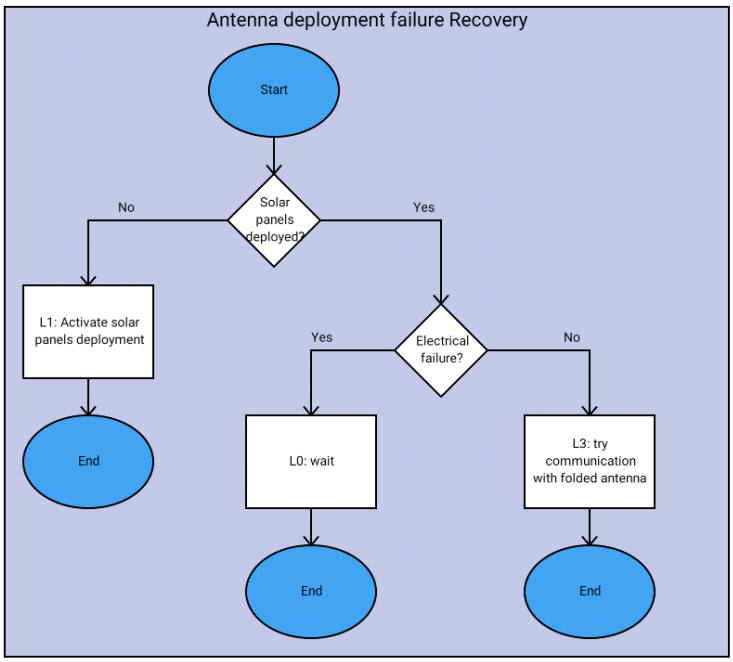

Figure 4-13: architecture of the "Recovery" state in case of failure of the antenna deployment

During Phase 0, two subsystems will be deployed: the antennas and the solar arrays. Thus, two different FDIR models will be built, one for each system: the architectures are shown in Figure 4-13 and Figure 4-14. An additional distinction is made inside each model, between electrical failure and mechanical failure. The other failures that can be contained in the FDI log of deployment system are “Malformed data package” and “Wrong failure triggering”: since they are both failures of the OBC, they will not be recovered within the FR of the deployment, but in the “Processors FR”.

The first module to be described is the deployment of the antenna, whose logic is shown in Figure 4-13. When the failure is triggered, three different situations might occur. The first operation to do is to check whether the solar panels were deployed or not; if not, the deployment of the solar panels should be prioritized, as it is a critical operation, necessary to have enough power to perform the other recovery actions. This recovery level will be called L1, despite it is not similar to the L1 used so far. If the panels are deployed, two distinct strategies can be applied: in case of an electrical failure, the only option is to wait for the batteries to charge, before trying the deployment again. Therefore, this recovery level will be called L0, for its similarity with the L0 used so far. In case of mechanical failure, the only recovery action possible is to begin communication and try to operate with the folded antenna, while keep trying to deploy it, e.g. once every orbit. Hence, since this situation is more critical and might affect significantly the operations, this recovery level will be called L3, despite it does not imply the use of Safe Configuration.

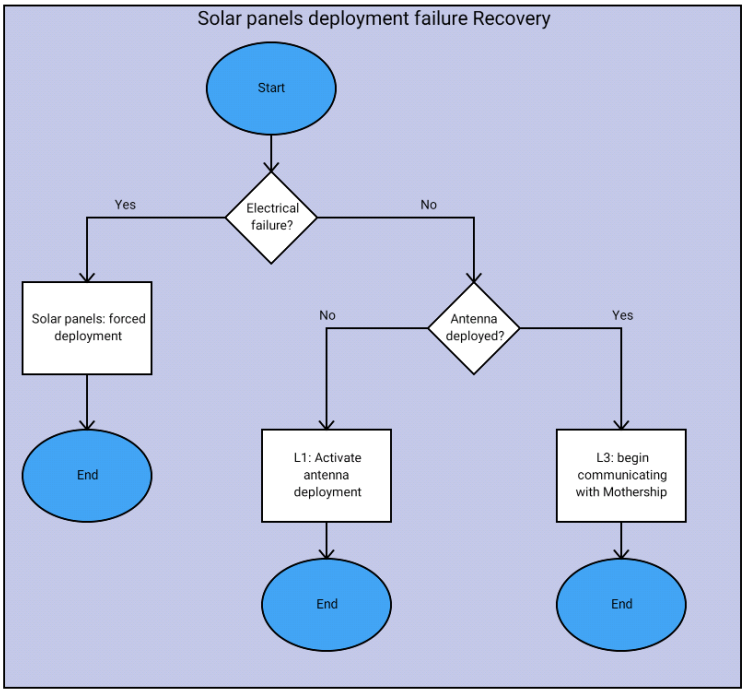

Figure 4-14: architecture of the "Recovery" state in case of failure of the solar panels deployment

In Figure 4-14, the recovery strategy for failures in the deployment of solar panels is shown. It can be seen that three different recovery levels are included, despite their application is exclusive and not consequential. In case of an electrical failure, the forced deployment of the solar arrays is tried. In fact, the electrical failure occurs when the battery voltage level is not enough to execute the deployment, and the timer to wait for the batteries to charge is over. Therefore, the only option is to force the deployment, i.e. to increase the battery DOD threshold in order to deploy the panels. This operation is an extreme measure, as it might cause the over-discharge of the batteries and compromise the rest of the mission, but it is the only way to avoid the loss of mission. In case the failure is not electrical, but mechanical, the state of antenna deployment is checked: if the antennas were not deployed yet, their deployment is prioritized, due to its criticality. This recovery level will be called L1, despite it is not similar to the L1 used so far. After that, recovery level L3 is applied: communication with the Mothership is started and the FDIR log is sent, and the S/C waits for the instructions from the Ground Station.

4.9.2 Stateflow model

Following the recovery actions strategy described above, the “Deployment FR” has been developed. The module is implemented in a Stateflow chart, which takes the FDI log as input and returns the FR log as an output. The FR log is an array of two elements [A, B]: A indicates which failure is being recovered, while B specifies the recovery action that is executed. The possible values of the FR log are explained in Table 4-8.

Table 4-8: FR log of the Deployment module

Element	Value	Meaning
A	0	No failure
	1	Antenna deployment – electrical failure
	2	Antenna deployment – mechanical failure
	3	Panels deployment – electrical failure
	4	Panels deployment – mechanical failure
B	0	No action
	1	Antenna: L0 / Panels: Forced deployment
	2	L1
	3	L3

In Figure 4-15 the high-level view of the Deployment FR model implemented in Stateflow is shown, while the detailed view of the two sub-modules is documented in Appendix D, in Figure D-8 and Figure D-9. It can be seen that the architecture described so far has been followed: there are two main sub-modules, one for the antennas and one for the solar panels, and their internal logic is based on three recovery levels, which are applied exclusively and not consequentially. Due to the early stage of the project, it is not possible to implement in the Simulink model the actions executed at these levels of recovery, as they require the interface with the satellite operations. Therefore, at this stage, the logic is implemented, and it is expected that in the future phases of the projects the recovery actions that were described so far will be implemented in the satellite operations and triggered based on the FDIR log.

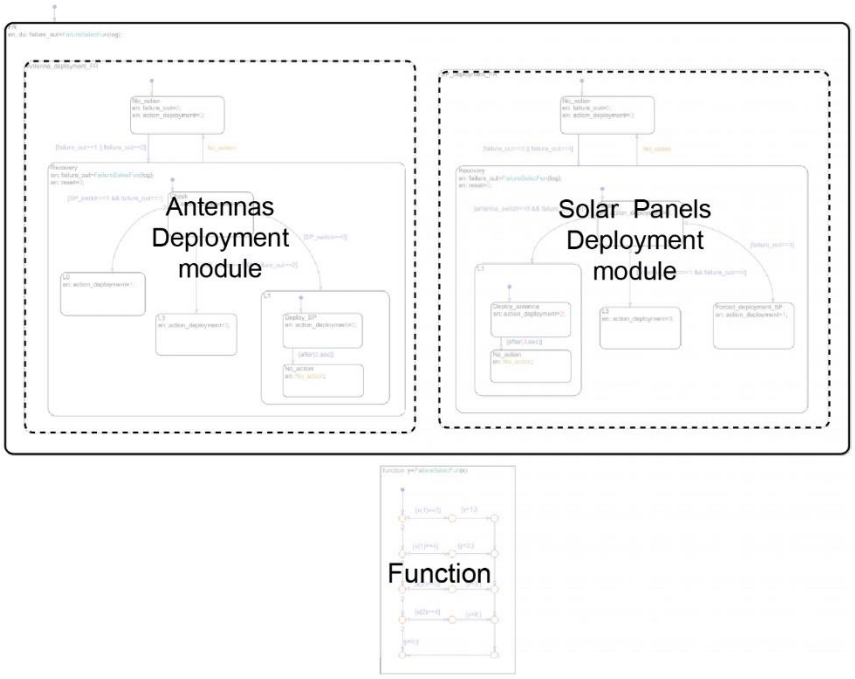

Figure 4-15: Stateflow model of the Deployment FR

4.10 Processors module

This section is dedicated to the description of the FR module aimed at the recovery of eventual failures of the processors.

4.10.1 Recovery strategy

The failures of the processors are treated in a single module; however, two additional sub-models must be added to deal with two particular failure scenarios that were categorised as OBC failures in the FMEA: human failures and time-keeping failures. It should be noticed that the phrase “human failure” in the FMECA refers to the scenario in which the ground crew inserts a wrong

parameter in the telemetry command; all the other possible human failures, e.g. manufacturing failures, are not considered, due to the absence of statistical data.

The division in three sub-modules is made due to the peculiarity of the recovery actions to be executed in response to these scenarios. In case one of the commands sent from the ground has an error, the only action that can be performed is to lock out the command and do not execute it. On top of this, the FDIR log shall be sent to the Mothership, to allow the ground crew to elaborate a new command. In case of time-keeping failure, instead, the only action to perform is to synchronize the time of LUMIO with the Mothership time and, again, send the FDIR log. Hence, a simplified logic has been applied for these two modules. On the other hand, the recovery actions in case of failure of one processor follow the general model described in Section 4.1, based on a sequence of recovery levels. L2 is included as well, as it is assumed that it will be possible to re-configure the satellite to operate with only 1 processor. However, the feasibility of this choice shall be analysed in the future phases of the project.

Finally, the definition of the recovery parameters is worth discussing. The recovery parameters used in this phase of the project are chosen arbitrarily: in case of malformed data package L0 time is set to 20 [s], while in case of negative validity flag it is reduced to 10 [s], due to the higher criticality of the latter scenario. The maximum number of resets during L1 is set arbitrarily to 4 for all the failures, and the reset-timer is set to 20 [s].

4.10.2 Stateflow model

The module is implemented in a Stateflow chart, which takes the FDI log as input and returns the FR log as an output. A peculiarity of the Processors FR module is that it does not only use the FDI log from the Processors FDI, but also from the Communication FDI, where there is the indication of human failure or time-keeping failure, and from the Deployment FDI, where there is indication of scenario “Wrong triggering of failure flag during deployment”. The FR log is an array of 4 elements [A, B, C, D, E]. A, B and C refer to the processors FR: A indicates which failure is being recovered, B specifies which processor and C indicates the recovery action that is executed. Element D refers to the recovery action to take in case of human failure in the command from the ground. Finally, element E indicates the recovery action to execute in case of time-keeping failures. The possible values of A and B are shown in Table 4-9.

The Stateflow model of the Processors FR is shown in Appendix D, in Figure D-10.

Table 4-9: FR log of the Processors module

Sub-Module	Element	Value	Meaning
Processors	A	0	No failure
		1	Malformed data package
		2	Negative validity flag
		3	Wrong triggering of failure flag during deployment (only for OBC)
	B	0	No failure
		1	Payload Processor failure
		2	OBC failure
		3	AOCS processor failure
		4	Multiple processors failure
	C	0	No action
		1	L0 – wait
		2	L1 - reset
		3	L1 – wait
		4	L2 – switch to redundancy
		5	L2 – wait
		6	L3
Human failure	D	0	No failure
		1	Lock command
Time-keeping failure (OBC)	D	0	No failure
		1	Synchronize time with Mothership

4.11 Communication module

This section is dedicated to the description of the FR module aimed at the recovery of eventual failures of the Communication subsystem.

4.11.1 Recovery strategy

The “Communication FR” is constituted of a single module, as the only failures that can be considered at this stage are the failures of the antennas. Other scenarios that were considered in the FDI cannot be recovered by LUMIO, e.g. malformed package from Mothership, or have been included in other modules, e.g. time-keeping bug.

The recovery actions in case of failure of the antennas follow the general model described in Section 4.1, based on a sequence of recovery levels. L2 is not included, as there is no redundancy in the current design. The recovery parameters used in this phase of the project are chosen arbitrarily: L0 time is set 20 [s]. The maximum number of resets during L1 is set arbitrarily to 4 for all the failures, and the reset-timer is set to 100 [s] since, in the current FDI design, the failures are detected through communication with the Mothership; hence, the expected time to wait before assessing if a failure was recovered or not is large.

4.11.2 Stateflow model

The module is implemented in a Stateflow chart, which takes the FDI log as input and returns the FR log as an output. The FR log is an array of 2 elements [A, B]: A indicates the failure that is being recovered, i.e. uplink or downlink antenna, and B indicates the recovery action that is executed. The possible values of A and B are shown in Table 4-10.

Table 4-10: FR log of the Communication module

Element	Value	Meaning
A	0	No failure
	1	Downlink failure
	2	Uplink failure
C	0	No action
	1	L0 – wait
	2	L1 - reset
	3	L1 – wait
	4	L3

The Stateflow model of the Communication FR is not reported, as it is implemented following a similar approach to the other modules, e.g. the main thruster module described in Section 4.2.

5 Simulation design

In the previous Chapters, the FDIR model developed in this Thesis was described, along with its implementation in MATLAB/Simulink coding environment. In this Chapter, the design of a simulation algorithm, aimed at producing the data packets coming from the satellite to the FDIR system, is proposed. In Section 5.1, the methodology is described. From Section 5.2 onwards, the Simulink models for each module of the simulation are explained in detail. Finally, Section 5.12 is dedicated to the description of the user interface that was developed to support the simulations. The results of the simulations performed with the algorithm introduced in this Chapter will be discussed in Chapter 6.

5.1 Methodology

In this section, the methodology used for the verification of the FDIR algorithm developed in this Thesis is described.

5.1.1 Introduction

In every space project, a significant effort is spent on verification and validation (V&V) of each component and subsystem and, finally, on the whole system. The verification is made to ensure that a product complies with its requirements. The validation is performed to prove that the product accomplishes its objective [49].

The FDIR system is no exception: according to the paper "*Testing Satellite On-Board Software - A Model Based Approach"* [50], approximatively 30% of the efforts in a space software project nowadays is spent on verification, of which about 30% is dedicated to the FDIR. In fact, FDIR is a critical system for dependability (reliability, availability and maintainability) and safety of the S/C; hence, proper verification and validation is fundamental before its final integration [9].

Due to the earliness of the FDIR design proposed in this Thesis, the only focus will be the verification of the algorithm produced so far. In fact, validation is not necessary at this stage, since the design is based on approaches commonly used in the industry for the definition of the failure scenario through the FMECA, the selection of the detection methods and the design of the recovery actions. Moreover, the absence of a finalised design and hardware makes it impossible to perform validation.

Since in this Thesis a simplistic FDIR, compatible with the current phase of the project (Phase A), was proposed, a simple sequence of simulations will be considered sufficient to achieve verification of the FDIR design at the current state. The simulations will have one objective:

To verify that the FDIR algorithm is able to detect and isolate the scenarios included in the FMECA at this phase and to produce the required recovery sequence.

In fact, the FDIR algorithm developed in this Thesis takes in some inputs (the data packets from the satellite) and returns the FDIR log as an output. Since the FDIR design is based on the FMECA, it is possible to use the scenarios of the FMECA worksheet as test cases; this is also the baseline for the study documented in [50]. The simplest option to test the FDIR algorithm is to manually create in MATLAB the data packages associated with each scenario; however, this approach is time-consuming and prone to human errors. Hence, a different methodology is followed: a test model is built in Simulink, which can reproduce all the possible behaviours of the system under test, i.e. all the failure scenarios. The only action performed by the operator is the selection of the scenario to simulate. Interfacing the simulation model with the FDIR algorithm

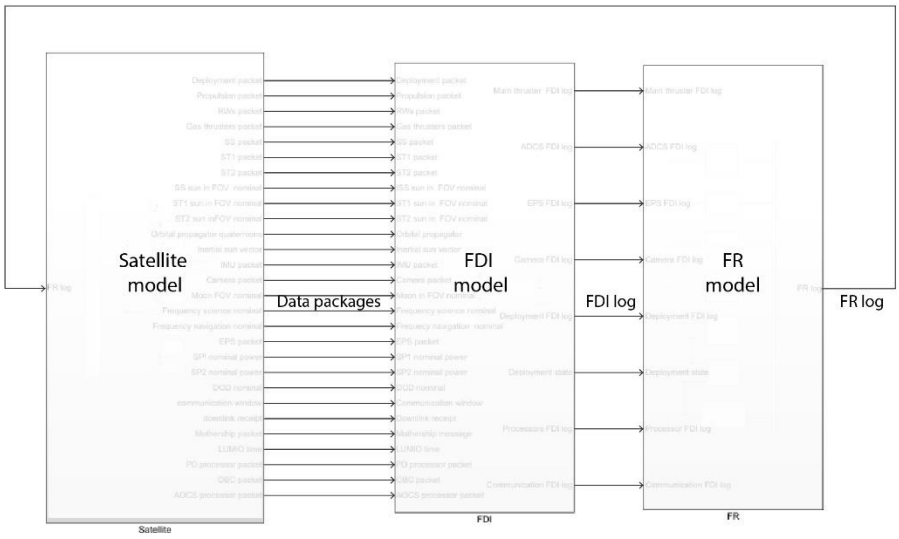

Figure 5-1: interaction between the satellite simulation model (left block), the FDI model (centre block) and the FR model (right block), in Simulink

allows verifying if, given the inputs associated with a certain scenario, the correct FDIR log is created. The interaction between the models in Simulink is shown in Figure 5-1. A similar approach was followed in the research described in the paper "*Simulink-Based FDI Simulator for Autonomous Low Earth Orbit Satellite"* [51], where the FDI algorithm developed by the authors for a LEO S/C was tested with the aid of a simulator built in the Simulink environment. However, the study in [51] was focused solely on the detection and isolation of ADCS failures; thus, a more detailed simulation model was designed, comprising the rigid body dynamics and the flight software; such level of detail was not reached in this Thesis, due to the earliness of the FDIR design.

The testing methodology used in this Thesis has several limitations, which are in conformity with the phase of the project. In the future phases, the first step will be the review and update of the simulator, based on the more detailed information about the S/C design. The implementation of the dynamics and the software in the simulation are recommended, as done in [51]. Later, one fundamental step will be to run the algorithm on the actual on-board processor, to prove its functionalities despite the hardware constraints (memory load, processing power, etc) [10]. Finally, in the more advanced phases of the project, hardware-in-the-loop tests will be crucial to validate the system before flight [9; 10].

5.1.2 Simulation model

As it was mentioned above, a model to simulate the satellite was built in Simulink. Following the architectures of the FDI and the FR, see Figure 3-1 and Figure 4-1, the satellite model was also divided into modules, one for each module of the FDIR.

The modules are built to simulate the behaviour of each subsystem/component of the S/C, with a level of detail in accordance with the analysis done in this Thesis. Each module produces as output a data package, built following the assumptions made in Chapter 3, and takes as inputs several **scenario parameters**, which are read directly from the MATLAB main workspace. Hence, instead of producing each packet, the user can simply change a limited number of parameters in MATLAB and run the Simulink simulation. There are two types of scenario parameters: some are used to specify certain variables of the simulation, e.g. the thrust command sent to the main thruster, others are used to inject a failure in the system, e.g. malformed data package.

In order to ease the FDI simulation process, a user interface was built, allowing the user to easily select which failure to inject, if any, and check the output of the FDI model. The flow of operations, from the decision of the scenario to inject until the observation of the simulation results, is shown in Figure 5-2. When a scenario is selected by the user, the user interface generates the

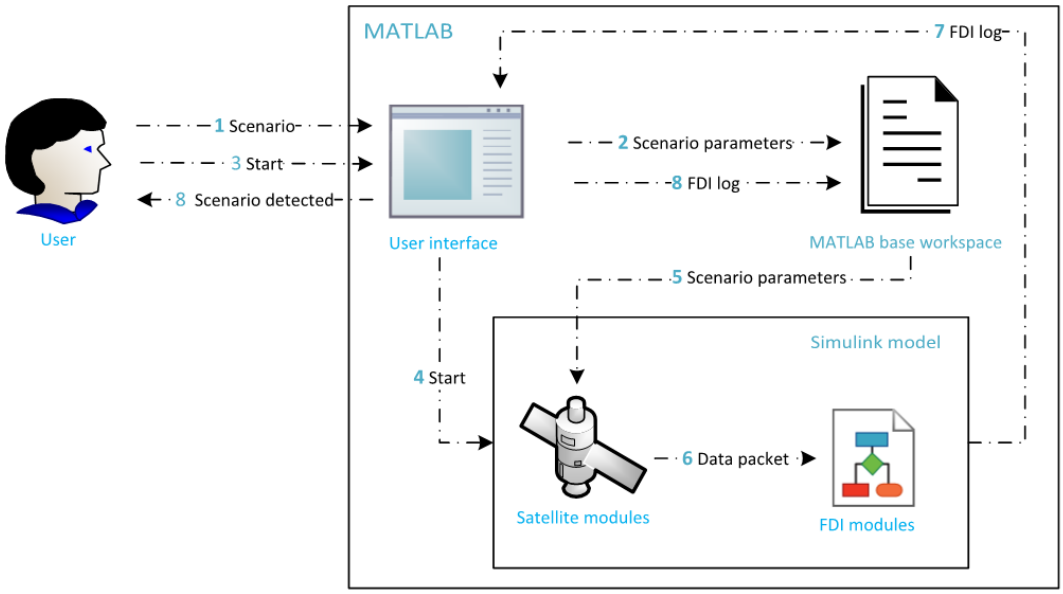

Figure 5-2: flow of operations of the FDI model tests, from the user's decision of the scenario to inject until the observation of the results. The user interacts with a user interface, which in turn exchanges data with the MATLAB base workspace and starts the Simul

corresponding *scenario parameters* and saves them in the MATLAB base workspace. Later, when the simulation is started, the Simulink model retrieves the values of these parameters from the workspace, then uses them to create the corresponding *data packets*. Finally, the FDI module receives the data packets and produces the *FDI log*, which is sent back to the user interface and the workspace: based on the log, the interface shows the scenario detected by the FDI system to the user, allowing to verify the correct detection of the injected failure. A more detailed description of the user-interface is given in Section 5.12.

As can be seen, the user interface can be used only for the verification of the FDI, not the FR; the reason is that verifying the FDI is easier, as it does not require to check the time evolution of the FDI log, apart from specific cases. In fact, the only important information is whether the FDI log indicates the right scenario or not. The verification of the FR, instead, relies on the analysis of the time evolution of the FR log, in order to understand if the correct recovery sequence was applied. The creation of a user interface for the FR is, therefore, a more complex activity, which was deemed unnecessary. In fact, while the proper detection of failures might be of interest for all the teams involved in LUMIO design, the verification of the recovery actions is considered a concern mainly for the team in charge of the FDIR design. To test the FR, the scenario parameters must be written manually in the MATLAB main workspace, and the desired FDIR log should be retrieved manually from the Simulink outputs.

5.1.3 Failure injection

The injection of a failure is performed through the scenario parameters. Normally, the parameters associated to failures have a value of 0; when the user wants to inject a specific failure scenario, the associated parameter is switched to a value of 1 (notice that even higher values would work, without any impact on the simulations). In fact, the value of the parameter enters in a Simulink “Switch” block, and an alternative path is activated in the model, based on the failure injected.

A critical point in the design of the satellite model is the implementation of feedback between the satellite and the FDIR, which is fundamental to the test the FR module. The feedback can be seen in Figure 5-1. In fact, it is necessary to verify that, in case a failure is recovered, the recovery actions produced by the FR are stopped consequently. Therefore, a Stateflow model was implemented for failure injection. The inputs of the model are the scenario parameter associated with the specific failure, the FR log and two additional parameters, called “solvable L0” and “solvable L1”. If “solvable L0” is switched to 1 by the user, the failure injected will recover within 10

[s] (the actual time is generated randomly), i.e. the failure parameter will be set to 0. If “solvable L1” is switched to 1 by the user, the failure will be recovered within 4 reset flags (the actual number of flags is generated randomly). Finally, if L2 is applicable and means to change the system configuration in order to solve a failure, the L2 flag will also switch the failure parameter to 0.

The inputs and output of the Stateflow chart used for the injection of a failure are shown in Figure 5-3, while the logic of the chart can be seen from Figure 5-4.

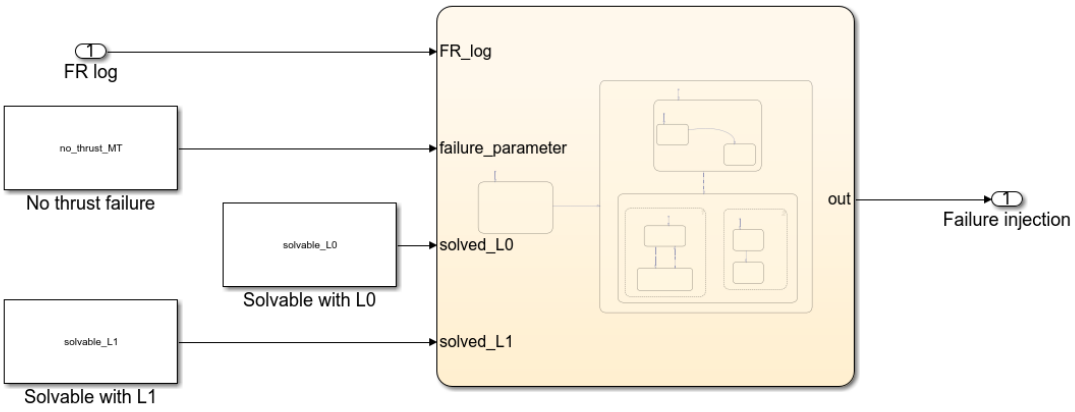

Figure 5-3: inputs and outputs of the Stateflow chart for failure injection

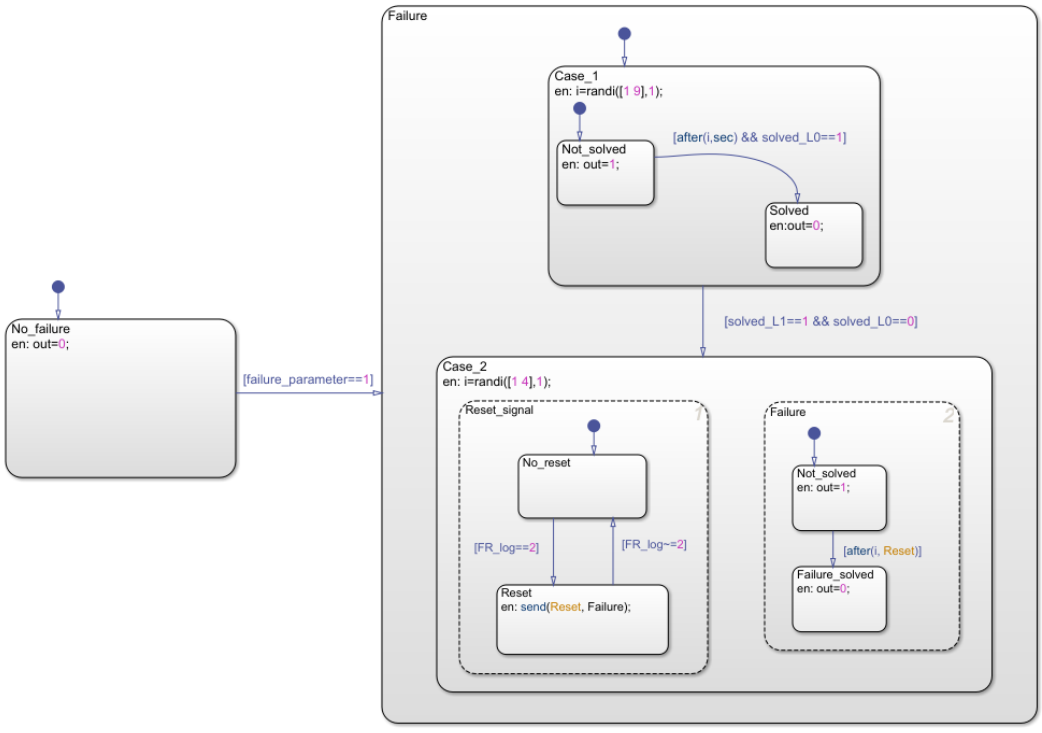

Figure 5-4: Stateflow chart used to inject a failure

5.2 Main thruster module

The “Main thruster” block creates the input coming from the satellite. To do so, firstly, it receives the parameters documented in Table 5-1, which define the scenario to represent. As can be seen, the parameters “command” and “temperature” are used to select the nominal scenario, while the

Table 5-1: scenario parameters needed by the "Main thruster" model

Input	Value	Comment
Malformed package MT	0/1	0 for normal packet, 1 for malformed packet.
Validity flag MT	0/1	
Command MT	0-1.5 [dN]	Later this input is converted in [cN] to represent it with 4 bits.
No thrust MT	0/1	1 to inject the “no thrust” failure scenario.
Locked output MT	0/1	1 to inject the “locked output” failure scenario.
Sensor failure valve MT	0/1	1 to inject a failure of the valve sensor.
Sensor failure propellant MT	0/1	1 to inject a failure in the propellant budget sensor
Temperature MT	+/-100 [°C]	Temperature of the thruster. Based on this value, the temperatures of the three sensors will be produced.
Frozen sensor MT 1	0/1	0 if normal sensor, 1 for frozen sensor.
Frozen sensor MT 2	0/1	0 if normal sensor, 1 for frozen sensor.
Frozen sensor MT 3	0/1	0 if normal sensor, 1 for frozen sensor.
General failure MT 1	0/1	0 if normal sensor, 1 if failure (the temperature has an offset of 50 °C with respect to the real one).
General failure MT 2	0/1	0 if normal sensor, 1 if failure (the temperature has an offset of 50 °C with respect to the real one).
General failure MT 3	0/1	0 if normal sensor, 1 if failure (the temperature has an offset of 50 °C with respect to the real one).
Sign(T)	-1/1	A function that depends on the value of the temperature of the main thruster: if T>=0, then sign(T)=1, otherwise Sign(T)=-1. It is used to produce the signal of the temperature sensors in case of failure.

others are all used to inject failures. Later, based on those parameters, a packet containing the information listed in Table 3-2, in Section 3.2, is created.

The data contained in the package are collected in a binary vector using the Simulink predefined function “Pack”, and control bits are appended to the vector, using the CRC generator block, a predefined function in Simulink. In order to simulate a malformed data package failure, the final package goes into a user-defined function, which randomly changes the value of one bit. This failure can be triggered by the user when switching the input parameter “malformed package propulsion” from 0 to 1. This sequence of operations can be seen from the high-level view of the Simulink model, in Figure 5-5.

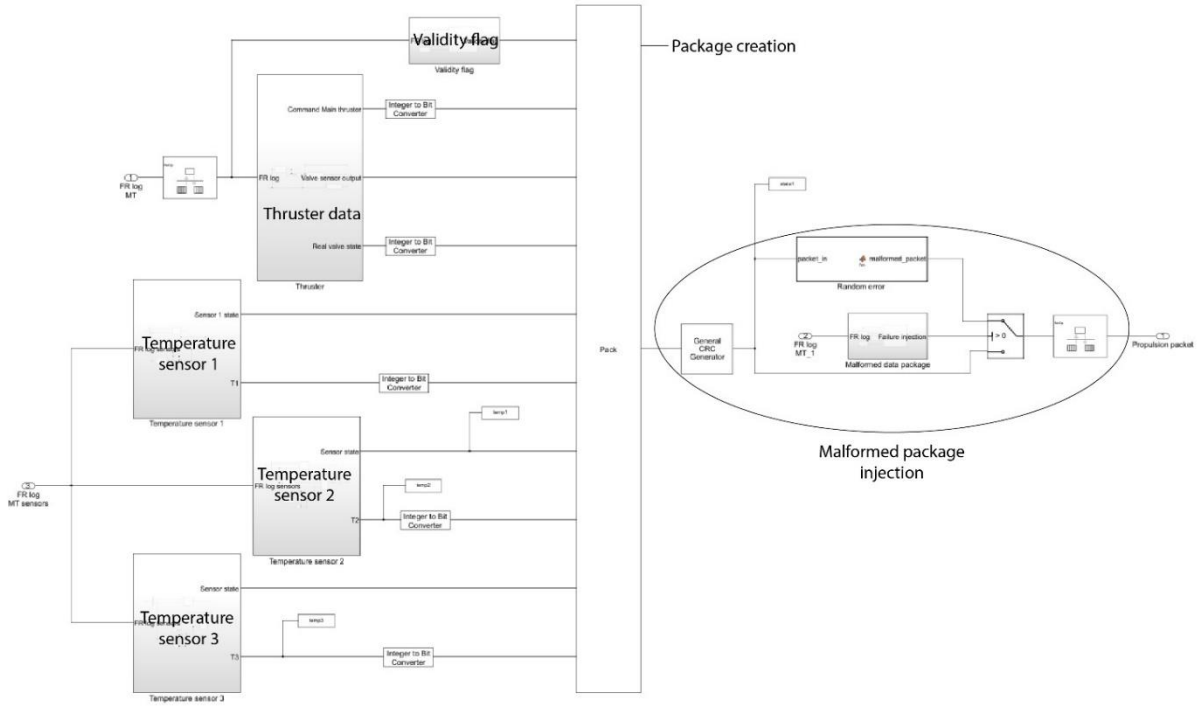

Figure 5-5: high-level structure of the Main Thruster simulation model

From Figure 5-5, it can be seen that different blocks are included in the Main Thruster model. The first is aimed at creating the validity flag, with a logic equal to the one described for failure injection in Section 5.1.3: if the validity flag selected by the user is 0, i.e. a failure occurred, the same value is generated unless the parameters “solvable L0” or “solvable L1” have a value of 1.

The second block creates the proper thruster’s data, namely the command, the valve state and the propellant budget. The structure of this block is seen in Figure F-1, documented in Appendix F. The command is selected by the user, and in case of command out of range the same logic explained for other failures apply, see Section 5.1.3. Based on the command, the valve state is generated, following the logic in Figure F-2, in Appendix F: in the nominal scenario, the valve is open if the command is greater than 0 [N], otherwise, it is closed. However, three failures can be injected. In the case of “No thrust” or “Locked output”, the valve state is opposite to the nominal. In the case of “Valve sensor failure”, the real valve state follows the nominal scenario, but the valve state written in the package is wrong. Finally, the propellant budget is created, based on the real valve state, as it is shown in Figure F-3. The signal of the propellant budget can be either constant or decreasing. If the valve is closed, the propellant budget is constant at an arbitrary value of 1950 [g]; if the valve is open, the propellant budget is represented with a ramp with an initial value of 1950 [g] and a slope of -1/20 [g/s]. Therefore, the signal decreases by 1 unit every 20 seconds. This value has been chosen based on the value of the mass flow rate of 0.05 [g/s], calculated from the reference [1]. A higher accuracy could have been used; however, this value was chosen to take into account the difficulties in measuring the propellant level on-orbit, mentioned in Section 3.2. In case the scenario “Propellant sensor failure” is injected, the signal will be the opposite: constant when the valve is open, decreasing when it is closed.

The final three blocks are aimed at producing the data of the temperature sensors. As can be seen from Table 5-1, the temperature of the thruster is selected by the user. However, in the nominal scenario, the temperature oscillates around the temperature of the thruster with arbitrary amplitude and frequency: amplitude of 10 [°C] and frequency of 1/20 [1/s]. This choice has been done to allow for the distinction between frozen sensor and general failure but has a significant drawback: the user does not have complete control over the temperature of the unit in the simulation. Therefore, if a temperature close to the limits of over/underheating is selected, the failure will not be triggered continuously; however, this can be solved simply by changing the temperature trend in the model.

The data from each temperature sensor are generated as shown in Figure F-4, in Appendix F: the initial state of the sensor is “on” and it is switched to “off” in case an L2 flag from the FDIR is received. The temperature measurement follows the real temperature unless a failure is selected. When the “frozen sensor” failure is injected, the signal stops oscillating after 10 seconds from the beginning of the simulation (arbitrary time) and remains constant. When the “general failure” is injected, the signal has an additional bias of 50 [°C] (arbitrary bias).

5.3 ADCS module – Reaction Wheels

The “Reaction Wheels” block creates the input coming from the satellite. To do so, firstly, it receives the parameters documented in Table 5-2, which define the scenario to represent. Later, based on those parameters, a packet containing the information listed in Table 3-5, in Section 3.3, is created. The data listed in the table are collected in a binary vector and control bits are appended to the vector, using the CRC generator block. In order to simulate a malformed data package failure, the same strategy explained for the main thruster in Section 5.2 has been used.

As it can be seen from Table 5-2, two parameters are used to change the nominal scenario: the RWs mode, which is used to distinguish between “idle” and “active” mode, and the temperature. The other parameters are used to inject failures.

The high-level structure of the Simulink module is shown in Figure F-5, in Appendix F. Different blocks can be distinguished. The first two blocks generate the validity flag and the response time,

Table 5-2: input parameters needed by the "Reaction wheels" model

Input	Value	Comment
Malformed package RW	0/1	0 for normal packet, 1 for malformed packet.
RW validity flag	0/1	0 if negative, 1 if positive.
Response time RW	0-10 [s]	The response time of the system.
RW mode	0/1	0 if idle, 1 if active.
RW1 undervoltage	0/1	1 if RW1 is undervoltage.
Locked speed RW1	0/1	1 if RW1 has locked speed.
RW1 general speed failure	0/1	1 if RW1 has a general failure.
RW2 undervoltage	0/1	1 if RW2 is undervoltage.
Locked speed RW2	0/1	1 if RW2 has locked speed.
RW2 general speed failure	0/1	1 if RW2 has a general failure.
RW3 undervoltage	0/1	1 if RW3 is undervoltage.
Locked speed RW3	0/1	1 if RW3 has locked speed.
RW3 general speed failure	0/1	1 if RW3 has a general failure.
Temperature RWs	+/-100 [°C]	Temperature of the RWs. Based on this value, the temperatures of the three sensors will be produced.
Frozen sensor RW1	0/1	0 if normal sensor, 1 for frozen sensor.
Frozen sensor RW2	0/1	0 if normal sensor, 1 for frozen sensor.
Frozen sensor RW3	0/1	0 if normal sensor, 1 for frozen sensor.
General failure RW1	0/1	0 if normal sensor, 1 if failure (the temperature has an offset of 50 °C with respect to the real one).
General failure RW2	0/1	0 if normal sensor, 1 if failure (the temperature has an offset of 50 °C with respect to the real one).
General failure RW3	0/1	0 if normal sensor, 1 if failure (the temperature has an offset of 50 °C with respect to the real one).
Sign(T) RW	-1/1	A function that depends on the value of the temperature of the RWs: if T>=0, then sign(T)=1, otherwise Sign(T)=-1. It is used to produce the signal of the temperature sensors in case of failure.

respectively, following the failure injection logic explained in Section 5.1.3.

Three blocks are dedicated to the generation of the signals coming from the RWs. The structure of one block is shown in Figure F-6. If the mode is “idle”, the voltage, commanded speed and speed of each wheel are set to 0. In case the RWs are active, instead, a different logic is applied and can be seen in Figure F-7. The reaction wheels are actuated through a brushless DC motor: to adjust the speed based on the command, the voltage applied to the motor is changed. Despite the existing correlation between the voltage applied to a RW and its speed, in the Simulink model, the two quantities are de-coupled, in order to simplify the verification of the FDI module. The voltage can be either 9 [V], which is in the nominal range, or 2 [V], which is under the nominal range; the value depends on the value of the parameter “RW undervoltage”. Since the FDI check is based on a simple threshold, i.e. voltage higher or lower than a threshold, this set-up is sufficient to verify the correct functioning of the FDI module. A more complex simulation, based on reproducing the actual control loop of a DC motor, would increase the complexity of the model but would not produce significant advantages at this stage, because the technical details of the RW motors are not known yet.

The reaction wheel command is produced as a ramp signal with a fixed slope, i.e. acceleration, of 0.5 [rpm/s]. In fact, the purpose of the RWs is to produce torque, which is achieved by changing their rotational speed; hence, the simplest way to simulate the system is to create a command with constant acceleration. In case there is no failure, the RW speed is created equal to the speed command. In case of “Locked speed” failure, the RW speed follows the command for 10 [s] (arbitrary value), but later it freezes at the last value and it is kept constant. In case of “General failure”, a bias of -200 [rpm] is added to the commanded speed.

Finally, the last blocks are related to the thermal control system. The models for the three temperature sensors follow the logic described in Section 5.2 for the main thruster. However, since the RWs are also equipped with a heater, a block for the heater is added and it is reported in

Appendix F, Figure F-8. In the nominal scenario, the heater voltage is set to 9 [V] when the heater is on (temperature below 20 [°C]); otherwise, it is set to 0 [°C]. In case a heater failure is injected, the values are switched.

5.4 ADCS module – Gas Thrusters

The “Gas thrusters” block creates the input coming from the satellite. To do so, firstly, it receives the parameters documented in Table 5-3, which define the scenario to represent. Later, based on those parameters, the packet containing the information listed in Table 3-8, in Section 3.4, is created. Since the four gas thrusters are part of the same system, it is assumed that their data will be contained by a unique packet. The data listed in the table are collected in a binary vector and control bits are appended to the vector, using the CRC generator block. In order to simulate a malformed data package failure, the same strategy explained for the main thruster in Section 5.2 has been used.

As can be seen from Table 5-3, the nominal scenario can be changed by changing the thrust command of each thruster; all the other parameters are used to inject failures.

Table 5-3: input parameters needed by the “Gas thrusters” model

Input	Value	Comment
Malformed package GT	0/1	0 in case there is no failure, 1 to inject the “malformed package” scenario.
Main thruster validity flag	0/1	0 in case there is no failure, 1 to inject the “negative validity flag” scenario.
Sensor failure propellant GT	0/1	0 in case there is no failure, 1 to inject the “GT propellant sensor failure” scenario.
Thrust command GT1	0-15 [mN]	The thrust command sent to GT1.
No thrust GT1	0/1	0 in case there is no failure, 1 to inject the “no thrust” scenario.
Locked output GT1	0/1	0 in case there is no failure, 1 to inject the “locked output” scenario.
Sensor failure valve GT1	0/1	0 in case there is no failure, 1 to inject the “GT1 valve sensor failure” scenario.
Thrust command GT2	0-15 [mN]	The thrust command sent to GT1.
No thrust GT2	0/1	0 in case there is no failure, 1 to inject the “no thrust” scenario.
Locked output GT2	0/1	0 in case there is no failure, 1 to inject the “locked output” scenario.
Sensor failure valve GT2	0/1	0 in case there is no failure, 1 to inject the “GT1 valve sensor failure” scenario.
Thrust command GT3	0-15 [mN]	The thrust command sent to GT1.
No thrust GT3	0/1	0 in case there is no failure, 1 to inject the “no thrust” scenario.
Locked output GT3	0/1	0 in case there is no failure, 1 to inject the “locked output” scenario.
Sensor failure valve GT3	0/1	0 in case there is no failure, 1 to inject the “GT1 valve sensor failure” scenario.
Thrust command GT4	0-15 [mN]	The thrust command sent to GT1.
No thrust GT4	0/1	0 in case there is no failure, 1 to inject the “no thrust” scenario.
Locked output GT4	0/1	0 in case there is no failure, 1 to inject the “locked output” scenario.
Sensor failure valve GT4	0/1	0 in case there is no failure, 1 to inject the “GT1 valve sensor failure” scenario.

The high-level structure of the Simulink model is shown in Figure F-9, in Appendix F. The model is similar to the main thruster model, but there are 4 thruster blocks, instead of one, and no thermal control blocks. The logic for the creation of the data of each thruster is the same as the one explained for the main thruster in Section 5.2, based on the command value to generate the nominal scenario unless a failure is selected. The generation of the propellant budget is also similar: if at least one valve is open, the propellant is decreasing; otherwise, it is constant. The propellant budget as an initial value of 210 [g]; when it decreases, it is a ramp with a slope of -1/20

[g/s]. Therefore, the signal decreases by 1 unit every 20 seconds. This value has been chosen based on the value of the mass flow rate of 0.05 [g/s], calculated from the reference [1].

5.5 ADCS module – Attitude determination

The “Attitude determination” block creates the input coming from the satellite. To do so, firstly, it receives the parameters documented in Table 5-4, which define the scenario to represent. Later, based on those parameters, the packets containing the information listed in Table 3-11, in Section 3.5, are created. To represent the real system, each sensor (SS and STs) is associated with its own packet. The data listed in the table are collected in a binary vector and control bits are appended to the vector, using the CRC generator block. In order to simulate a malformed data package failure, the same strategy explained for the main thruster in Section 5.2 has been used.

In addition to the three aforementioned packets, two additional information will be sent: the real attitude quaternions and the Sun-vector in the inertial reference frame. These latter data are needed by the FDI module to perform cross-checks. In the real system, the expected attitude quaternions will be sent by the orbital propagator; at this stage, the orbital propagator will be modelled as an infallible system that sends the real attitude. On the other hand, $r_{sun}|N$, the Sun-vector in the inertial reference frame, will be calculated by the orbital propagator based on the time epoch and the position of the S/C; at this stage, a constant and random value for this vector is assumed.

From Table 5-2, it can be seen that several parameters are used to select the nominal scenario. Firstly, the initial value of the angular speed ω of the S/C can be chosen, along with its trend. The two options are constant angular speed or changing angular speed: both these scenarios are implemented to make the simulations more realistic but have in principle no impact on the correct functioning of the sensors. The nominal orientation of the sensors with respect to the Sun can also be changed: it can be selected if, in the nominal scenario, the Sun sensor and the star-trackers have the Sun in their field of view. Finally, the temperatures of ST1 and ST2 can be selected.

It is paramount to show how the data of the packets described in Table 3-11 are created in the “Attitude determination” block, starting from the scenario parameters of Table 5-4, which are selected by the user. In Figure F-10, in Appendix F, the high-level composition of the block is shown: it is divided into several subsystems, characterized by different colours.

The first subsystem is the “Real attitude kinematic” block, coloured in blue in Figure F-10. This block is aimed at the simulation of the real S/C kinematics, i.e. the change of its attitude with time and can be seen in Figure F-11, in Appendix F. Firstly, it generates the normalized Sun-vector in the inertial reference frame, $r_{sun}|N$, which is later sent to the FDIR system by the orbital propagator. In this phase, an arbitrary vector is used. Secondly, it generates the expected value of the “Sun in FOV” parameter for each sensor: this value is important to distinguish between failure scenarios, e.g. the Sun sensor does not detect the Sun due to an internal failure, and nominal scenarios, e.g. the Sun sensor does not detect the Sun because the satellite is experiencing an eclipse. These parameters are directly generated from the scenario parameters “SS Sun FOV nominal” and “ST Sun FOV nominal”, listed in Table 5-4. Finally, the block generates the real attitude of the S/C, based on the value of the initial angular velocity ω, whose components are generated from the scenario parameters “Omega 1”, “Omega 2” and “Omega 3”, and of the parameter “Angular speed mode”. The attitude is represented with a vector of three Euler angles $[\theta_x\ \theta_y\ \theta_z]$, which represent the rotation from the body reference frame B to the inertial frame N. The Euler angles were chosen to describe the real attitude since they are the most intuitive representation method. However, in order to integrate the S/C kinematic equations, the Euler angles are converted into quaternions since they allow to avoid a singularity that is encountered when $\theta_y = 90°$ [36]. Therefore, the kinematic is computed in the model by the integration of Equation (30), starting from an arbitrary initial attitude.

Table 5-4: input parameters needed by the “Attitude determination” model

Input	Value	Comment
Omega 1	+/-400 [deg/s]	S/C initial rotational speed around its x-axis.
Omega 2	+/-400 [deg/s]	S/C initial rotational speed around its y-axis.
Omega 3	+/-400 [deg/s]	S/C initial rotational speed around its z-axis.
Angular speed mode	0/1	0 for changing rotational speed, 1 for constant rotational speed.
SS Sun FOV nominal	0/1	1 if Sun sensor has Sun in FOV in the nominal scenario, 0 if the Sun is expected to be out of FOV.
ST1 Sun FOV nominal	0/1	1 if ST1 has Sun in FOV in the nominal scenario, 0 if the Sun is expected to be out of FOV.
ST2 Sun FOV nominal	0/1	1 if ST2 has Sun in FOV in the nominal scenario, 0 if the Sun is expected to be out of FOV.
Malformed package SS	0/1	1 if packet from SS is malformed, 0 if not.
Validity flag SS	0/1	1 if the validity flag is positive, 0 if negative.
SS sun FOV	0/1	0 if Sun is out of FOV, 1 if it is in the FOV
SS frozen	0/1	1 if SS if frozen.
SS general failure	0/1	1 if SS has a general failure (offset of 5° in one of its measurements).
Malformed package ST1	0/1	1 if packet from ST1 is malformed, 0 if not.
Validity flag ST1	0/1	1 if the validity flag is positive, 0 if negative.
ST1 sun FOV	0/1	0 if Sun is out of FOV, 1 if it is in the FOV
ST1 frozen	0/1	1 if ST1 if frozen.
ST1 general failure	0/1	1 if ST1 has a general failure (offset of 5° in the measurement of the x-axis attitude).
Temperature ST1	+/-100 [°C]	Temperature of ST1. Based on this value, the temperatures of the three sensors will be produced.
Frozen temperature sensor-1 ST1	0/1	0 if normal sensor, 1 for frozen sensor.
Frozen temperature sensor-2 ST1	0/1	0 if normal sensor, 1 for frozen sensor.
Frozen temperature sensor-3 ST1	0/1	0 if normal sensor, 1 for frozen sensor.
General failure temperature-1 ST1	0/1	0 if normal sensor, 1 if failure (the temperature has an offset of 50 °C with respect to the real one).
General failure temperature-2 ST1	0/1	0 if normal sensor, 1 if failure (the temperature has an offset of 50 °C with respect to the real one).
General failure temperature-3 ST1	0/1	0 if normal sensor, 1 if failure (the temperature has an offset of 50 °C with respect to the real one).
Sign(T) ST1	-1/1	A function that depends on the value of the temperature of the ST1: if T>=0, then sign(T)=1, otherwise Sign(T)=-1. It is used to produce the signal of the temperature sensors in case of failure.
ST1 heater failure	0/1	1 to inject failure, 0 if ok.
Malformed package ST2	0/1	1 if packet from ST2 is malformed, 0 if not.
Validity flag ST2	0/1	1 if the validity flag is positive, 0 if negative.
ST2 sun FOV	0/1	0 if Sun is out of FOV, 1 if it is in the FOV
ST2 frozen	0/1	1 if ST2 if frozen.
ST2 general failure	0/1	1 if ST2 has a general failure (offset of 5° in the measurement of the x-axis attitude).
Temperature ST2	+/-100 [°C]	Temperature of ST2. Based on this value, the temperatures of the three sensors will be produced.
Frozen temperature sensor-1 ST2	0/1	0 if normal sensor, 1 for frozen sensor.
Frozen temperature sensor-2 ST2	0/1	0 if normal sensor, 1 for frozen sensor.
Frozen temperature sensor-3 ST2	0/1	0 if normal sensor, 1 for frozen sensor.
General failure temperature-1 ST2	0/1	0 if normal sensor, 1 if failure (the temperature has an offset of 50 °C with respect to the real one).
General failure temperature-2 ST2	0/1	0 if normal sensor, 1 if failure (the temperature has an offset of 50 °C with respect to the real one).
General failure temperature-3 ST2	0/1	0 if normal sensor, 1 if failure (the temperature has an offset of 50 °C with respect to the real one).
Sign(T) ST2	-1/1	A function that depends on the value of the temperature of the ST2: if T>=0, then sign(T)=1, otherwise Sign(T)=-1. It is used to produce the signal of the temperature sensors in case of failure.
ST2 heater failure	0/1	1 to inject failure, 0 if ok.

$$\begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \\ \dot{q}_3 \\ \dot{q}_4 \end{bmatrix} = \frac{1}{2} \begin{bmatrix} 0 & \omega_3 & -\omega_2 & \omega_1 \\ -\omega_3 & 0 & \omega_1 & \omega_2 \\ \omega_2 & -\omega_1 & 0 & \omega_3 \\ -\omega_1 & -\omega_2 & -\omega_3 & 0 \end{bmatrix} \begin{bmatrix} q_1 \\ q_2 \\ q_3 \\ q_4 \end{bmatrix} \quad (30)$$

After the integration, the quaternions are used to create the orbital propagator output, i.e. the vector of the real attitude quaternions, but they are also converted back in a vector of Euler angles, which is sent to the blocks that simulate the attitude sensors. It should be noticed that the vector is re-arranged in the form of $[\theta_z\ \theta_x\ \theta_y]$, because this order is needed by the Simulink functions in case the rotation sequence ZYX is chosen.

The yellow subsystem in Figure F-10 generates the output from the Sun sensor. The Simulink model can be seen in detail in Figure F-12, in Appendix F. Firstly, the values of the validity flag and Sun in FoV flag are created, based on the scenario parameters “Validity flag SS” and “SS Sun in FOV”. Secondly, the SS outputs, namely the angles alpha and beta, are created. To do so, the Sun vector in the body reference frame, $r_{sun}|B$, is computed, from the rotation of the Sun vector in the inertial frame $r_{sun}|N$. The vector produced in this way is represented in Cartesian coordinated; thus, later it is converted into spherical coordinates since the Sun sensor measures its azimuth and elevation. These two values are used to create the packet. In the case of “Frozen sensor” failure, alpha and beta follow the real attitude for 10 [s] (arbitrary value), but later they freeze at their last values and are kept constant. In the case of “General failure”, a bias of 5 [°] is added to the azimuth angle.

The red and green subsystems in Figure F-10 generates the output from ST1 and ST2, respectively. The detailed model is shown in Figure F-13. Firstly, the values of the validity flag and Sun in FoV flag are created, based on the scenario parameters “Validity flag ST” and “ST Sun in FOV”. Later, the attitude quaternions, which describe the rotation from the body reference frame to the inertial one, are created from the Euler angles coming from the “real attitude” subsystem. An additional bias of 3 [sterad] (arbitrary value) is added to one of the Euler angles prior to the conversion in quaternions, in order to simulate the inaccuracy of the sensors. In the case of “Frozen sensor” failure, the quaternions follow the real attitude for 10 [s] (arbitrary value), but later freeze at their last values and are kept constant. In the case of “General failure”, a bias of 5 [°] is added to one of the Euler angles, before the conversion in quaternions.

In the star-tracker blocks, the signals from the temperature sensors and the heaters are created as well. The logic is the one described for the Reaction Wheels module, in Section 5.3.

5.6 ADCS module – IMU

The IMU module creates the input coming from the satellite. To do so, firstly, it receives the parameters documented in Table 5-5, which define the scenario to represent. In addition to those, it also receives data from the “Attitude determination” block: ω, the real angular velocity of the S/C. Later, based on those parameters, a packet containing the information listed in Table 3-13, in Section 3.6, is created. The data listed in the table are collected in a binary vector and control bits are appended to the vector, using the CRC generator block. In order to simulate a malformed data package failure, the same strategy explained for the main thruster in Section 5.2 has been used.

All the scenario parameters in Table 5-5 are related to a failure scenario; in fact, the nominal

Table 5-5: input parameters needed by the "IMU" model

Input	Value	Comment
Malformed package IMU	0/1	0 1 to inject the failure scenario.
IMU validity flag	0/1	0 if negative, 1 if positive.
IMU gyro general failure	0/1	1 to inject the failure scenario.
IMU gyro frozen	0/1	1 to inject the failure scenario.

scenario is specified in the scenario parameters for the Attitude determination module, where the initial value and the trend of the S/C angular speed can be selected, see Table 5-4.

The structure of the IMU model is shown in Figure F-14, reported in Appendix F. The first block generates the validity flag following the failure injection logic explained in Section 5.1.3. The second block creates the data from the accelerometers. Since no cross-check options are available in the current design for the accelerometers, the FDI logic does not use the information provided by these sensors, see Section 3.6; therefore, the output of the three accelerometers is generated as an arbitrary vector with values [1, 2, 3] [g].

The final block in Figure F-14 creates the data package from the gyroscopes and is documented in Figure F-15. The starting point for the generation of the output of the gyroscopes is the angular velocity vector ω, generated in the blue block of the "Attitude determination" system, shown in Figure F-10. It is fundamental to use the same attitude scenario used for the star-trackers since the two units are cross-checked in the FDI model. Starting from the real angular speed, the signal is generated following the logic illustrated in Figure F-15. Firstly, an arbitrary bias of 0.05 [°/s] is added to ω_1, to simulate the accuracy of the sensor. Later, two different failures can be injected, if the scenario parameters "IMU gyro general failure" or "IMU gyro frozen" are set to 1: a bias of 0.8 [°/s] can be added to ω_1, to simulate a general failure, or the signal can be frozen.

5.7 Power module

The Power module creates the input coming from the satellite. To do so, firstly, it receives the parameters documented in Table 5-6, which define the scenario to represent. Later, based on

Table 5-6: input parameters needed by the "EPS" model

Input	Value	Comment
Malformed package EPS	0/1	0 for normal packet, 1 for malformed packet.
EPS validity flag	0/1	0 if negative, 1 if positive.
Failure SP1	0/1	1 to inject the failure scenario: "SP1 loss of power".
Failure SADA SP1	0/1	1 to inject the failure scenario: "SP1 locked SADA".
Failure SP2	0/1	1 to inject the failure scenario: "SP2 loss of power".
Failure SADA SP2	0/1	1 to inject the failure scenario: "SP2 locked SADA".
EPS attitude failure	0/1	1 to inject the failure of the solar panels attitude.
Battery mode	0/1	0 if constant DOD, 1 if battery is discharging
DOD nominal	0-100 [%]	The value of the nominal DOD of the batteries
DOD B1	0-100 [%]	The value of the DOD of battery 1
Failure DOD measurement B1	0/1	1 to inject the failure in the DOD measurement of battery 1
Failure V measurement B1	0/1	1 to inject the failure in the voltage measurement of battery 1
DOD B2	0-100 [%]	The value of the DOD of battery 2
Failure DOD measurement B2	0/1	1 to inject the failure in the DOD measurement of battery 2
Failure V measurement B2	0/1	1 to inject the failure in the voltage measurement of battery 2
Temperature B	+/-100 [°C]	Temperature of the battery system. Based on this value, the temperatures of the three sensors will be produced.
Frozen sensor B1	0/1	0 if normal sensor, 1 for frozen sensor.
Frozen sensor B2	0/1	0 if normal sensor, 1 for frozen sensor.
Frozen sensor B3	0/1	0 if normal sensor, 1 for frozen sensor.
General failure B1	0/1	0 if normal sensor, 1 if failure (the temperature has an offset of 50 °C with respect to the real one).
General failure B2	0/1	0 if normal sensor, 1 if failure (the temperature has an offset of 50 °C with respect to the real one).
General failure B3	0/1	0 if normal sensor, 1 if failure (the temperature has an offset of 50 °C with respect to the real one).
Sign(T) B	-1/1	A function that depends on the value of the temperature of the batteries: if T>=0, then sign(T)=1, otherwise Sign(T)=-1. It is used to produce the signal of the temperature sensors in case of failure.
B heater failure	0/1	1 to inject failure.

those parameters, a packet containing the information listed in Table 3-15, in Section 3.7, is created. The data listed in the table are collected in a binary vector and control bits are appended to the vector, using the CRC generator block. In order to simulate a malformed data package failure, the same strategy explained for the main thruster in Section 5.2 has been used.

As it can be seen from Table 5-6, several parameters are used to change the nominal scenario of the batteries DOD: the nominal DOD can be chosen, along with its trend (constant or decreasing). The different trends have been implemented to make the simulations more realistic but have in principle no impact on the correct functioning of the FDI. On top of this, the DOD of battery 1 and battery 2 can also be chosen by the user. Finally, the temperature of the battery package can be selected. All the other scenario parameters are instead used to inject a failure.

The Power module is divided into different subsystems, which create different parts of the packet and can be seen in Figure F-16, documented in Appendix F. The first block generates the validity flag following the failure injection logic explained in Section 5.1.3.

The green block creates the data of the SADA and is shown in Figure F-17. Two failures can be injected by switching to 1 the value of the scenario parameter “SP SADA failure”, one for the SADA of the SP1 and one for the SP2. They can also be injected at the same time, to simulate a complete SADA failure. The validity flag of the SADA is 1 if no failure is injected, or 0 if at least one failure is selected.

The yellow block in Figure F-16 creates the data for the solar panels. Its logic is documented in Figure F-18, in Appendix F. The value of the nominal power produced by each solar panel is fixed at a value of 14 [W], which corresponds to half of the minimum average power needed by the S/C according to [1]. In case no failure is selected, the actual power produced by each panel will be 15 [W]. When a failure is selected, this value is diminished by 5 [W]. Three types of failures can be selected. Firstly, if the “EPS attitude failure” is switched to 1, it simulates a failure of the ADCS system; it is recommended to select this scenario while injecting a failure in one of the ADCS modules, in order to make the simulation consistent. Secondly, a solar panel failure can be simulated by switching to 1 the value of the “SP failure” parameter. Finally, a SADA failure can be injected, as explained above.

In the red block of Figure F-16, shown in Figure F-19, the temperatures of the panels are created. If any attitude failure occurred to the panel, i.e. if a SADA failure or an ADCS failure was injected, the temperature is at 10 [°C]; otherwise, the value is 80 [°C]. These arbitrary values were chosen to represent two extreme situations and will be used by the FDI only as an extreme cross-check in case of indecision.

The remaining blocks of Figure F-16 are related to the battery system. The grey blocks are related to the thermal control and produce the temperature measurements and the heater state. The logic is the one described for the Reaction Wheels module, in Section 5.3. In the blue block, instead, the relevant data about the batteries are produced.

The Simulink model of the batteries is shown in Appendix F, in Figure F-20. Firstly, the states of the batteries are created: at the beginning of a simulation they are both “on”, but if an L2 command is received from the FDIR the selected battery is switched off. Later, the DOD values of the batteries are generated, from the user’s input. If one battery is off, the nominal DOD and the other battery’s DOD will be doubled, to simulate the real behaviour of the system more realistically. When the battery mode is set to 1, i.e. discharging batteries, the DOD starts at the input value and changes with a slope of 1/20 [%/s], in order to simulate a discharging process.

Starting from the DOD of each battery, the voltage level is computed. In order to create the voltage level corresponding to a DOD state, the simple relation of Equation (29), documented in Section 3.7, is used, to make the simulation consistent with the FDI model. After the batteries DOD and voltage values are created, one of two failures can be injected for each battery. When the parameter “voltage measurement failure” is set to 1, an additional arbitrary offset of 1 [V] is added

to the voltage measurement. If the parameter “DOD measurement failure” is set to 1, an offset of 20 [%] is added to the DOD measurement. In this way, a measurement failure can be simulated.

5.8 Camera module

The Camera module creates the input coming from the satellite. The data are divided between the Camera packet, which contains the data received from LUMIO-Cam, and other data, which represent the nominal scenario and are used within the FDI module to check the behaviour of the unit. To create these packets, the block receives the parameters documented in Table 5-7, which define the scenario to represent. Later, based on those parameters, a packet containing the information listed in Table 3-17, in Section 3.8, is created. The data listed in the table are collected in a binary vector and control bits are appended to the vector, using the CRC generator block. In order to simulate a malformed data package failure, the same strategy explained for the main thruster in Section 5.2 has been used.

As it can be seen from Table 5-7, three parameters are used to change the nominal scenario: the actual navigation frequency and the navigation mode, which is used to select the nominal scenario between low frequency, high frequency and no navigation. The other parameters are used to inject failures.

Table 5-7: input parameters needed by the “Camera” model

Input	Value	Comment
Malformed package CAM	0/1	0 for normal packet, 1 for malformed packet.
CAM validity flag	0/1	0 if negative, 1 if positive.
Moon FOV nominal	0/1	Attitude of the camera according to the nominal scenario: 1 if Moon is in the FOV, 0 if not.
Science mode	0/1	1 if science mode is on, 0 if off.
Navigation mode	0/1/2	0 if navigation mode is off, 1 if low-frequency navigation, 2 for high-frequency navigation.
Disturbance failure CAM	0/1	1 to inject the failure.
Moon FOV failure CAM	0/1	1 to inject the failure.
Science frequency failure CAM	0/1	1 to inject the failure.
Navigation frequency	0-17 [mHz]	The frequency of image acquisition during navigation. It should be 0 when navigation is off, around 2 during low-frequency acquisition and around 17 during high-frequency acquisition.

The Simulink model for the creation of the camera packet is shown in Figure F-21, in Appendix F. The first block generates the validity flag, following the failure injection logic explained in Section 5.1.3.

The second block is aimed at creating the camera parameters and is shown in Figure F-22. The saturation, SNR and contrast of the camera have arbitrary nominal values of 80 %, 5 and 80 %, respectively. When the parameter “Disturbance failure CAM” is set to 1, the value of one of these parameters, randomly chosen, is changed to a value which is considered out of the nominal range, e.g. the saturation value becomes 20 %.

The third block is used to create the “Moon in FOV flag”, which has the same value of the nominal scenario unless a failure is triggered by setting the “Moon FOV failure CAM” parameter to 1: in that case, the value is the opposite of the nominal one. The same principle applies to the science frequency, which is created in the fourth block, shown in Figure F-23: the real frequency is equal to the nominal one unless the parameter “science frequency failure CAM” is set to 1.

The final block creates the data about the image acquisition frequency during navigation, following the logic shown in Figure F-24, in Appendix F. The nominal frequency is selected by the user; the possible values of the frequency are 0, 2 or 17 [mHz], which correspond to three possible

scenarios: the navigation is off, the navigation is done at low-frequency or the navigation is done at high-frequency. However, the actual frequency can be set by the user to any other value in the interval, e.g. 8 [mHz], in order to simulate a general failure. The chosen frequency value enters in a Stateflow chart similar to the one used to inject a failure and documented in Figure 5-4, in order to allow the simulation of the failure recovery.

5.9 Deployment module

The Deployment module creates the input coming from the satellite. To do so, firstly, it receives the parameters documented in Table 5-8, which define the scenario to represent. Later, based on those parameters, a packet containing the information listed in Table 3-20, in Section 3.9, is created. The data listed in the table are collected in a binary vector and control bits are appended to the vector, using the CRC generator block. In order to simulate a malformed data package failure, the same strategy explained for the main thruster in Section 5.2 has been used.

Table 5-8: input parameters needed by the "Deployment" model

Input	Value	Comment
Malformed package deployment	0/1	0 for normal packet, 1 for malformed packet.
Failure deployment antenna	0/1	0 if OK, 1 to inject failure in the antenna deployment.
OBC wrong failure flag deployment antenna	0/1	0 if OK, 1 to inject failure in the OBC: the OBC flags antenna deployment failure but there is no failure.
Failure deployment SP	0/1	0 if OK, 1 to inject failure in the SP deployment.
OBC wrong failure flag deployment SP	0/1	0 if OK, 1 to inject failure in the OBC: the OBC flags SP deployment failure but there is no failure.

The Simulink model used for the simulation is divided into two blocks, one for the antennas and one for the solar panels, as can be seen from Figure F-25, in appendix F.

The logic used to create the model for the simulation of the antenna and solar panels deployment is the same and it is shown in Figure F-26. In the nominal scenario, the switches are not pressed (the appendages are deployed) and there is no error flag sent by the OBC. If the deployment failure is injected, the switch value is set to 1 (switch is pressed) and the OBC flag is activated. If the OBC failure scenario “Wrong failure flag” is injected, the OBC flag is sent, but the switches are not pressed.

5.10 Processors module

The Processors module creates the input coming from the satellite. The data are divided into three packets, one for each processor. To create these packets, the block receives the parameters documented in Table 5-9, which define the scenario to represent. Later, based on those parameters, a packet containing the information listed in Table 3-22, in Section 3.10, is created. For each packet, the data are collected in a binary vector and control bits are appended to the vector, using the CRC generator block. In order to simulate a malformed data package failure, the same strategy explained for the main thruster in Section 5.2 has been used.

Table 5-9: input parameters needed by the “Processors" model

Input	Value	Comment
Malformed package PD processor	0/1	0 for normal packet, 1 for malformed packet.
Validity flag PD processor	0/1	0 for negative validity flag, 1 for positive validity flag.
Malformed package OBC	0/1	0 for normal packet, 1 for malformed packet.
Validity flag OBC	0/1	0 for negative validity flag, 1 for positive validity flag.
Malformed package AOCS processor	0/1	0 for normal packet, 1 for malformed packet.
Validity flag AOCS processor	0/1	0 for negative validity flag, 1 for positive validity flag.

The Simulink implementation of the Processors module is not reported: the only data in each package are the validity flag and the state of a processor, which are created following the same approach used for the other modules, e.g. the main thruster in Section 5.2.

5.11 Communication module

The Communication module creates the input coming from the satellite. The data are divided between the Mothership packet, which contains the data received from the Mothership, and other data, which represent the nominal scenario and are used within the FDI module. In particular, it is paramount to define whether the S/C is in the communication window or not and if the Mothership message was received or not. To create these packets, the “Communication” block receives the parameters documented in Table 5-10, which define the scenario to represent. Later, based on those parameters, a packet containing the information listed in Table 3-24, in Section 3.11, is created. The data listed in the table are collected in a binary vector and control bits are appended to the vector, using the CRC generator block. In order to simulate a malformed data package failure, the same strategy explained for the main thruster in Section 5.2 has been used.

Table 5-10: input parameters needed by the “Communication" model

Input	Value	Comment
Malformed package Mothership	0/1	0 for normal packet, 1 for malformed packet.
Communication window	0/1	0 if the S/C is not in the communication window, 1 if it is.
Downlink failure	0/1	0 if there is no failure, 1 if there is a downlink failure, i.e. the S/C does not receive a message during the communication window.
Ground command failure	0/1	0 if there is no failure, 1 if there is a ground command failure.
Time failure	0/1/2	0 if there is no failure, 1 to inject a failure of LUMIO on-board time-keeping.
Uplink failure	0/1	0 if there is no failure, 1 if there is an uplink failure, i.e. the Mothership di not receive the message from the S/C.

The Simulink model is shown in Figure F-27, in Appendix F. The first block in the model creates the nominal scenario as shown in Figure F-28: if the S/C is in the communication window a message is received unless a downlink failure is injected.

The Mothership package consists of three different data, as can be seen in Table 3-24. The first is the command: it is set to 20 if no failure is injected, i.e. if the parameter “Ground command failure” is set to 0; otherwise, the command is 80. The values are arbitrary and based on the assumption that the maximum command that can be received from the ground is 50. The Mothership time is created with the “clock” Simulink function, which returns the current simulation time. LUMIO time is the same, but an offset of 30 [s] is added if the “Time failure” parameter is set to 1, as can be seen from Figure F-29. Finally, the uplink receipt is created. In the nominal scenario the value is 1, i.e. message received, but it is switched to 0 if a failure is injected using the scenario parameter “Uplink failure”.

5.12 User interface

Once the simulation model has been described, the user-interface mentioned in Section 5.1.2 can be described more in detail. The interface was created using the MATLAB App Designer tool, which allows to interact with MATLAB/Simulink models, and its purpose is to ease the simulations of the FDI model. The file of the interface should be placed in the same folder of the FDIR Simulink model, to work properly.

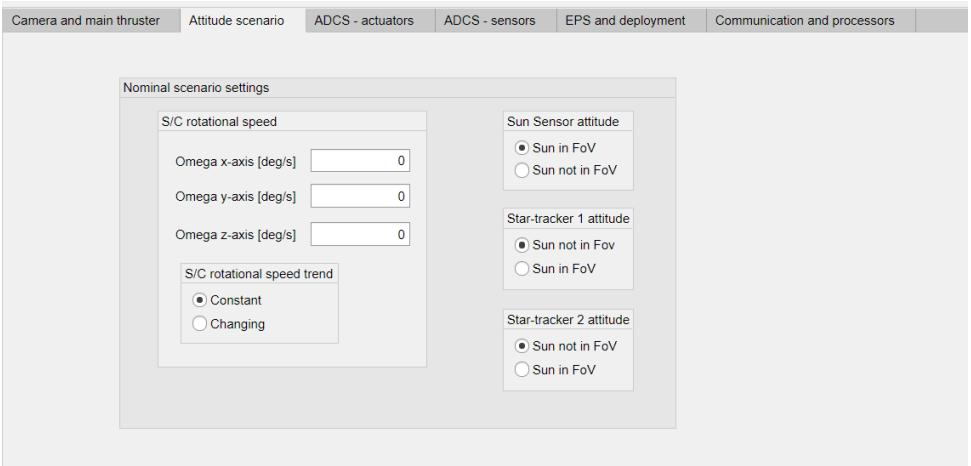

Figure 5-6: second tab of the user interface, dedicated to the selection of the nominal attitude scenario

Due to the conspicuous number of scenarios to simulate, the interface is divided into multiple tabs, each dedicated to certain modules. The picture of the second tab is shown in Figure 5-6, while the other in tabs are documented in Appendix F. The scenario parameters described in the previous chapters can be selected through thick boxes or multiple choices selection; the default selection is always the nominal scenario. Each box corresponds to one scenario parameter. The result of the FDI log, instead, is displayed with lamps, which help to discriminate between the absence of failure (lamp named “OK”) and the detection of a failure. All the possible scenarios treated in the FDI are included in the user interface. Once the scenario has been selected, the simulation can be run by pushing the “Start simulation” button, in the bottom right of the interface.

The first tab is dedicated to failures of the camera and the main thruster and is documented in Figure F-30. Each module is enclosed in a well-defined area, to ease the distinction. In the camera interface, a distinction is clear between the parameters that define the nominal scenario and those that define the failure to inject. In the main thruster module, the scenarios related to the thruster and to the thermal control are clearly distinguished. The thrust command and the temperature can be typed directly in the boxes; later, the failure can be injected. Regarding the temperature sensors, it should be noticed that the possibility of simulating the “frozen sensor” and “general failure” scenarios together has been included: the reason is that, since the temperature of the unit is periodic (see Section 5.2 for a description of the main thruster temperature in the simulation), the frozen sensor might not be detected as a failure, unless a bias is added.

The second tab is not dedicated to a particular system, but to the selection of the attitude scenario, as it was explained in Section 5.5: the user can select the initial rotational speed and its trend, and the nominal position of the Sun with respect to the field of view of the attitude determination sensors. It is shown above, in Figure 5-6.

The third tab is shown in Figure F-31 and is dedicated to failures of the ADCS actuators. The box on the left allows selecting failure for the reaction wheels, divided between proper wheel’s failures and failures of the thermal control. The second box is dedicated to the gas thrusters and follows the same principle of the main thruster interface. It should be noticed that different areas are created, one for each thruster: when a failure lamp inside one area is active, it means that the failure related to that specific unit occurred. If a lamp outside the areas lights on, instead, the failure involves the whole system, e.g. malformed data package.

Figure F-32 represents the next tab, which is used to inject failures of the ADCS sensors. For the attitude sensors, the real orientation with respect to the Sun can be chosen, which can be different from the nominal one, selected in the second tab. On top of this, the other scenarios can be selected, also those related to the thermal control of the reaction wheels. For the IMU, only gyroscope failures can be injected.

The fifth page of the interface is dedicated to the Power system and the deployment and is shown in Figure F-33. In the Power part, different areas are created for the panels, the SADA and the batteries. A clear distinction between battery 1, battery 2 and battery thermal control is made. The deployment, instead, is divided between antennas and solar panels.

The last tab, in Figure F-34, is aimed at selecting the failures for the Communication module and the Processors module. Due to the limited failure scenarios, the interface is simple. In case a Communication failure is simulated, the box “Communication window” should be ticked first, and later the failure can be selected.

6 Simulations and results

In the previous Chapters, the FDIR algorithm was designed and implemented on Simulink and a simulator was built in the same environment. In this Chapter, the results obtained through the simulations of the FDIR model will be described.

6.1 Test procedure and success criteria

The methodology used for the verification of the FDIR algorithm was explained in Chapter 5, where the design of a Simulink model that simulates the packages coming from the satellite was described. The model has been used to perform several simulations. As it was mentioned in Section 5.1, the test-cases were defined following the FMECA: one test for each failure scenarios. Moreover, one test-case for the nominal scenario has been included, to ensure the absence of false positives.

For each test-case, three simulations must be performed, in order to assess the correct application of the recovery sequence in case a failure is not recovered, is recovered with L0 or is recovered with L1. The list of the simulations is documented in Table 6-1; an explanation for the scenario parameters “solvable L0” and “solvable L1” is given in Section 5.1.3.

Table 6-1: list of the simulations performed for each test-case

Simulation nr	Parameters	Rationale
1	Solvable L0 = 0 Solvable L1 = 0	To test the application of the complete recovery sequence.
2	Solvable L0 = 1 Solvable L1 = 0	To test the application of the correct recovery sequence when a failure is solved with L0 (when applicable).
3	Solvable L0 = 0 Solvable L1 = 1	To test the application of the correct recovery sequence when a failure is solved with L1 (when applicable).

The procedure used for each test is simple and made of well-defined steps:

- The scenario-parameters associated with the nominal scenario are written in MATLAB base workspace; parameters “solvable L0” and “solvable L1” have a value of 0.
- The test-case is chosen.
- The scenario parameters related to the test case are overwritten in MATLAB base workspace.
- The simulation is run in Simulink, with a simulation time of 200 [s], which is sufficient to test the recovery of all the possible failures.
- The outputs of the simulation (FDI log and FR log of the afferent module) are examined.
- The parameter “solvable L0” is overwritten in MATLAB base workspace, with a value of 1.
- The simulation is run in Simulink, with a simulation time of 200 [s].
- The outputs of the simulation (FDI log and FR log of the afferent module) are examined.
- The parameter “solvable L0” is overwritten in MATLAB base workspace, with a value of 0, and the parameter “solvable L1” is overwritten with a value of 1.
- The simulation is run in Simulink, with a simulation time of 200 [s].
- The outputs of the simulation (FDI log and FR log of the afferent module) are examined.

In the test procedure described above, the nominal scenario is mentioned. The parameters associated with this scenario are reported in Table 6-2. As it can be seen, no manoeuvre is executed with the propulsion system, the RWs are on, all the attitude sensors have the right orientation with respect to the Sun vector, and the S/C is not rotating. Moreover, the Camera is

Table 6-2: default scenario used in the simulations

Parameter	Nominal value	Description
Main thrust command	0 [N]	The main thrust is in idle
Gas thrusters command	0 [N]	The gas thrusters are in idle
RW mode	1	Reaction wheels are active
Angular speed	0 [°/s]	S/C is not rotating
Angular speed mode	1	Constant angular speed
SS Sun FOV nominal	1	Sun in FOV of the SS
ST Sun FOV nominal	0	Sun not in FOV of the ST
Battery DOD	0 %	Batteries are not discharged
Battery mode	0	Constant DOD
Moon FOV nominal	1	Moon in camera FOV
Navigation frequency mode	1	Low frequency navigation
Science mode	0	No science
Temperature (all units)	20 [°C]	Temperature in range
Communication window	1	S/C is in communication window

performing navigation, the battery DOD in constant and the S/C is in a communication window. It should be noticed that this scenario is not intended to represent a specific operational mode, but only to be used as a default case for the simulations. Reproducing a specific mode is not possible with the simulation model developed in this study, as all the modules are independent of each other, apart from exceptional cases. In case a specific failure is influenced by one or more parameters, additional test-cases are added, e.g. overheating can be simulated when a unit is operational and when a unit is in “idle” mode.

Once the simulation is performed, the success criteria listed in Table 6-3 are used to distinguish between a successful test and a failed test.

Table 6-3: success criteria for the FDIR algorithm simulations

Criteria	Description
SIM.1	No bug occurs during the simulation.
FDIR.1	The FDI log contains the detection flag of the failure associated to the test.
FDIR.2	The FR log contains the correct recovery sequence for the failure associated to the test.

6.2 Results summary

The FMECA at the current state is made of approximatively 75 scenarios; for each of them, at least 3 simulations were performed. Due to the vast number of simulations, it is not possible to describe each of them in detail. Hence, a list of all the tests is documented in Appendix G, in Table G-1, and a summary will be presented, in this section.

For each test-case, the following information is reported in Table G-1:

- Test-case ID (based on FMECA ID)
- Test-case description
- Scenario parameters to insert
- Test result

The inclusion of the scenario parameters used for a test-case is fundamental, as it allows repeatability of the simulation of interest. It should be noticed that, as mentioned in the test procedure of Section 6.1, at the beginning of each test all the scenario parameters must be set to

the nominal case of Table 6-2; hence, only those that need to be actively changed for each test-case are reported in Table G-1. Finally, it should be remembered that each test comprises three simulations, in which the parameters “solvable L0” and “solvable L1” are changed according to the test procedure.

The result of each test-case is documented in Table G-1. As it was expected, all the results are positive; in fact, the simulation model is tailored to the FDIR logic that was developed, and no unexpected scenario could be tested. Nevertheless, some interesting outcomes were obtained and are discussed in this section. In Section 6.2.1, the results of a successful test are illustrated in detail, as an example of all the successful simulations that were performed. Later, in Section 6.2.2 the criticalities that were found are discussed.

6.2.1 TEMP.1 test-case results

In order to provide an example of a simulation, the test case TEMP.1, from Table G-1, was selected. In the FMECA, this ID corresponds to a frozen sensor failure of the temperature sensor. The reasons for this choice are manifold. Firstly, the temperature sensors are installed on many components and therefore this scenario is important for several modules. Secondly, it is a scenario which is easy to represent and intuitive to understand. Finally, the presence of redundancy allows showing how to simulate not only the recovery levels L0 and L1, but also L2.

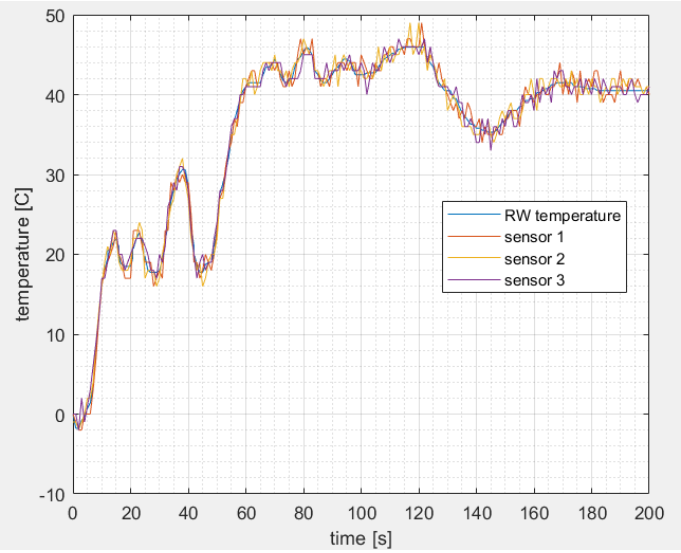

Figure 6-1: nominal scenario used for the simulations of a temperature sensor failure

The temperature sensors of the reaction wheels were selected for this test, but the same results can be obtained with the other modules, as the FDIR logic is the same. Unlike the model described in Section 5.3, where a periodic temperature trend was created for the temperature, a customized trend was used to perform the simulations, in order to highlight different features of the FDIR algorithm. On top of this, an accuracy of 1 [°C] (1-σ) is added to each sensor, in order to make the simulation more realistic. The nominal scenario is shown in Figure 6-1. The scenario represented is not realistic, as the RWs experience a fast temperature increase; however, the focus of the simulation is not the temperature, but the behaviour of the temperature sensors.

Following the test procedure described in Section 6.1, three simulations are performed. In the first, the failure is not recovered, as the parameters “solvable L0” and “solvable L1” are set to 0. The trend of the RW temperature and sensors output can be seen in the top diagram of Figure 6-2: from time 20 [s] of the simulation, the “frozen sensor” failure is injected to sensor 2 and is never recovered.

The second diagram of Figure 6-2 reports the FDI log created by the FDIR algorithm. Since the detection method is based on a cross-check between the three sensors, the failure is not detected continuously, but only when the real temperature differs from the frozen value. This aspect represents a limitation of the detection method; however, it is accepted in the current design, as it does not affect the thermal control of the unit (the temperature measurement is still close to reality, despite the failure). An additional consideration can be made regarding the value of the FDI log: at first, when a failure is detected, the “general failure” flag is created (FDI log equal to 5) and, after 10 [s] of frozen signal, the flag is changed to “frozen sensor”. Thus, the FDI log follows the logic that was designed in Section 3.3, and criterion FDIR.1 is satisfied.

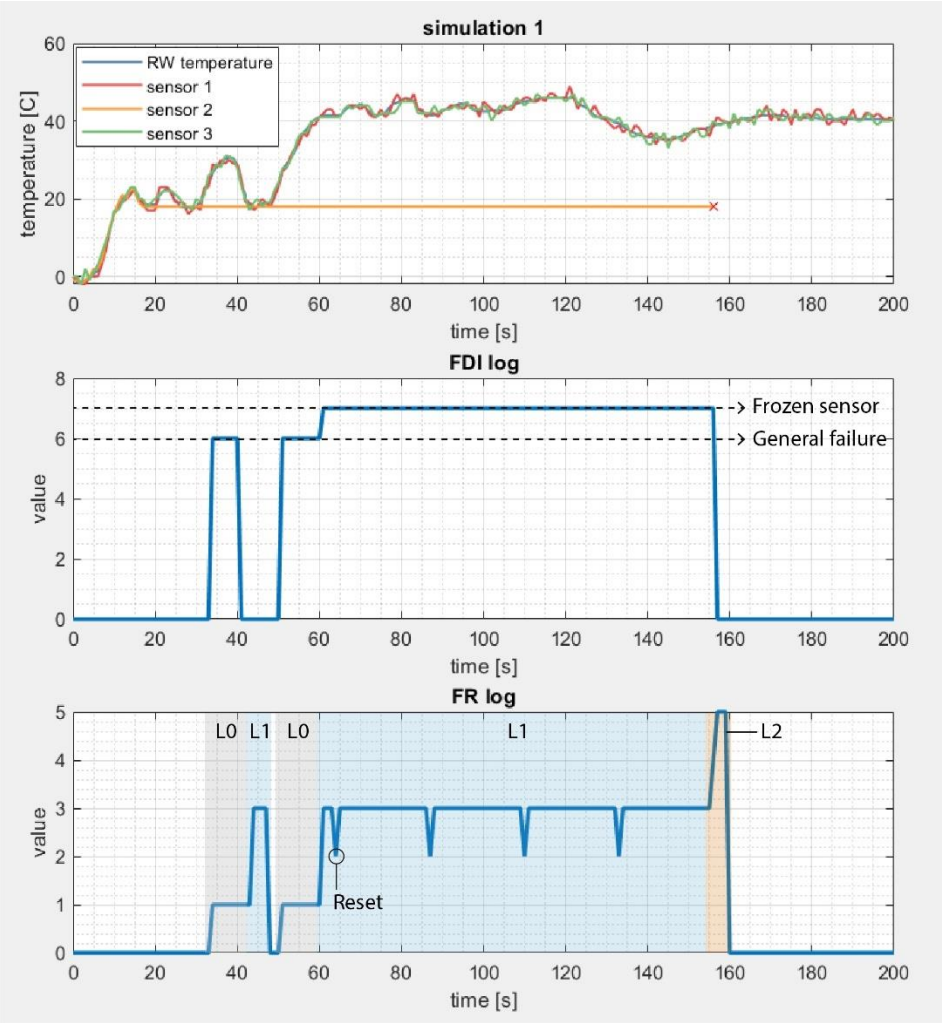

Figure 6-2: results of the first simulation. In the top diagram, the outputs of the sensors are shown. Below, the trend FDI log and FR log are shown

Finally, the evolution of the FR log is shown in the bottom diagram of Figure 6-2. The FDIR begins the recovery when the failure is detected the first time, but at around 45 [s] the recovery is stopped before reaching the reset since the detection flag is changed. However, from 50 [s] of simulation time, the full recovery sequence is executed, from L0 to L1 (4 resets) to L2. In fact, since the failure was not recovered, the unit is switched off and the thermal control is assigned to the other two sensors: from that moment, the detection FDI also indicates the absence of failure. Therefore, the FR log follows the logic that was designed in Section 4.3, and the criterion FDIR.2 is satisfied.

The following simulation was executed with the parameter “solvable L0” set to 1. Hence, the failure recovers itself after a proper amount of time (set to 15 [s] in the current simulation). The results are shown in Figure 6-3. The trend of the RW temperature and sensors output can be seen in the top diagram: from second 45 of the simulation, the “frozen sensor” failure is injected to sensor 2 and is recovered after about 15 [s].

The second diagram of Figure 6-3 shows the FDI log, which follows the logic described in Section 3.3: when the sensor is frozen and its value differs from the nominal, the failure is detected. However, it is flagged as a “general failure” since the failure is recovered before 10 [s] have passed. Hence, the success criterion FDIR.1 is satisfied.

The FR log is shown in the bottom diagram of Figure 6-3. As can be seen, only recovery level L0 is fully executed. In fact, L1 is entered but the reset is not performed since the FDI indicates the absence of failure for more than 3 [s] is a row. Hence, the correct recovery sequence is applied and success criterion FDIR.2 is satisfied.

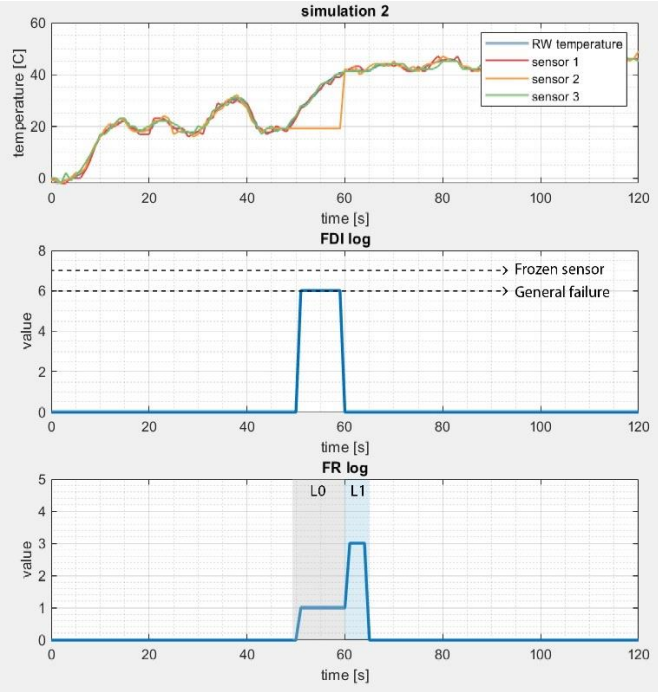

Figure 6-3: results of the second simulation. In the top diagram, the outputs of the sensors are shown. Below, the trend FDI log and FR log are shown

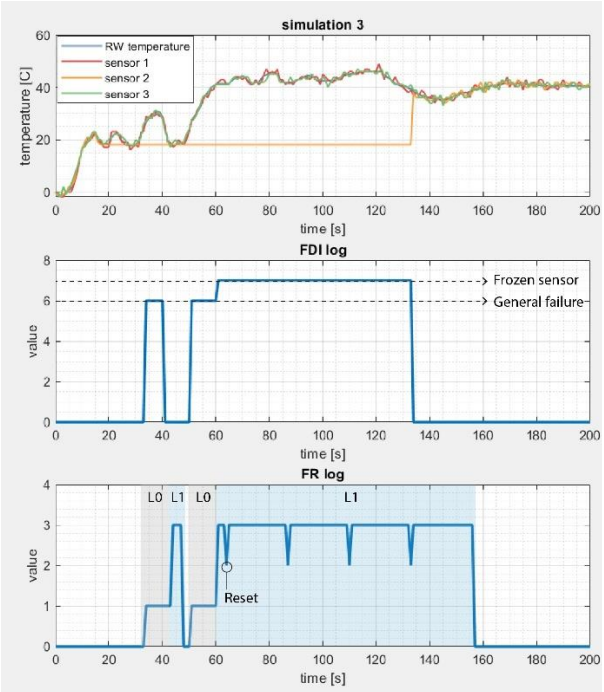

Figure 6-4: results of the third simulation. In the top diagram, the outputs of the sensors are shown. Below, the trend FDI log and FR log are shown

Finally, a third simulation is executed, with parameter “solvable L0” set to 0 and parameter “solvable L1” set to 1. The results are documented in Figure 6-4: after 20 [s] of simulation, the “frozen sensor” failure is injected to sensor 2 and is recovered after 4 resets signals from the FDIR.

The second diagram of Figure 6-4 represents the evolution of the FDI log. Since a similar situation described for the first simulation occurs, the success criterion FDIR.1 is satisfied. The FR log values, instead, are shown in the bottom diagram. As it happened in the first simulation, the recovery is begun, then stopped due to the absence of failure detection, and finally restarted from L0. In this case, after the application of 4 resets, the failure is recovered. Therefore, the FR system exits from the recovery sequence, instead of proceeding to L2; hence, the correct FR logic is applied and the success criterion FDIR.2 is satisfied.

Since all the simulations were successful, the test is considered successful as well. Thus, the FDIR algorithm can detect, isolate and recover the scenario TEMP.1 from the FMECA. The same result obtained for this scenario was obtained for the other failures, as indicated in Table G-1.

6.2.2 Delays in thrusters FDI

From Table G-1 it can be seen that, despite all the tests are categorized as successful, some criticalities were found for certain scenarios that were labelled as “partially positive”. Most of these scenarios are related to the main thruster or the gas thrusters; hence, it is worth discussing them more in detail.

The reasons why the FDIR for several failures of these units is considered critical after the simulations are the significant detection delay and the frequency of false positives, situations in which a failure is detected when no failure occurred. The causes of these phenomena are two: the use of the propellant trend as a parameter for failure detection (see Section 3.2) and the low accuracy of the propellant budget’s measurements that were chosen for the simulation (see Section 4.2). As a result, the time needed to detect a decreasing propellant level is 10 [s], during which the propellant budget is considered constant, leading to a wrong FDI outcome.

An example is documented, for clarity. In case the main thruster’s command is positive, the simulation model opens the valve and starts the propellant flow. The mass flow is 20 [g/s] and the

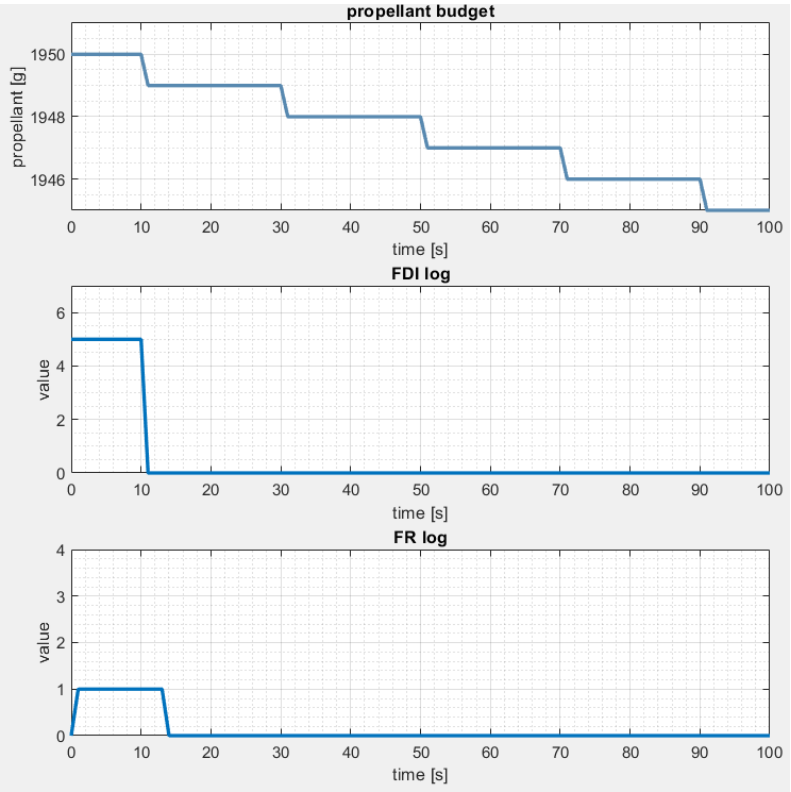

Figure 6-5: simulation of the nominal scenario in which the thruster is fired. At the top, the propellant budget trend is shown. Below, the FDI log and FR log are represented

resolution of the measurement is 1 [g]; thus, only after 10 [s] the FDI will detect a propellant decrease. Until then, a failure flag is created. The situation is represented in Figure 6-5.

The same situation of the example occurs also in all the other scenarios categorized as “partially positive” in Table G-1. The only ways to tackle this issue and prevent the wrong detection to cause a chain of failures is, therefore, to increase the waiting time during L0. Since, at this stage, the detection delay is assumed to be 10 [s], an L0 of 15-20 [s] is sufficient to confirm the FDI log and take a decision. However, the drawback is that it is not possible to implement a faster response to failures such as “Locked output”, which were assigned a high criticality in the FMECA, due to the risk of collision.

It is foreseen that the issue described for the thrusters will be solved in the next phases of the project. In fact, it will be possible to counteract this problem with the use of a more accurate measurement of the propellant consumed, if possible, and with the implementation of the dynamics, which will allow performing an alternative check. However, identifying this potential design pitfall makes it is possible to draw general conclusions that might be useful in the future FDIR design, also for other components: in case the trend of a parameter is needed for failure detection, an available measurement with the correct resolution and accuracy should be available. The correct accuracy depends on the parameter of interest, but, in principle, it is the accuracy that allows fast detection of eventual changes., e.g. within 1-5 [s]. In case such measurement is not available, a higher L0 time shall be implemented, or alternative detection approaches shall be sought.

6.3 Cross-check tests

The tests described so far were aimed at the verification of the FDIR algorithm, by injecting the input associated with a failure scenario and checking the FDIR output. In this section, a different type of tests will be described, which are aimed at the verification of the cross-check methodology that was described in 3.1.3 and used in this Thesis. In particular, the application of the methodology to the cross-check between the sensors of the ADCS was tested.

6.3.1 Test procedure and success criteria

In the simulations performed so far, the failures of the ADCS sensors (star-trackers, Sun sensor, IMU) considered in the FMECA were tested: frozen sensor and general failure. The frozen sensor leads to a growing error, while the general failure was simulated by injecting a large bias in the measurement of a unit. The tests for both these scenarios were successful and confirmed the possibility of using the cross-check between two units to detect large errors. However, it is considered important to verify whether the error function and the threshold that were selected in

this Thesis work as intended. In Section 3.1.3, where the cross-check methodology is described, three main aspects were deemed relevant: the rate of false positive detection, the rate of detection of loss of accuracy and the rate of detection of bias. Therefore, all these aspects must be properly tested.

In order to obtain relevant results, a more detailed model of the attitude sensors and the IMU must be used: in Chapter 5, the simulation model is described, and it can be noticed that to represent the inaccuracy of the units a small bias is added to their measurements. This method is considered sufficient for the FMECA tests performed so far, but a better model is needed to test the cross-check methodology. Therefore, an accuracy error is added to the measurements, in accordance with the assumptions that were made in Chapter 3. The accuracy error is modelled as a statistical variable, with mean 0 and standard deviation in line with the values used in this study. The error is created in MATLAB and retrieved by the Simulink model from the main workspace. A summary of the values used for the accuracy is reported in Table 6-4.

Table 6-4: accuracy values used for the simulations of the cross-checks of ADCS sensors

Unit	Measurement	Accuracy (1-σ)
Star-tracker	θ_1	40 [arcsec]
	θ_2	3.3 [arcsec]
	θ_3	3.3 [arcsec]
Sun sensor	α	0.16 [°]
	β	0.16 [°]
IMU	ω_1	0.05 [°/s]
	ω_2	0.05 [°/s]
	ω_3	0.05 [°/s]

The values in Table 6-4 are used to simulate the nominal scenario, which allows assessing the rate of false positives. To simulate a loss of accuracy, it is sufficient to increase the value of the afferent accuracy parameter. To simulate a bias, an additional, constant bias is added to the measurement.

In order to test the three relevant aspects of the threshold, the following procedure will be used, for each cross-check under test:

- The scenario-parameters associated to the nominal scenario are written in MATLAB base workspace; parameters “solvable L0” and “solvable L1” have a value of 0.
- The nominal accuracy error vector is created in MATLAB.
- The simulation is run in Simulink, with a simulation time of 200 [s].
- The outputs of the simulation (FDI log and FR log of the afferent module) are examined.
- The accuracy error vector associated with “loss of accuracy” is created in MATLAB.
- The simulation is run in Simulink, with a simulation time of 200 [s].
- The outputs of the simulation (FDI log and FR log of the afferent module) are examined.
- The nominal accuracy error vector is created in MATLAB.
- An additional bias is created in MATLAB for the measurement of interest.
- The simulation is run in Simulink, with a simulation time of 200 [s].
- The outputs of the simulation (FDI log and FR log of the afferent module) are examined.

As the methodology that was developed was based on statistical considerations, a proper number of simulations must be run, using the procedure above. For every scenario, e.g. loss of accuracy, 100 simulations will be performed: the number was chosen as it allows to draw relevant conclusions while keeping the time consumed restrained.

The test procedure was applied to simulate the three different cross-checks (between STs, ST and SS, ST and IMU) that were developed in this Thesis. Once a simulation is performed, the success criteria listed in Table 6-5 are used to distinguish between a successful test and a failed test. They refer to the minimum failures, which are based on the analysis done in Chapter 3.

Table 6-5: success criteria for the cross-check tests

Criteria	Description
SIM.1	No bug occurs during the simulation.
CK.1	Less than 5% of the simulations present detection of false positives.
CK.2	The minimum loss of accuracy is detected in at least 95% of the simulations.
CK.3	The minimum bias is detected in at least 95% of the simulations.

6.3.2 Star-trackers cross-check test

In Section 3.5.1.1, a threshold of was selected for the cross-check of each Euler angle measured by the STs: for each Euler angle, $trs_i = 2.5\sqrt{2}\sigma_i$.

Following the test procedure, the first step regards the detection of false positives. The nominal scenario (no failures injected) was simulated, and the FR log was checked: a failure is considered detected in case a reset is triggered, according to the definition that was given in Section 3.1.3. It should be noticed that the nominal scenario was already tested in the context of the simulations described in the previous sections; however, in this case, the accuracy error is added, and extensive simulations are performed (100 simulations, lasting 200 [s] each). The result is as expected: with the selected threshold, no false positive was detected in any simulation. Hence, criterion CK.1 is satisfied.

The following simulations were finalised at investigating the detection rate of loss of accuracy. A loss of accuracy failure was injected to one of the measurements of ST1, and extensive simulations were performed. If a reset is triggered in the FR log, the failure is considered detected. However, another parameter was analysed: if the failure was isolated or not. In fact, several times the FDIR detects the failure but triggers the reset of the wrong unit (ST2) since it is not able to isolate it. Despite this issue is not caused by the cross-check methodology between two STs, which is used only for the detection, it is still relevant to understand which level of loss of accuracy can be detected and isolated by the algorithm at this stage. The results are collected in Table 6-6. It can be seen that the minimum loss of accuracy that was theorized in Section 3.5.1.1 ($(\sigma_\theta)_{degraded} = 7\sigma_\theta$), is detected 97 % of the times; hence, the success criterion CK.2 is satisfied. Higher degradations were detected in 100% of the simulations. However, a good rate of successful isolation was never obtained, not even for a loss of accuracy factor of 50. The reason is that the isolation is performed through cross-check with the Sun sensor, which is not accurate enough for the purpose. Thus, the test allowed to assess the necessity of an alternative isolation method, for the star-trackers.

Finally, the detection of bias was tested, and the results are shown in Table 6-7. Recalling Section 3.5.1.1, it was hypothesized that the minimum bias that could be detected without false negatives is $bias_i = 2.8\sigma_\theta$. However, only for values higher than $4\sigma_i$ the detection is achieved, and detection

Table 6-6: results of the loss of accuracy tests for the cross-check between star-trackers

Loss of accuracy injected	Nr of simulations	Rate detection	Rate detection + isolation
$\sigma_f = 6\sigma_i$	100	90%	50%
$\sigma_f = 7\sigma_i$	100	97%	55%
$\sigma_f = 10\sigma_i$	100	100%	52%
$\sigma_f = 15\sigma_i$	100	100%	43%
$\sigma_f = 17\sigma_i$	100	100%	52%
$\sigma_f = 20\sigma_i$	100	100%	58%
$\sigma_f = 25\sigma_i$	100	100%	57%
$\sigma_f = 50\sigma_i$	100	100%	56%

Table 6-7: results of the bias tests for the cross-check between star-trackers

Bias injected	Nr of simulations	Rate detection	Rate detection + isolation
$bias = 2.2\sigma_i$	100	0%	0%
$bias = 3\sigma_i$	100	0%	0%
$bias = 4\sigma_i$	100	53%	25%
$bias = 5\sigma_i$	100	100%	60%
$bias = 10\sigma_i$	100	100%	58%
$bias = 15\sigma_i$	100	100%	56%
$bias = 20\sigma_i$	100	100%	59%

rates higher than 95 % are obtained only for a bias higher than $5\sigma_i$. Thus, despite success criterion CK.3 is not satisfied, the minimum bias that can be detected is still considered satisfactory. In fact, considering the specifications of the STs of LUMIO, the minimum bias is of about 0.005 [deg] for yaw and pitch axes, and 0.06 [deg] for roll axis, which is line with the required pointing accuracy of the ADCS, reported as 0.1 [deg] in [1]. As in the case of loss of accuracy, a good isolation rate is not achieved, due to the little accuracy of the Sun sensor.

In conclusion, the test allowed to assess that the cross-check methodology developed in this Thesis for the star-trackers work as intended. In particular, the rate of false positives is in line with the requirements, hence it is not necessary to increase the threshold. Regarding the detection of failures, good results are obtained. However, the hypotheses that were done in Section 3.5.1.1 are too optimistic, as the isolation of the failure was not considered; therefore, the minimum loss of accuracy and bias that can be detected and isolated without significant rates of false negatives are determined by the cross-check with the Sun sensor, rather than the check between star-trackers. In the future phases of the mission, additional information, as the orbital propagator and the S/C dynamics, shall be integrated to improve the failure isolation.

6.3.3 Star-tracker and Sun sensor cross-check test

In Section 3.5.1.2, a threshold of was selected for the cross-check of a star-tracker and a Sun sensor: $trs_1 = 3\sigma_\alpha$ and $trs_2 = 3\sigma_\beta$, which are used for the error function of α and β, respectively.

Following the test procedure, the first step regards the detection of false positives. The nominal scenario (no failures injected) was simulated, and the FR log was checked: a false positive is detected in case a reset is triggered, according to the definition that was given in Section 3.1.3. The result is as expected: with the selected threshold, no false positive was detected in any simulation. Hence, criterion CK.1 is satisfied.

The following simulations were finalised at investigating the detection rate of loss of accuracy. It should be remembered that, in Section 3.5.1.2, it was concluded that this check can be used to detect Sun sensors failures, but is not accurate to detect minimal failures of the star-tracker; hence, only the Sun sensor failures were tested. A loss of accuracy failure was injected to one of the measurements of the SS, and extensive simulations were performed. If a reset is triggered in the FR log, the failure is considered detected. The results are collected in Table 6-8. Some interesting

Table 6-8: results of the loss of accuracy tests for the cross-check between a star-tracker and the Sun sensor

Loss of accuracy injected	Nr of simulations	Rate detection
$\sigma_f = 4\sigma_i$	100	91%
$\sigma_f = 4.5\sigma_i$	100	99%
$\sigma_f = 5\sigma_i$	100	100%
$\sigma_f = 5.5\sigma_i$	100	100%
$\sigma_f = 6\sigma_i$	100	100%
$\sigma_f = 9\sigma_i$	100	100%

results arise. It can be seen that the minimum loss of accuracy that was theorized in Section 3.5.1.2 ($\sigma_f = 6\sigma_i$), is detected 100 % of the times; hence, the success criterion CK.2 is satisfied. The actual minimum degradation of accuracy that can be detected is $(\sigma_\alpha)_{degraded} = 5\sigma_\alpha$, but good results are obtained also for lower values. In this case, the isolation was not studied, as the isolation is performed through the cross-check with another star-tracker or the orbital propagator, which at this stage is considered infallible; hence, a rate of isolation of 100% is always obtained.

Finally, the detection of bias was tested, and the results are shown in Table 6-9. Recalling Section 3.5.1.2, it was hypothesized that the minimum bias that could be detected without false negatives is $bias_\alpha = 3.3\sigma_\alpha$. However, a better result was obtained, since a 100% detection rate was obtained also for a bias of $2\sigma_\alpha$.

Table 6-9: results of the bias tests for the cross-check between a star-tracker and a Sun sensor

Bias injected	Nr of simulations	Rate detection
$bias = 2\sigma_i$	100	100%
$bias = 3\sigma_i$	100	100%
$bias = 4\sigma_i$	100	100%

In conclusion, the tests on the cross-check between a star-tracker and a Sun sensor were successful, according to all the criteria in Table 6-5. The detection rate is indeed more accurate than it was foreseen, confirming that this cross-check is a fundamental tool to detect failures of the Sun sensor. However, it cannot be used for effective detection and isolation of failures of the star-tracker, due to the excessive difference in their accuracy.

6.3.4 Star-tracker and IMU cross-check

In Section 3.6.1.1, the cross-check between the star-tracker and the IMU was described. The resulting threshold, for each error function, is $trs = \frac{3\sqrt{3}}{2}\sigma_\omega$, where σ_ω is the gyroscope accuracy in [rad/s].

Following the test procedure, the first step regards the detection of false positives. The nominal scenario (no failures injected) was simulated, and the FR log was checked: a false positive is detected in case a reset is triggered according to the definition that was given in Section 3.1.3. The result is as expected: with the selected threshold, no false positive was detected in any simulation. Hence, criterion CK.1 is satisfied.

The second test is focused on the detection of a loss of accuracy. In Section 3.6.1.1, a minimum value of $(\sigma_\omega)_{degraded} = 3\sqrt{3}\sigma_\omega \sim 5\sigma_\omega$ was hypothesised. However, from the simulations it results that the minimum loss of accuracy that can be detected without false negatives with a probability higher than 95 % is $(\sigma_\omega)_{degraded} = 14\sigma_\omega = 0.7\ [°/s]$, as can be seen from Table 6-10; it can be noticed that, despite this value is not high in the context of manoeuvres as de-tumbling the S/C, it is still too large to achieve detection with the star-tracker in the real operations, as the ST can operate only until an angular speed of 1 [°/s]. Thus, the criterion CK.2 is not satisfied.

Table 6-10: results of the loss of accuracy tests for the cross-check between a star-tracker and the IMU

Loss of accuracy injected	Nr of simulations	Rate detection
$\sigma_f = 4\sigma_\omega$	100	0%
$\sigma_f = 6\sigma_\omega$	100	1%
$\sigma_f = 8\sigma_\omega$	100	11%
$\sigma_f = 10\sigma_\omega$	100	51%
$\sigma_f = 12\sigma_\omega$	100	82%
$\sigma_f = 14\sigma_\omega$	100	98%

Finally, the detection of bias in the measurement of a gyroscope is tested. In Section 3.6.1.1, a value of $bias = (\frac{3\sqrt{3}}{2} + 0.3)\sigma_\omega \sim 3\sigma_\omega$ was supposed as the minimum bias that can be detected; however, from the simulations, summarized in Table 6-11, it can be concluded that the minimum value of the bias with the current threshold selection is $bias = 6\sigma_\omega = 0.3\ [°/s]$. Despite this value is more satisfactory than the one obtained for the loss of accuracy, the cross-check is still considered not accurate enough. Thus, criterion CK.3 is not satisfied.

Table 6-11: results of the bias tests for the cross-check between a star-tracker and the IMU gyroscopes

Bias injected	Nr of simulations	Rate detection
$bias = 2\sigma_\omega$	100	0%
$bias = 4\sigma_\omega$	100	0%
$bias = 6\sigma_\omega$	100	100%
$bias = 8\sigma_\omega$	100	100%

In conclusion, the tests of the cross-check between the IMU and the star-tracker were not successful. Despite the check shows good performance in avoiding false positives and reaches acceptable values for the detection of biases, its feasibility is limited by the fact that the two units can be cross-checked only for small angular rates. Hence, it is recommended to pursue alternative ways to check the IMU, e.g. checks based on the dynamics of the S/C. The addition of a redundant IMU is suggested, to perform a more accurate cross-check and to allow for the implementation of an L2 recovery level.

7 Conclusions

This Chapter is intended to draw conclusions for the work performed in this Thesis and provide recommendations for the follow-up projects.

7.1 Conclusions

The aim of the Thesis was summarised in one main objective:

Designing and developing a highly logical Fault Detection Isolation and Recovery architecture for LUMIO mission with simplistic and coherent MATLAB/Simulink implementation.

This objective was translated into a series of research questions that established the methodology used for the work. Some conclusive answers can be drawn for each research question:

1. *"What inputs for the Fault Detection Isolation and Recovery development can be obtained through the Failure Modes Effects Analysis of LUMIO mission?"*
   Amongst the list of potential scenarios that were identified during the Thesis, those that could be treated at the current phase of the project were selected and analysed following the methodology of the Failure Modes Effects (and Criticality) Analysis. The preliminary criteria for the allocation of the severity and probability classification were also proposed, from the guidelines given in the ECSS standards and the statistical studies of the past satellite failures. The allocation of the probability level represents a relevant result, as it allows to have a baseline also for the future phases, in case more accurate information on the failure rates will not be available. As a result, a version of the FMECA worksheet compatible with the current phase of the mission (Phase A) was produced.
   The benefits of the FMECA table in the context of LUMIO are manifold. Firstly, it allows summarizing the failure scenarios to be handled by the FDIR, thus simplifying the prosecution of the work in the future phase. Secondly, it represents significant progress for the project, whose main pitfall highlighted by ESA was the earliness of the dependability and safety analysis. In fact, in the ECSS standards, it is recommended to perform the FMECA from Phase A and to update it in the following stages, in order to support the design trade-offs, to keep track of the impact of design changes on system safety, and to update the critical items list, the fault tree analysis and the FDIR consequently [11]. Therefore, the FMECA led to the identification of the main criticalities in the current design of LUMIO and the proposal of potential compensating provisions: it was established that the main thruster and the camera are the most critical units of the satellite, as they represent single point failures with a potential risk of collision.

2. *"How can the Fault Detection and Isolation activities be integrated within the satellite operations?"*
   The integration of the FDIR functionalities in the satellite operations was achieved, with a level of detail consistent with the earliness of the design. The main obstacle was the absence of a clear definition of the operational sequences of LUMIO, in the current phase of the project; hence, a preliminary version was produced, based on the mission design. The main novelty introduced in this work is the addition of a Safe Mode within the satellite operations, which is triggered when a failure is detected or the voltage level drops below a limit. This mode was not included in the design of LUMIO but is fundamental to permit a successful integration of the FDIR algorithm in the mission: whenever a failure is detected, Safe Mode is triggered to avoid the failure to affect the satellite operations and propagate; later, the recovery actions can be executed.
   The other relevant result obtained through the definition of the satellite operational modes is the identification of new potential failures, which differ significantly from those found from the

analysis of each component of the S/C: they are not detected directly by the FDIR but are triggered by the OBC as part of the satellite operations. Some of these scenarios, those related to deployment failures, were included in the current FDIR design, while the others were left for future stages, despite the moment in which the FDIR should take over was identified and a preliminary version of recovery sequence was proposed.

3. *"What is the Fault Detection Isolation and Recovery architecture of LUMIO mission?"*
   The design of an FDIR architecture for LUMIO mission was successfully accomplished. The high-level architecture of the algorithm is based on the division between the FDI, which produces a log containing information on failure detection and isolation, and the FR that elaborates the recovery action.
   The FDI was divided into different modules, in order to ease the design and testing of the algorithm. One module for each component was included; the components of the ADCS were grouped together, like those of the Power system. Although it is expected that the architecture will undergo significant changes in the future, it represents a valid baseline for its simplicity and logic. The detailed block diagram for every group was produced in Chapter 3. The logic for failure detection and isolation is simple and based on different types of checks; the most used is the cross-check, for which a methodology was outlined. The study performed allowed to identify some simple guidelines for the design of cross-checks between units, which were successfully applied in this Thesis, but represent also a useful baseline for the future FDI design.
   The high-level architecture of the FR follows that of the FDI, with the division into modules. A detailed diagram for failure recovery was produced in Chapter 4 and the implementation of this logic was accomplished for each module. The recovery sequence is based on the common FDIR methods, but the detailed decision logic for the less critical recovery levels (L0, L1, L2) is a result of the work of this Thesis, which paves the way for the future expansion of the higher levels of the FR (L3, L4), in the next phases.
   Finally, in this Thesis a successful implementation of the FDIR logic in Simulink was achieved; in particular, the suitability of the Stateflow environment was assessed, as it allows to incorporate the decision logic described so far in transition diagrams, with the aid of custom MATLAB functions and graphic functions. The simulations performed using the FDIR Simulink model verified the architecture developed in this Thesis, confirming the ability to detect, isolate and recover the failure scenarios included in the FMECA at this stage.

In conclusion, answering to the three main research question led to the accomplishment of the main objective of the Thesis. The principal achievements of this work are the definition of a methodology for the design of the FDIR system, which can be applied also in more advanced phases of the project, and the development of a preliminary version of the algorithm, in line with the current stage of LUMIO mission. The main limitation of this Thesis is the earliness of the algorithm produced: only a limited number of failure scenarios could be studied, and only simple detection and recovery methods could be developed, due to the uncertainties in the S/C design; nonetheless, it represents a baseline for the future detailed development. The completion of this work has great relevance for LUMIO mission, since the on-board FDIR algorithm will increase the autonomy of the S/C and the overall reliability and availability of the mission; on top of this, the main criticalities in the current S/C design were identified, providing a concrete start point for the future design phases.

7.2 Recommendations and future works

Performing the FMECA of LUMIO mission allowed to point out some criticalities in the current design of the S/C and the mission, which leads to the following recommendations for the follow-up design:

- The mitigation of the risks associated with the most critical failures in Table 2-4 is recommended. The most impactful compensating provisions that can be undertaken are

the addition of a redundant thruster for orbital manoeuvres or the change of its whole design, and the implementation of a redundant navigation method, e.g. using ground-based radiometric tracking through ESA's network [1; 18] or laser tracking [19]. On top of this, the addition of redundant reaction wheel and IMU is also considered a priority. Finally, the redundancies already included in the design should be exploited. For instance, the possibility of operating with only two on-board processors should be guaranteed.

- It is highly recommended to include in the orbital design the investigation about potential safe orbits for LUMIO, in order to implement effectively a Safe Configuration for the satellite. Moreover, the design of a collision avoidance manoeuvre is also recommended.

The FDIR is a core system engineering activity, which starts at the beginning of the design phase and ends at decommissioning. Therefore, the design of the FDIR for LUMIO mission will proceed in the next phases of the projects. It is recommended that the future development of the algorithm follows the same methodology of this Thesis, thus starts with the update of the FMECA. The following can be recommended regarding the follow-up FMECA:

- The current FMECA worksheet should be revised based on the design changes of the S/C: outdated failures should be removed, e.g. in case a unit is removed from the design, and scenarios related to additional components should be included.

- The failures scenarios that were discarded in this preliminary phase (see Table A-1 in Appendix A) should be taken into consideration, if considered still applicable. In particular, scenarios related to navigation/dynamics shall be included.

- Following the guidelines in [11], several FMECA should be performed, from sub-system level to system level. In this way, a more relevant assignation of the severity of a failure could be done: a failure with a critical severity at subsystem level might have a milder severity level when propagated to system level.

- A more detailed investigation of the probability of the failure scenarios should be carried out. In case a COTS unit is used, data about the failure rates should be asked to the supplier of the component. In case of custom design, other methodologies can be used, as the definition of proper testing. In case failure rates are not available, a reasonable probability number should be assigned, based on engineering judgment. The preliminary statistical analysis used in this Thesis can be used as a baseline, when necessary.

Regarding the definition of the operational modes, the following can be recommended:

- The operational modes should be updated based on the changes in the mission design. Interactions with the teams in charge of LUMIO design are necessary to confirm the logic of the modes and to clarify the most dubious parts, e.g. the transitions during manoeuvres.

- It is recommended to search for new possible failure scenarios associated with the satellite operations and to define appropriate recovery sequences, to be implemented in the FDIR.

The following can be recommended for the future development of the FDIR algorithm:

- In the current phase, the S/C dynamics was not considered in the development of the FDI. Therefore, it is highly recommended to take this aspect into account, when the design will be more advanced. The implementation of the dynamics for the following units is recommended:

    - Main thruster: the knowledge of the S/C position and velocity can be used to check eventual failures, in particular, the most critical as "locked output".

  - Reaction wheels: it will be possible to detect failures of the wheels using the approximation of the actuator's dynamics through transfer functions [33].
  - Gas thrusters: the knowledge of the S/C dynamics can be used to check eventual failures during de-tumbling manoeuvres,
  - Attitude sensors: the orbital propagator can be effectively integrated to cross-check the output of the attitude sensors and the IMU.
  - Camera and OBPDP: all the failures related to navigation can be included in the FDIR algorithm.

- A more detailed research on the data packages that are received by the FDIR is recommended, in terms of which data will be received and what communication protocol is used. In this way, the FDI logic can be updated: in case inapplicable checks were included in the current design, e.g. a check that uses information which is not available in the real system, they should be removed, and eventual additional checks should be included.

- Regarding the FDI for the thrusters (main thruster and gas thrusters), the availability of measurement of the propellant consumption should be investigated. In case it is not, the design of an alternative logic is recommended, e.g. using the S/C dynamics.

- Despite in this Thesis the second Sun sensor of LUMIO is considered as a cold redundancy, it is recommended to use it continuously (hot redundancy), in order to have an additional cross-check option for the attitude sensors.

- The cross-check methodology that was developed for the star-trackers should be updated, based on the mission requirements: in case the requirement for the accuracy of attitude determination is less stringent, a higher threshold can be used for the cross-check. Similar actions are recommended for the other cross-checks proposed in this Thesis.

- Regarding the FR design, more detailed research about the recovery parameters for each failure is recommended. At this phase, only a preliminary proposal of L0 timer, L1 timer and number of resets was done, but a proper analysis is necessary to avoid excessive time spent in recovery, or excessive waiting time before recovering a critical failure. On top of this, it is recommended to adapt the recovery parameters to each mission phase, following the preliminary proposal done in the FMECA.

- The implementation of recovery actions that were not detailed in this Thesis is recommended. Firstly, the application of L2 when it implies switching to a functional redundancy should be developed, e.g. in case of gas thruster's failures during de-tumbling, the reaction wheels can be used to perform the manoeuvre. Secondly, it is fundamental to proceed with the implementation of recovery levels L3 and L4.

- In the future development of the FDIR, it is recommended to review and update the simulation models, based on the more detailed information about the S/C design. More accurate simulations can be performed using advanced tools (e.g. STK) that allow to reproduce the on-orbit environment with more fidelity in terms of dynamics, power production, temperature, etc. Later, one fundamental step will be to run the algorithm on the actual on-board processor, to prove its functionalities despite the hardware constraints (memory load, processing power, etc). Finally, in the more advanced phases of the project, hardware-in-the-loop tests will be crucial to validate the system before flight.

Bibliography

1. **F. Topputo, M. Massari, J. Biggs, D.A. Dei Tos, S. Ceccherini, K. Mani, V. Franzese, A. Cervone, P. Sundaramoorthy, S. Mestry, S. Speretta, A. do Carmo Cipriano, A. Ivanov, D. Labate, A. Jochemsen, Q. Leroy, R. Furfaro, and K. Jacquinot.** *"Lunar Meteoroid Impacts Observer, A CubeSat at EarthMoon L2, Challenge Analysis".* November 2017.

2. **ESTEC Concurrent Design Facility Team.** *"LUMIO Study".* Noordwijk, The Netherlands : European Space Agency, February 2018.

3. *"LUMIO: achieving autonomous operations for Lunar exploration with a CubeSat".* **S. Speretta, F. Topputo, J. Biggs, P. Di Lizia, M. Massari, K. Mani, D. Dei Tos, S. Ceccherini, V. Franzese, A. Cervone, P. Sundaramoorthy, R. Noomen, S. Mestry, A. Cipriano, A. Ivanov, D. Labate, L. Tommasi, A. Jochemsen, J. Gailis, R. Furfaro, V. Reddy.** s.l. : American Institute of Aeronautics and Astronautics, May 2018. SpaceOps Conferences.

4. **National Aeronautics and Space Administration.** *"Fault Management Handbook - Draft 2".* Washington, DC : NASA, April 2012. NASA-HDBK-1002.

5. *"Generic AOCS/GNC Techniques & Design Framework for FDIR".* **Airbus Defence & Space, University Stuttgart (IFR), Astos Solutions GmbH.** Noordwijk, The Netherlands : ESA-ESTEC, April 2016.

6. *"FDIR Process at Airbus DS, FDIR Workshop".* **Lautenschläger, G.** Noordwijk, The Netherlands : ESA-ESTEC, April 2016.

7. *"SENER experience in AOCS FDIR".* **Rodríguez, G.** Noordwijk, The Netherlands : ESA-ESTEC, April 2016.

8. *"FDIR Development and Lessons Learned on Various Missions".* **Radu, S.** Delft, The Netherlands : TU Delft Lecture, 2018.

9. *"FDI(R) for satellites: How to deal with high availability and robustness in the space domain?".* **Olive, X.** s.l. : International Journal of Applied Mathematics and Computer Science, 2012, Vol. vol. 22 (1).

10. *Innovative Fault Detection, Isolation and Recovery Strategies On-board Spacecraft: State of the Art and Research Challenges.* **A. Wander, R. Förstner.** Neubiberg, Germany : Bundeswehr University Munich, 2013.

11. **European Cooperation for Space Standardization.** *"Space Product Assurance - Failure modes effects (and criticality) analysis (FMEA/FMECA)".* Noordwijk, The Netherlands : ECSS Secretariat, March 2009. ECSS-Q-ST-30-02C.

12. **Vesely, M. Stamatelatos and W.** *Fault Tree Handbook with Aerospace Applications.* Washington DC, USA : NASA, August 2002.

13. **Ducard, G.** *"Fault-tolerant Flight Control and Guidance Systems".* London, UK : Springer, 2009.

14. **J. Marzat, H. Piet-Lahanier, F. Damongeot, E. Walter.** Model-based fault diagnosis for aerospace systems: a survey. *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering.* January 2012, Vols. 226, Issue 10.

15. **Eichkoff, J.** *On board Computers, Onboard Software and Satellite Operations.* Berlin (DE) : Springer, 2012. ISBN 978-3-642-25170-2.

16. *"Collision Avoidance for Satellites in Formation Flight".* **G. L. Slater, S. M. Byram, and T. W. Williams.** Cincinnati, Ohio : Journal of Guidance, Control and Dynamics, September-October 2006, Vol. 29(5).

17. *A study of on-orbit spacecraft failures.* **S. Tafazoli., Canadian Space Agency.** s.l. : Acta Astronautica, 2009, Vols. 64(2-3), pp. 195-205.

18. *Improving tracking systems for deep-space navigation.* **L. Iess, F. Budnik, C. Colamarino, A. Corbelli, M. Di Benedetto, V. Fabbri, A. Graziani, R. Hunt, N. James, M. Lanucara, R. Maddè, M. Marabucci, G. Mariotti,M. Mercolino, P. Racioppa, L. Simone, P. Tortora, M. Westcott, M. Zannoni.** ESA-ESOC : 6th ESA International Workshop on Tracking, Telemetry and Command Systems for Space Applications, 10-13 September 2013.

19. **C. Bettanini, E.C. Lorenzini, G. Colombatti, A. Aboudan, M. Massironi.** CUTIE: A cubesats tether-inserted mission for moon exploration. *Acta Astronautica.* November 2018, Vol. 152, PAGES 580-587.

20. **Gertler, J.** Analytical Redundancy Methods in Fault Detection and Isolation - Survey and Synthesis. *IFAC Fault Detection, Supervision and Safety for Tech. Process.* January 1992, Vol. 6.

21. **Isermann, R.** *Fault-diagnosis system - An introduction from Fault Detection to Fault Tolerance.* Darmstadt : Springer, 2006. ISBN 978-3-540-24112-6.

22. *Threshold Selection for Timely Fault Detection in Feedback Control Systems.* **T. Hsiao, M. Tomizuka.** Portland, Oregon (USA) : American Control Conference, June 2005.

23. **B. Ochoa, S. Belongie.** *Covariance Propagation for Guided Matching.* s.l. : 3rd Workshop on Statistical Methods in Multi-Image and Video Processing, 2006.

24. **Basseville, M.** On-Board Component Fault Detection and Isolation Using the Statistical Local Approach. *Automatica.* 1998, Vol. 34(11), pp. 1391-1416.

25. **A. N. Najam, R. Muhammad.** Simulator Development of an Attitude Determination and Control Subsystem of an Earth Observation Satellite. *3rd International Conference on Recent Advances on Space Technologies.* 2007, pp. 448-452.

26. **Ziemer, Rodger E. and Tranter, William H.** *Principles of communication : systems, modulation, and noise (seventh edition).* Hoboken, New Jersey : John Wiley & Sons, Inc, 2015. ISBN 9781118078914.

27. **Peterson, W. Wκ. and Brown, D. T.** *Cyclic codes for error detection.* s.l. : Proceedings of the IRE, 1961.

28. **D. Krejci, P. Lozano.** Space Propulsion Technology for Small Spacecraft. *Proceedings of the IEEE.* 2018, Vol. 106(3), 362-378.

29. *Review of Propellant Gauging Methods.* **Yendler, Dr. B.** Reno, Nevada (USA) : 44th AIAA Aerospace Sciences Meeting and Exhibit, 9 - 12 January 2006.

30. *Comparative Assesment of Gauging Systems and Description of a Liquid Level Gauging.* **B. Hufenbach, R. Brandt, G. André, R. Voeten,P. van Put, B. Sanders.** Noordwijk, the Netherlands : Proceedings of the Second European Spacecraft Propulsion Conference, May 1997.

31. **Bradford.** Ultrasonic Flow Meter. *Bradforfd products - Propulsion system.* [Online] [Cited: 5 September 2019.] http://bradford-space.com/products-propulsion-ultrasonic-flow-meter.php.

32. **R. J. Patton, F. J. Uppal, S. Simani, B. Polle.** Robust FDI applied to thruster faults of a satellite system. *Control Engineering Practice.* September 2010, Vols. 18, Issue 9, pp. 1093, 1109.

33. **S. Radu, C. Cherciu, D. Costea, A. M. Stoica, C. G. Dragasanu, M. F. Trusculescu, M. Balan.** Individual fault detection and isolation system for Cubesats using neural networks and multimodel algorithms. *66th International Astronautical Congress.* International Astronautical Federation, 2015.

34. **ICS Engineering Inc.** "Types of Redundancy". *ICS Engineering Inc.* [Online] 2017. [Cited: 28 September 2018.] http://www.icsenggroup.com/types-of-redundancy.shtml.

35. **Basseville, M.** Detecting changes in signals and systems - A survey. *Automatica.* 1998, Vol. 24(3), p. 309-326.

36. **Dr. Chu, Q. P.** Spacecraft Attitude Dynamics and Control, Lecture sides. Delft, The Netherlands : Faculty of Aerospace Engineering, TU Delft, 2018.

37. **Hyperion Technologies.** Products - ST400. *Hyperion Technologies.* [Online] 2018. [Cited: 16 August 2019.] https://hyperiontechnologies.nl/products/st400-star-tracker/.

38. **MathWorks.** cart2sph. *MathWorks.com.* [Online] [Cited: 25 July 2019.] https://nl.mathworks.com/help/matlab/ref/cart2sph.html.

39. **Ku, H. H.** Notes on the use of propagation for error formulas. *Journal of Research of the National Bureau of Standards.* October 1966, 70C (4).

40. **Solar MEMS Technologies.** Space products - nanoSSOC-D60. *Solar MEMS Technologies.* [Online] 2017. [Cited: 16 August 2019.] http://www.solar-mems.com/space-equipment/.

41. **Atkinson, Kendall E.** *An Introduction to Numerical Analysis (2nd ed.).* New York : John Wiley & Sons, 1989. ISBN 978-0-471-50023-0.

42. **Sensonor.** Product - STIM300. *Sensonor.* [Online] 2019. [Cited: 16 August 2019.] https://www.sensonor.com/products/inertial-measurement-units/stim300/.

43. **N. Venkateswaran, M. Siva, P. Goel.** Analytical redundancy based fault detection of gyroscopes in spacecraft applications. *Acta Astronautica.* May 2002, Vol. 50(9), pp. 535-545.

44. *Preliminary Design Analysis Methodology for Electric Multirotor.* **M. Gatti, F. Giulietti.** Bologna, Italy : IFAC Proceedings Volumes (IFAC-PapersOnline). 2. 58-63., 2013. 10.3182/20131120-3-FR-4045.00038..

45. **GomSpace.** "GomSpace Battery Datasheet". *GomSpace.* [Online] 6 April 2018. [Cited: 16 October 2018.] https://gomspace.com/Shop/subsystems/batteries/nanopower-bpx.aspx.

46. *"Deployable Techniques for Small Satellite".* **Miyazaki, Y.** Tokio, Japan : EIEE, 2018.

47. *Smart architecture for highly available, robust and autonomous satellite.* **X. Olive, S. Clerc, D. Losa.** Milano (Italy) : The International Federation of Automatic Control, Proceedings of the 18th World Congres, August 28 - September 2, 2011.

48. *Hierarchical FDIR Concepts in S/C Systems .* **R. Gessner, B. Kosters, A. Hefler, R. Eilenberger, J. Hartmann, M. Schmidt.** Montreal (Canada) : Proceedings of the 8th International Conference on Space Operations (SpaceOps), May 2004.

49. **Defense Acquisition University Press.** *Systems Engineering Fundamentals.* Virginia, USA : s.n., January 2001.

50. *Testing Satellite On-Board Software - A Model Based Approach.* **M. Zeuner, H. Gogl, H.-J. Herpel, G. Willich.** Wurzburg, Germany : 19th IFAC Symposium on Automatic Control in Aerospace, September 2013.

51. *Simulink-Based FDI Simulator for Autonomous Low Earth Orbit Satellite.* **Jun Kyu Lim, Won Hee Lee and Chan Gook Park.** Milano (Italy) : The International Federation of Automatic Control - Proceedings of the 18th World Congress, August 28 - September 2, 2011.

52. **VACCO.** Hybrid ADN Delta-V / RCS System. *CubeSat Propulsion Systems from VACCO Industries.* [Online] 2012. [Cited: 13 August 2019.] https://www.cubesat-propulsion.com/hybrid-adn-delta-v-rcs-system/.

A List of discarded scenarios

This Appendix is dedicated to the possible failure scenarios of LUMIO mission that were identified during the literature review but were discarded from the FMECA. In Section 1.3.3 the principles that were used to identified possible scenarios are described, while in Section 2.1.1 the criteria used for the selection of the failures to consider in this Thesis are explained. However, documenting the discarded scenarios is crucial for the future prosecution of the FDIR design, since in the further phases of the project it might be possible to implement detection and recovery methods for them.

The scenarios are collected in Table A-1. For each scenario, a brief description is given, along with the rationale behind its exclusion.

Table A-1: list of failure scenarios that were not considered at this stage of the FMECA

Component	Scenario	Description	Rationale
LUMIO-Camera	Insufficient frequency	Meteor shower on the Moon, characterized by a higher impact frequency than expected; therefore, some impacts are not detected. This is a relevant issue for scientific purposes.	Detection method unclear at the moment, since it can be based only on the number of flashes detected.
LUMIO-Camera	No detection	No impacts detected: the assumed impact intensity differs from reality. This is a serious issue for the scientific output.	Detection method unclear at the moment, since it can be based only on the number of flashes detected.
Payload Processor	Recoverable image processing bug	Recoverable bug in the image processing for optical navigation, which leads to a wrong estimation of position. This kind of failure might affect the whole mission, because the autonomous operation is a core system requirement.	There are not sufficient data to detect the failure.
Payload Processor	Unrecoverable image processing bug	Unrecoverable bug in the image processing for optical navigation, which leads to a wrong estimation of position. This kind of failure might affect the whole mission, because the autonomous operation is a core system requirement.	There are not sufficient data to detect the failure.
Payload Processor	Recoverable loss of data	Recoverable science algorithm bug, which can lead to the loss of scientific data (a light flash is not recognised as impact). This could compromise the science output of the mission, because less scientific data than expected are produced.	Detection method unclear at the moment, since it is based only on the number of flashes detected.
Payload Processor	Unrecoverable loss of data	Unrecoverable science algorithm bug, which can lead to the loss of scientific data (a light flash is not recognised as impact). This could compromise the science output of the mission, because less scientific data than expected are produced.	Detection method unclear at the moment, since it is based only on the number of flashes detected.
Reaction Wheel	Mechanical failure	Reaction wheels mechanical failure, due to friction, vibration. Reaction wheels are the main attitude actuator in LUMIO CubeSat, so this is an important issue.	This failure can be addressed only when the dynamics of the satellite is known.
Reaction Wheel	Excessive rate of change	Maximum rate of change exceeded: the actuators start accelerating more than allowed by its technical specification, therefore there is the risk to cause a mechanical damage to the unit.	This failure can be addressed only when the dynamics of the satellite is known.
Reaction Wheel	Loss of effectiveness	Loss of effectiveness: the torque created by the component differs from the desired torque, with a lower accuracy than expected.	This failure can be addressed only when the dynamics of the satellite is known.
Reaction Wheel	Jitter	Jitter higher than expected. This can affect the pointing precision of the camera and cause its performance to degrade.	The detection of S/C jittering requires the knowledge of the S/C

Component	Scenario	Description	Rationale
			dynamics or complex image processing algorithms.
Cold Gas Thruster	Reduced thrust efficiency	Reduction of thrust efficiency: the thrust created by the component differs from the desired value, with a lower accuracy than the nominal.	With the sensors that are assumed to be installed on the S/C (propellant budget sensor and valve status sensor) there is not enough data to detect this failure.
Cold Gas Thruster	Wrong thrusting direction	Failure in the control of the thrusting direction, which can cause orbital deviation and cause risk of collisions.	In order to detect this failure, the satellite dynamic is necessary, since the knowledge of the position and velocity error is necessary.
Sun Sensor	Sensor bias	Sensor bias: constant error between the real value of the sun angle and the measured value.	With two sun sensors, it is possible to detect the failure, but not to isolate it, i.e. distinguish it from other Sun sensor failures.
Sun Sensor	Loss of accuracy	Loss of accuracy: the measurement accuracy is lower than expected, due to an internal malfunctioning.	With two sun sensors, it is possible to detect the failure, but not to isolate it, i.e. distinguish it from other Sun sensor failures.
Sun Sensor	Sensor drift	Sensor drift: the output of the sensor differs from the real value with a growing error.	With two sun sensors, it is possible to detect the failure, but not to isolate it, i.e. distinguish it from other Sun sensor failures.
Star-Tracker	Sensor bias	Sensor bias: constant error between the real value of the star angle and the measured value.	With two star-trackers, it is possible to detect the failure, but not to isolate it, i.e. distinguish it from other star-trackers failures.
Star-Tracker	Loss of accuracy	Loss of accuracy: the measurement accuracy is lower than expected, due to a wrong calibration or an internal malfunctioning.	With two star-trackers, it is possible to detect the failure, but not to isolate it, i.e. distinguish it from other star-trackers failures..
Star-Tracker	Sensor drift	Sensor drift: the output of the sensor differs from the real value with a growing error.	With two star-trackers, it is possible to detect the failure, but not to isolate it, i.e. distinguish it from other star-trackers failures.
IMU	Gyroscope bias	Gyroscope measurements bias: constant error between the real value of the inertial rotational speed and the measured value.	This failure cannot be detected without knowing the satellite's dynamic or having a redundant IMU.
IMU	Loss of gyroscope accuracy	Loss of gyroscope accuracy: the gyroscope measurement accuracy is lower than expected, due to an internal malfunctioning of one of the components.	This failure cannot be detected without knowing the satellite's dynamic or having a redundant IMU.
IMU	Accelerometer bias	Accelerometer bias: constant error between the real value of the inertial rotational acceleration and the measured value.	This failure cannot be detected without knowing the satellite's dynamic or having a redundant IMU.
IMU	Loss of accelerometer accuracy	Loss of accelerometer accuracy: the accelerometer measurement accuracy is lower than expected, due to an internal malfunctioning of one of the components.	This failure cannot be detected without knowing the satellite's dynamic or having a redundant IMU.
AOCS processor	Recoverable navigation software failure	Recoverable navigation software bug, which leads to a wrong estimation of the attitude, position and/or velocity. This is an important issue, which might compromise the autonomous navigation.	This failure cannot be detected without a redundant navigation method and/or the data from the orbital propagator.
AOCS processor	Unrecoverable navigation software failure	Unrecoverable navigation software bug, which leads to a wrong estimation of the attitude, position and/or velocity. This is an important issue, which might compromise the autonomous navigation.	This failure cannot be detected without a redundant navigation method and/or the data from the orbital propagator.
Main Thruster	Reduced thrust efficiency	The thrust created by the thruster differs from the desired value, with a lower accuracy than the nominal.	With the sensors that are assumed to be installed on the S/C (propellant budget sensor and valve status sensor) there is not enough data to detect this failure.

Component	Scenario	Description	Rationale
Main Thruster	Wrong thrusting direction	Failure in the control of the thrusting direction, which can cause orbital deviation and cause risk of collisions.	In order to detect this failure, the satellite dynamic is necessary, since the knowledge of the position and velocity error is necessary.
RF power amplifier	Loss of gain	The component is not able to amplify the signal with the required gain, due e.g. to an internal failure.	The RF amplifier was not designed yet; therefore, there is not sufficient data to handle this scenario.
UHF antenna	No uplink – GEO	It is not possible to send data to the GEO satellites, due e.g. to communication disturbances. If this happens, the data package cannot be dropped, and the ground centre would not know the S/C state.	This scenario is based on the implementation of GEO communication, not present at the current stage.
UHF antenna	No downlink - GEO	Data from the GEO satellites are not received, due to e.g. communication disturbances. If this happens, commands from ground could not be received.	This scenario is based on the implementation of GEO communication, not present at the current stage.
Temperature sensor	Loss of accuracy	Loss of accuracy: the measurement accuracy is lower than expected, due to a wrong calibration or an internal malfunctioning.	The failure can be detected and isolated with 3 temperature sensors, but the effort to distinguish between different failures of the temperature sensors is not considered reasonable at this stage; hence, it is grouped under “general failure”.
Temperature sensor	Sensor drift	Sensor drift: the output of the sensor differs from the real value with a growing error.	The failure can be detected and isolated with 3 temperature sensors, but the effort to distinguish between different failures of the temperature sensors is not considered reasonable at this stage; hence, it is grouped under “general failure”.
S/C ejection system	Late deployment	The deployment is done too late after the release from the Mothership. This failure is important to be considered, because the S/C will be in a different orbit than the one designed or can be in danger of collision with a NEO.	To detect this failure, it is necessary to have the navigation output (knowledge of the position error).

B FMECA

This Appendix is dedicated to the FMECA worksheet, in Table B-1.

Table B-1: Phase-A FMECA worksheet for LUMIO mission

ID number	Item	Function	Failure Mode	Failure Cause	Failure effects		Severity Classification	Failure detection / symptoms	Compensating provisions / recommendations	SN	PN	CN
					Local Effects	End Effects						
CAM.1	LUMIO Camera (payload)	Science and Navigation	Camera image processing disturbances	Camera malfunctioning, e.g. electronics or software failure	No navigation and/or science data available	-Divergence of navigation - Malfunction of GNC (w/o risk of collision) - not possible to produce science product.	4	No signals from Camera; background radiation, insufficient light, brightness contrast, saturation inadequacy.	L0: wait 20 seconds. L1: the camera is reset 4 times. -Request health-check to component. -Send packet with FDIR log. L2: N/A	1	3	3
						-Divergence of navigation - Malfunction of GNC (with risk of collision) - not possible to produce science product.	1	No signals from Camera; background radiation, insufficient light, brightness contrast, saturation inadequacy.	Check risk of collision based on step-by-step scenario and orbital propagator. If the current risk is high, do range checks and cross-check distances with ISL. L3: if the risk of collision is low, wait for ground instructions. If risk will increase in the next X orbits: -go to safe configuration -send packet to Mothership with FDIR log -wait for Ground Station instructions. L4: -activate CAM -send packet with FDIR log -wait for Ground Station instructions.	4	3	12
CAM.2	LUMIO Camera (payload)	Science and Navigation	Moon outside Field of View	Camera malfunctioning, e.g. electronics or software failure	No navigation and/or science data available	-Divergence of navigation - Malfunction of GNC (w/o risk of collision) - not possible to produce science product.	4	During the image processing, no Moon is detected in the picture taken by the camera.	Cross-check with the ADCS to locate the failure. Also, check risk of collision (if high risk, activate L4). L0: wait 1 minute (phase 1 and 3) or 3 minutes (phase 2, when there is the change from Earth-based navigation to Moon-based navigation). L1: the camera is reset 4 times. -request health-check. -send packet with FDIR log. L2: N/A	1	3	3

ID number	Item	Function	Failure Mode	Failure Cause	Failure effects		Severity Classification	Failure detection / symptoms	Compensating provisions / recommendations	SN	PN	CN
					Local Effects	End Effects						
						-Divergence of navigation - Malfunction of GNC (with risk of collision) - not possible to produce science product.	1	During the image processing, no Moon is detected in the picture taken by the camera.	Check risk of collision based on step-by-step scenario and orbital propagator. If the current risk is high, do range checks and cross-check distances with ISL. L3: if the risk of collision is low, wait for ground instructions. If risk will increase in the next X orbits: -go to safe configuration -send packet to Mothership with FDIR log -wait for Ground Station instructions. L4: -activate CAM -send packet with FDIR log -wait for Ground Station instructions.	4	3	12
CAM.3	LUMIO Camera (payload)	Science and Navigation	Frozen navigation frequency	Camera malfunctioning, e.g. electronics or software failure	No navigation and/or science data available	-Divergence of navigation - Malfunction of GNC (w/o risk of collision) - not possible to produce science product. -excessive power consumption.	4	The acquisition frequency is locked at a certain value, different from the commanded value.	L0: wait 20 seconds. L1: the camera is reset 4 times. -Request health-check to component. -Send packet with FDIR log. L2: N/A	1	3	3
						-Divergence of navigation - Malfunction of GNC (with risk of collision) - not possible to produce science product. -excessive power consumption.	1	The acquisition frequency is locked at a certain value, different from the commanded value.	Check risk of collision based on step-by-step scenario and orbital propagator. If the current risk is high, do range checks and cross-check distances with ISL. L3: if the risk of collision is low, wait for ground instructions. If risk will increase in the next X orbits: -go to safe configuration -send packet to Mothership with FDIR log -wait for Ground Station instructions. L4: -activate CAM -send packet with FDIR log -wait for Ground Station instructions.	4	3	12
CAM.4	LUMIO Camera (payload)	Science and Navigation	General failure - navigation frequency	Camera malfunctioning, e.g. electronics or software failure	No navigation and/or science data available	-Divergence of navigation - Malfunction of GNC (w/o risk of collision) - not possible to produce science product. -excessive power consumption.	4	The acquisition frequency is different from the commanded value.	L0: wait 20 seconds. L1: the camera is reset 4 times. -Request health-check to component. -Send packet with FDIR log. L2: N/A	1	3	3
						-Divergence of navigation - Malfunction of GNC (with risk of collision)	1	The acquisition frequency is different from the commanded value.	Check risk of collision based on step-by-step scenario and orbital propagator. If the current risk is high, do range checks and cross-check distances with ISL. L3: if the risk of collision is low, wait for ground instructions. If risk will increase in the next X orbits: -go to safe configuration	4	3	12

ID number	Item	Function	Failure Mode	Failure Cause	Failure effects		Severity Classification	Failure detection / symptoms	Compensating provisions / recommendations	SN	PN	CN
					Local Effects	End Effects						
						- not possible to produce science product. -excessive power consumption.			-send packet to Mothership with FDIR log -wait for Ground Station instructions. L4: -activate CAM -send packet with FDIR log -wait for Ground Station instructions.			
CAM.5	LUMIO Camera (payload)	Science and Navigation	Low science frequency	Camera malfunctioning, e.g. electronics or software failure	The science image acquisition frequency is lower than nominal	Not possible to produce science product.	2	Check camera frequency value (lower than an acceptable threshold).	L0: wait 20 seconds. L1: the camera is reset 4 times. -Request health-check to component. -Send packet with FDIR log. L2: N/A If L0/L1 do not resolve the failure, wait for Ground Station instructions in the current orbit.	3	3	9
CAM.6	LUMIO Camera (payload)	Science and Navigation	High science frequency	Camera malfunctioning, e.g. electronics or software failure	The science image acquisition frequency is high, despite it should be 0 [Hz]	-Excessive power consumption - OBPDP memory saturated	2	Check camera frequency value (higher than 0 when it should be 0)	L0: wait 20 seconds. L1: the camera is reset 4 times. -Request health-check to component. -Send packet with FDIR log. L2: N/A If L0/L1 do not resolve the failure, wait for Ground Station instructions in the current orbit.	3	3	9
CAM.7	LUMIO Camera (payload)	Science and Navigation	Malformed data package	Camera malfunctioning, e.g. electronics or software failure	No navigation and/or science data available	Divergence of navigation - Malfunction of GNC (w/o risk of collision) - not possible to produce science product. -excessive power consumption.	4	Check camera packet	L0: wait 20 seconds. L1: the camera is reset 4 times. -Request health-check to component. -Send packet with FDIR log. L2: N/A	1	1	1
						-Divergence of navigation - Malfunction of GNC (with risk of collision) - not possible to produce science product. -excessive power consumption.	1	Check camera packet	Check risk of collision based on step-by-step scenario and orbital propagator. If the current risk is high, do range checks and cross-check distances with ISL. L3: if the risk of collision is low, wait for ground instructions. If risk will increase in the next X orbits: -go to safe configuration -send packet to Mothership with FDIR log -wait for Ground Station instructions. L4: -activate CAM -send packet with FDIR log -wait for Ground Station instructions.	4	3	12

ID number	Item	Function	Failure Mode	Failure Cause	Failure effects		Severity Classification	Failure detection / symptoms	Compensating provisions / recommendations	SN	PN	CN
					Local Effects	End Effects						
OBPDP.1	Payload processor (OBPDP)	Science and Navigation	Recoverable hardware failure	Overheating, high voltage spikes, overclocking or external radiation	No navigation and/or science data are produced	-Divergence of navigation - Malfunction of GNC (w/o risk of collision) - Not possible to produce science product.	4	Negative validity flag from payload processor	L0: wait 10 seconds. L1: reset the processor 4 times. - request health check -send packet with FDIR log.	1	2	2
OBPDP.2	Payload processor (OBPDP)	Science and Navigation	Unrecoverable hardware failure	Overheating, high voltage spikes, overclocking or external radiation	No scientific and navigation data are produced	-Divergence of navigation - Malfunction of GNC (w/o risk of collision) - Not possible to produce science product. -less computational power on-board	3	Negative validity flag from payload processor.	L2: switch the payload processor's functions to the AOCS processor. .	4	2	8
OBPDP.3	Payload processor (OBPDP)	Science and Navigation	Malformed data package	Radiation, power fluctuations.	Images not available or readable by the software	-Divergence of navigation - Malfunction of GNC (w/o risk of collision) - Not possible to produce science product.	3	Check package from payload processor	L0: wait 20 seconds. L1: reset the processor 4 times. - request health check -send packet with FDIR log. L2: switch the payload processor's functions to the AOCS processor.	2	2	4
OBC.1	On-Board Computer (CDHS)	Autonomous operation, ground command execution, telemetry collection, communication management, time keeping	Recoverable hardware failure	Overheating, high voltage spikes, overclocking or external radiation	OBC cannot perform processing and housekeeping tasks.	-Not possible to switch S/C modes - Malfunction of GNC (w/o risk of collision) -No time keeping	4	Negative validity flag from OBC	L0: wait 10 seconds. L1: reset the processor 4 times. - request health check -send packet with FDIR log.	1	2	2

ID number	Item	Function	Failure Mode	Failure Cause	Failure effects		Severity Classification	Failure detection / symptoms	Compensating provisions / recommendations	SN	PN	CN
					Local Effects	End Effects						
OBC.2	On-Board Computer (CDHS)	Autonomous operation, ground command execution, telemetry collection, communication management, time keeping	Unrecoverable hardware failure	Overheating, high voltage spikes, overclocking or external radiation	OBC cannot perform processing and housekeeping tasks.	-Not possible to switch S/C modes - Malfunction of GNC (w/o risk of collision) -No time keeping -less computational power on-board	3	Negative validity flag from OBC	L2: switch OBC's functions to payload processor.	2	2	4
OBC.3	On-Board Computer (CDHS)	Autonomous operations, ground command execution, telemetry collection, communication management, time keeping	Malformed data package from Mothership	Communication disturbances with Mothership	The OBC is not able to read the ground commands	-It is not possible to execute ground commands.	4	The OBC receives malformed data from Mothership.	L0: send packet with FDIR log -wait for new Mothership data package. If the new data package is malformed too, activate L1. L1: restart the unit (and the communication system) 4 times -request health check -wait for Ground Station instructions. L2: N/A	1	2	2
OBC.4	On-Board Computer (CDHS)	Autonomous operation, ground command execution, telemetry collection, communication management, time keeping	Human failure	Human error (ground crew)	The OBC receives a wrong command from ground	-Wrong mode is triggered -Wrong manoeuvre is executed	4	Parameters in the ground commands exceed pre-defined ranges.	L0: N/A L1: lock out command -send packet with FDIR log -wait for new Mothership data package L2: N/A	1	1	1

ID number	Item	Function	Failure Mode	Failure Cause	Failure effects		Severity Classification	Failure detection / symptoms	Compensating provisions / recommendations	SN	PN	CN
					Local Effects	End Effects						
OBC.5	On-Board Computer (CDHS)	Autonomous operation, ground command execution, telemetry collection, communication management, time keeping	Malformed data package	External radiation, electrical failure.	Errors in the command from OBC.	-Not possible to switch ADCS between modes - Malfunction of GNC (w/o risk of collision)	3	Check package from OBC	L0: wait 20 seconds. L1: reset the OBC 4 times -request health-check -send packet with FDIR log. L2: switch OBC's functions to payload processor.	2	2	4
OBC.6	On-Board Computer (CDHS)	Autonomous operation, ground command execution, telemetry collection, communication management, time keeping	Wrong failure flag during deployment	Software problem	The FDIR for the deployment is triggered when not necessary	-Excessive time spent In recovery -possible chain of failures	4	Cross-check the failure flag from the OBBC with the signal from the deployment switch.	L0: wait 20 seconds. L1: reset the OBC 4 times -request health-check -send packet with FDIR log. L2: switch OBC's functions to payload processor.	1	1	1
OBC.7	On-Board Computer (CDHS)	Autonomous operation, ground command execution, telemetry collection, communication management, time keeping	Time-keeping bug	Software problem.	The time of the OBC is wrong	Wrong timestamp added to the scientific product.	4	The time keeping of the Mothership and of LUMIO is not coherent.	L0: N/A L1: send packet with FDIR log -adjust the time keeping: synchronize with Mothership. L2: N/A	1	1	1
RW.1	Reaction Wheel (ADCS)	Attitude actuator (precision pointing and slew rate)	Speed controller failure	Electrical or software problem	RW degrade performance	- Malfunction of ADCS (less pointing accuracy) - Malfunction of GNC (w/o risk of collision)	4	RW's response time degradation, negative validity flag	L0: wait 20 seconds. L1: reset the unit 4 times - request health-check -send packet with FDIR log.	1	2	2

ID number	Item	Function	Failure Mode	Failure Cause	Failure effects		Severity Classification	Failure detection / symptoms	Compensating provisions / recommendations	SN	PN	CN
					Local Effects	End Effects						
						- Malfunction of ADCS (less pointing accuracy) - Malfunction of GNC (with risk of collision)	1	RW's response time degradation, negative validity flag	Check risk of collision based on step-by-step scenario and orbital propagator. If the current risk is high, do range checks and cross-check distances with ISL. L3: if the risk of collision is low, wait for ground instructions. If risk will increase in the next X orbits: -go to safe configuration -send packet to Mothership with FDIR log -wait for Ground Station instructions. L4: -activate CAM -send packet with FDIR log -wait for Ground Station instructions.	2	2	4
RW.2	Reaction Wheel (ADCS)	Attitude actuator (precision pointing and slew rate)	Under voltage	Electrical problem	RW degrade performance	- Malfunction of ADCS (less pointing accuracy) - Malfunction of GNC (w/o risk of collision)	4	RW's voltage drop.	L0: wait 20 seconds. L1: reset the unit 4 times - request health-check -send packet with FDIR log.	1	2	2
						- Malfunction of ADCS (less pointing accuracy) - Malfunction of GNC (with risk of collision)	1	RW's voltage drop.	Check risk of collision based on step-by-step scenario and orbital propagator. If the current risk is high, do range checks and cross-check distances with ISL. L3: if the risk of collision is low, wait for ground instructions. If risk will increase in the next X orbits: -go to safe configuration -send packet to Mothership with FDIR log -wait for Ground Station instructions. L4: -activate CAM -send packet with FDIR log -wait for Ground Station instructions.	2	2	4
RW.3	Reaction Wheel (ADCS)	Attitude actuator (precision pointing and slew rate)	Locked speed	Loss of feedback	No control torque produced.	- Malfunction of ADCS (less pointing accuracy) - Malfunction of GNC (w/o risk of collision)	4	RW's speed is constant despite changes in the command	L0: wait 20 seconds. L1: reset the unit 4 times - request health-check -send packet with FDIR log.	1	2	2
						- Malfunction of ADCS (less pointing accuracy) - Malfunction of GNC (with risk of collision)	1	RW's speed is constant despite changes in the command	Check risk of collision based on step-by-step scenario and orbital propagator. If the current risk is high, do range checks and cross-check distances with ISL. L3: if the risk of collision is low, wait for ground instructions. If risk will increase in the next X orbits: -go to safe configuration -send packet to Mothership with FDIR log -wait for Ground Station instructions. L4: -activate CAM -send packet with FDIR log -wait for Ground Station instructions.	2	2	4

ID number	Item	Function	Failure Mode	Failure Cause	Failure effects		Severity Classification	Failure detection / symptoms	Compensating provisions / recommendations	SN	PN	CN
					Local Effects	End Effects						
RW.4	Reaction Wheel (ADCS)	Attitude actuator (precision pointing and slew rate)	General failure	Loss of feedback, software failure, mechanical failure, e.g. excessive friction	Wrong control torque produced	- Malfunction of ADCS (less pointing accuracy) - Malfunction of GNC (w/o risk of collision)	4	RW's speed is different from the command	L0: wait 20 seconds. L1: reset the unit 4 times - request health-check -send packet with FDIR log.	1	2	2
						- Malfunction of ADCS (less pointing accuracy) - Malfunction of GNC (with risk of collision)	1	RW's speed is different from the command	Check risk of collision based on step-by-step scenario and orbital propagator. If the current risk is high, do range checks and cross-check distances with ISL. L3: if the risk of collision is low, wait for ground instructions. If risk will increase in the next X orbits: -go to safe configuration -send packet to Mothership with FDIR log -wait for Ground Station instructions. L4: -activate CAM -send packet with FDIR log -wait for Ground Station instructions.	2	2	4
RW.5	Reaction Wheel (ADCS)	Attitude actuator (precision pointing and slew rate)	Overheating	Wrong internal functioning (e.g. excessive friction) or external causes	RW's temperature higher than allowed by its technical specification	- Malfunction of ADCS (less pointing accuracy) - Malfunction of GNC (w/o risk of collision) -other S/C subsystems overheated	4	RW's temperature increase, heater functioning correctly	*If the failure does not happen during a critical manoeuvre:* L0: stop actuation of RW -wait 5 minutes *If the failure happens during a critical manoeuvre:* L0: wait until the end of the manoeuvre -stop actuation of RW L1: reset the unit 4 times -request health-check -send packet with FDIR log.	1	2	2
						- Malfunction of ADCS (less pointing accuracy) - Malfunction of GNC (with risk of collision) -other S/C subsystems overheated	1	RW's temperature increase, heater functioning correctly	Check risk of collision based on step-by-step scenario and orbital propagator. If the current risk is high, do range checks and cross-check distances with ISL. L3: if the risk of collision is low, wait for ground instructions. If risk will increase in the next X orbits: -go to safe configuration -send packet to Mothership with FDIR log -wait for Ground Station instructions. L4: -activate CAM -send packet with FDIR log -wait for Ground Station instructions.	2	2	4
RW.6	Reaction Wheel (ADCS)	Attitude actuator (precision pointing and slew rate)	Underheating	Design failure	RW's temperature lower than allowed by its technical	- Malfunction of ADCS (less pointing accuracy) - Malfunction of GNC (w/o risk of collision)	4	RW's temperature decrease, heater functioning correctly	*If the failure does not happen during a critical manoeuvre:* L0: stop actuation of RW -wait 5 minutes *If the failure happens during a critical manoeuvre:* L0: wait until the end of the manoeuvre -stop actuation of RW L1: reset the unit 4 times	1	1	1

ID number	Item	Function	Failure Mode	Failure Cause	Failure effects		Severity Classification	Failure detection / symptoms	Compensating provisions / recommendations	SN	PN	CN
					Local Effects	End Effects						
					specification				-request health-check -send packet with FDIR log.			
						- Malfunction of ADCS (less pointing accuracy) - Malfunction of GNC (with risk of collision)	1	RW's temperature decrease, heater functioning correctly	Check risk of collision based on step-by-step scenario and orbital propagator. If the current risk is high, do range checks and cross-check distances with ISL. L3: if the risk of collision is low, wait for ground instructions. If risk will increase in the next X orbits: -go to safe configuration -send packet to Mothership with FDIR log -wait for Ground Station instructions. L4: -activate CAM -send packet with FDIR log -wait for Ground Station instructions.	2	1	2
RW.7	Reaction Wheel (ADCS)	Attitude actuator (precision pointing and slew rate)	Malformed data package	External radiation, electrical failure.	No data regarding the RWs is available	- Malfunction of ADCS (less pointing accuracy) - Malfunction of GNC (w/o risk of collision)	4	Check RWs packet	*If the failure does not happen during a critical manoeuvre:* L0: stop actuation of RW -wait 20 seconds *If the failure happens during a critical manoeuvre:* L0: wait until the end of the manoeuvre -stop actuation of RW L1: reset the unit 4 times -request health-check -send packet with FDIR log.	1	2	2
						- Malfunction of ADCS (less pointing accuracy) - Malfunction of GNC (with risk of collision)	1	Check RWs packet	Check risk of collision based on step-by-step scenario and orbital propagator. If the current risk is high, do range checks and cross-check distances with ISL. L3: if the risk of collision is low, wait for ground instructions. If risk will increase in the next X orbits: -go to safe configuration -send packet to Mothership with FDIR log -wait for Ground Station instructions. L4: -activate CAM -send packet with FDIR log -wait for Ground Station instructions.	2	2	4
GT.1	Cold gas thruster (ADCS)	Satellite de-tumbling and RW de-saturation	Locked output	Stuck valve	The thruster keeps losing propellant and	-S/C rotational speed increases - Malfunction of GNC (w/o risk of collision)	2	Decreasing propellant budget and open valve, despite command is 0	L0: wait for 5 seconds. L1: command to reset the unit (until it works) -send packet with FDIR log. L2: N/A	3	3	9

ID number	Item	Function	Failure Mode	Failure Cause	Failure effects		Severity Classification	Failure detection / symptoms	Compensating provisions / recommendations	SN	PN	CN
					Local Effects	End Effects						
					creating thrust	-S/C rotational speed increases - Malfunction of GNC (with risk of collision)	1	Decreasing propellant budget and open valve, despite command is 0	Check risk of collision based on step-by-step scenario and orbital propagator. If the current risk is high, do range checks and cross-check distances with ISL. L3: if the risk of collision is low, wait for ground instructions. If risk will increase in the next X orbits: -go to safe configuration -send packet to Mothership with FDIR log -wait for Ground Station instructions. L4: -activate CAM -send packet with FDIR log -wait for Ground Station instructions.	4	3	12
GT.2	Cold gas thruster (ADCS)	Satellite de-tumbling and RW de-saturation	Command out of range	Guidance Navigation and Control failure	The gas thruster receives a wrong command	-The manoeuvre cannot be executed -Divergence of navigation	4	Commanded thrust higher than maximum	L0: wait for 20 seconds. L1: block the command -request health-check to GNC -send packet with FDIR log - wait for Ground Station instructions in the current orbit.	1	3	3
GT.3	Cold gas thruster (ADCS)	Satellite de-tumbling and RW de-saturation	No thrust	Stuck valve	No thrusting action	-S/C detumbling not possible -RW's desaturation not possible - Malfunction of GNC (w/o risk of collision)	3	No propellant consumption, constant propellant budget, despite command is higher than 0	L0: wait 20 seconds. L1: reset the unit 4 times - request health-check -send packet with FDIR log. L2: switch to redundant configuration	2	3	6
GT.4	Cold gas thruster (ADCS)	Satellite de-tumbling and RW de-saturation	Propellant sensor failure	Electrical failure, radiation	No accurate data on the propellant budget trend	-Not possible to control the operation of the thruster - Malfunction of GNC (w/o risk of collision)	4	Crosscheck command, propellant budget and vale state	L0: stop the use of the thruster -wait 5 minutes L1: reset the unit 4 times - request health-check -send packet with FDIR log. L2: N/A	1	3	3
						-Not possible to control the operation of the thruster - Malfunction of GNC (with risk of collision))	1	Crosscheck command, propellant budget and vale state	Check risk of collision based on step-by-step scenario and orbital propagator. If the current risk is high, do range checks and cross-check distances with ISL. L3: if the risk of collision is low, wait for ground instructions. If risk will increase in the next X orbits: -go to safe configuration -send packet to Mothership with FDIR log -wait for Ground Station instructions. L4: -activate CAM -send packet with FDIR log -wait for Ground Station instructions.	2	3	6

ID number	Item	Function	Failure Mode	Failure Cause	Failure effects		Severity Classification	Failure detection / symptoms	Compensating provisions / recommendations	SN	PN	CN
					Local Effects	End Effects						
GT.5	Cold gas thruster (ADCS)	Satellite de-tumbling and RW de-saturation	Valve sensor failure	Electrical failure, radiation	No data about the thruster's functioning	-Not possible to control the operation of the thruster - Malfunction of GNC (w/o risk of collision)	3	The internal sensors of the thruster (propellant level and valve sensor) have conflicting output	L0: wait 20 seconds. L1: reset the unit 4 times - request health-check -send packet with FDIR log. L2: switch to redundant configuration	2	3	6
GT.6	Cold gas thruster (ADCS)	Satellite de-tumbling and RW de-saturation	Malformed data package	External radiation, electrical failure.	Errors in the command from OBC.	-Not possible to control the operation of the thruster - Malfunction of GNC (w/o risk of collision)	4	Check data package from GTs	L0: stop the use of the thruster -wait 5 minutes L1: reset the unit 4 times - request health-check -send packet with FDIR log. L2: N/A	1	3	3
						-Not possible to control the operation of the thruster - Malfunction of GNC (with risk of collision))	1	Check data package from GTs	Check risk of collision based on step-by-step scenario and orbital propagator. If the current risk is high, do range checks and cross-check distances with ISL. L3: if the risk of collision is low, wait for ground instructions. If risk will increase in the next X orbits: -go to safe configuration -send packet to Mothership with FDIR log -wait for Ground Station instructions. L4: -activate CAM -send packet with FDIR log -wait for Ground Station instructions.	2	3	6
SS.1	Sun sensor (ADCS)	Absolute attitude estimation	Frozen sensor	Internal electrical failure, due to overheating, high voltage spikes or external radiation	No attitude data	-AOCS cannot determine S/C attitude - Malfunction of GNC (w/o risk of collision)	4	Negative validity flag, frozen output, cross-check with the STs	L0: wait 1 minute. L1: reset the unit 4 times - request health-check -send packet with FDIR log. L2: switch to redundant unit.	1	1	1
SS.2	Sun sensor (ADCS)	Absolute attitude estimation	General failure	Internal electrical failure, due to overheating, high voltage spikes or external radiation	No attitude data	-AOCS cannot determine S/C attitude - Malfunction of GNC (w/o risk of collision)	4	Negative validity flag, cross-check with the STs	L0: wait 1 minute. L1: reset the unit 4 times - request health-check -send packet with FDIR log. L2: switch to redundant unit.	1	1	1

ID number	Item	Function	Failure Mode	Failure Cause	Failure effects		Severity Classification	Failure detection / symptoms	Compensating provisions / recommendations	SN	PN	CN
					Local Effects	End Effects						
SS.3	Sun sensor (ADCS)	Absolute attitude estimation	Sun not in FoV	Internal electrical failure, due to overheating, high voltage spikes or external radiation	No attitude data	-AOCS cannot determine S/C attitude - Malfunction of GNC (w/o risk of collision)	4	Negative validity flag, cross-check with expected attitude.	L0: wait 1 minute. L1: reset the unit 4 times - request health-check -send packet with FDIR log. L2: switch to redundant unit.	1	1	1
SS.4	Sun sensor (ADCS)	Absolute attitude estimation	Malformed data package	Internal electrical failure, external radiation	No attitude data	-AOCS cannot determine S/C attitude - Malfunction of GNC (w/o risk of collision)	4	Check SS data package.	L0: wait 1 minute. L1: reset the unit 4 times - request health-check -send packet with FDIR log. L2: switch to redundant unit.	1	1	1
ST.1	Star-Tracker (ADCS)	Absolute attitude estimation	Frozen sensor	Internal electrical failure, due to overheating, high voltage spikes or external radiation	No attitude data	-AOCS cannot determine S/C attitude - Malfunction of GNC (w/o risk of collision)	3	Negative validity flag, frozen output, cross-check with the other ST and the SS.	L0: wait 20 seconds. L1: reset the unit 4 times - request health-check -send packet with FDIR log. L2: switch to redundant configuration	2	1	2
ST.2	Star-Tracker (ADCS)	Absolute attitude estimation	General failure	Internal electrical failure, due to overheating, high voltage spikes or external radiation	No attitude data	-AOCS cannot determine S/C attitude - Malfunction of GNC (w/o risk of collision)	3	Negative validity flag, cross-check with the other ST and the SS.	L0: wait 20 seconds. L1: reset the unit 4 times - request health-check -send packet with FDIR log. L2: switch to redundant configuration	2	1	2
ST.3	Star-Tracker (ADCS)	Absolute attitude estimation	Sun in FoV	Sun exclusion angle less than nominal or internal electrical failure.	No attitude data	-AOCS cannot determine S/C attitude - Malfunction of GNC (w/o risk of collision)	3	Negative validity flag, cross-check with expected attitude	L0: wait 20 seconds. L1: reset the unit 4 times - request health-check -send packet with FDIR log. L2: switch to redundant configuration	2	1	2

ID number	Item	Function	Failure Mode	Failure Cause	Failure effects		Severity Classification	Failure detection / symptoms	Compensating provisions / recommendations	SN	PN	CN
					Local Effects	End Effects						
ST.4	Star-Tracker (ADCS)	Absolute attitude estimation	Overheating	Wrong internal functioning or external causes	ST temperature higher than allowed by its technical specification	-AOCS cannot determine S/C attitude - Malfunction of GNC (w/o risk of collision)	3	Temperature of the unit, cross-check with the heater	L0: wait 20 seconds. L1: reset the unit 4 times - request health-check -send packet with FDIR log. L2: switch to redundant configuration	2	1	2
ST.5	Star-Tracker (ADCS)	Absolute attitude estimation	Underheating	Design failure	ST temperature lower than allowed by its technical specification	-AOCS cannot determine S/C attitude - Malfunction of GNC (w/o risk of collision)	3	Temperature of the unit, cross-check with the heater	L0: wait 20 seconds. L1: reset the unit 4 times - request health-check -send packet with FDIR log. L2: switch to redundant configuration	2	1	2
ST.6	Star-Tracker (ADCS)	Absolute attitude estimation	Malformed data package	Radiation, electrical failure	ST data cannot be retrieved	-AOCS cannot determine S/C attitude - Malfunction of GNC (w/o risk of collision)	3	Check ST data package	L0: wait 20 seconds. L1: reset the unit 4 times - request health-check -send packet with FDIR log. L2: switch to redundant configuration	2	1	2
IMU.1	IMU (ADCS)	Relative attitude estimation (angular velocity and acceleration)	Frozen gyroscope	Internal electrical failure, due to overheating, high voltage spikes or external radiation	No S/C angular velocity data	-AOCS cannot determine S/C attitude - Malfunction of GNC (w/o risk of collision)	4	Negative validity flag, cross-check with ST, frozen output.	L0: wait 20 seconds. L1: reset the unit 4 times - request health-check -send packet with FDIR log.	1	2	2
						-AOCS cannot determine S/C attitude - Malfunction of GNC (with risk of collision)	1	Negative validity flag, cross-check with ST, frozen output.	Check risk of collision based on step-by-step scenario and orbital propagator. If the current risk is high, do range checks and cross-check distances with ISL. L3: if the risk of collision is low, wait for ground instructions. If risk will increase in the next X orbits: -go to safe configuration -send packet to Mothership with FDIR log -wait for Ground Station instructions. L4: -activate CAM -send packet with FDIR log -wait for Ground Station instructions.	4	2	8
IMU.2	IMU (ADCS)	Relative attitude estimation (angular velocity and	General failure - gyroscope	Internal electrical failure, due to overheating, high	No S/C angular velocity data	-AOCS cannot determine S/C attitude - Malfunction of GNC (w/o risk of collision)	4	Negative validity flag, cross-check with ST	L0: wait 20 seconds. L1: reset the unit 4 times - request health-check -send packet with FDIR log.	1	2	2

ID number	Item	Function	Failure Mode	Failure Cause	Failure effects		Severity Classification	Failure detection / symptoms	Compensating provisions / recommendations	SN	PN	CN
					Local Effects	End Effects						
		acceleration)		voltage spikes or external radiation		-AOCS cannot determine S/C attitude - Malfunction of GNC (with risk of collision)	1	Negative validity flag, cross-check with ST	Check risk of collision based on step-by-step scenario and orbital propagator. If the current risk is high, do range checks and cross-check distances with ISL. L3: if the risk of collision is low, wait for ground instructions. If risk will increase in the next X orbits: -go to safe configuration -send packet to Mothership with FDIR log -wait for Ground Station instructions. L4: -activate CAM -send packet with FDIR log -wait for Ground Station instructions.	4	2	8
IMU.3	IMU (ADCS)	Relative attitude estimation (angular velocity and acceleration)	Malformed data package	Internal electrical failure, due to overheating, high voltage spikes or external radiation	No S/C acceleration data	-AOCS cannot determine S/C attitude - Malfunction of GNC (w/o risk of collision)	4	Check IMU data package	L0: wait 20 seconds. L1: reset the unit 4 times - request health-check -send packet with FDIR log.	1	2	2
						-AOCS cannot determine S/C attitude - Malfunction of GNC (with risk of collision)	1	Check IMU data package	Check risk of collision based on step-by-step scenario and orbital propagator. If the current risk is high, do range checks and cross-check distances with ISL. L3: if the risk of collision is low, wait for ground instructions. If risk will increase in the next X orbits: -go to safe configuration -send packet to Mothership with FDIR log -wait for Ground Station instructions. L4: -activate CAM -send packet with FDIR log -wait for Ground Station instructions.	4	2	8
AOCS.1	AOCS processor (ADCS)	Navigation and attitude estimation, attitude control, propulsion control	Recoverable hardware failure	Overheating, high voltage spikes, overclocking or external radiation	AOCS processor cannot perform attitude estimation and control.	-No attitude control -No science produced - Malfunction of GNC (w/o risk of collision) less computational power on-board	4	Negative validity flag from AOCS processor	L0: wait 10 seconds. L1: reset the processor 4 times. - request health check -send packet with FDIR log.	1	2	1
AOCS.2	AOCS processor (ADCS)	Navigation and attitude estimation, attitude control, propulsion control	Unrecoverable hardware failure	Overheating, high voltage spikes, overclocking or external radiation	AOCS processor cannot perform its tasks.	-No attitude control -No science produced - Malfunction of GNC (with risk of collision) less computational power on-board	3	Negative validity flag from AOCS processor	L2: switch AOCS processor's functions to OBC.	2	2	4

ID number	Item	Function	Failure Mode	Failure Cause	Failure effects		Severity Classification	Failure detection / symptoms	Compensating provisions / recommendations	SN	PN	CN
					Local Effects	End Effects						
AOCS.3	AOCS processor (ADCS)	Navigation and attitude estimation, attitude control, propulsion control	Malformed data package	External radiation, electrical failure, software failure	Errors in the AOCS and GNC	- Malfunction of GNC (w/o risk of collision) -wrong system mode is triggered	3	Check package from AOCS processor	L0: wait 20 seconds. L1: reset the OBC 4 times -request health-check -send packet with FDIR log. L2: switch AOCS functions to OBC.	2	2	4
DEP.1	S/C Deployment System (Structure)	Deployment of the S/C after the release from the Mothership	Excessive tumbling rate	Deployment system mechanical failure, due e.g. to vibrational loads during launch	S/C tumbling rate higher than expected	-Excessive propellant consumption -Excessive navigation error -De-tumble cannot be executed	2	The gyroscopes in the IMU measure a S/C rotational speed higher than 30 deg/s.	L0: activate also the RWs to de-tumble -wait for 30 minutes L1: reset the entire satellite 2 times -request health-check to all subsystems -wait up to 30 minutes. L3: try deployment of the antennas (up to 4 times), -send packet with FDIR log - wait for Ground Station instructions.	3	1	3
MT.1	Main thruster (propulsion)	Navigation (execution of the manoeuvres)	Commands out of range	Guidance Navigation and Control failure	The thruster receives a wrong command	-The manoeuvre cannot be executed -Divergence of navigation	4	Commanded thrust higher than maximum	L0: wait for 20 seconds. L1: block the command -request health-check to GNC -send packet with FDIR log - wait for Ground Station instructions in the current orbit.	1	3	3
MT.2	Main thruster (propulsion)	Navigation (execution of the manoeuvres)	No thrust	Stuck valve	No thrusting action	-The manoeuvre cannot be executed -Divergence of navigation - Malfunction of GNC (w/o risk of collision)	3	Cross-check valve state, propellant trend, thrust command	L0: wait 20 seconds. L1: reset the unit 4 times - request health-check -send packet with FDIR log.	2	3	6
						-The manoeuvre cannot be executed -Divergence of navigation - Malfunction of GNC (with risk of collision)	1	Cross-check valve state, propellant trend, thrust command	L3: wait for Ground commands in the current orbit	4	3	12
MT.3	Main thruster	Navigation (execution of the	Locked output	Stuck valve	The thruster keeps	- Decrease of propellant budget	2	Cross-check valve state, propellant	L0: wait for 10 seconds. L1: command to reset the unit (until it works) -send packet with FDIR log.	3	3	9

ID number	Item	Function	Failure Mode	Failure Cause	Failure effects		Severity Classification	Failure detection / symptoms	Compensating provisions / recommendations	SN	PN	CN
					Local Effects	End Effects						
	(propulsion)	manoeuvres)			losing propellant and creating thrust	-Divergence of navigation - Malfunction of GNC (w/o risk of collision)		trend, thrust command				
						- Decrease of propellant budget -Divergence of navigation - Malfunction of GNC (with risk of collision)	1	Cross-check valve state, propellant trend, thrust command	Check risk of collision based on step-by-step scenario and orbital propagator. If the current risk is high, do range checks and cross-check distances with ISL. L3: if the risk of collision is low, wait for ground instructions. If risk will increase in the next X orbits: -go to safe configuration -send packet to Mothership with FDIR log -wait for Ground Station instructions. L4: -activate CAM -send packet with FDIR log -wait for Ground Station instructions.	4	3	12
MT.4	Main thruster (propulsion)	Navigation (execution of the manoeuvres)	Propellant sensor failure	Electrical failure, radiation	No data about the thruster's functioning	-Not possible to control the operation of the thruster - Malfunction of GNC (w/o risk of collision)	3	Cross-check valve state, propellant trend, thrust command	L0: wait for 10 seconds. L1: command to reset the unit (until it works) -send packet with FDIR log.	2	3	6
						-Not possible to control the operation of the thruster - Malfunction of GNC (with risk of collision)	1	Cross-check valve state, propellant trend, thrust command	Check risk of collision based on step-by-step scenario and orbital propagator. If the current risk is high, do range checks and cross-check distances with ISL. L3: if the risk of collision is low, wait for ground instructions. If risk will increase in the next X orbits: -go to safe configuration -send packet to Mothership with FDIR log -wait for Ground Station instructions. L4: -activate CAM -send packet with FDIR log -wait for Ground Station instructions.	4	3	12
MT.5	Main thruster (propulsion)	Navigation (execution of the manoeuvres)	Valve sensor failure	Electrical failure, radiation	No data about the thruster's functioning	-Not possible to control the operation of the thruster - Malfunction of GNC (with risk of collision)	3	Cross-check valve state, propellant trend, thrust command	L0: wait for 10 seconds. L1: command to reset the unit (until it works) -send packet with FDIR log.	2	3	6
						-Not possible to control the operation of the thruster - Malfunction of GNC (w/o risk of collision)	1	Cross-check valve state, propellant trend, thrust command	Check risk of collision based on step-by-step scenario and orbital propagator. If the current risk is high, do range checks and cross-check distances with ISL. L3: if the risk of collision is low, wait for ground instructions. If risk will increase in the next X orbits: -go to safe configuration -send packet to Mothership with FDIR log -wait for Ground Station instructions. L4: -activate CAM -send packet with FDIR log -wait for Ground Station instructions.	4	3	12

ID number	Item	Function	Failure Mode	Failure Cause	Failure effects		Severity Classification	Failure detection / symptoms	Compensating provisions / recommendations	SN	PN	CN
					Local Effects	End Effects						
MT.6	Main thruster (propulsion)	Navigation (execution of the manoeuvres)	Overheating	Excessive friction or external causes	Thruster deformation and/or performance degradation	-Increased propellant consumption -not possible to perform orbital manoeuvres - overheating of other S/C units	4	Thruster's temperature increases over threshold	L0: stop manoeuvre -wait 40 seconds L1: reset the unit 4 times -request health-check -send packet with FDIR log.	1	3	3
						-Increased propellant consumption -overheating of other S/C units -not possible to perform orbital manoeuvres	3	Thruster's temperature increases over threshold	L3: wait for Ground commands in the current orbit	2	3	6
MT.7	Main thruster (propulsion)	Navigation (execution of the manoeuvres)	Underheating	Design failure	Problems to the electronics	-not possible to perform orbital manoeuvres	4	Thruster's temperature under threshold	L0: wait 20 seconds. L1: reset the unit 4 times - request health-check -send packet with FDIR log.	1	1	1
							3	Thruster's temperature under threshold	L3: wait for Ground commands in the current orbit	2	1	2
MT.8	Main thruster (propulsion)	Navigation (execution of the manoeuvres)	Malformed package	External radiation, electronic failure, software failure	Thruster's data are not available	-Not possible to control the operation of the thruster - Malfunction of GNC (with risk of collision)	4	Check data package from thruster	L0: wait for 20 seconds. L1: command to reset the unit (until it works) -send packet with FDIR log.	1	3	3
						-Not possible to control the operation of the thruster - Malfunction of GNC (w/o risk of collision)	1	Check data package from thruster	Check risk of collision based on step-by-step scenario and orbital propagator. If the current risk is high, do range checks and cross-check distances with ISL. L3: if the risk of collision is low, wait for ground instructions. If risk will increase in the next X orbits: -go to safe configuration -send packet to Mothership with FDIR log -wait for Ground Station instructions. L4: -activate CAM -send packet with FDIR log -wait for Ground Station instructions.	4	3	12

ID number	Item	Function	Failure Mode	Failure Cause	Failure effects		Severity Classification	Failure detection / symptoms	Compensating provisions / recommendations	SN	PN	CN
					Local Effects	End Effects						
SP.1	Solar panel (power)	Production of power	Recoverable loss of power	Excessive loads at launch, external radiations	A cell or an array of cells produce less power than expected	-Decrease of the S/C power budget	4	Decreased solar panels current, decreased S/C power budget	L0: wait 1 minute. L1: reset the unit 4 times - request health-check -send packet with FDIR log.	1	4	4
SP.2	Solar panel (power)	Production of power	Unrecoverable loss of power	Excessive loads at launch, external radiations	A cell or an array of cells produce less power than expected	-Decrease of the S/C power budget	2	Decreased solar panels current, decreased S/C power budget	L3: if the risk of collision is low, wait for ground instructions. If risk will increase in the next X orbits: -go to safe configuration -send packet to Mothership with FDIR log -wait for Ground Station instructions.	3	4	12
EPS.1	EPS (power)	Power distribution to the subsystems	Malformed data package	External radiation, software failure, electrical failure	Errors in the EPS data	-Wrong power budget estimation -wrong triggering of S/C modes	4	Check data package from EPS	L0: wait 20 seconds. L1: reset the unit 4 times - request health-check -send packet with FDIR log.	1	1	1
							1	Check data package from EPS	Check risk of collision based on step-by-step scenario and orbital propagator. If the current risk is high, do range checks and cross-check distances with ISL. L3: if the risk of collision is low, wait for ground instructions. If risk will increase in the next X orbits: -go to safe configuration -send packet to Mothership with FDIR log -wait for Ground Station instructions. L4: -activate CAM -send packet with FDIR log -wait for Ground Station instructions.	4	1	4
EPS.2	EPS (power)	Power distribution to the subsystems	Recoverable hardware failure	Overheating, voltage spikes, external loads (e.g. during launch)	Powerline failure	-No sufficient power to certain subsystems/units - Malfunction of GNC (w/o risk of collision)	4	Negative validity flag	L0: wait 20 seconds. L1: reset the unit 4 times - request health-check -send packet with FDIR log.	1	1	1
EPS.3	EPS (power)	Power distribution to the	Unrecoverable hardware failure	Overheating, voltage spikes,	Powerline failure	-No sufficient power to certain subsystems/units	2	Negative validity flag	L3: send packet with FDIR log -wait for Ground instructions in the current orbit.	3	1	3

ID number	Item	Function	Failure Mode	Failure Cause	Failure effects		Severity Classification	Failure detection / symptoms	Compensating provisions / recommendations	SN	PN	CN
					Local Effects	End Effects						
		subsystems		external loads (e.g. during launch)		- Malfunction of GNC (with risk of collision)						
BT.1	Battery (power)	Power storage and power supply	Overheating	Internal or external short-circuit	Battery temperature higher than allowed by its technical specification, dangerous chemical reactions might start	-Decrease of the S/C power budget -Damage to other subsystems	4	Temperature over threshold	*If the failure does not happen during a critical manoeuvre*: L0: switch-off the unit -wait 2 minutes L1: reset the unit 4 times - request health-check -send packet with FDIR log. -L2: N/A *If the failure happens during a critical manoeuvre*: L0: end the manoeuvre, if temperature below a critical threshold -switch-off the unit L1: reset the unit 4 times - request health-check -send packet with FDIR log. L2: N/A	1	2	2
						-Decrease of the S/C power budget -Damage to other subsystems	3	Temperature over threshold	L3: send packet with FDIR log -wait for Ground instructions in the current orbit.	2	2	4
BT.2	Battery (power)	Power storage and power supply	Underheating	Thermal design failure	Battery temperature lower than allowed by its technical specification, life cycle is reduced	-Decrease of the S/C power budget -Decrease of the mission lifetime	4	Temperature under threshold	*If the failure does not happen during a critical manoeuvre*: L0: switch-off the unit -wait 2 minutes L1: reset the unit 4 times - request health-check -send packet with FDIR log. -L2: N/A *If the failure happens during a critical manoeuvre*: L0: end the manoeuvre, if temperature above a critical threshold. Later: -switch-off the unit L1: reset the unit 4 times - request health-check -send packet with FDIR log. L2: N/A	1	2	2
						-Decrease of the S/C power budget -Decrease of the mission lifetime	3	Temperature under threshold	L3: send packet with FDIR log -wait for Ground instructions in the current orbit.	2	2	4
BT.3	Battery (power)	Power storage and power supply	Over-discharged battery	Excessive power consumption by other	Reduction of the electrolyte,	-Decrease of the S/C power budget -Decrease of the mission lifetime	4	Battery DOD lower than minimum	*If the failure does not happen during a manoeuvre*: L0: switch-off the unit -wait 20 seconds L1: reset the unit 4 times - request health-check	1	2	2

ID number	Item	Function	Failure Mode	Failure Cause	Failure effects		Severity Classification	Failure detection / symptoms	Compensating provisions / recommendations	SN	PN	CN
					Local Effects	End Effects						
				units, design error	production of combustible gas, with potential security risks and performance degradation	-Damage to other subsystems			-send packet with FDIR log L2: switch to redundancy. *If the failure happens during a (non-critical) manoeuvre*: L2: switch to redundancy -fix the failure before Phase 2			
							2	Battery DOD lower than minimum	*During Phase 2:* L3: send packet with FDIR log -wait for Ground instructions in the current orbit.	3	2	6
BT.4	Battery (power)	Power storage and power supply	Voltage measurement failure	Software failure, electrical failure	The voltage measurement is not accurate.	-It is not possible to determine accurately the DOD of the battery	4	Battery voltage and DOD measurements are not coherent	L0: wait 20 seconds L1: -reset the unit 4 times - request health-check -send packet with FDIR log. L2: switch to redundancy -fix the failure before Phase 2	1	2	2
							2	Battery voltage and DOD measurements are not coherent	During Phase 2: L3: send packet with FDIR log -wait for Ground instructions in the current orbit.	3	2	6
BT.5	Battery (power)	Power storage and power supply	DOD measurement failure	Software failure, electrical failure	The DOD measurement is not accurate	-It is not possible to determine accurately the DOD of the battery	4	Battery voltage and DOD measurements are not coherent	L0: wait 20 seconds L1: -reset the unit 4 times - request health-check -send packet with FDIR log. L2: switch to redundancy -fix the failure before Phase 2	1	2	2
							2	Battery voltage and DOD measurements are not coherent	During Phase 2: L3: send packet with FDIR log -wait for Ground instructions in the current orbit.	3	2	6
COMM .1	UHF Antenna (communication)	Communication	No uplink to Mothership	Antenna electrical problem	Uplink antenna does not send data to Mothership	-Not possible to send all the telemetry and science data	4	Error flag from mothership	L0: wait 20 seconds L1: reset the unit 10 times -request health-check -send packet with FDIR log.	1	3	3
						-Not possible to send telemetry and science data	2	Error flag from mothership	L3: try to send packet with FDIR log -wait for Ground instructions in the current orbit.	3	3	9
COMM .2	UHF Antenna (communication)	Communication	No downlink from Mothership	Antenna electrical problem	Downlink antenna does not receive data from Mothership	-Not possible to receive messages from Mothership	4	No message received from Mothership	L0: wait 20 seconds. L1: reset the unit 10 times -request health-check -send packet with FDIR log.	1	3	3
						-Not possible to receive messages from Mothership	3	No message received from Mothership	L3: send packet with FDIR log -wait for Ground instructions in the current orbit	2	3	6

ID number	Item	Function	Failure Mode	Failure Cause	Failure effects		Severity Classification	Failure detection / symptoms	Compensating provisions / recommendations	SN	PN	CN
					Local Effects	End Effects						
DEP.2	Antenna deployment system (structure)	Deployment of the antenna	No deployment - electrical	Electrical problem	The antenna is not deployed	-Reduced or impossible communication with Mothership	2	Deployment switch still pressed, failure flag from OBC	L0: -try deployment of solar panels -wait until battery DOD increases	3	3	9
DEP.3	Antenna deployment system (structure)	Deployment of the antenna	No deployment - mechanical	Vibrational loads at launch	The antenna is not deployed)	-Reduced or impossible communication with Mothership	2	Deployment switch still pressed, error flag from OBC	L3: start transmission -try to send packet to Mothership with FDIR log -try deployment each orbit -wait for Ground Station instructions.	3	3	9
DEP.4	Sola panel deployer (structure)	Deployment of the solar panels	No deployment - electrical	Electrical problem	The solar panels are not deployed	-Power budget decreased	2	Deployment switch still pressed, error flag from OBC	Try forced deployment (use battery above their max DOD level)	3	4	12
DEP.5	Sola panel deployer (structure)	Deployment of the solar panels	No deployment - mechanical	Vibrational loads at launch	The solar panels are not deployed	-Power budget decreased	2	Deployment switch still pressed, error flag from OBC	L3: -deploy antenna --start transmission -try to send packet to Mothership with FDIR log -try deployment each orbit -wait for Ground Station instructions.	3	4	12
SADA.1	SADA (structure)	Power maximization	Recoverable locked SADA	Vibrational loads at launch	SADA is locked and do not track the Sun	-Power budget decreased - Malfunction of GNC (w/o risk of collision)	4	Negative validity flag, reduced power from SPs	L0: wait 20 seconds. L1: reset the unit 4 times -request health-check -send packet with FDIR log.	1	4	4
SADA.2	SADA (structure)	Power maximization	Unrecoverable locked SADA	Vibrational loads at launch	The solar panels are not deployed SADA is locked and do not track the Sun	-Power budget decreased - Malfunction of GNC (w/o risk of collision)	1	No signal from SADA, reduced power budget	Check risk of collision based on step-by-step scenario and orbital propagator. If the current risk is high, do range checks and cross-check distances with ISL. L3: if the risk of collision is low, wait for ground instructions. If risk will increase in the next X orbits: -go to safe configuration -send packet to Mothership with FDIR log -wait for Ground Station instructions. L4: -activate CAM -send packet with FDIR log -wait for Ground Station instructions.	3	4	12
HEAT.1	Heater (thermal)	Thermal control (heat production)	No heating	Electrical problem	The heater produces no heating power	- The temperature of the unit associated to the heater might drop below the minimum allowed	4	Cross-check the heater current and the unit's temperature	L0: wait 20 seconds. L1: reset the unit 4 times -request health-check -send packet with FDIR log. L3: switch-off the heater	1	1	1

ID number	Item	Function	Failure Mode	Failure Cause	Failure effects		Severity Classification	Failure detection / symptoms	Compensating provisions / recommendations	SN	PN	CN
					Local Effects	End Effects						
HEAT.2	Heater (thermal)	Thermal control (heat production)	Locked output	Thermal control software problem	The heater heats unit the allowable range	-Overheating of the unit associated to the heater	4	High current input to the heater, high temperature of the unit	L0: wait 20 seconds. L1: reset the unit 4 times -request health-check -send packet with FDIR log. L3: switch-off the interested unit until the temperature increase again over threshold	1	1	1
TSENS.1	Temperature sensor (thermal)	Thermal control (temperature measurement)	Frozen sensor	Internal electrical failure, due to overheating, high voltage spikes or external radiation	No reliable temperature data	-Activation of heater when not necessary -Overheating of sensitive units	4	Cross-check with the other temperature sensors, constant output	L0: wait 20 seconds. L1: reset the unit 4 times L2: switch to redundant configuration	1	1	1
TSENS.2	Temperature sensor (thermal)	Thermal control (temperature measurement)	General failure	Internal electrical failure, due to overheating, high voltage spikes or external radiation	No reliable temperature data	-Activation of heater when not necessary -Overheating of sensitive units	4	Cross-check with the other temperature sensors	L0: wait 20 seconds. L1: reset the unit 4 times L2: switch to redundant configuration	1	1	1

C FDI Simulink model screenshots

In this Appendix, relevant screenshots of the Simulink and Stateflow FDI model will be documented, in support to the description made in Chapter 3. The Appendix is divided into sub-sections, each one dedicated to a particular module.

C.1 Main Thruster module

In this Section, relevant screenshots of the Simulink model of the Main Thruster FDI, described in Section 3.2, are reported.

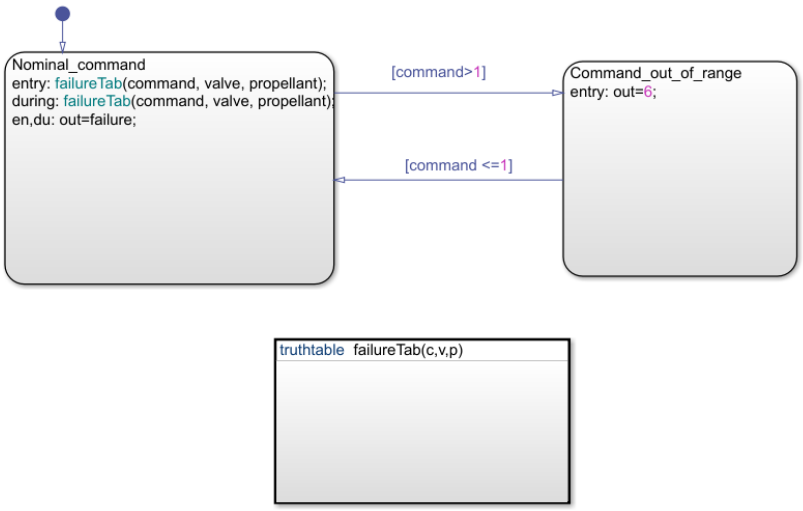

Figure C-1: Stateflow implementation of the thruster check

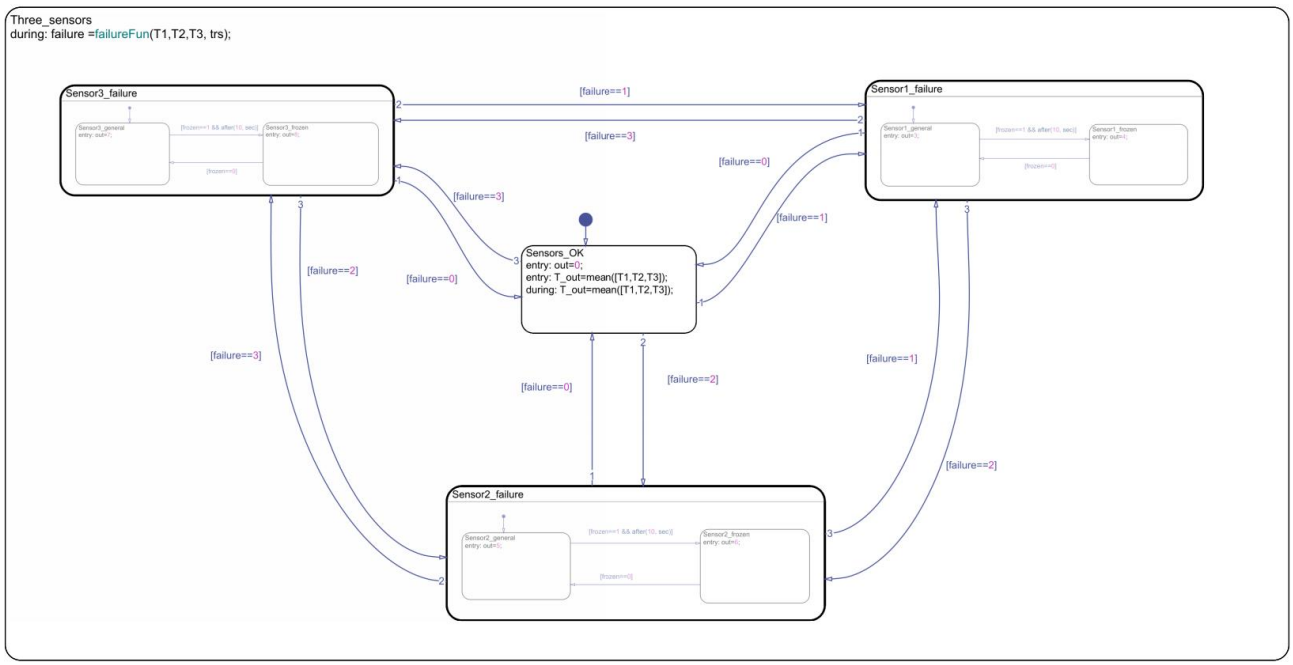

Figure C-2: Stateflow implementation of the cross-check between three temperature sensors

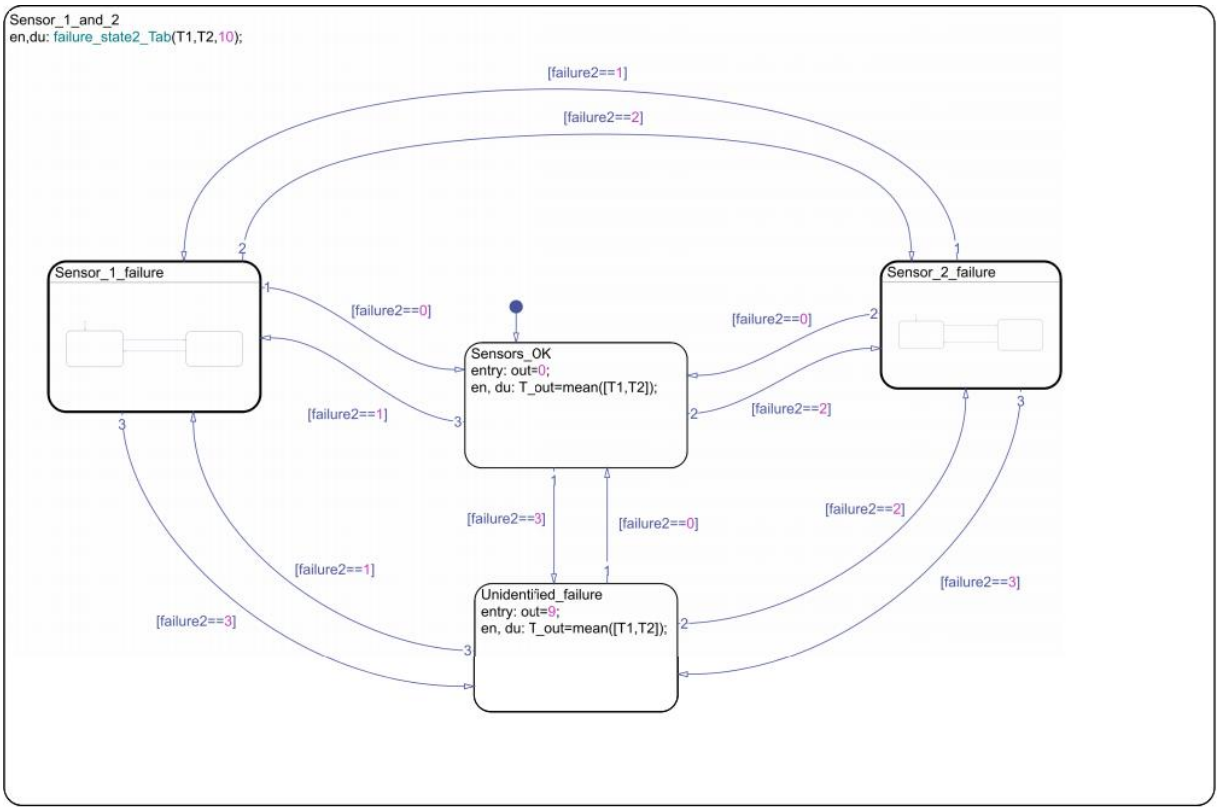

Figure C-3: Stateflow implementation of the cross-check between two temperature sensors

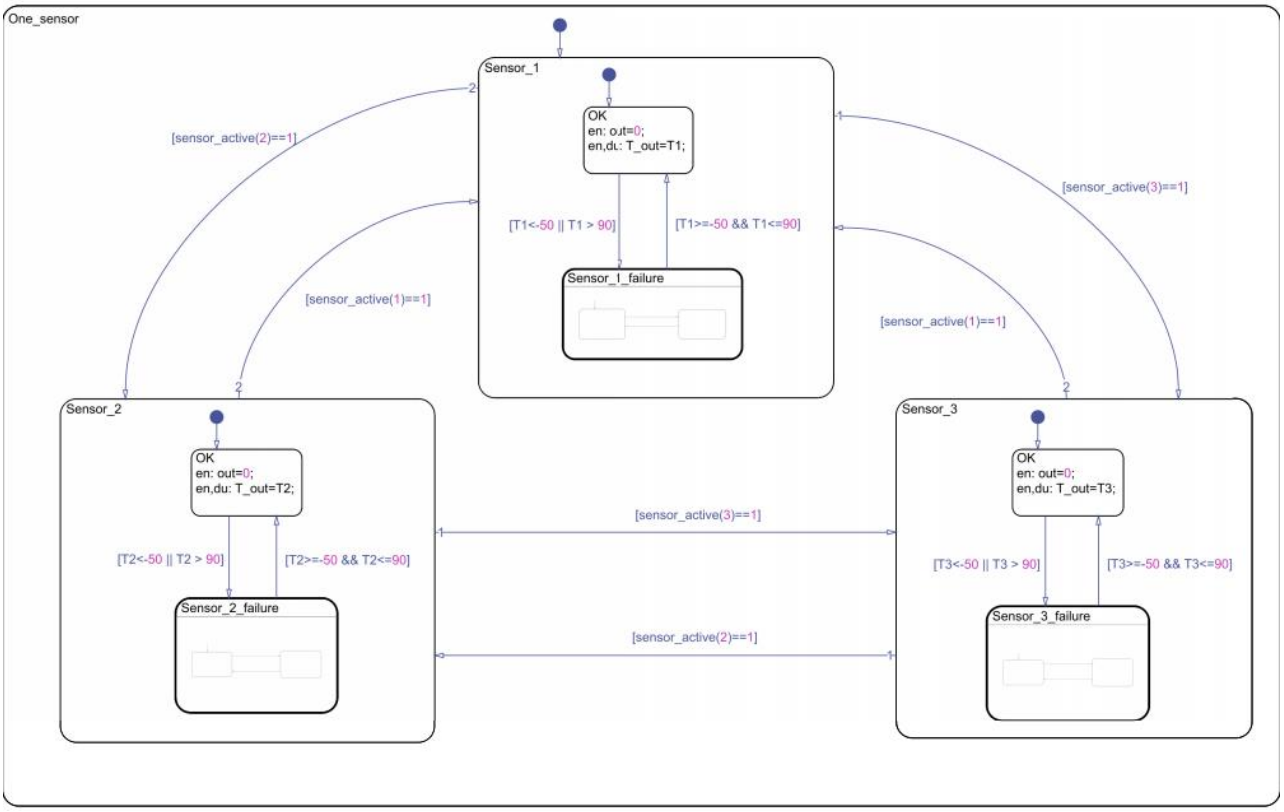

Figure C-4: Stateflow implementation of the check in case only one temperature sensor is available

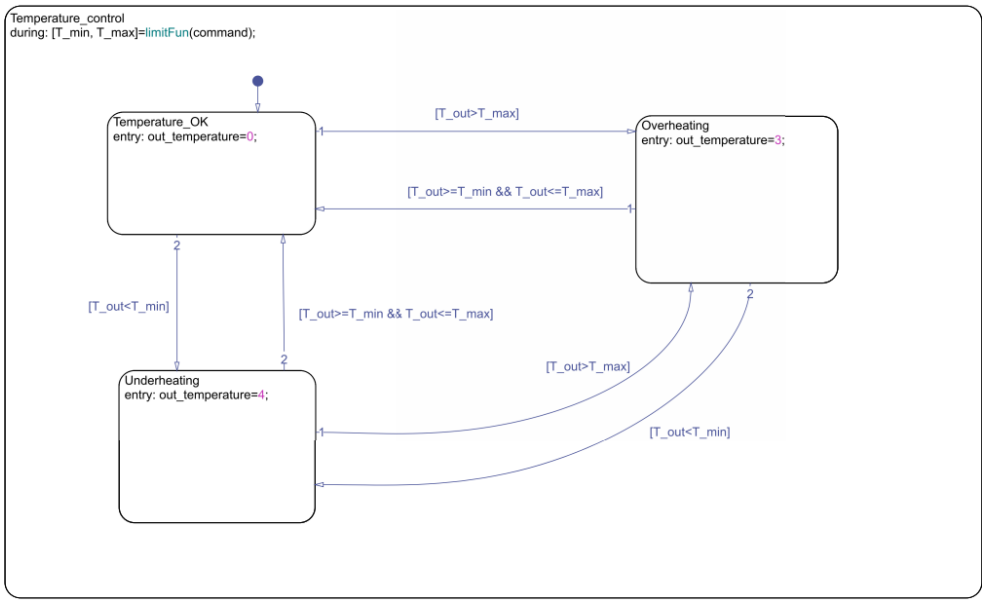

Figure C-5: Stateflow implementation of the temperature check for the Main Thruster FDI

C.2 Reaction Wheel module

In this Section, relevant screenshots of the Simulink model of the Reaction Wheel FDI, described in Section 3.3, are reported.

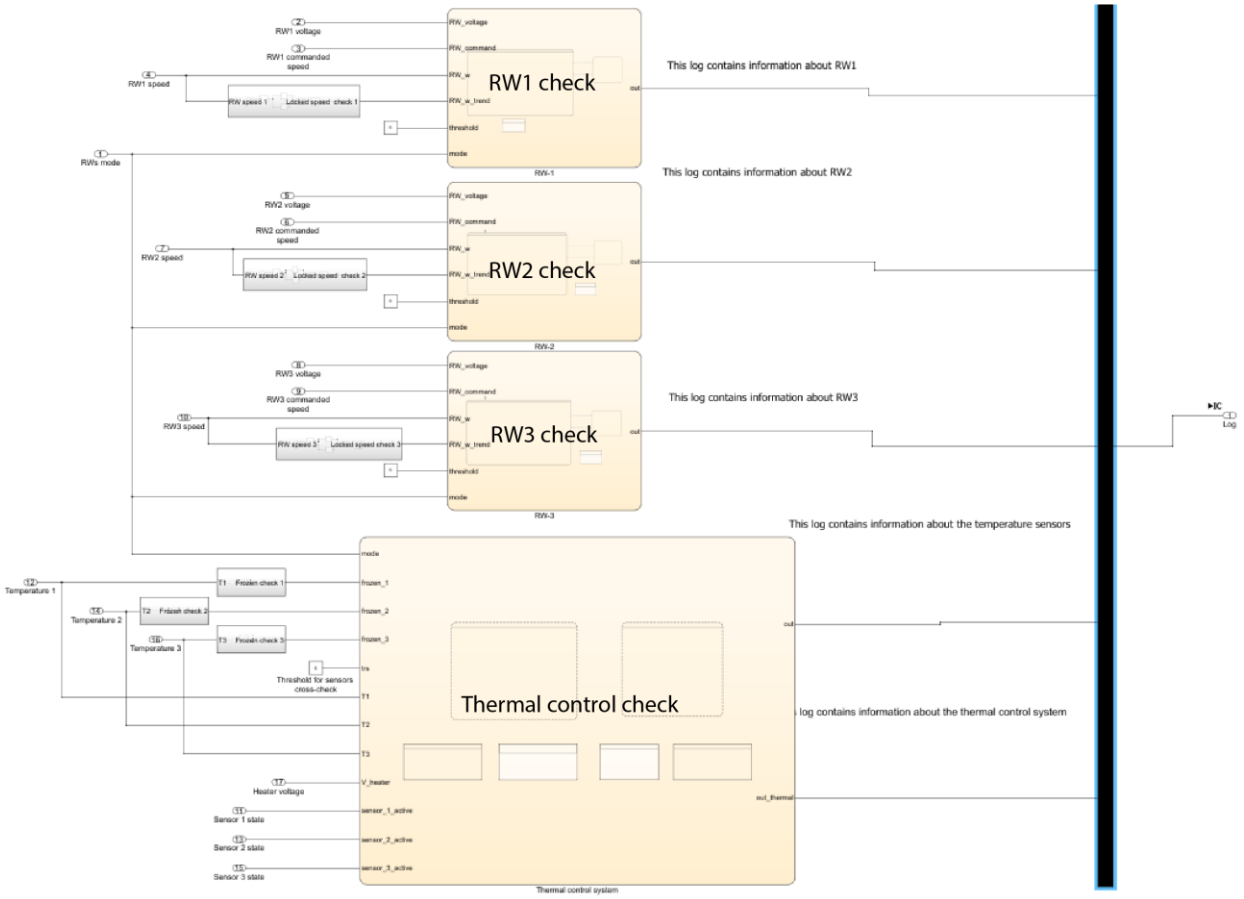

Figure C-6: high-level architecture of the checks in the RW FDI module

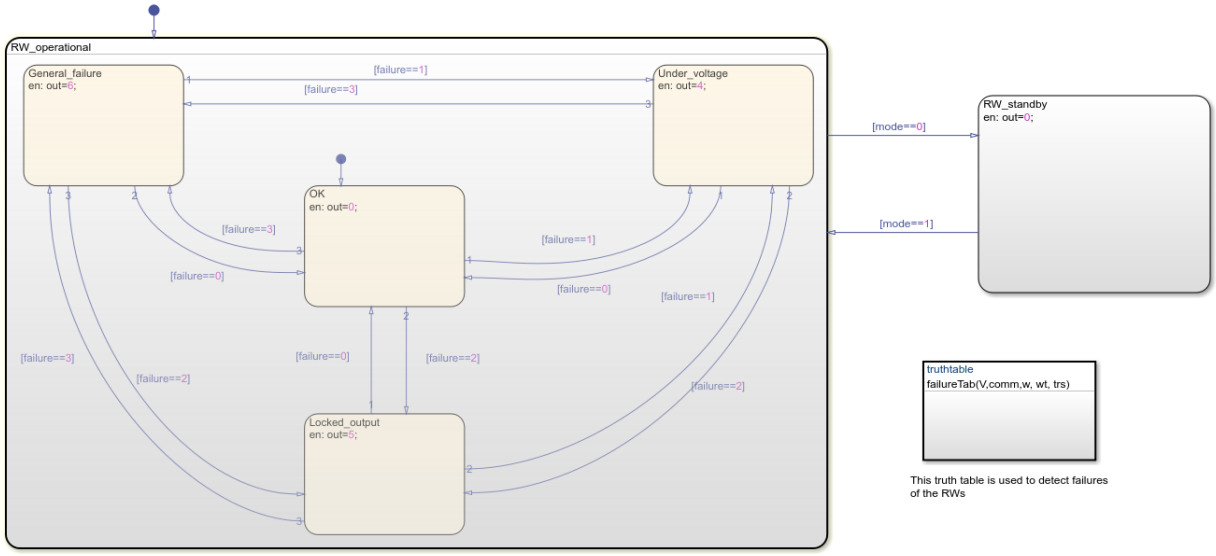

Figure C-7: Stateflow implementation of the RW check, in the RW FDI model

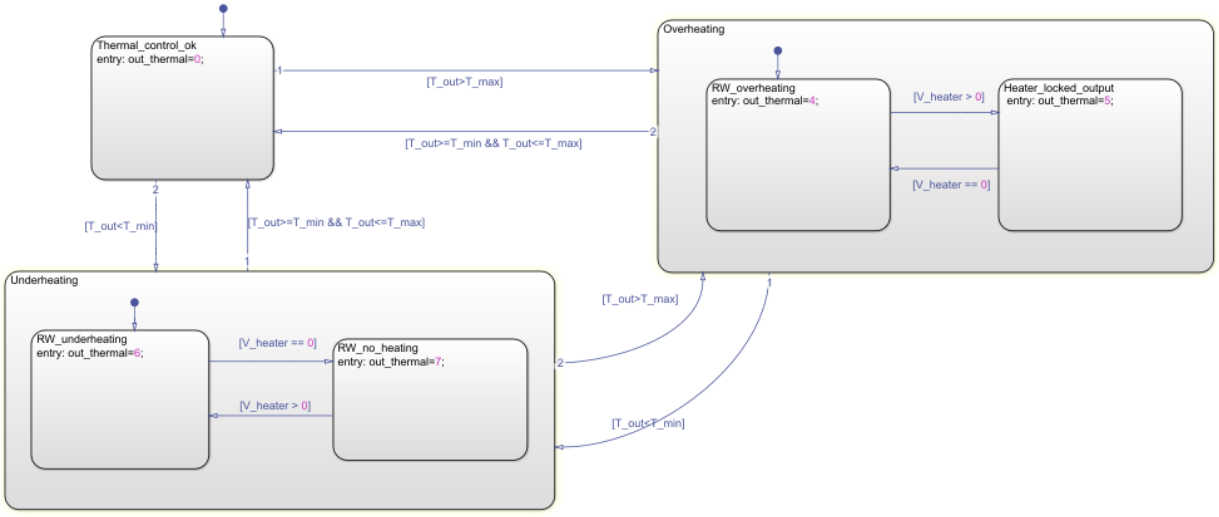

Figure C-8: Stateflow implementation of the temperature check in the RWs FDI model

C.3 Gas Thrusters module

In this Section, the relevant screenshot of the Simulink model of the Gas Thrusters FDI, described in Section 3.4, is reported.

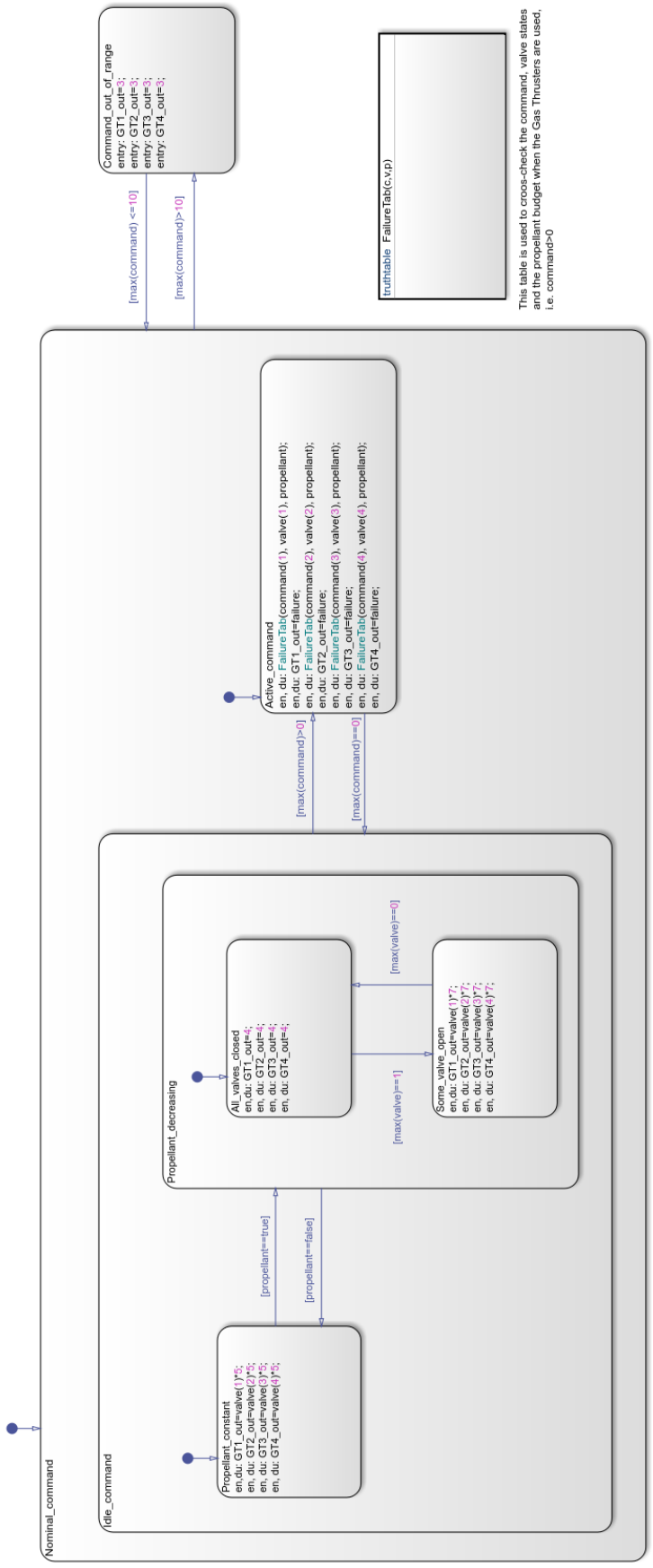

Figure C-9: Stateflow implementation of the thrusters check in the Gas Thrusters FDI model

C.4 Attitude determination module

In this Section, the relevant screenshots of the Simulink model of the Attitude determination FDI, described in Section 3.5, are reported.

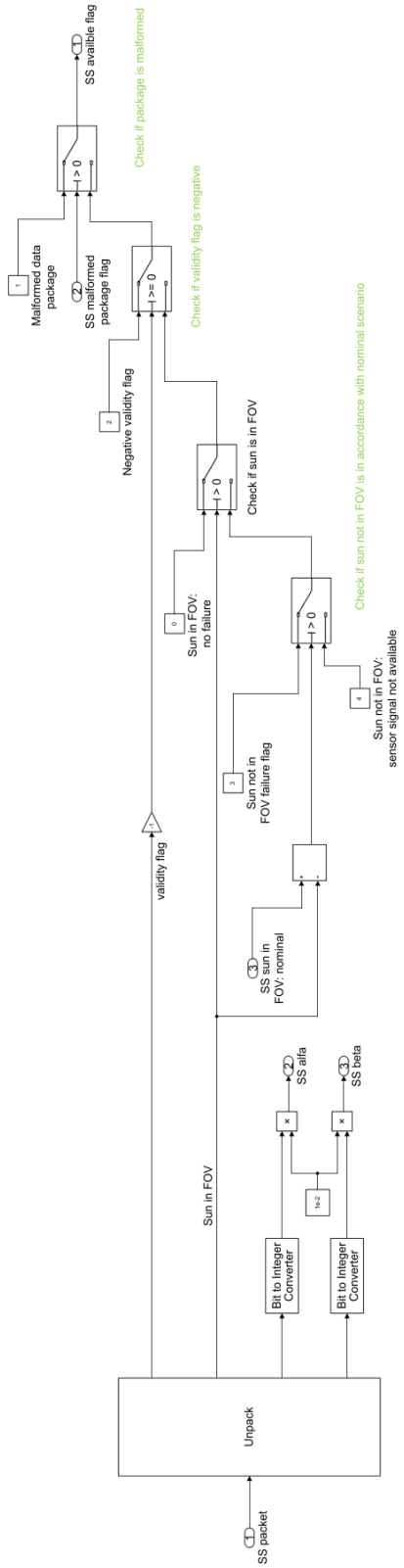

Figure C-10: Simulink implementation of the Sun sensor availability check, in the Attitude determination FDI

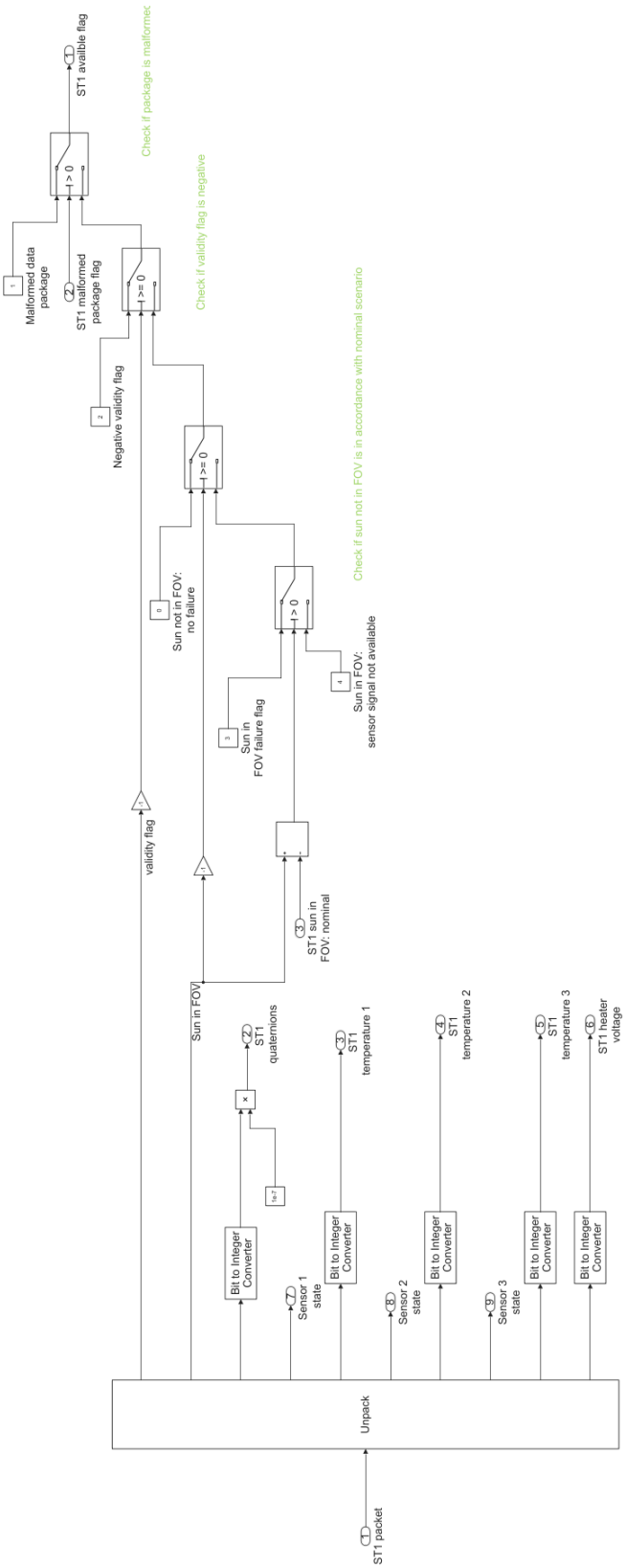

Figure C-11: Simulink implementation of the star-tracker availability check, in the Attitude determination FDI

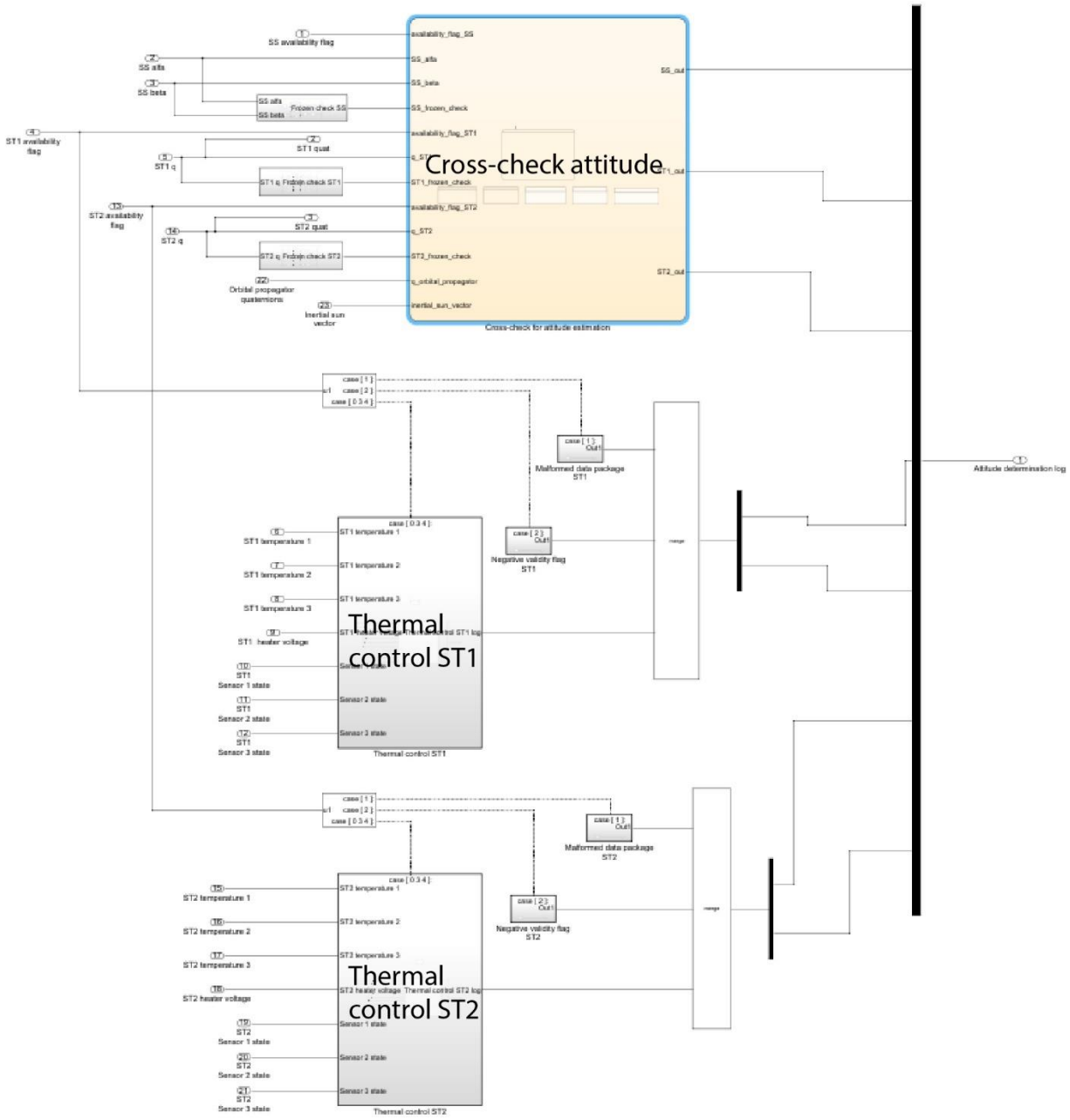

Figure C-12: high-level architecture of the checks in the Attitude determination FDI module

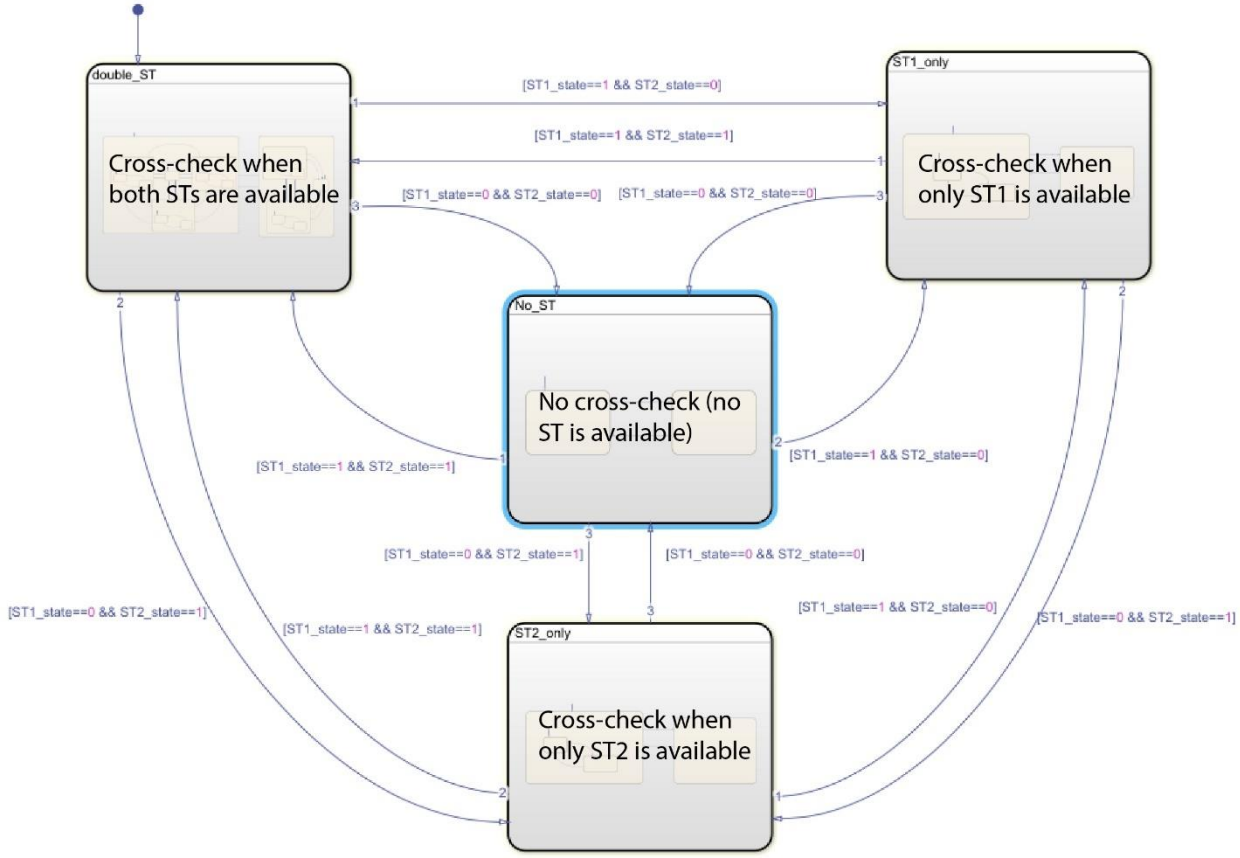

Figure C-13: high-level logic of the Stateflow chart that implements the cross-check between the attitude determination units

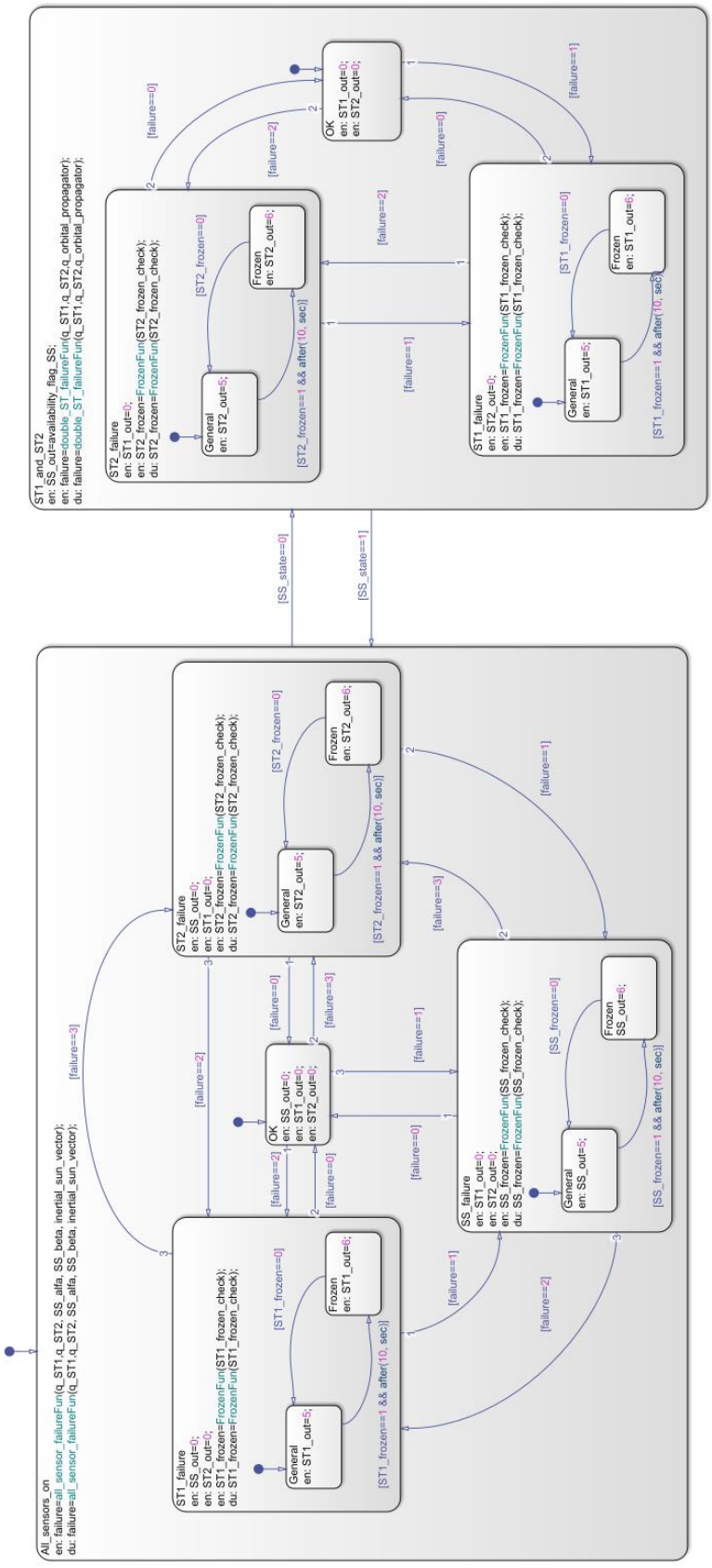

Figure C-14: Stateflow implementation of the cross-check when two star-trackers are available

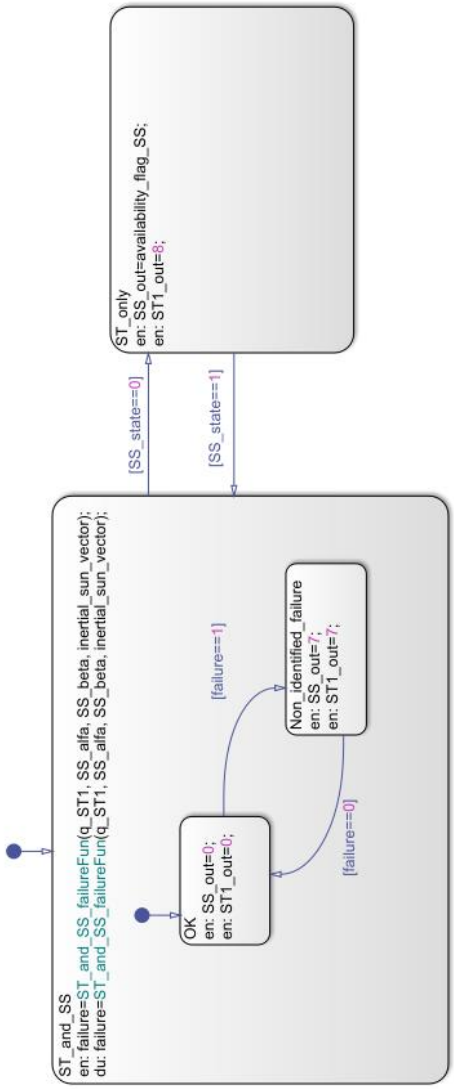

Figure C-15: Stateflow implementation of the cross-check in the case in which only one star-tracker is available

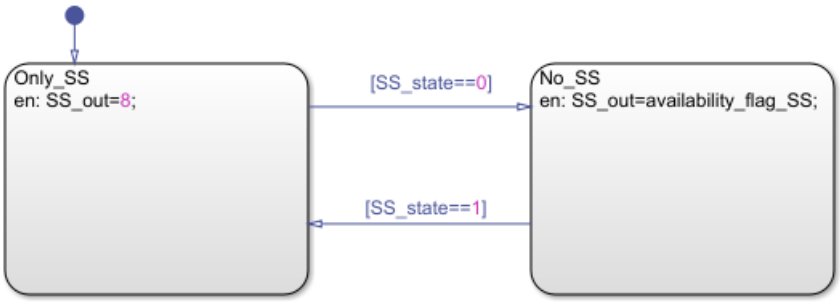

Figure C-16: Stateflow implementation of the cross-check when no star-tracker is available

C.5 IMU module

In this Section, the relevant screenshots of the Simulink model of the IMU FDI, described in Section 3.6, are reported.

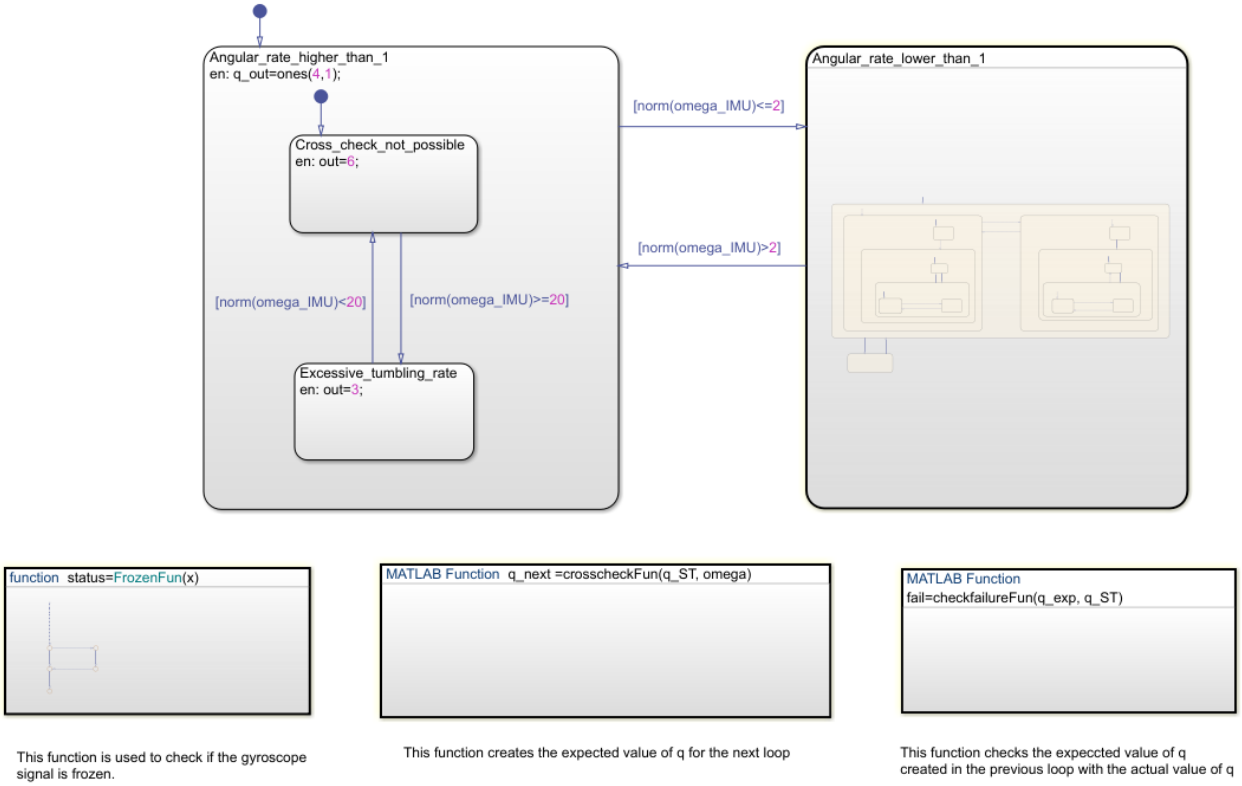

Figure C-17: Stateflow implementation of the IMU FDI

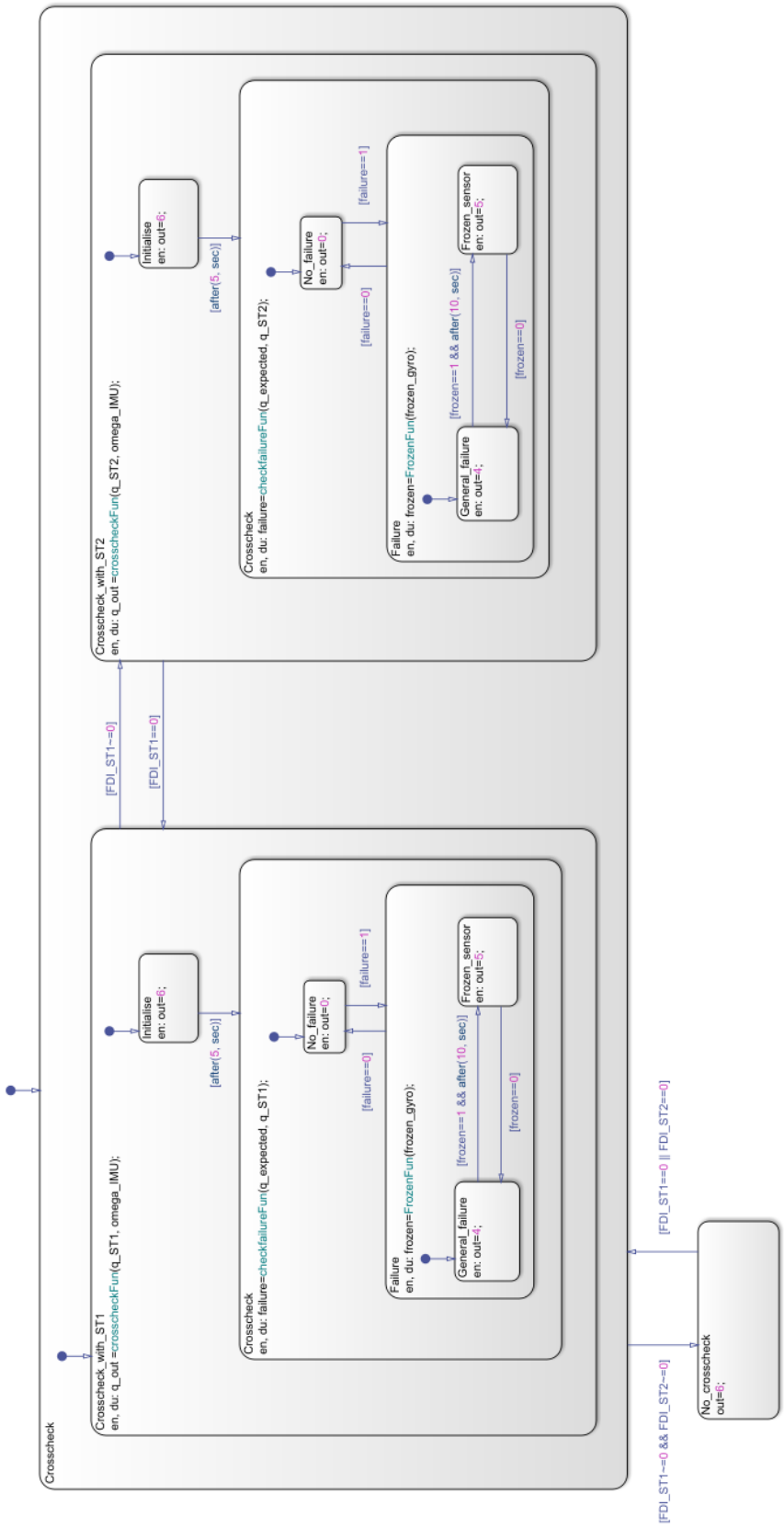

Figure C-18: Stateflow implementation of the cross-check between the IMU and the star-tracker

C.6 Power module

In this Section, the relevant screenshots of the Simulink model of the Power FDI, described in Section 3.7, are reported.

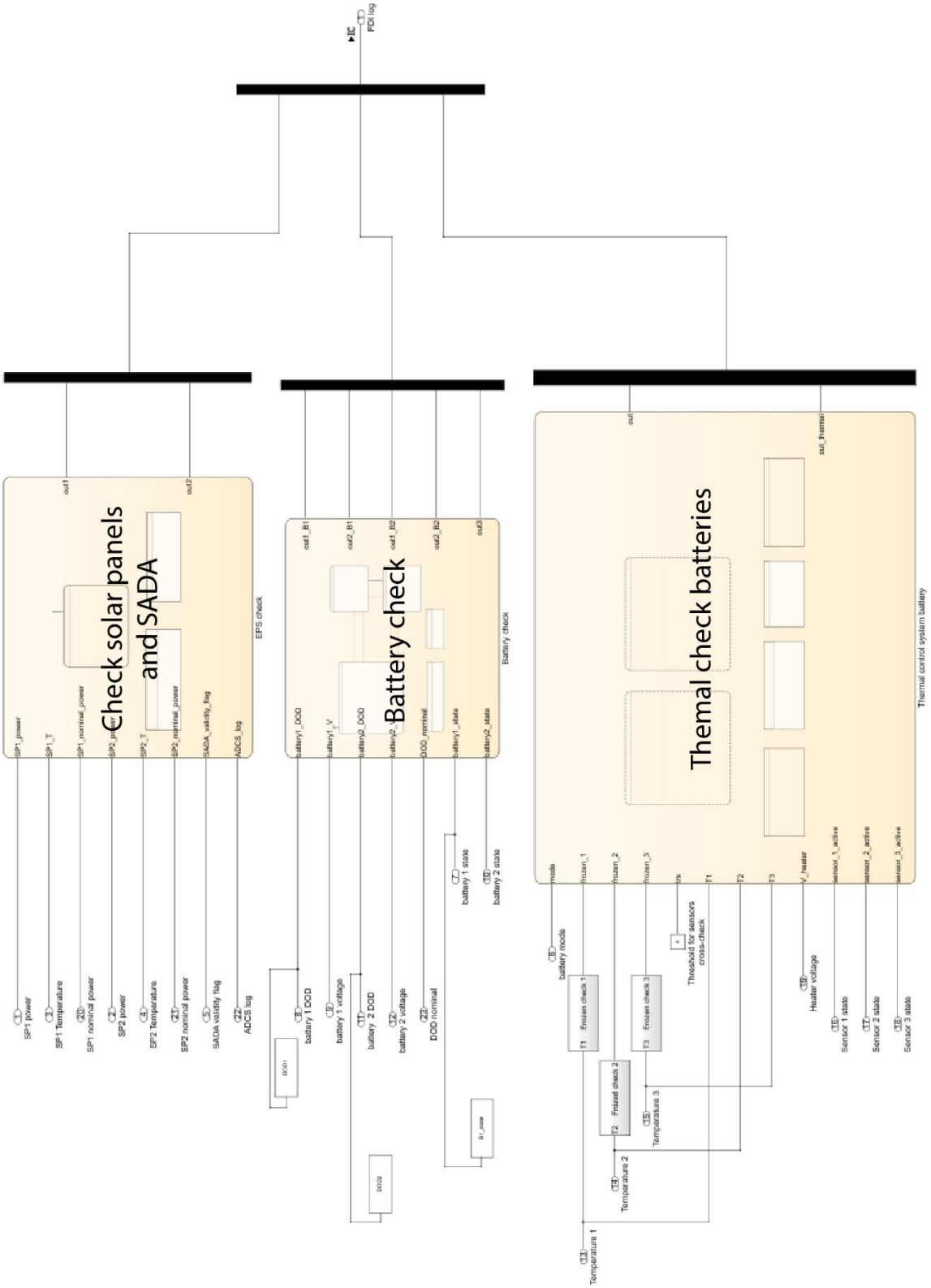

Figure C-19: high-level architecture of the checks in the Power FDI module

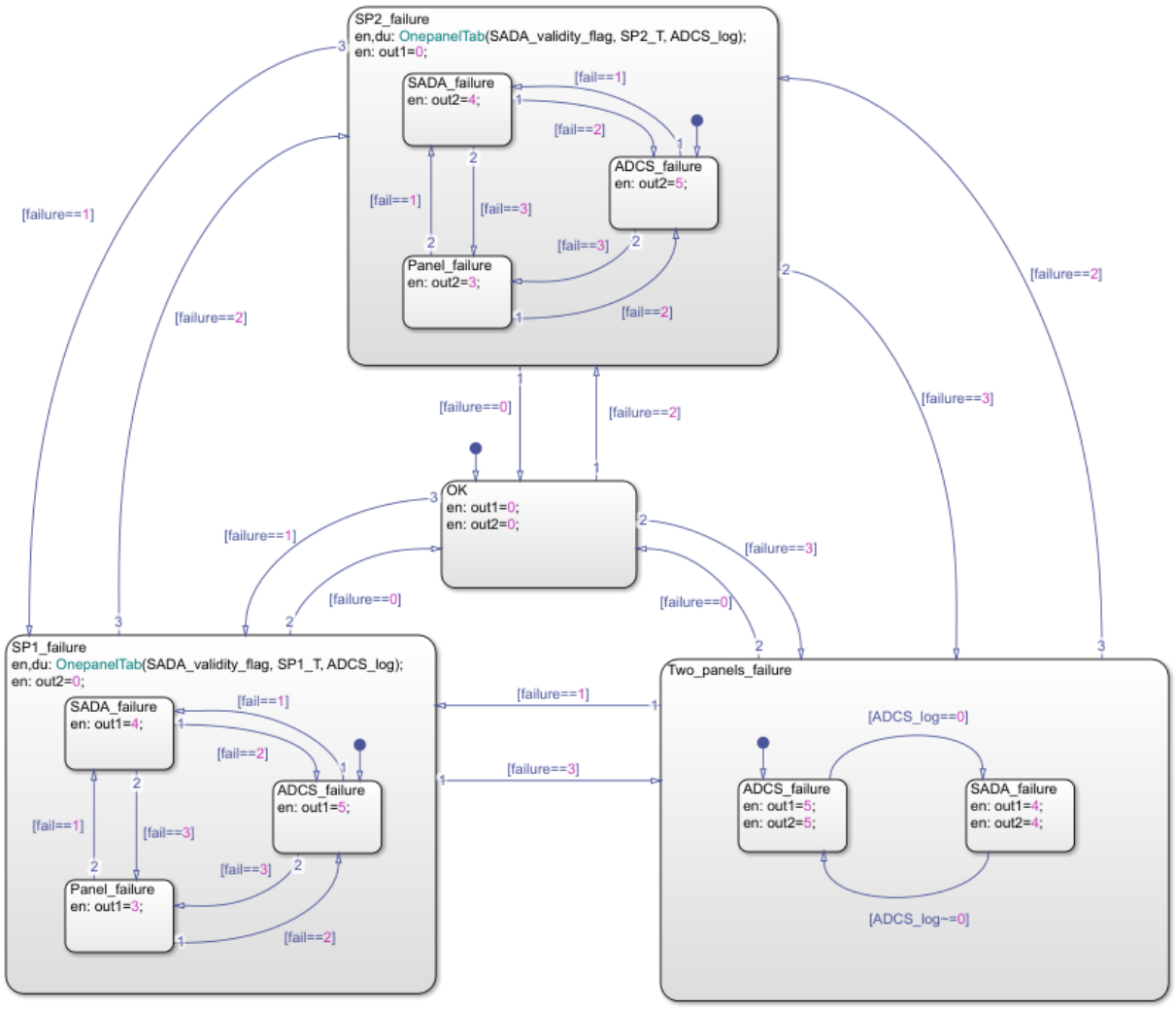

Figure C-20: Stateflow implementation of the solar panels and SADA check

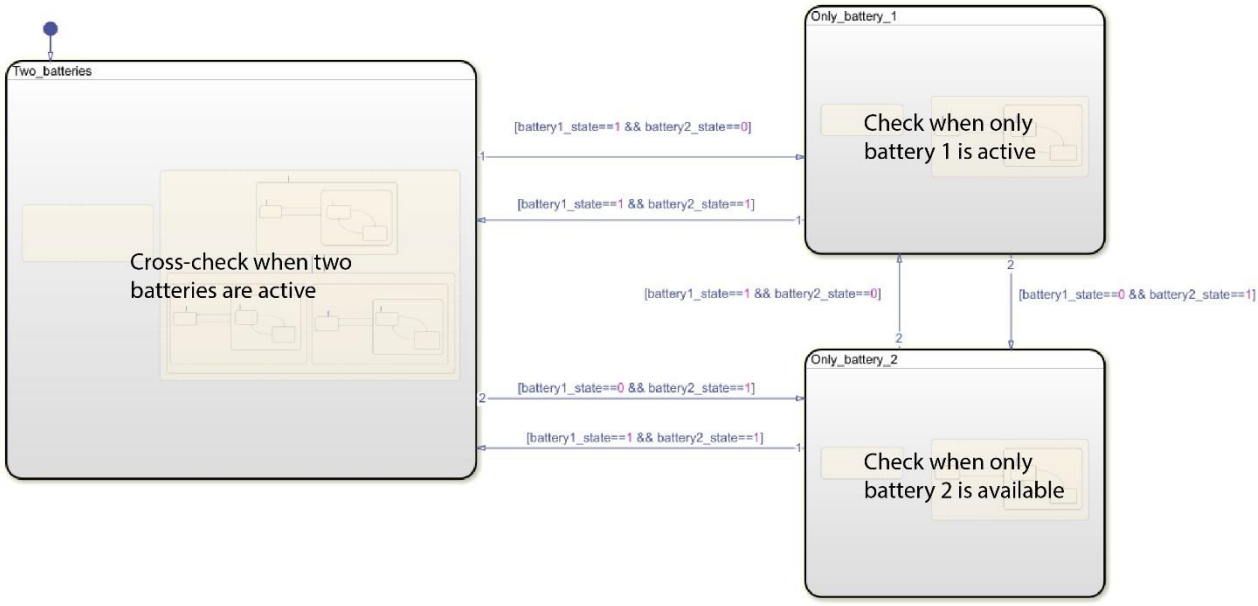

Figure C-21: high-level Stateflow implementation of the batteries FDI

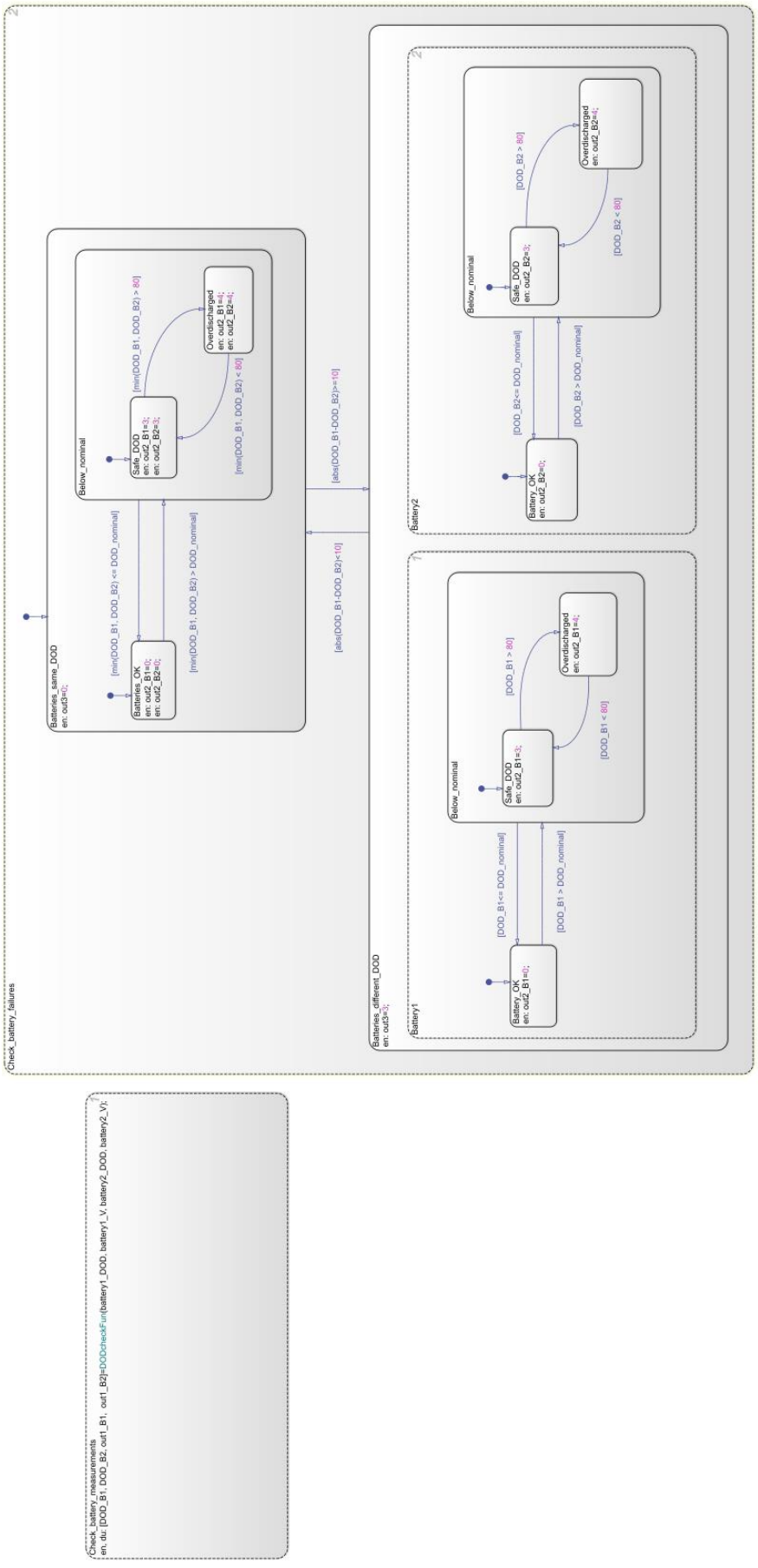

Figure C-22: Stateflow implementation of the battery FDI, when two batteries are available

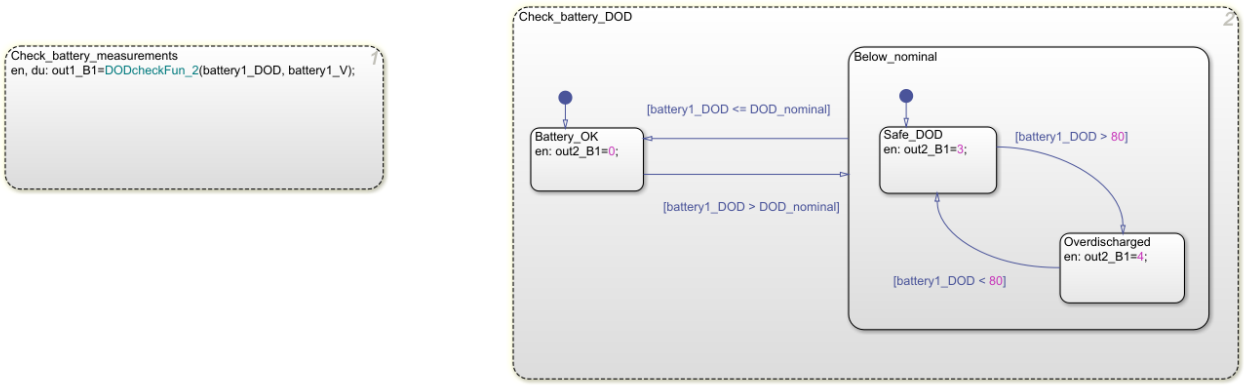

Figure C-23: Stateflow implementation of the battery FDI, when only one battery is available

C.7 Camera module

In this Section, the relevant screenshot of the Simulink model of the Camera FDI, described in Section 3.8, is reported.

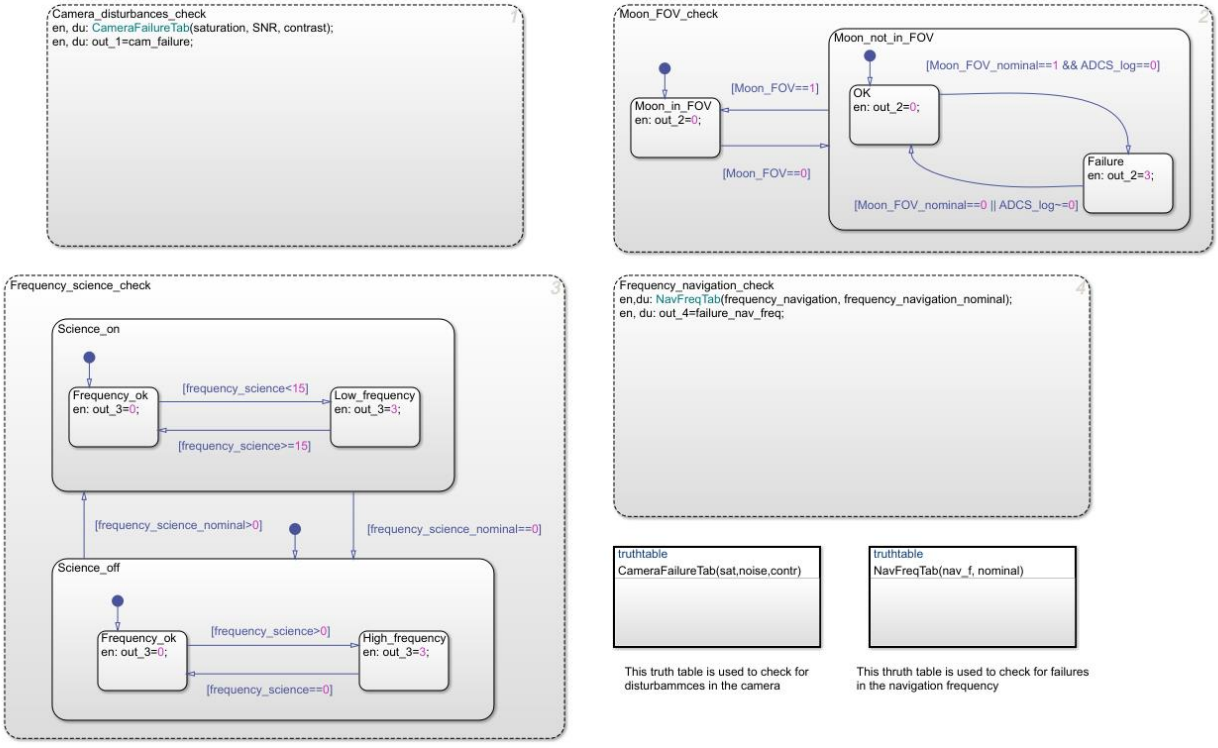

Figure C-24: Stateflow implementation of the Camera FDI

C.8 Deployment module

In this Section, the relevant screenshot of the Simulink model of the Deployment FDI, described in Section 3.9, is reported.

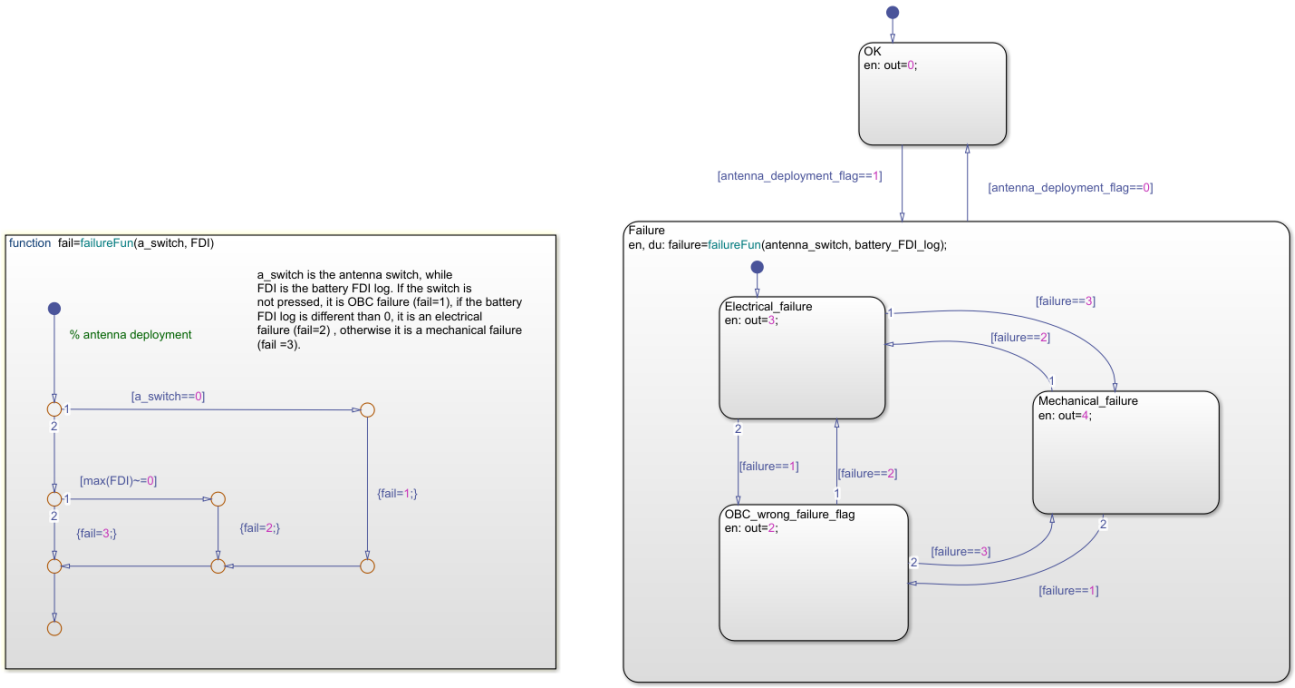

Figure C-25: Stateflow implementation of the Antenna Deployment check, in the Deployment FDI

C.9 Communication module

In this Section, the relevant screenshots of the Simulink model of the Communication FDI, described in Section 3.11, are reported.

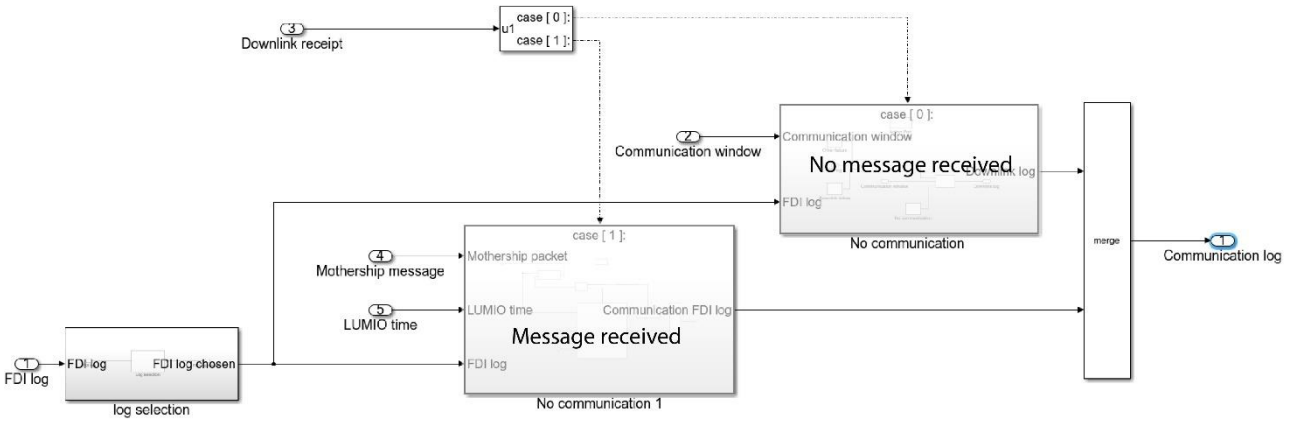

Figure C-26: Simulink model of the Communication FDI - detail 1: message receipt check

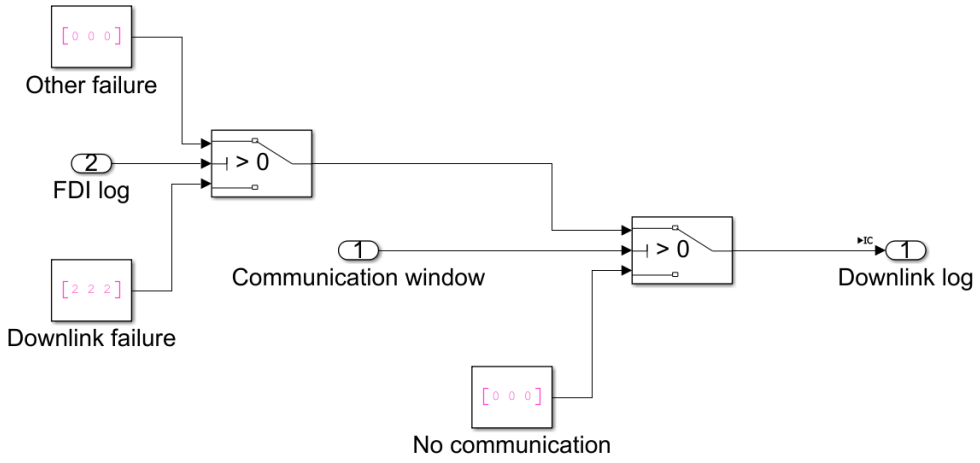

Figure C-27: Simulink model of the Communication FDI - detail 2: downlink check

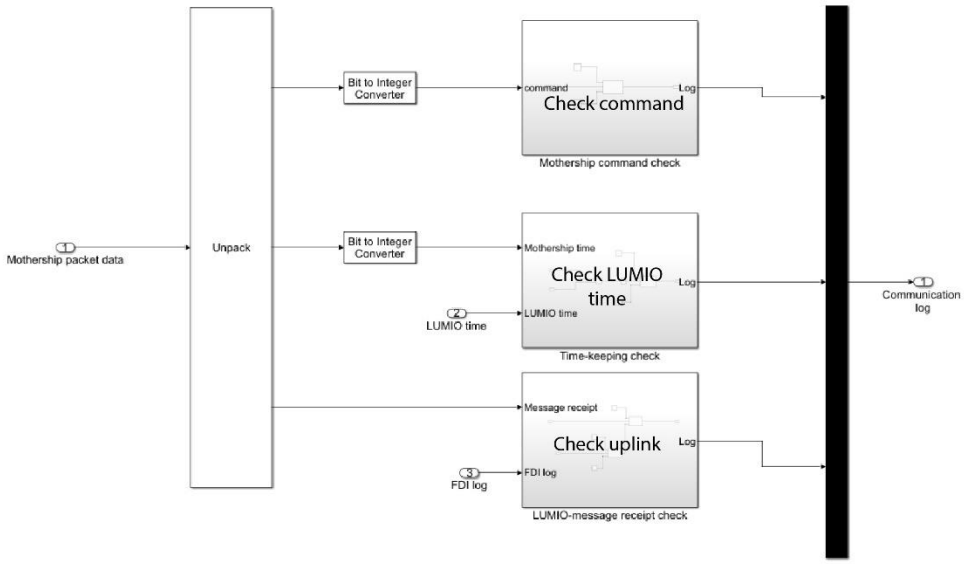

Figure C-28: Simulink model of the Communication FDI - detail 3: division of the Mothership package into three checks

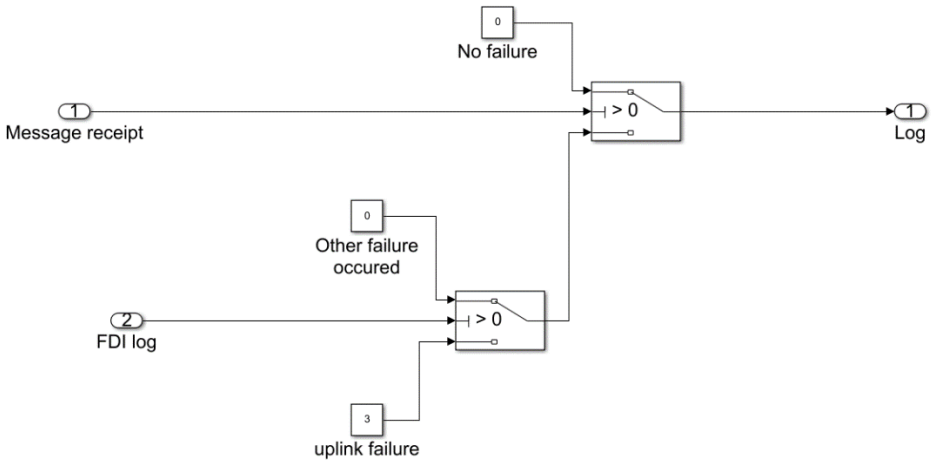

Figure C-29: Simulink model of the Communication FDI - detail 4: uplink check

D FR Stateflow model screenshots

In this Appendix, relevant screenshots of the Stateflow FR model will be documented, in support to the description made in Chapter 4. The Appendix is divided into sub-sections, each one dedicated to a particular module.

D.1 Reaction Wheels module

In this Section, the relevant screenshots of the Stateflow model of the Reaction Wheels FR, described in Section 4.3, are reported. They include the FR model for the Reaction Wheels and the model for the Heater FR.

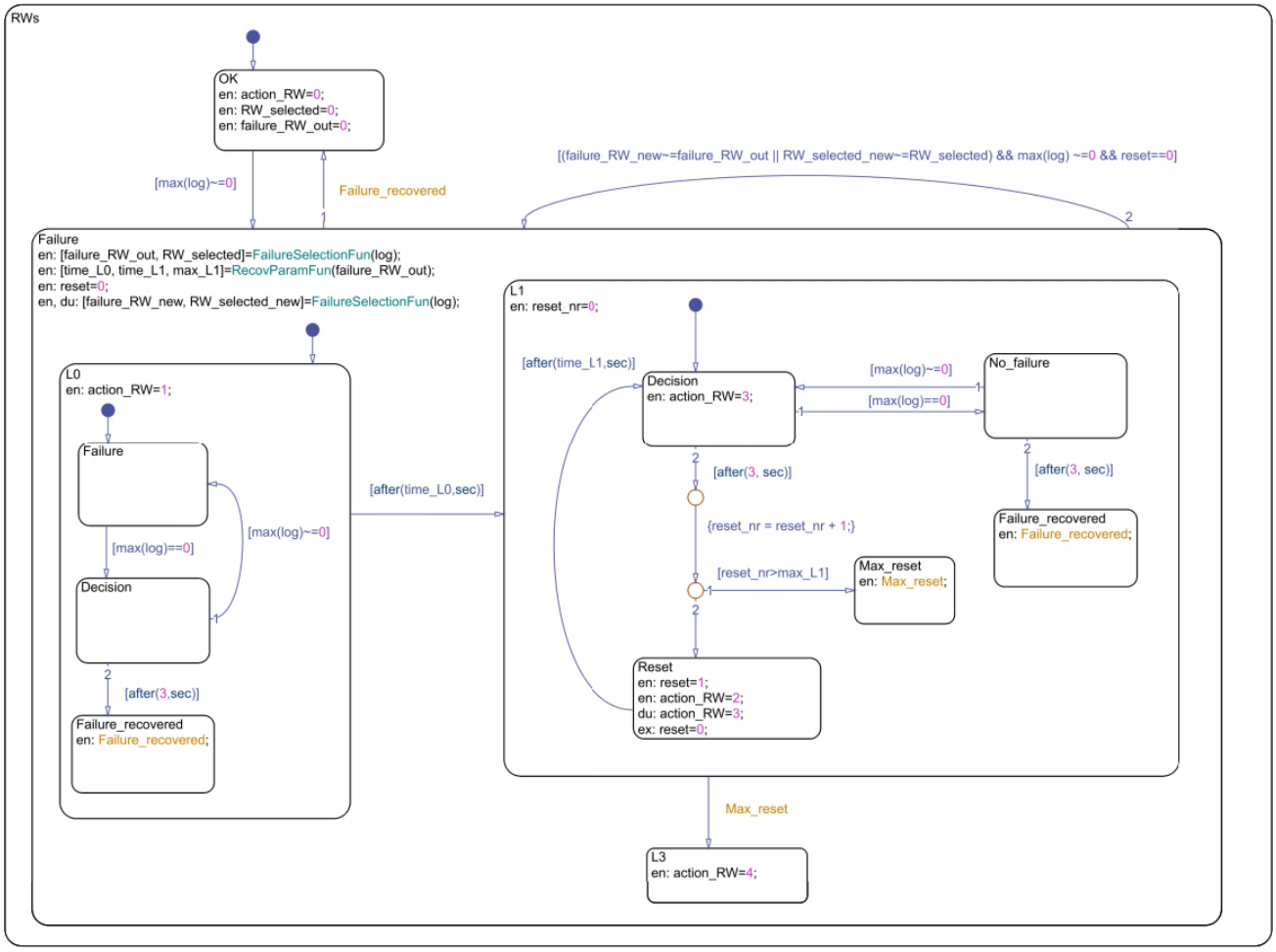

Figure D-1: Stateflow model of the Reaction Wheel FR – detail 1

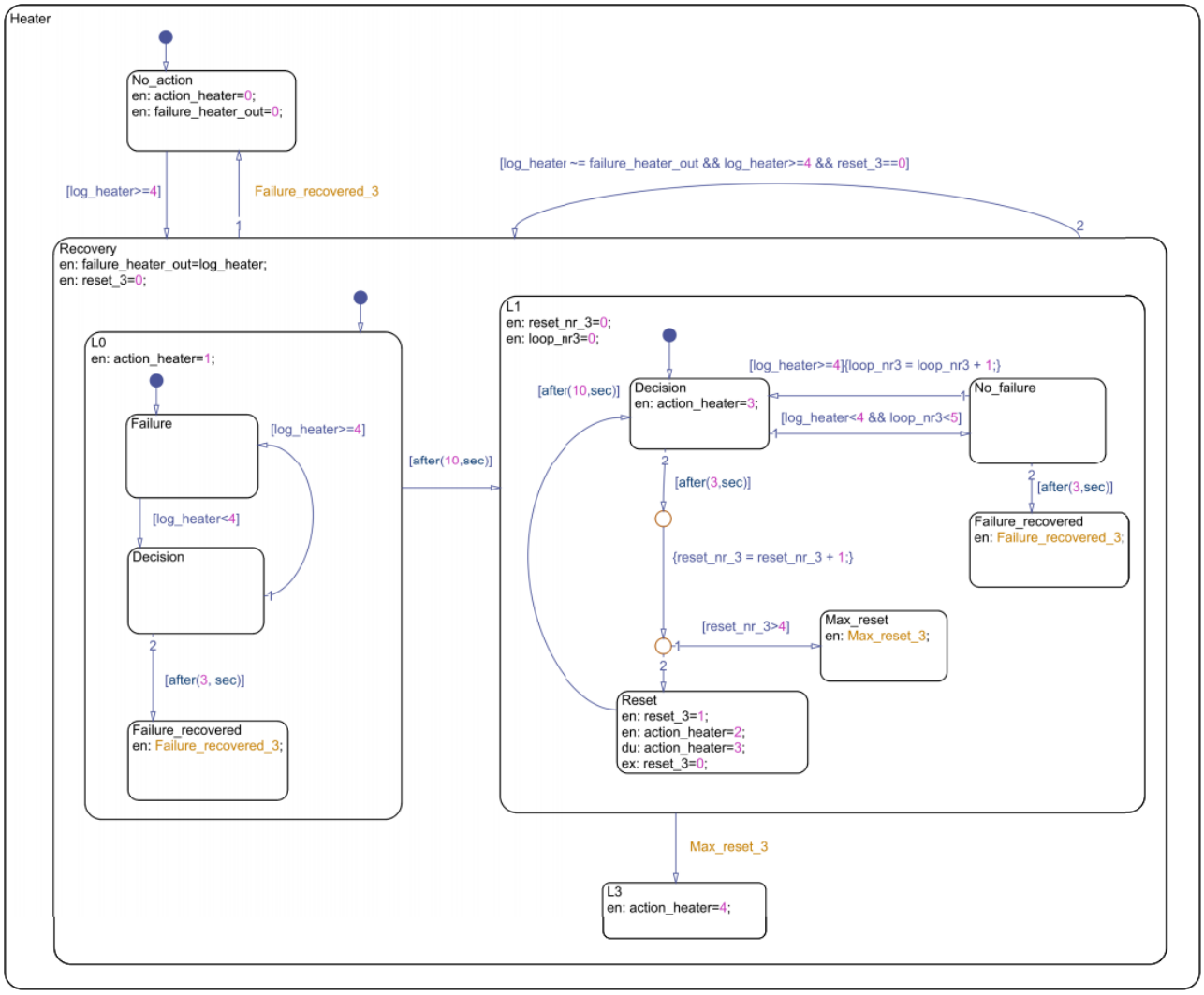

Figure D-2: Stateflow model of the Reaction Wheel FR – detail 2

D.2 Gas Thrusters module

In this Section, the relevant screenshot of the Stateflow model of the Gas thrusters FR, described in Section 4.4, is reported.

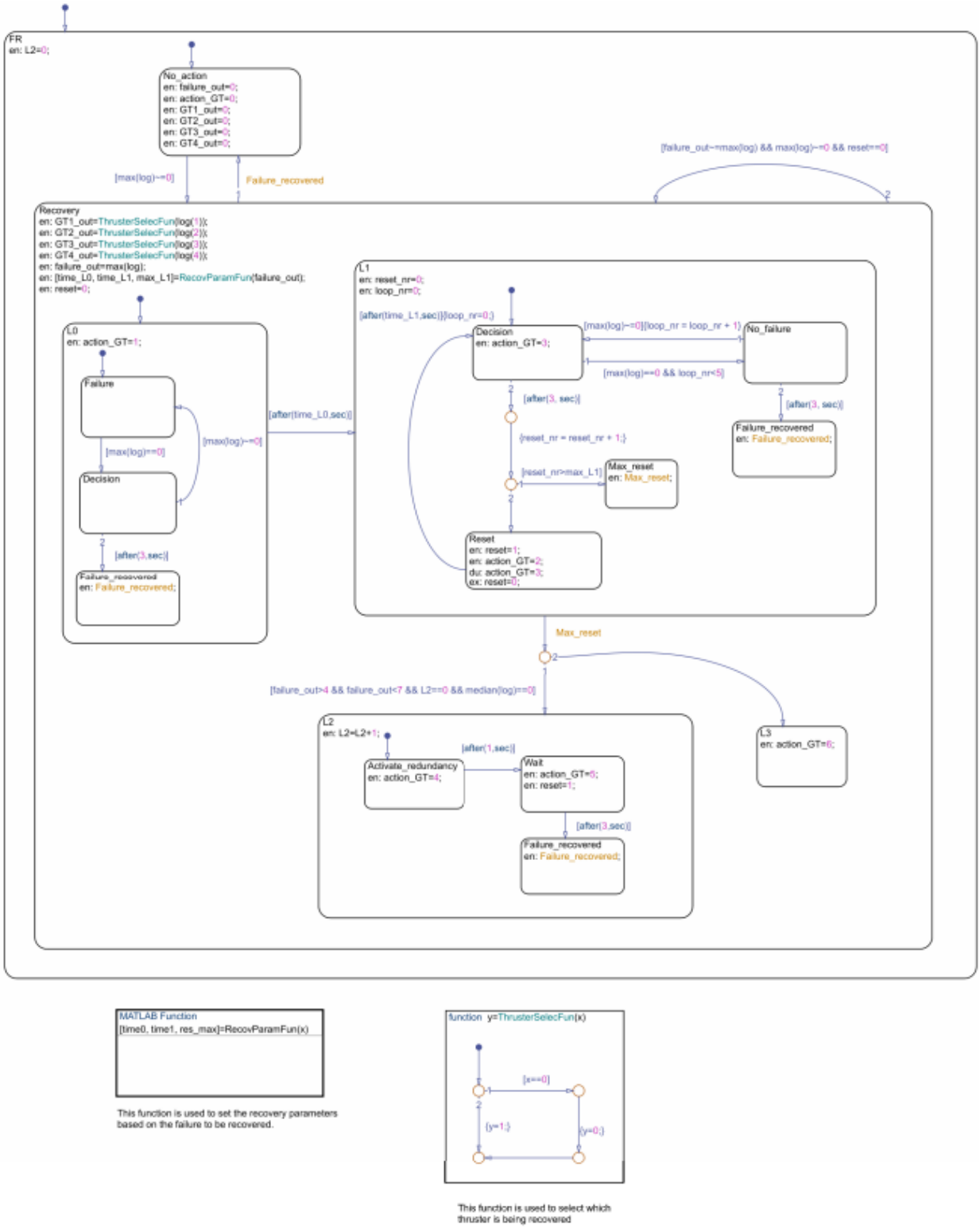

Figure D-3: Stateflow model of the Gas Thrusters FR

D.3 Attitude determination module

In this Section, the relevant screenshots of the Stateflow model of the Attitude determination FR, described in Section 4.5, are reported. They include the FR for the star-trackers and the FR for the Sun sensors.

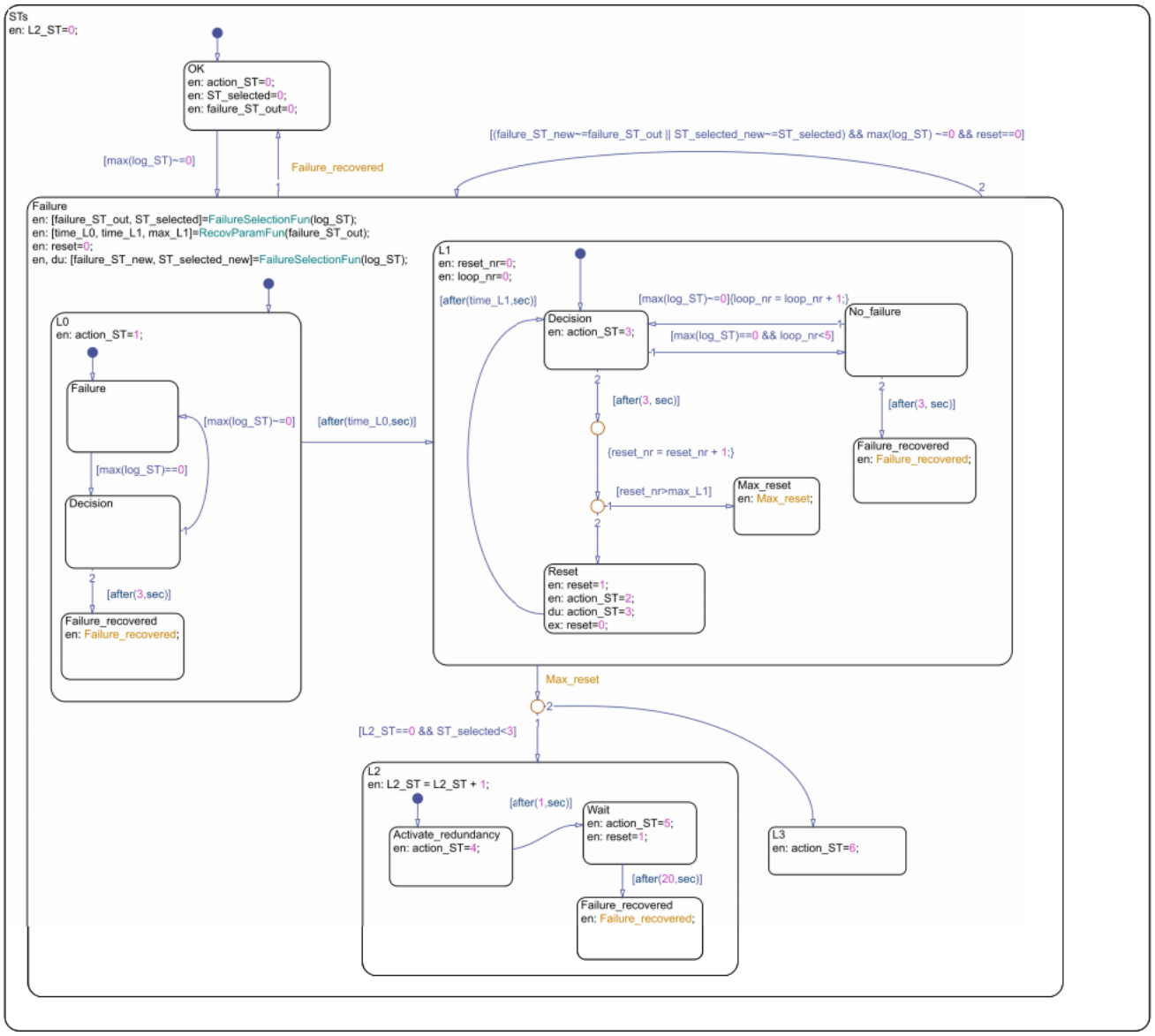

Figure D-4: Stateflow model of the star-tracker FR

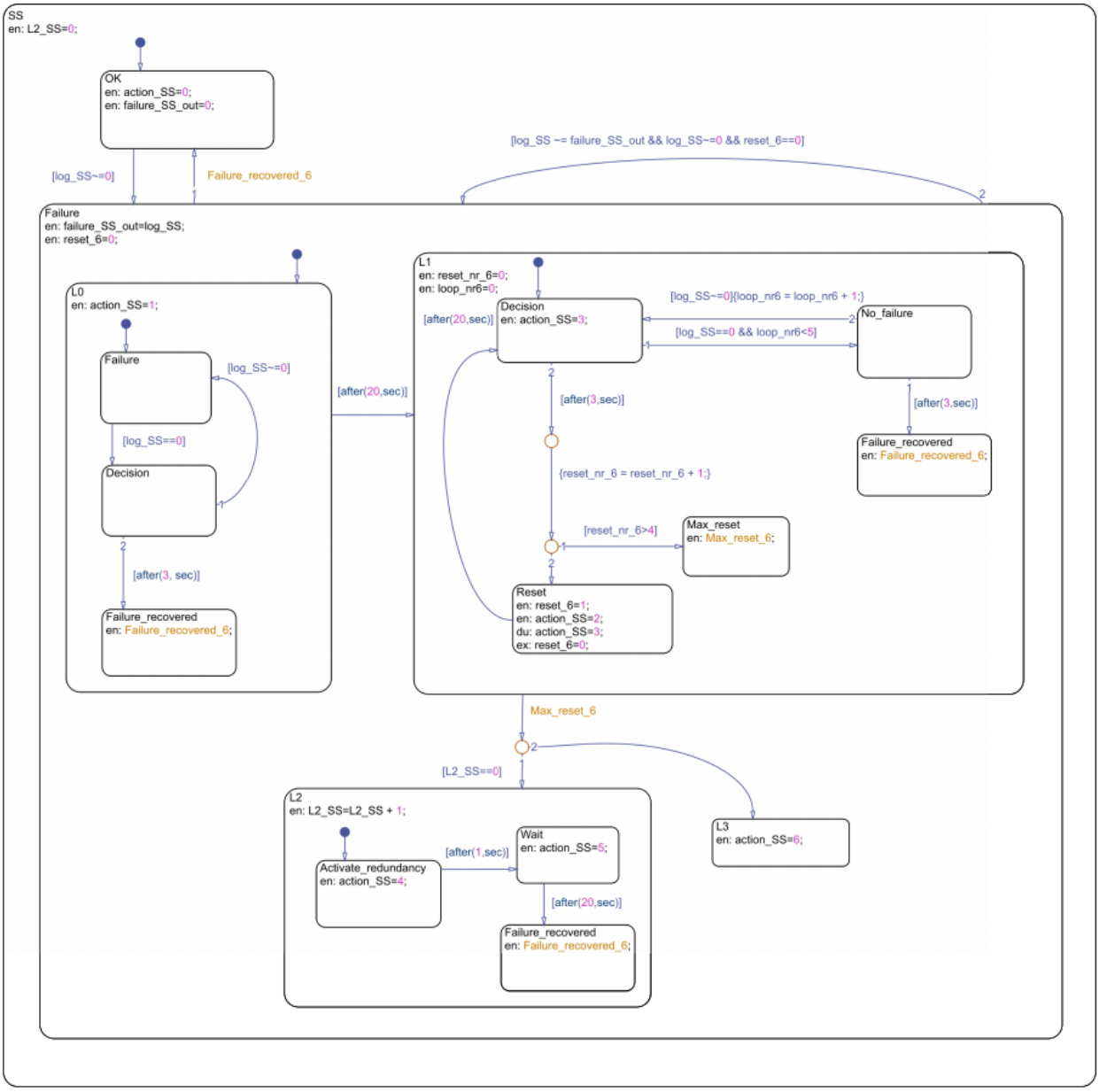

Figure D-5: Stateflow model of the Sun sensor FR

D.4 Power module

In this Section, the relevant screenshots of the Stateflow model of the Power FR, described in Section 4.7, are reported. They include the FR for the EPS, and the FR for the batteries.

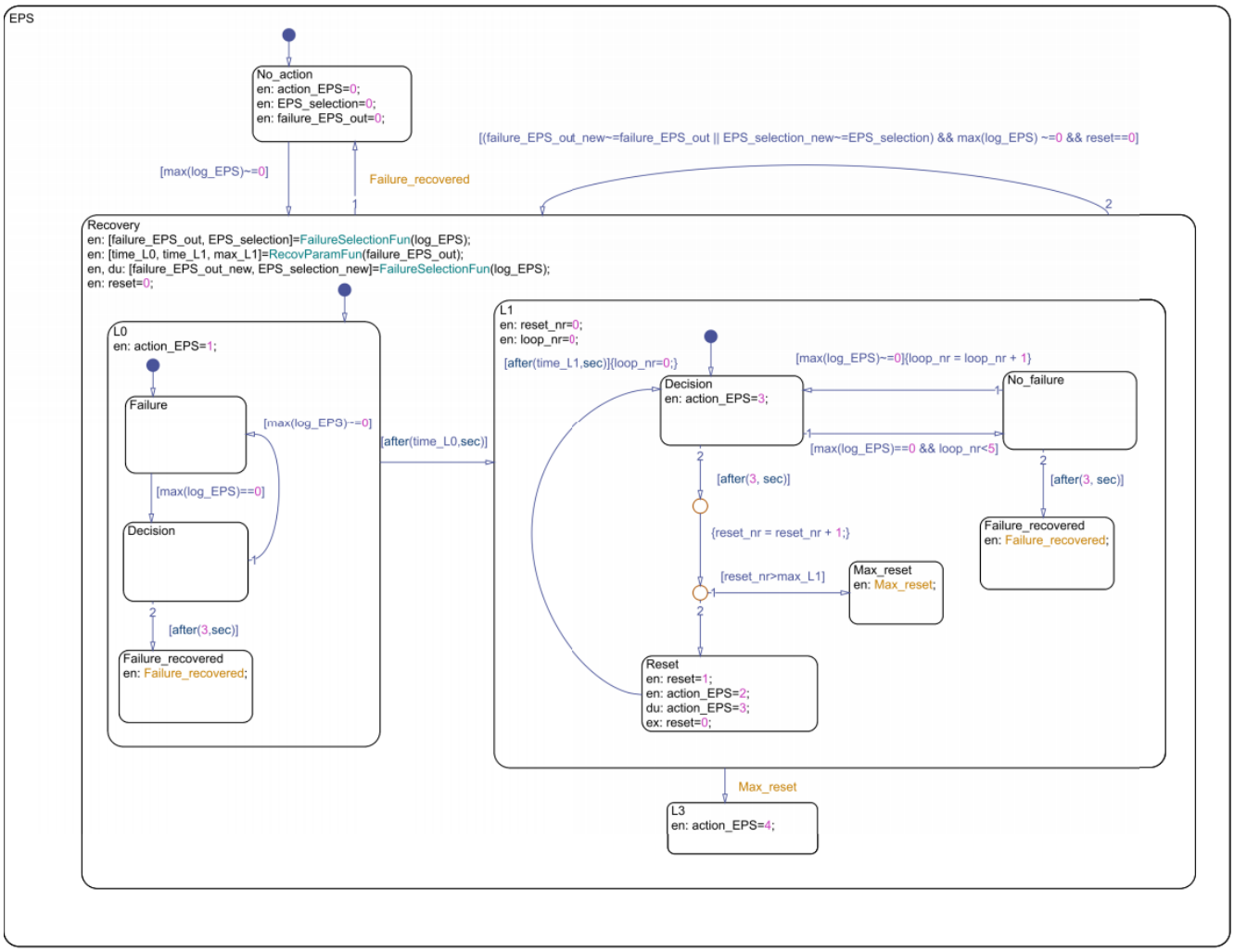

Figure D-6: Power Stateflow model - detail 1: EPS sub-module

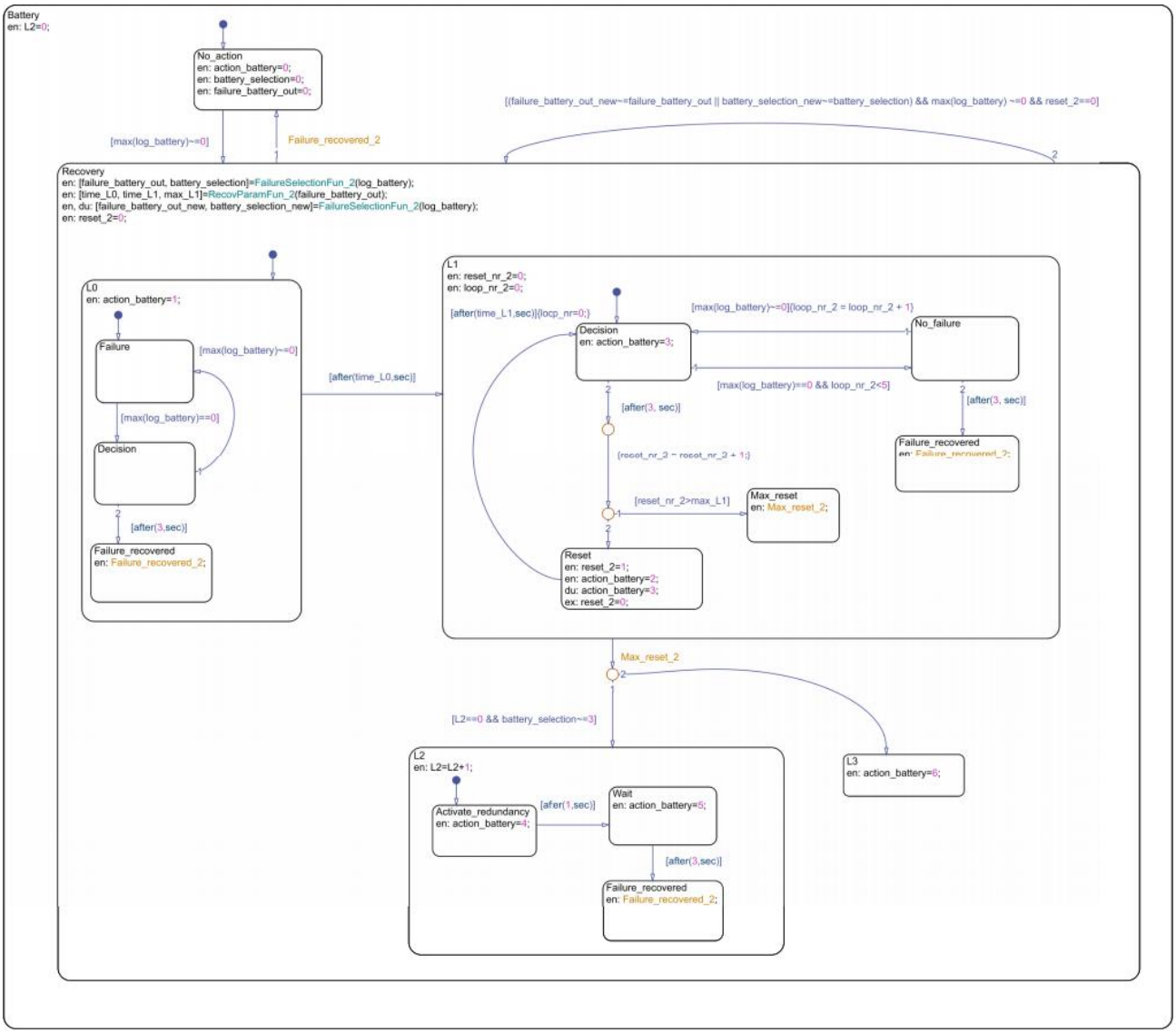

Figure D-7: Power Stateflow model - detail 2: battery sub-module

D.5 Deployment module

In this Section, the relevant screenshots of the Stateflow model of the Deployment FR, described in Section 4.9, are reported. They include the FR for the Antenna deployment and the FR for the Solar Panels Deployment.

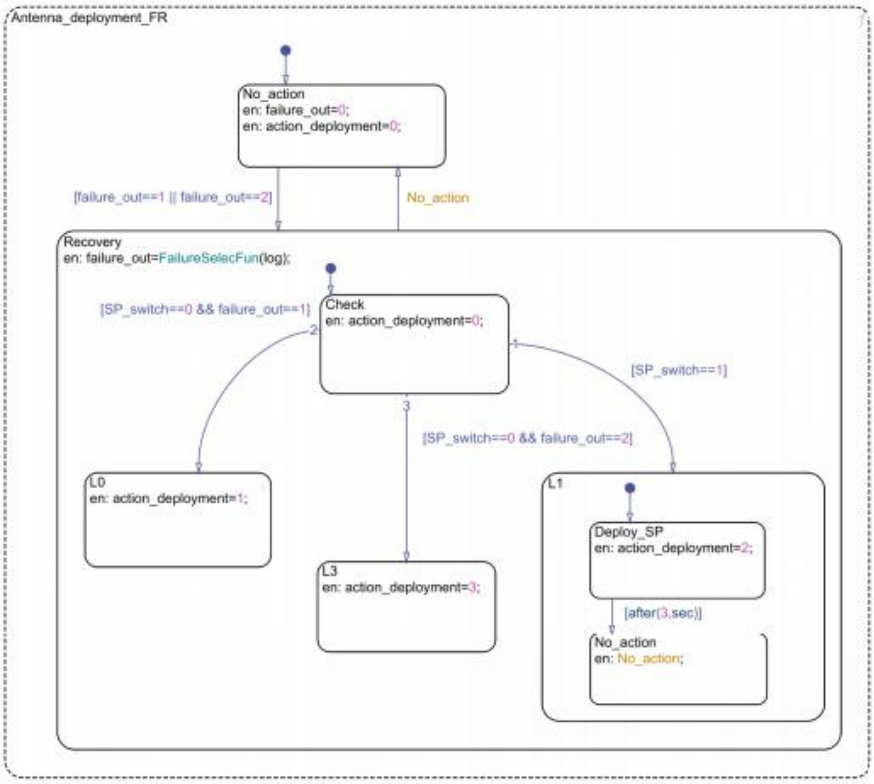

Figure D-8: Stateflow model of antennas deployment FR

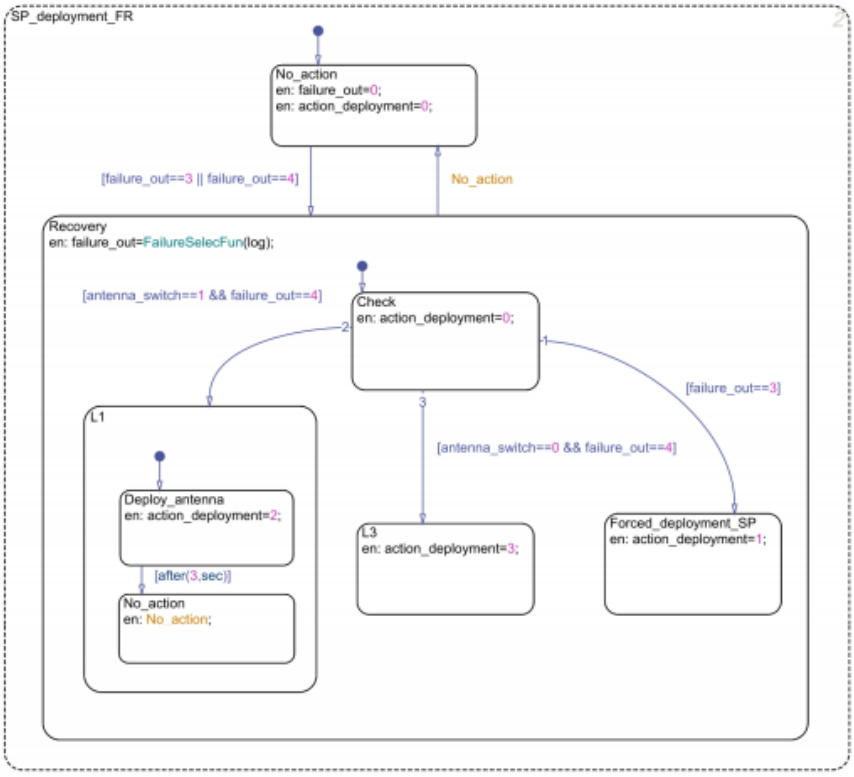

Figure D-9: Stateflow model of the solar panels deployment FR

D.6 Processors module

In this Section, the relevant screenshot of the Stateflow model of the Processors FR, described in Section 4.10, is reported.

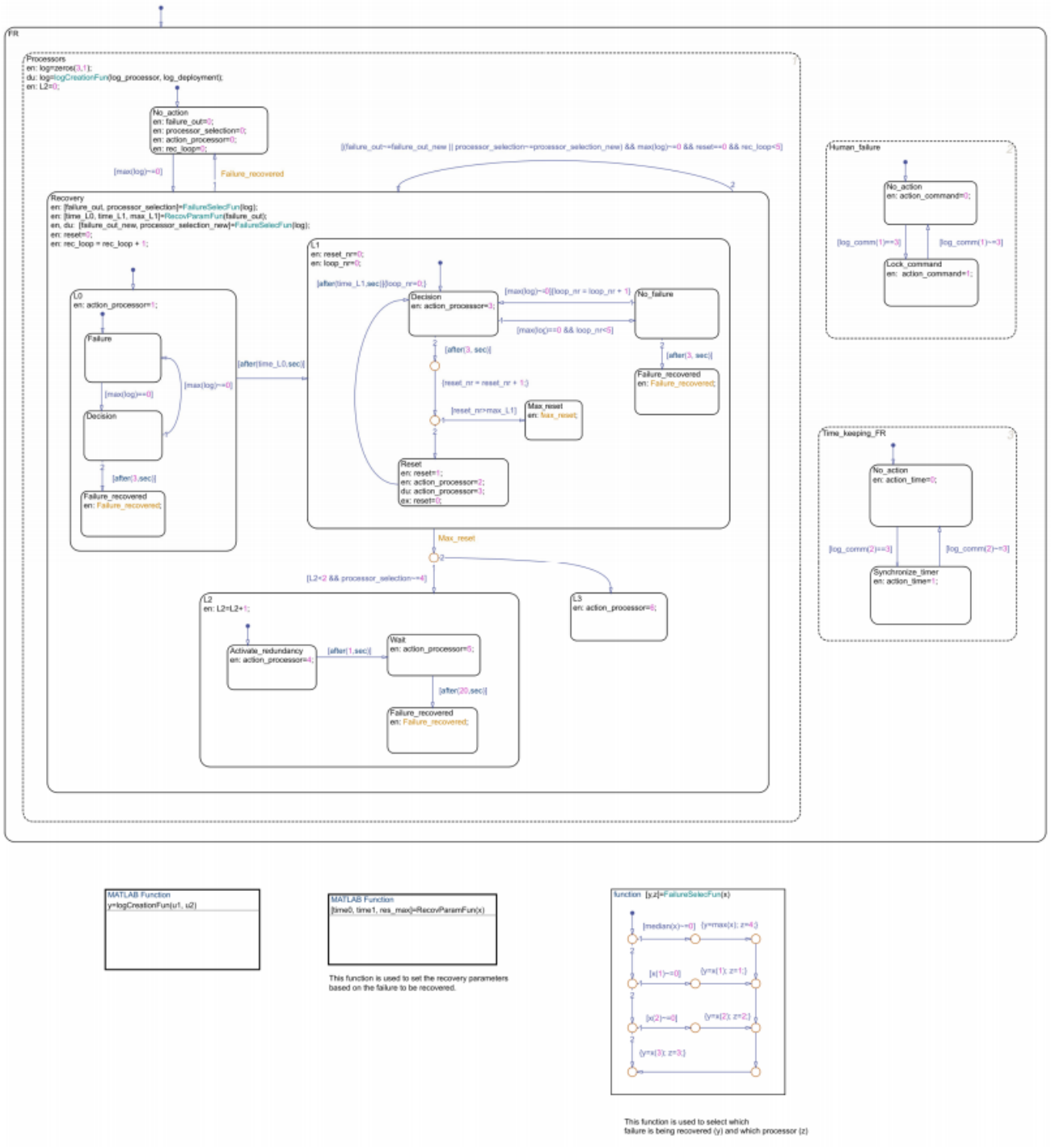

Figure D-10: Stateflow model of the processors FR

E Preliminary analysis on the gas thrusters redundancy

This Appendix is finalised at demonstrating the possibility of operating the system of gas thrusters in case one of them fails. The demonstration is considered preliminary because it is only aimed at showing that, in principle, the system is still able to create a 3D torque with only three thrusters; however, a further analysis, based on the expected values of torque to be produced, will be needed in the next phases of the project, to assess if the operations of LUMIO with only 3 gas thrusters will actually be feasible.

Let us have 4 gas thrusters in a pyramidal configuration. Let the thrusters be placed all in the same plane, parallel to plane x-y and at a distant *c* to plane z, as illustrated in Figure E-1. The base of the pyramid is assumed to be a rectangle with sides measuring *2a* and *2b*. To demonstrate how many thrusters are necessary to produce the desired torque, it is necessary to compute the total torque produced by the system, as the sum of the torques produced by each thruster.

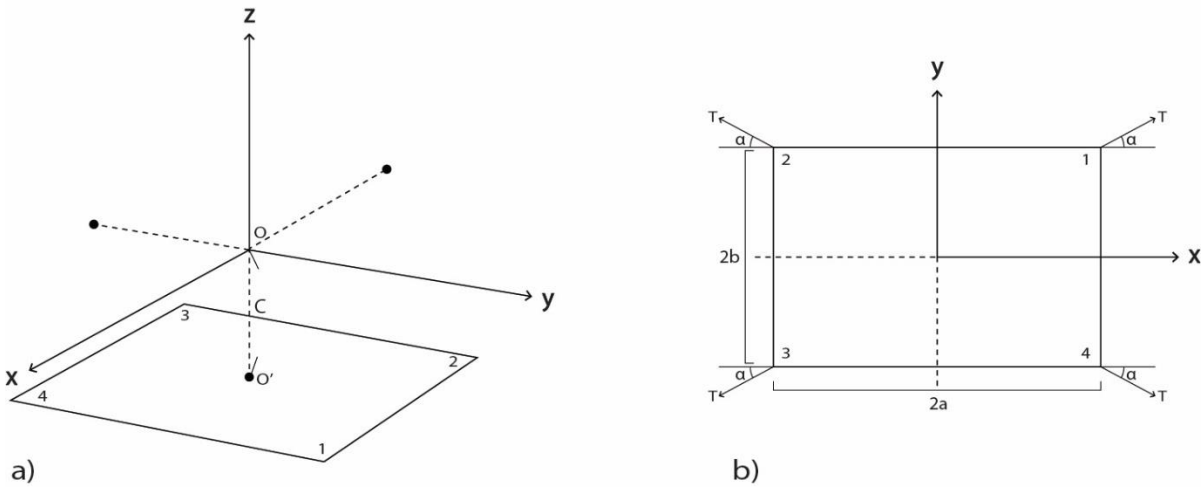

Figure E-1: position of the 4 gas thrusters in the selected reference frame (a) and orientation of the thrust vectors (b)

The torque *M* created by a thruster around point O, which is assumed to be the centre of mass of the whole S/C, can be calculated as the cross product between the position vector *r*, which starts in O and ends in the point where the thrust is applied, and the force *T*.

$$\bar{M} = \bar{r} \times \bar{T} \tag{31}$$

According to the pyramidal configuration that was described above, it is possible to write the position vector for each one of the thrusters:

$$\bar{r_1} = \begin{bmatrix} a \\ b \\ -c \end{bmatrix} \tag{32}$$

$$\bar{r_2} = \begin{bmatrix} -a \\ b \\ -c \end{bmatrix}$$

$$\overline{r_3} = \begin{bmatrix} -a \\ -b \\ -c \end{bmatrix}$$

$$\overline{r_4} = \begin{bmatrix} a \\ -b \\ -c \end{bmatrix}$$

In the same reference frame, the value of the force shall be written for each thruster. It is assumed that also the thrust of each thruster is parallel to plane x-y and the direction forms a general angle α with axis x, as shown in Figure E-1. These assumptions are a good representation of the configuration used in the chosen propulsion system and documented in the drawing in Figure E-2. Therefore, the expressions of the thrust vectors of each thruster can be found:

$$\begin{aligned} \overline{T_1} &= T_1 \begin{bmatrix} cos(\alpha) \\ \sin(\alpha) \\ 0 \end{bmatrix} \\ \overline{T_2} &= T_2 \begin{bmatrix} -cos(\alpha) \\ \sin(\alpha) \\ 0 \end{bmatrix} \\ \overline{T_3} &= T_3 \begin{bmatrix} -cos(\alpha) \\ -\sin(\alpha) \\ 0 \end{bmatrix} \\ \overline{T_4} &= T_4 \begin{bmatrix} cos(\alpha) \\ -\sin(\alpha) \\ 0 \end{bmatrix} \end{aligned} \tag{33}$$

Using the definition of torque, the torque created by each thruster can be computed.

$$\begin{aligned} M_1 &= T_1 \begin{bmatrix} c\ sin(\alpha) \\ -\mathrm{c}\ \cos(\alpha) \\ a\ sin(\alpha) - b\ cos(\alpha) \end{bmatrix} \\ M_2 &= T_2 \begin{bmatrix} c\ sin(\alpha) \\ c\ cos(\alpha) \\ -a\ sin(\alpha) + b\ cos(\alpha) \end{bmatrix} \\ M_3 &= T_3 \begin{bmatrix} -c\ sin(\alpha) \\ c\ cos(\alpha) \\ a\ sin(\alpha) - b\ cos(\alpha) \end{bmatrix} \\ M_4 &= T_4 \begin{bmatrix} -c\ sin(\alpha) \\ -c\ cos(\alpha) \\ -a\ sin(\alpha) + b\ cos(\alpha) \end{bmatrix} \end{aligned} \tag{34}$$

Adding the torques produced by each thruster, it is possible to obtain the total torque produced by the system along the three axes. The torque depends on the thrust level of each thruster, which can be properly tuned from 0 [N] to a maximum of 10 [mN], according to [1].

$$\begin{cases} M_x = c\ \sin(\alpha)(T_1 + T_2 - T_3 - T_4) \\ M_y = b\ \cos(\alpha)(-T_1 + T_2 + T_3 - T_4) \\ M_z = (\sin(\alpha) - \cos(\alpha))\ (T_1 - T_2 + T_3 - T_4) \end{cases} \tag{35}$$

When 4 thrusters are available, a fourth equation can be added to define the thrust produced by each unit; different optimization strategies can be applied, e.g. minimizing the propellant consumption for a manoeuvre. However, in case one unit is not available, it is still possible to produce the desired torque, by solving three equations with three unknown variables. For instance, in case of failure of thruster 2, it would be necessary to solve the set of equations below, which has a unique solution.

$$\begin{cases} M_x = c\sin(\alpha)(T_1 - T_3 - T_4) \\ M_y = b\cos(\alpha)(-T_1 + T_3 - T_4) \\ M_z = (\sin(\alpha) - \cos(\alpha))\,(T_1 + T_3 - T_4) \end{cases} \tag{36}$$

Therefore, in principle, it is possible to operate the system in case a thruster fails. However, some additional analysis is needed: in case only three thrusters are available, the maximum torque value that can be obtained is inferior to the nominal scenario with 4 operating units. For instance, when thruster 2 fails, it can be seen from the equations above that the maximum torque that can be achieved in certain directions is half of the nominal, e.g. the positive torque around axis x. Moreover, in case the system will be characterized by blow-down operations, i.e. the propellant pressure in the tank will decrease during the mission, problems might arise during the final phases of the mission, when the maximum torque that each thruster will be able to produce will be lower than 10 [mN]. Hence, additional investigations on the topic will be needed in the future phases of the mission: it will be fundamental to estimate in advance the required torques that will be needed during the mission, in order to assess if the operations with only 3 units will be possible. At this stage, it is reasonable to make the following assumption: as the mission can be categorised as a "deep space" mission, the expected disturbance torque is lower to a LEO mission, due to the absence of atmospheric drag; hence, it can be assumed that the operation with three thrusters will be possible, at least during the initial phases of the mission, e.g. during de-tumbling.

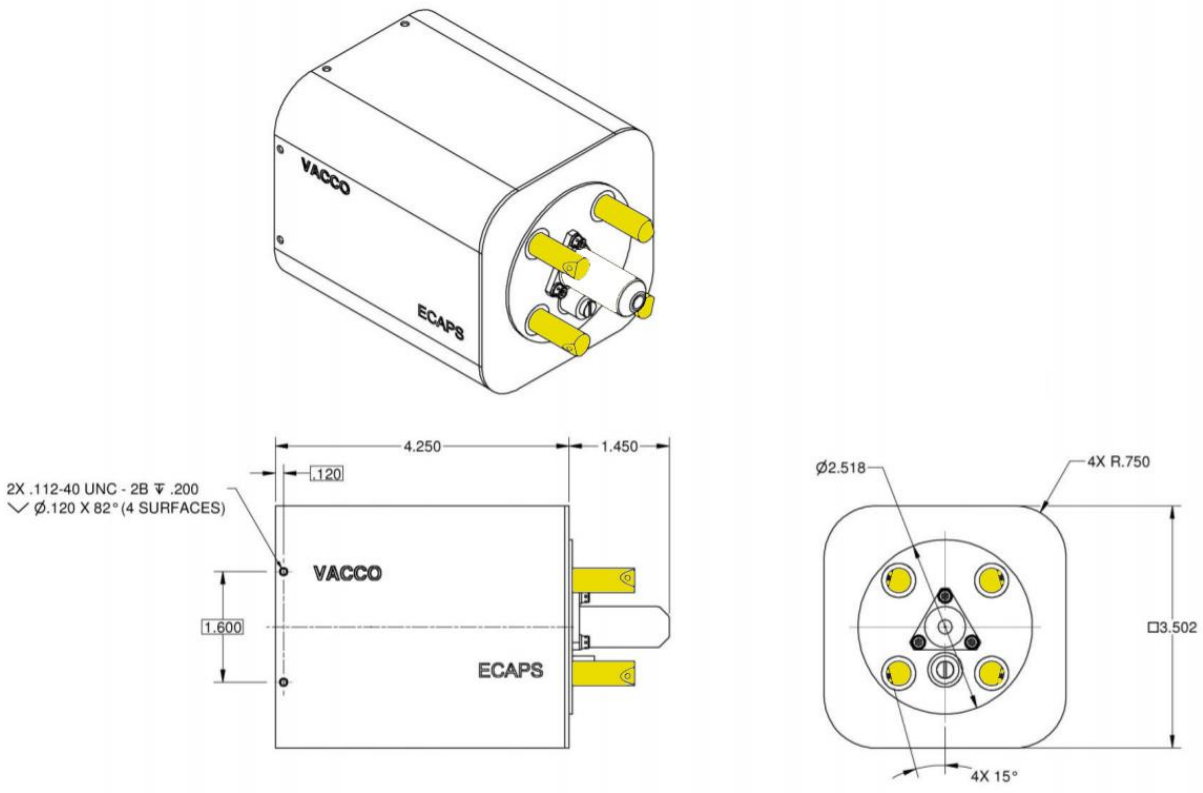

Figure E-2: CAD drawing of the propulsion system; the yellow elements are the gas thrusters. Credits: [52]

F Simulink satellite model screenshots

In this Appendix, relevant screenshots of the Simulink satellite model will be documented, in support to the description made in Chapter 5. The Appendix is divided into sub-sections, each one dedicated to a particular module.

F.1 Main Thruster module

In this Section, the relevant screenshots of the Simulink model of the Main Thruster, described in Section 5.2, are reported. They include the model for the creation of the thruster data and the model for the temperature sensors.

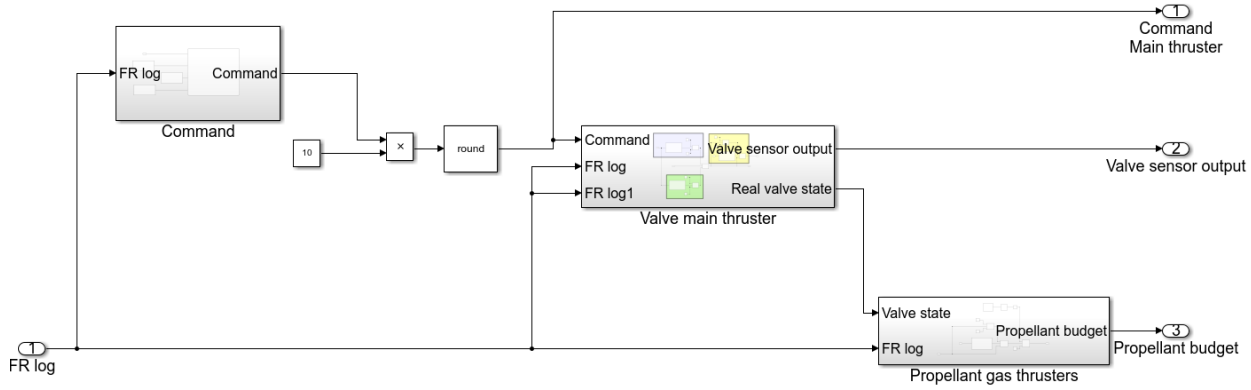

Figure F-1: Simulink block used to create the thruster's data (command, valve state, propellant budget)

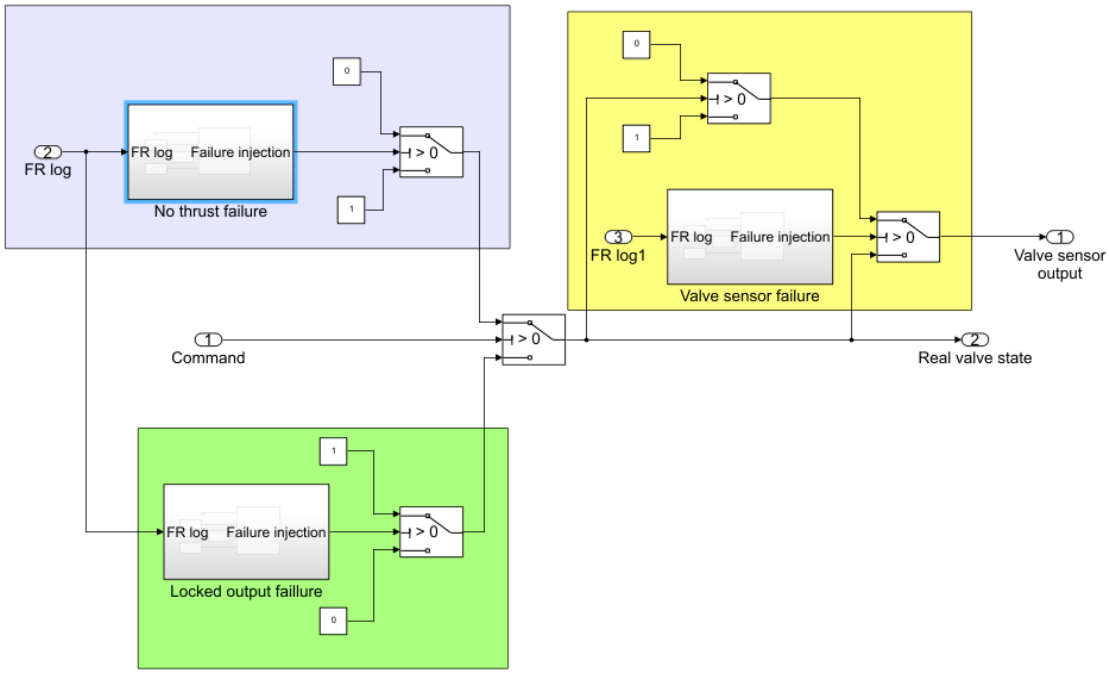

Figure F-2: Simulink logic used to create the valve state. The following failures can be injected: "No thrust" (blue area), "Locked output" (green area) and valve sensors failure" (yellow area)

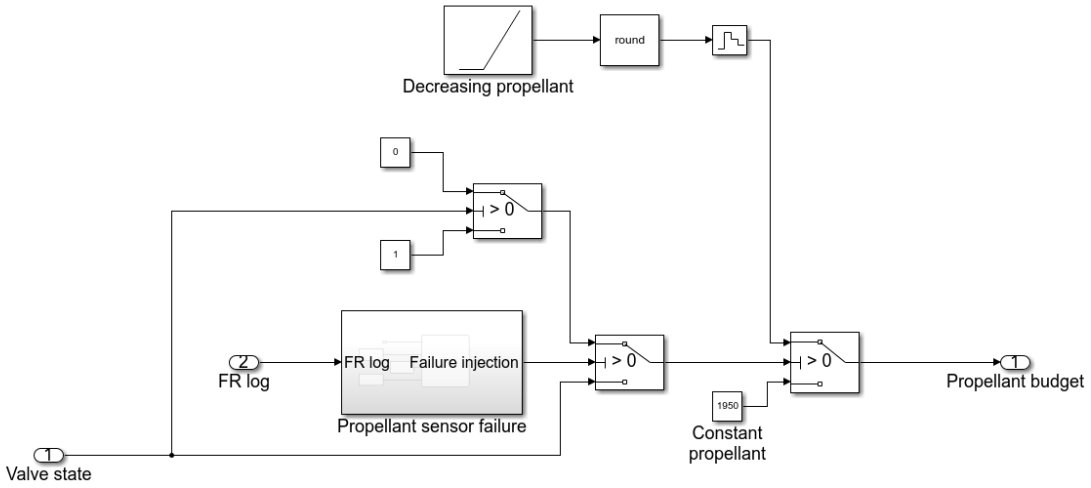

Figure F-3: Simulink logic used to create the propellant budget. The failure "Propellant sensor failure" can be injected

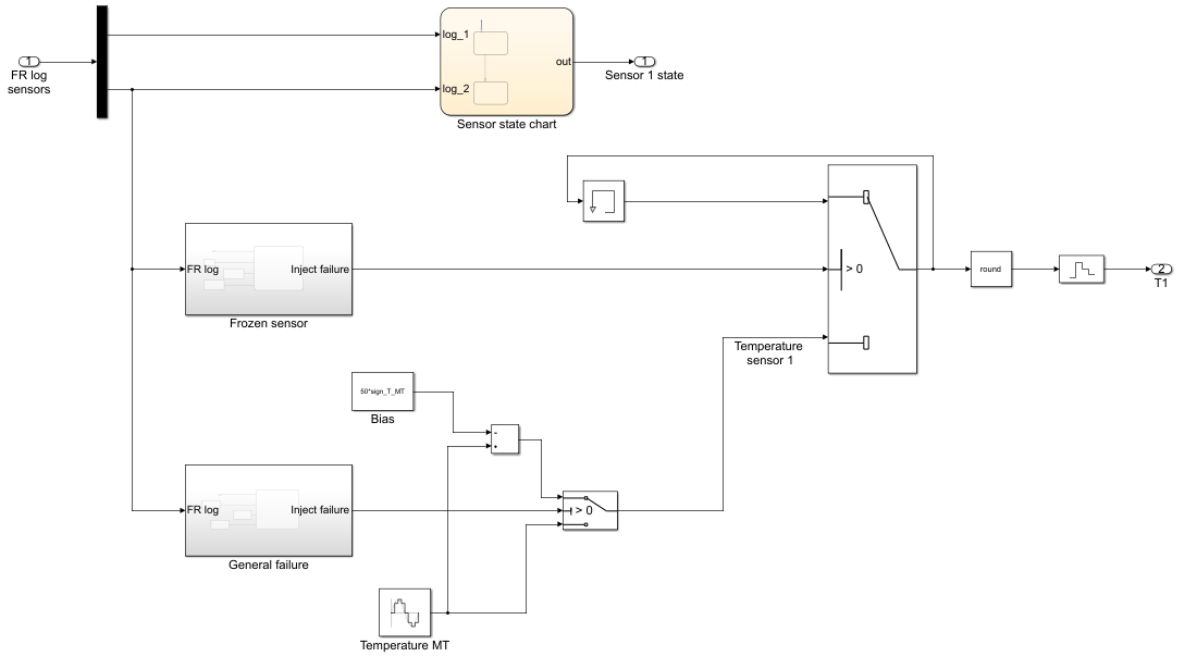

Figure F-4: Simulink logic used for the creation of the data of a temperature sensor. Two failures can be injected: "Frozen sensor" and "General failure"

F.2 Reaction Wheels module

In this Section, the relevant screenshots of the Simulink model of the reaction wheels, described in Section 5.3, are reported. They include the model for the creation of the wheels data and the model for the heater.

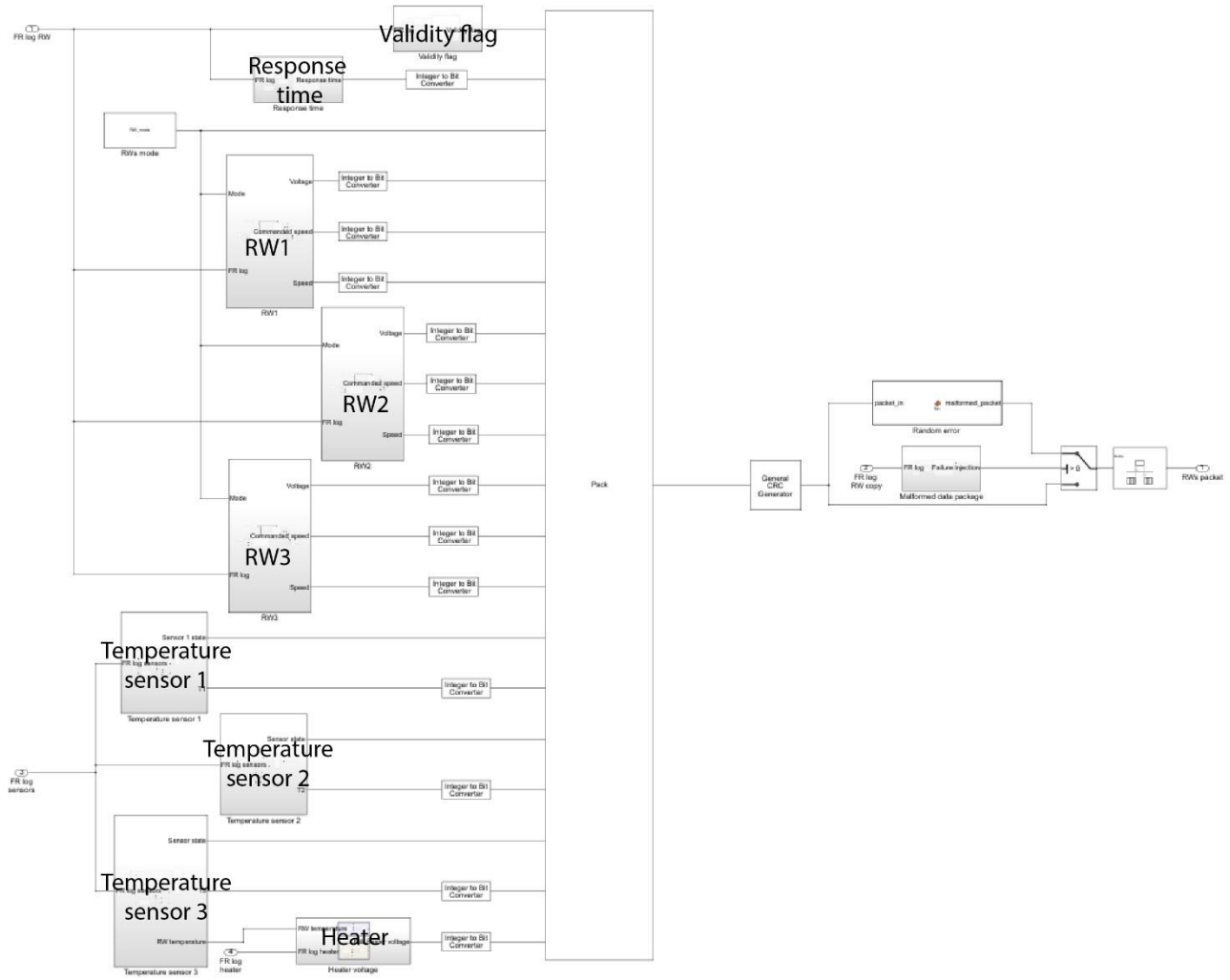

Figure F-5: high-level structure of the Reaction Wheel model

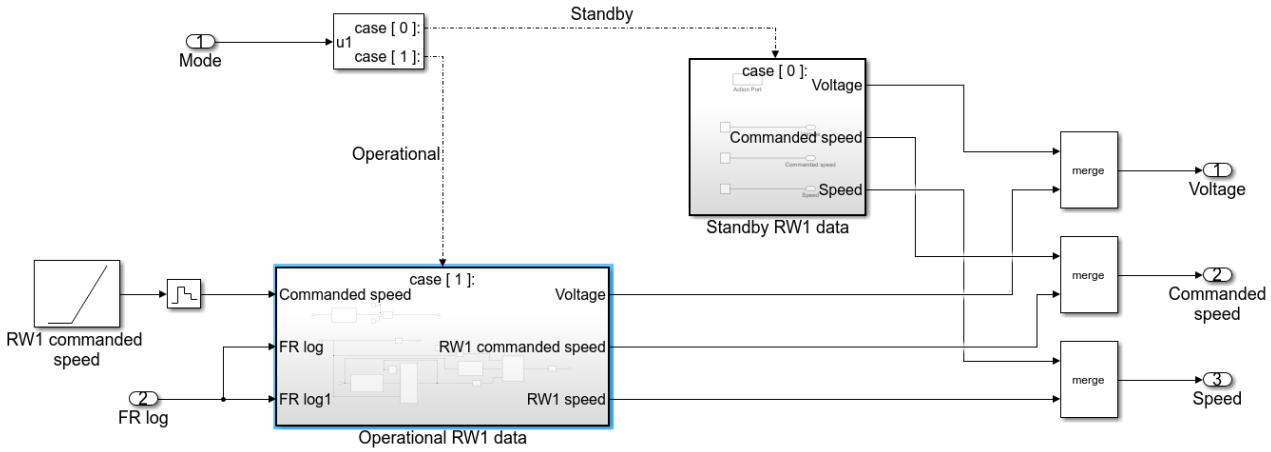

Figure F-6: Simulink logic used to create the RW data (voltage, commanded speed, speed)

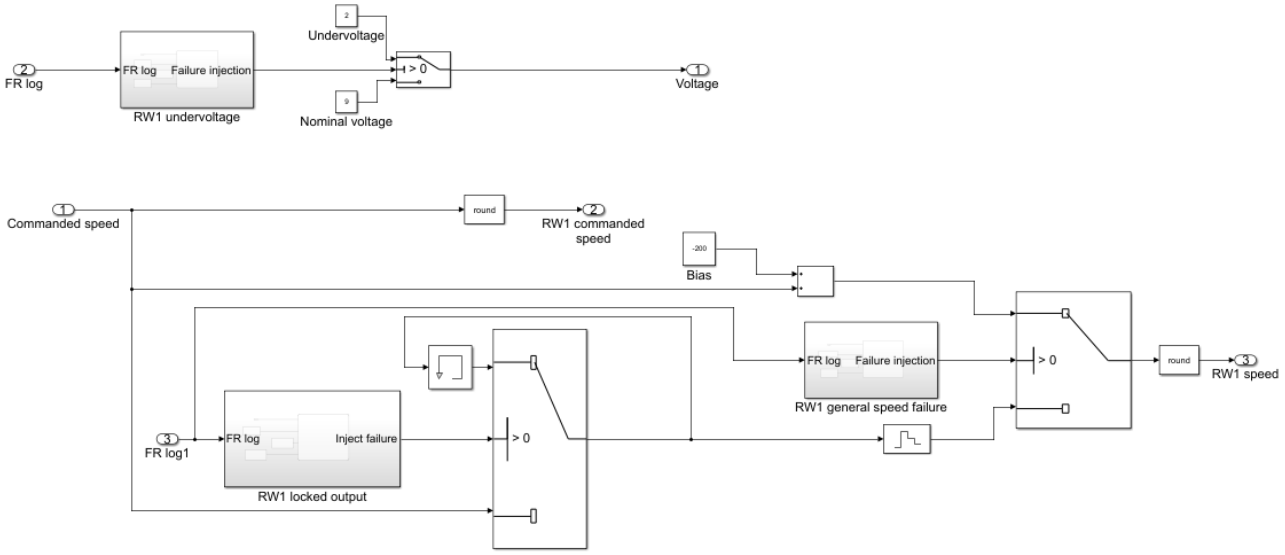

Figure F-7: Simulink logic used to create the data of a reaction wheel during "active" mode. Three failures can be injected: “Undervoltage”, “Locked output” and “General failure”

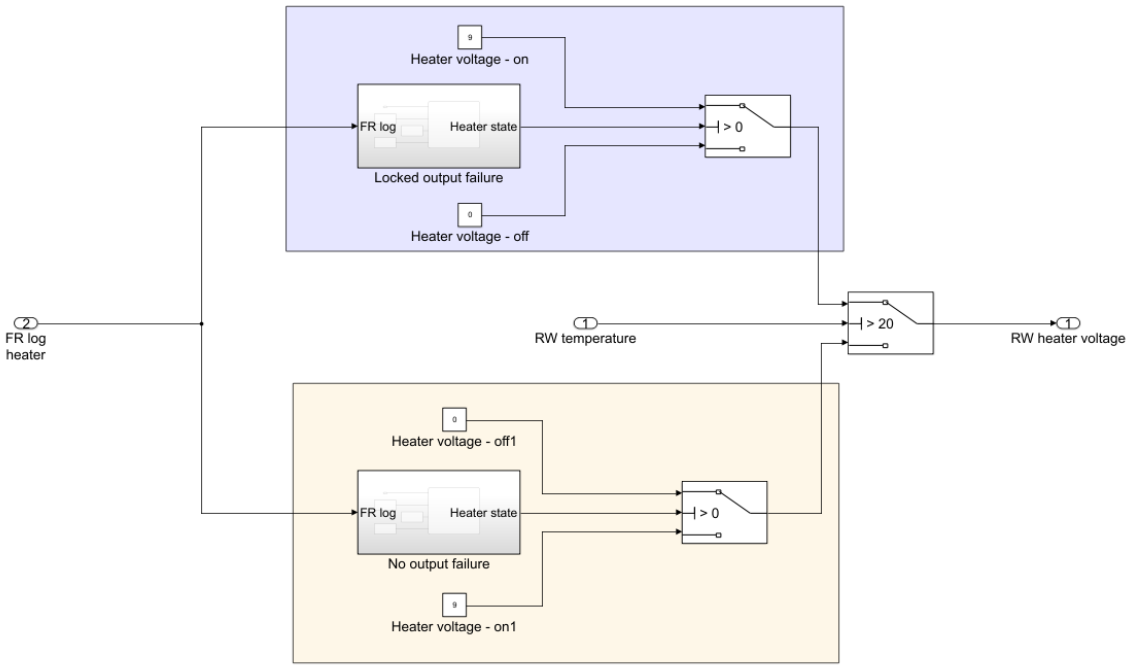

Figure F-8: Simulink logic used to create the heater data. In case a failure is injected, it can be a "Locked heater" if the temperature is high (blue area), or a "No heat" if the temperature is low (pink area)

F.3 Gas Thrusters module

In this Section, the relevant screenshot of the Simulink model of the gas thrusters, described in Section 5.4, is reported.

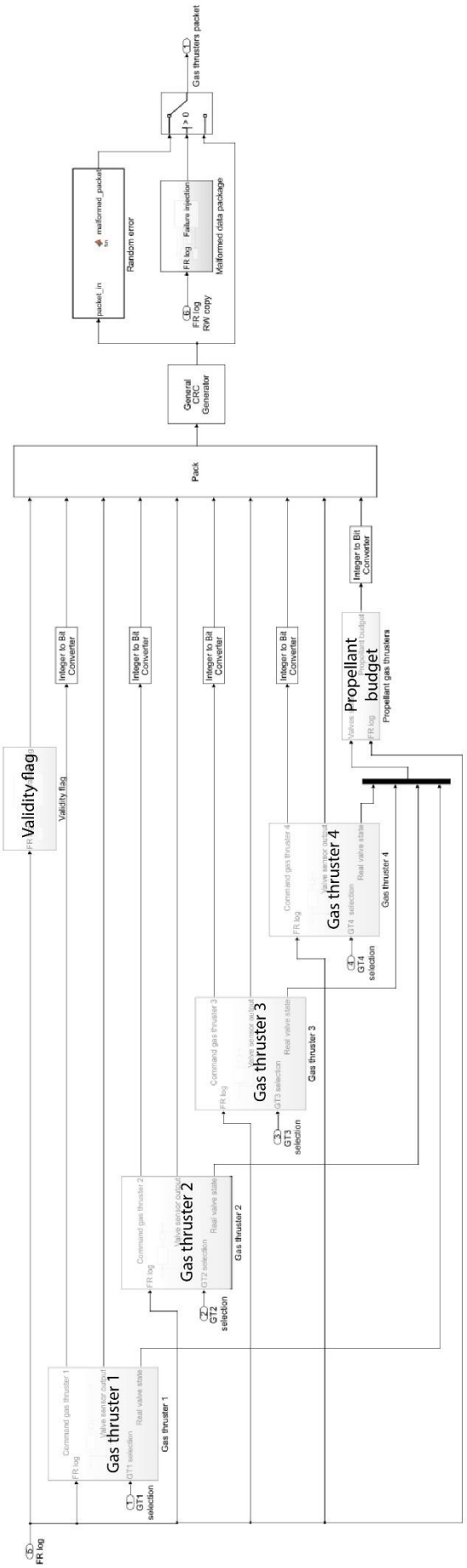

Figure F-9: high-level structure of the Simulink model which creates the packet coming from the gas thrusters

F.4 Attitude determination module

In this Section, the relevant screenshots of the Simulink model of the Attitude determination sensors, described in Section 5.5, are reported. They include the model for the creation of the attitude kinematic (the attitude quaternions and the S/C angular speed), the model for the Sun sensor and the model for a star-tracker.

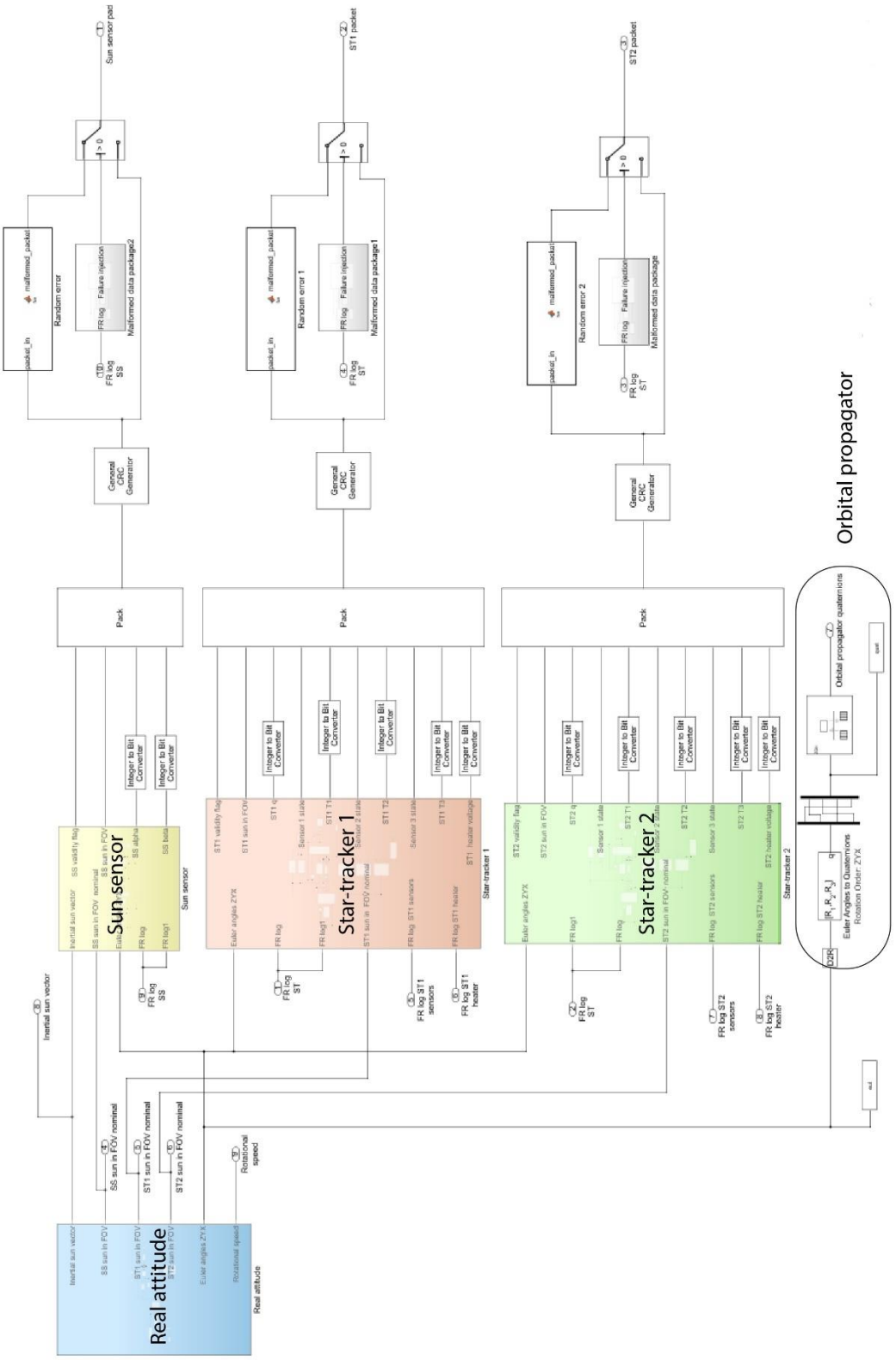

Figure F-10: high-level structure of the Attitude determination Simulink model. The blue block creates the real attitude data, the others create the data from: Sun sensor (yellow), star-tracker 1 (red) and star-tracker 2 (green).

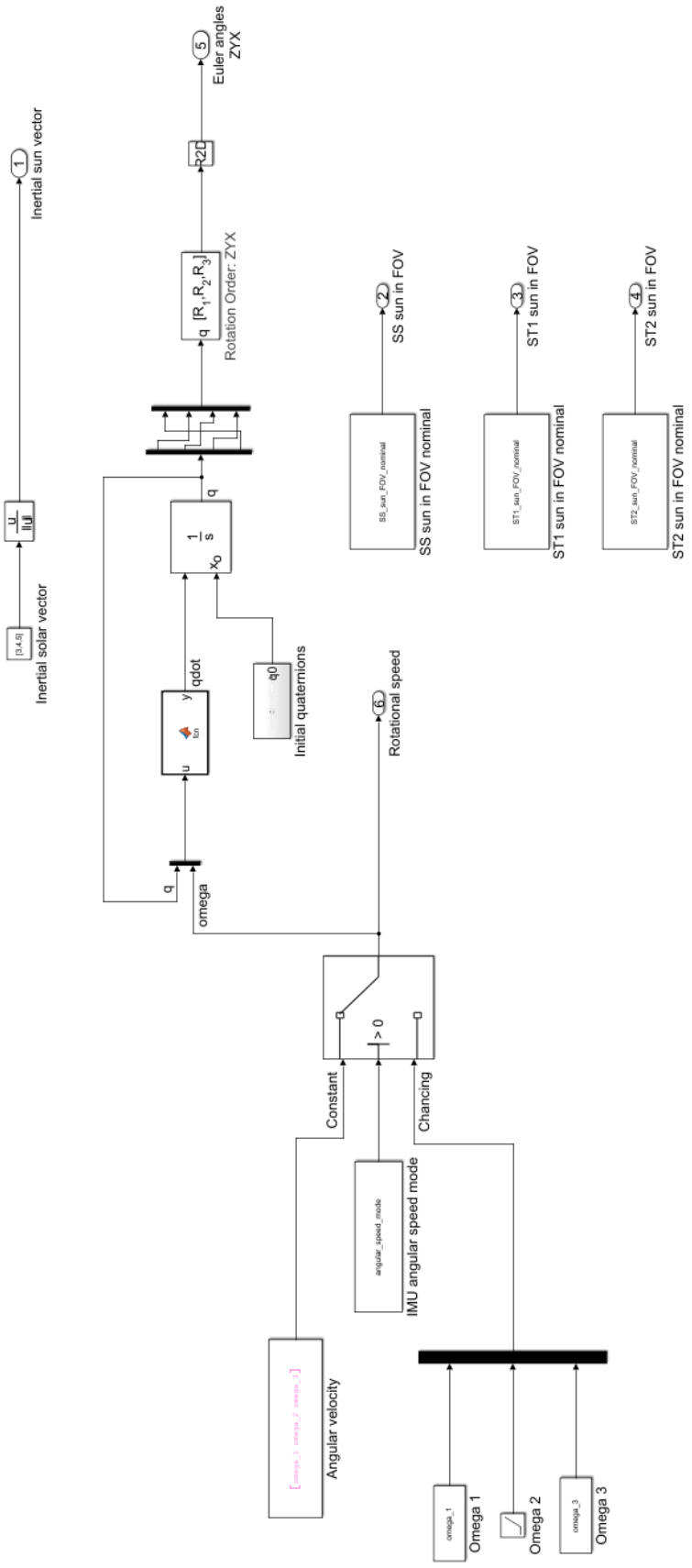

Figure F-11: Simulink model that creates the real atittude data

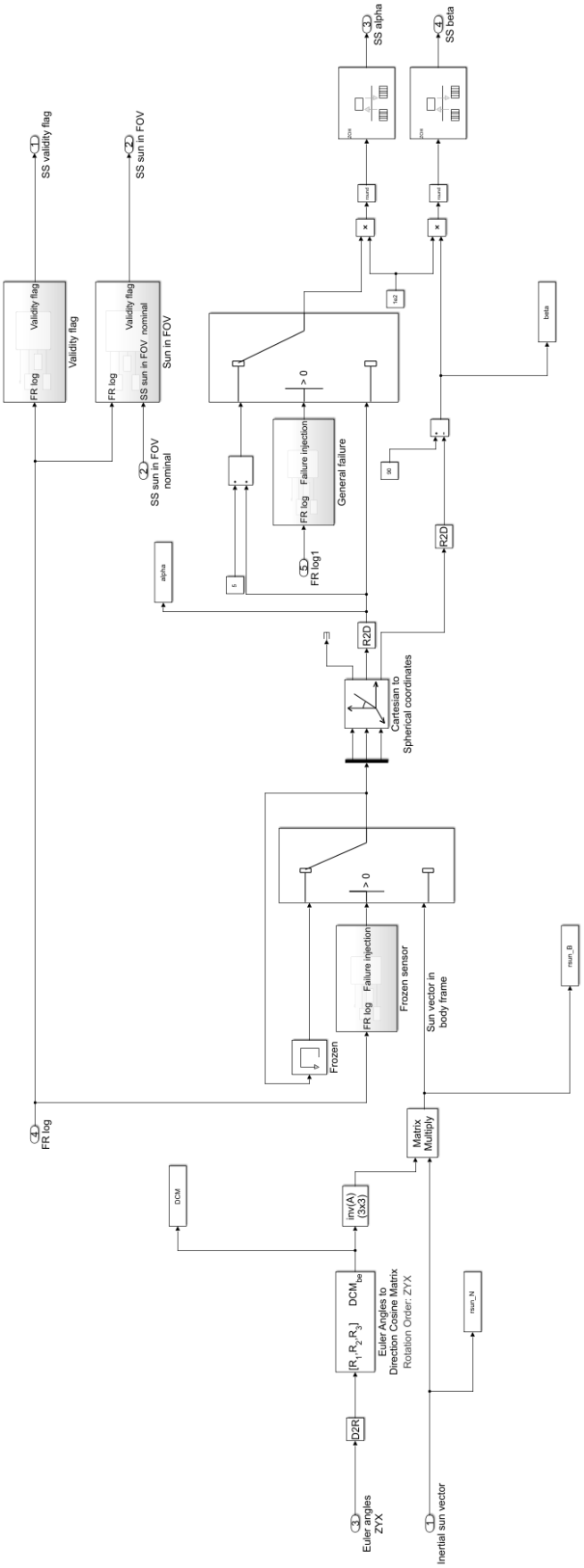

Figure F-12: Simulink model used to create the data package from the Sun sensor. The sun vector in the body reference frame is computed from the nominal attitude and converted into spherical coordinates, to generate the angles alpha and beta. Two failures can be injected: “Frozen sensor” and “General failure”.

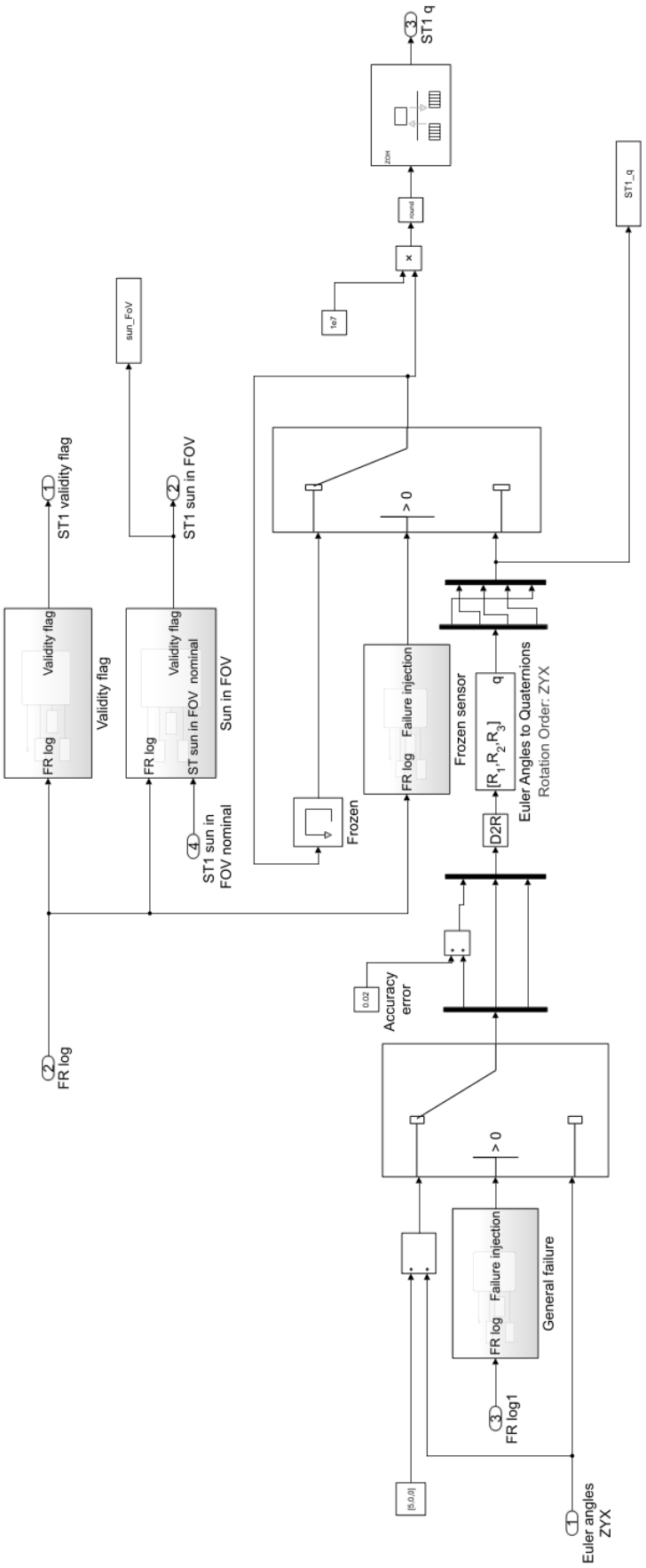

Figure F-13 Simulink model used to create the data package from a star-tracker. The Euler angles from the real attitude are converted into quaternions; two failures can be injected: “Frozen sensor” and “General failure”

F.5 IMU module

In this Section, the relevant screenshots of the Simulink model of the IMU, described in Section 5.6, are reported. They include the high-level view of the module and the detailed view of the block aimed at the creation of the gyroscope measurements.

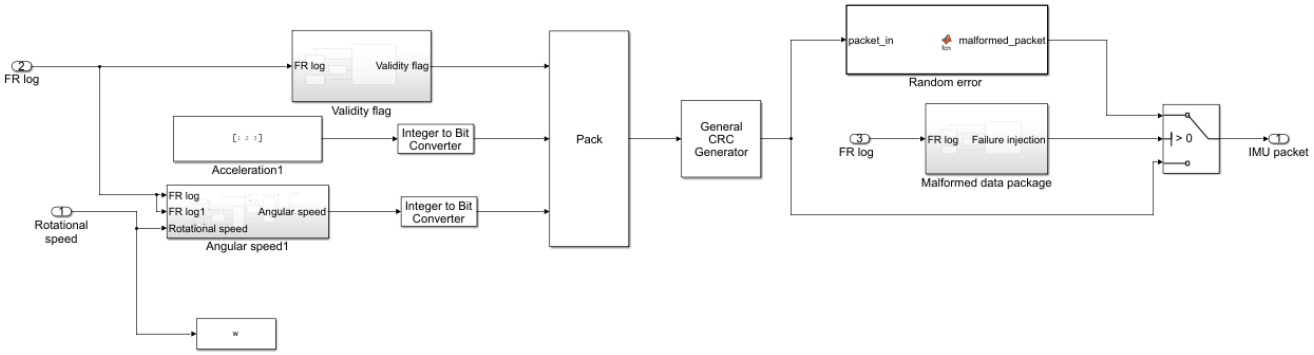

Figure F-14: high-level structure of the Simulink model used to create the data package from the IMU

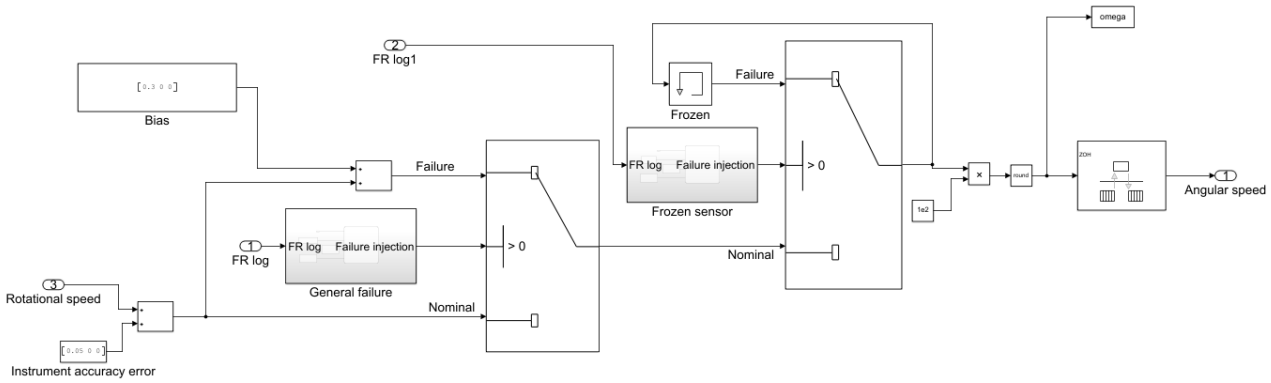

Figure F-15: Simulink model used to generate the measurements of the gyroscopes. Two failure scenarios can be injected: "Frozen sensor" and "general failure"

F.6 Power module

In this Section, the relevant screenshots of the Simulink model of the Power module, described in Section 5.7, are reported. They include the high-level view of the module and the detailed view of the blocks of the SADA, the solar panels and the batteries.

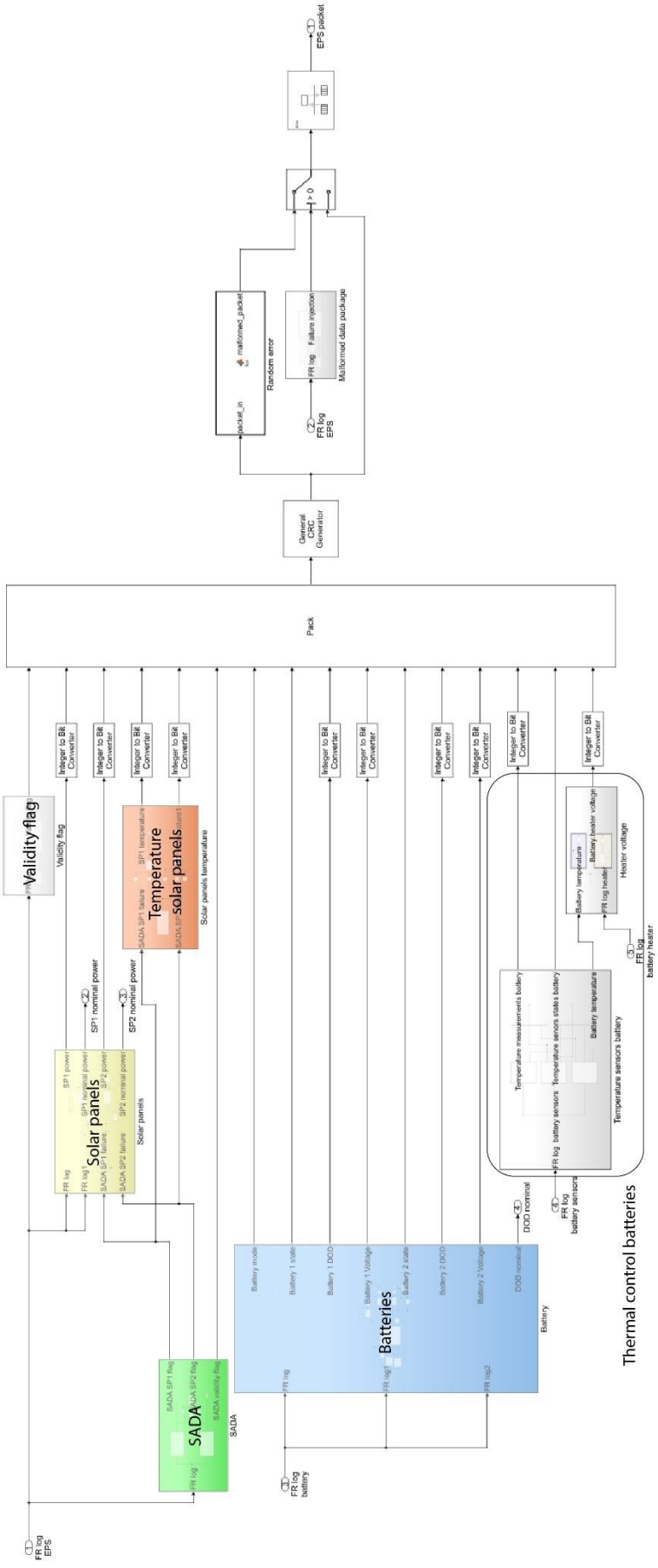

Figure F-16: high-level structure of the Simulink model for the creation of the Power data package. The following blocks can be distinguished: SADA (green), solar panels (yellow) solar panels temperature (red), batteries (blue)

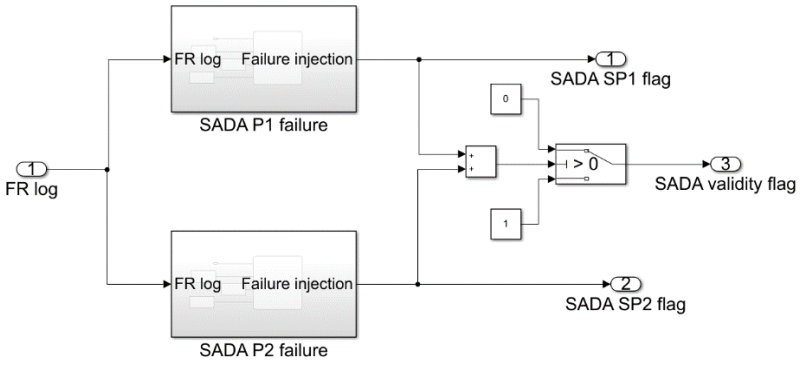

Figure F-17: detailed view of the Simulink block aimed at the creation of the SADA validity flag, in green in Figure F-16

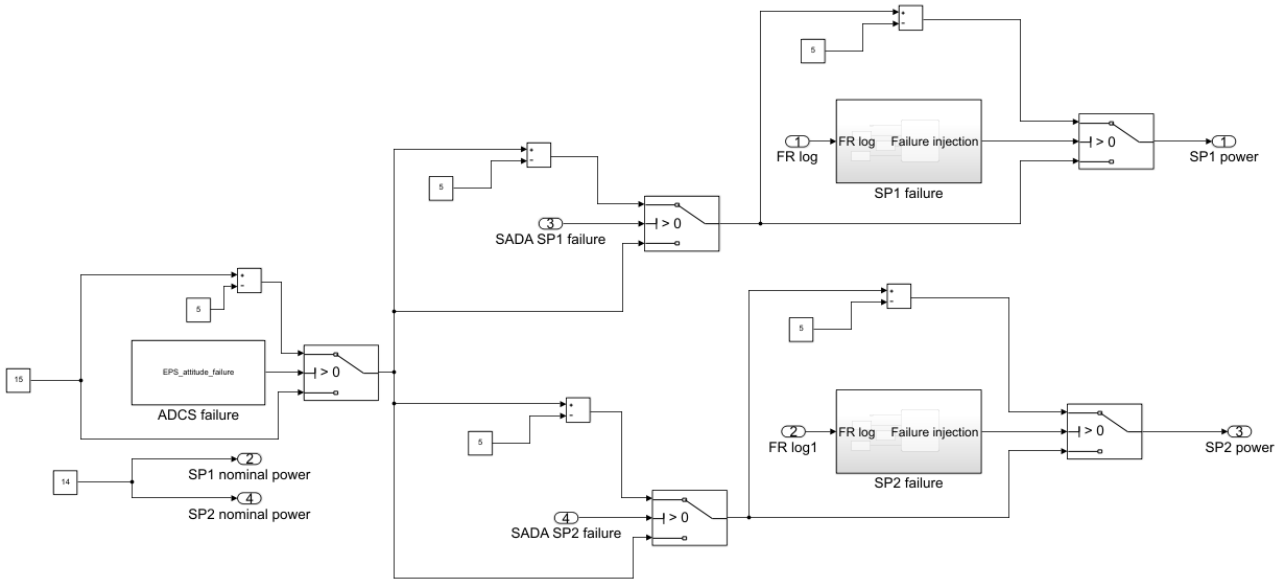

Figure F-18: Simulink block aimed at creating the data about solar panels power, in yellow in Figure F-16. The power value can be affected by SADA failures, panels failures or ADCS failures

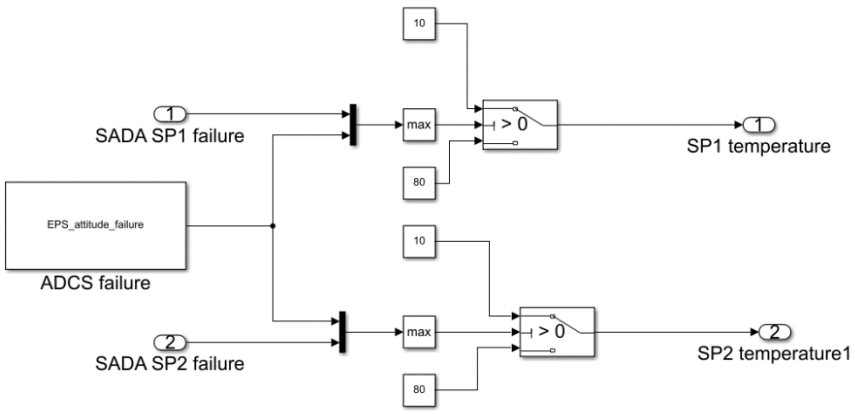

Figure F-19: Simulink block aimed at creating the data about solar panels temperature, in red in Figure F-16

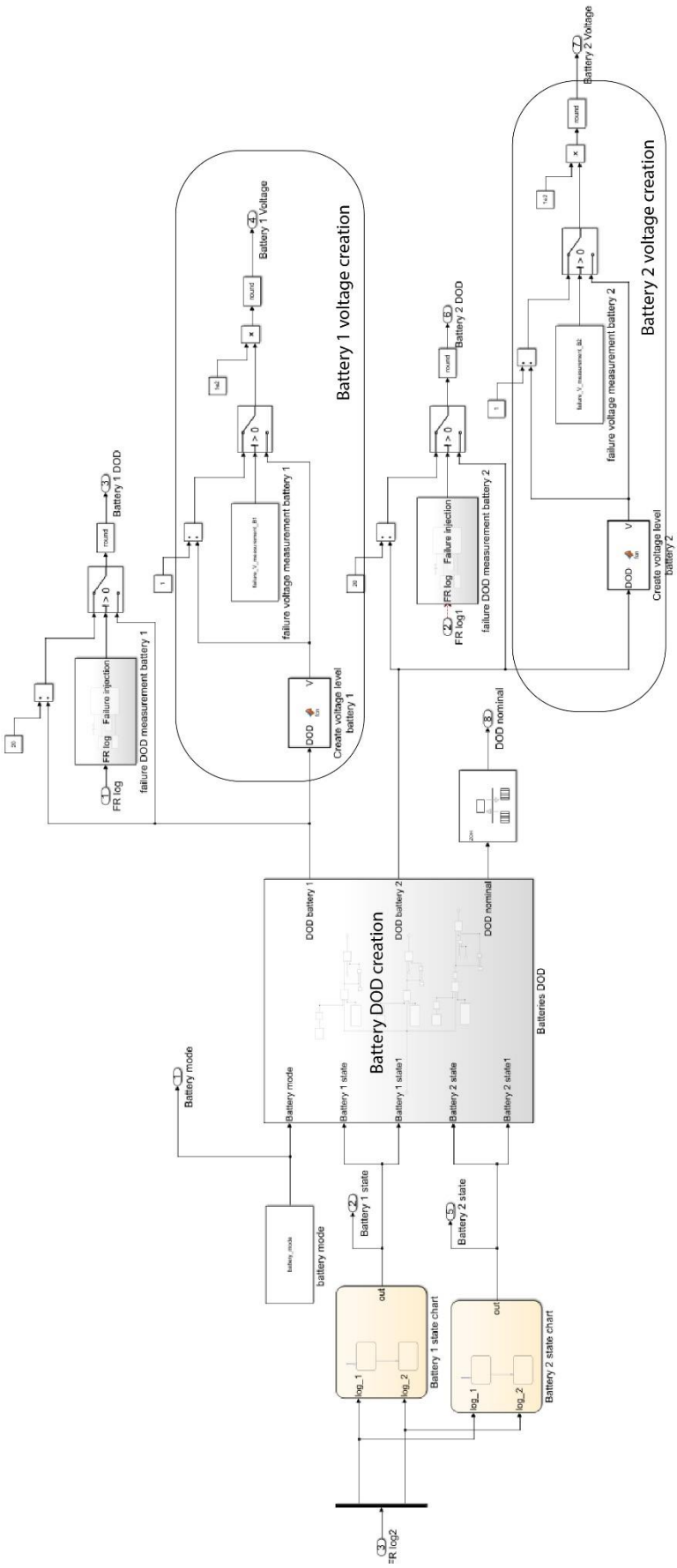

Figure F-20: Simulink block aimed at creating the data package from the batteries, in blue in Figure F-16. First, the DOD is created; later, the voltage level is calculated from the DOD. Failures of the DOD measurement or voltage measurement can be injected

F.7 Camera module

In this Section, the relevant screenshots of the Simulink model of the Camera module, described in Section 5.8, are reported. They include the high-level view of the module and the detailed view of its blocks.

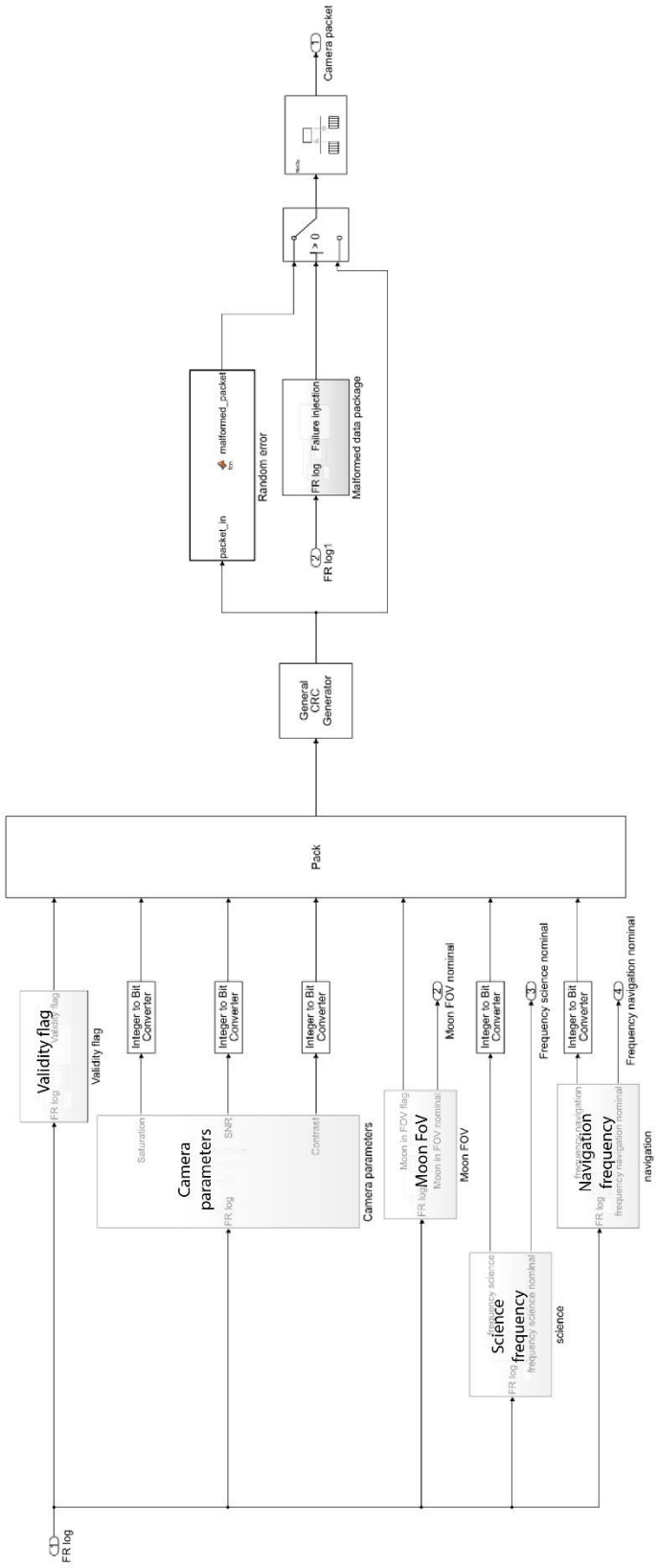

Figure F-21: high-level structure of the Simulink model for the creation of the Camera data package

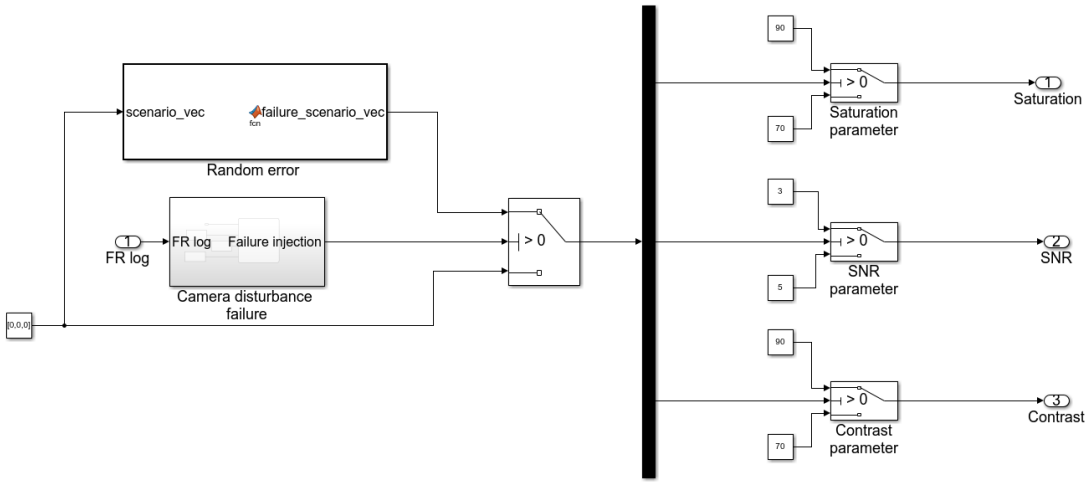

Figure F-22: detailed view of the Simulink model that creates the Camera parameters. The "Camera disturbances" failure can be injected

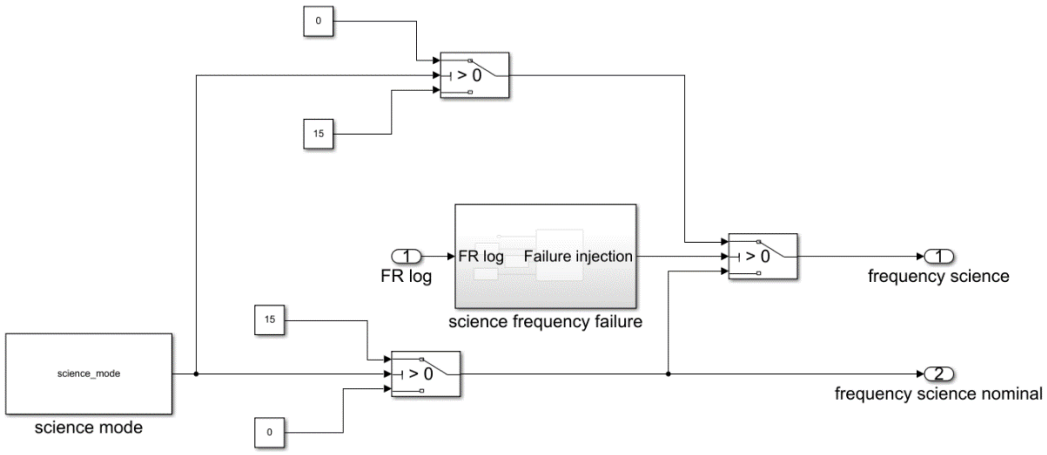

Figure F-23: detailed view of the Simulink model used to generate the camera acquisition frequency during science operations. The scenario “Science frequency failure can be injected”

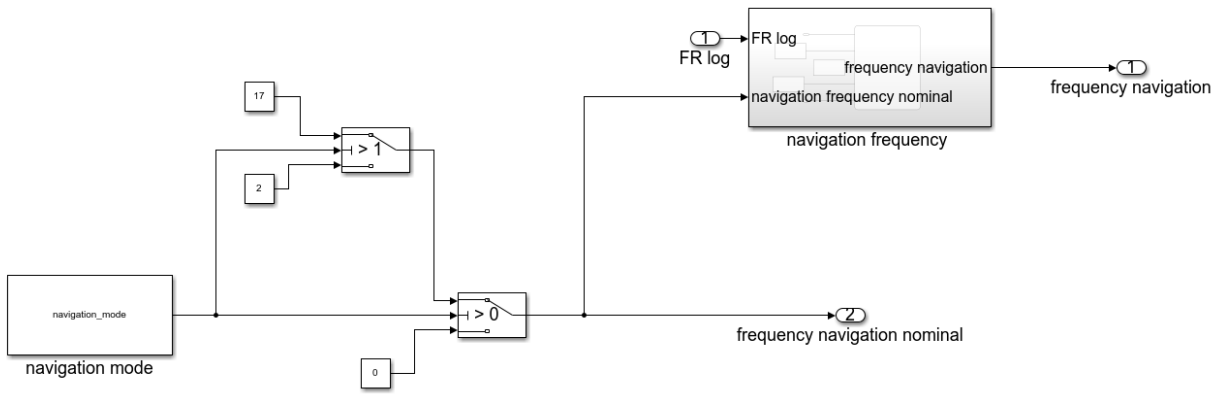

Figure F-24: detailed view of the Simulink model used to generate the camera acquisition frequency during navigation

F.8 Deployment module

In this Section, the relevant screenshots of the Simulink model of the Deployment module, described in Section 5.9, are reported. They include the high-level view of the module and the detailed view of one block.

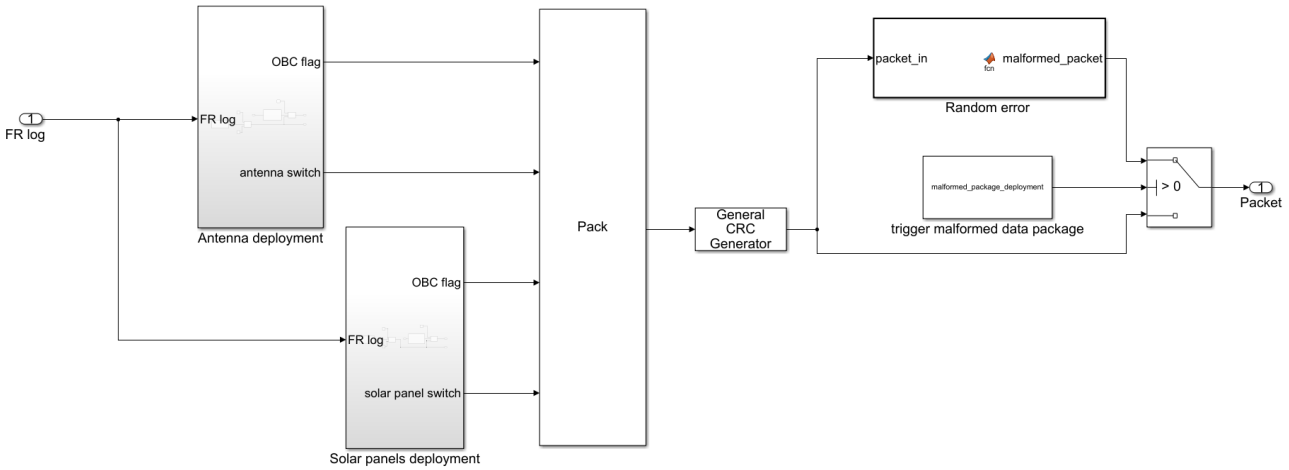

Figure F-25: high-level view of the Simulink model aimed at the creation of the Deployment data package

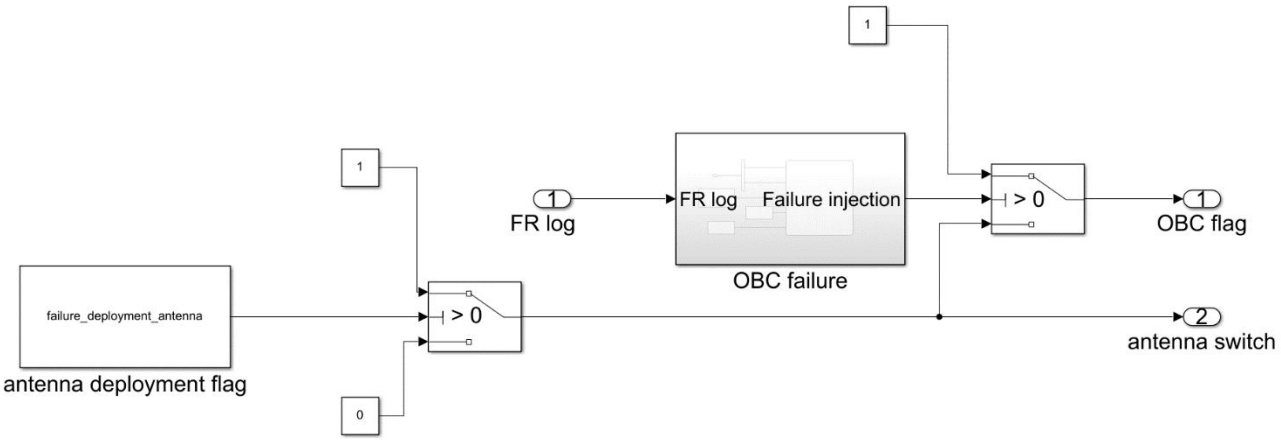

Figure F-26: logic used in the Simulink model for the antenna deployment. Two failures can be injected: "Deployment failure" and "OBC failure"

F.9 Communication module

In this Section, the relevant screenshots of the Simulink model of the Communication module, described in Section 5.10, are reported. They include the high-level view of the Simulink model and the detailed view of its blocks.

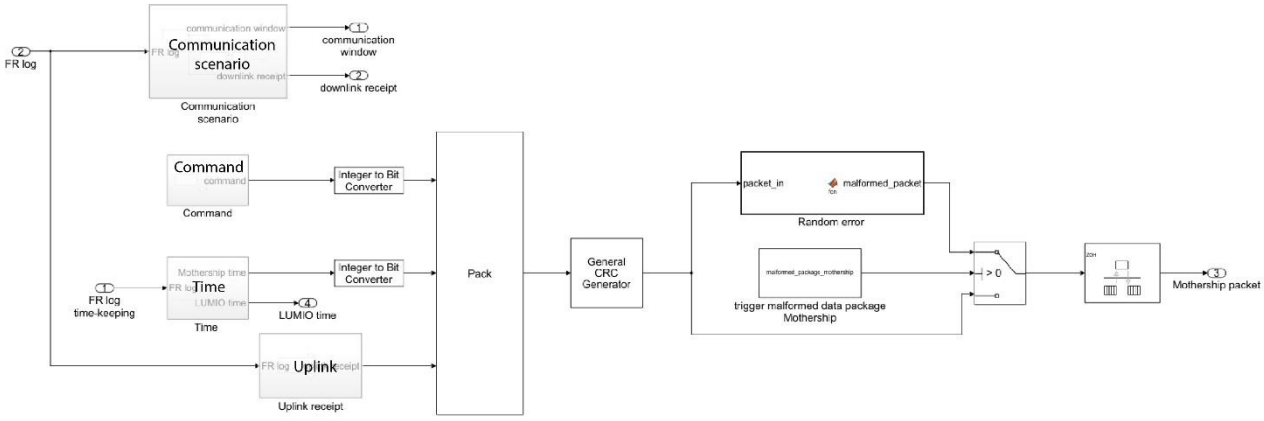

Figure F-27: high-level structure of the Simulink model aimed at the creation of the Communication data package

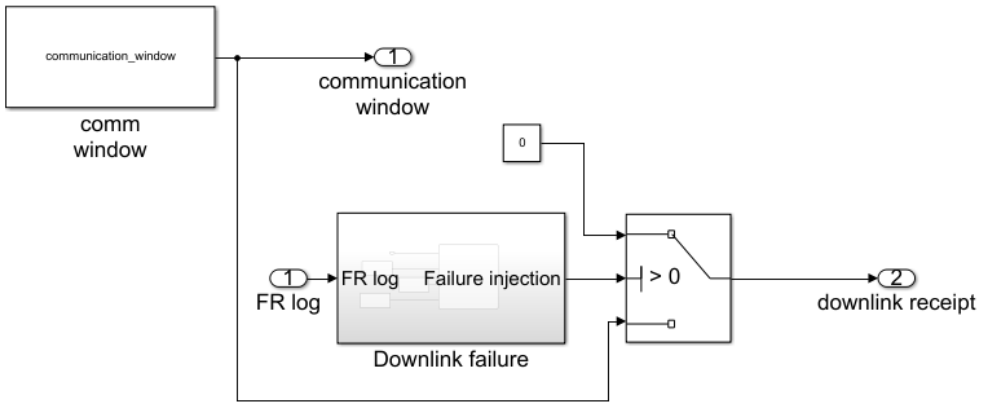

Figure F-28: logic used for the creation of the nominal scenario in the Communication module

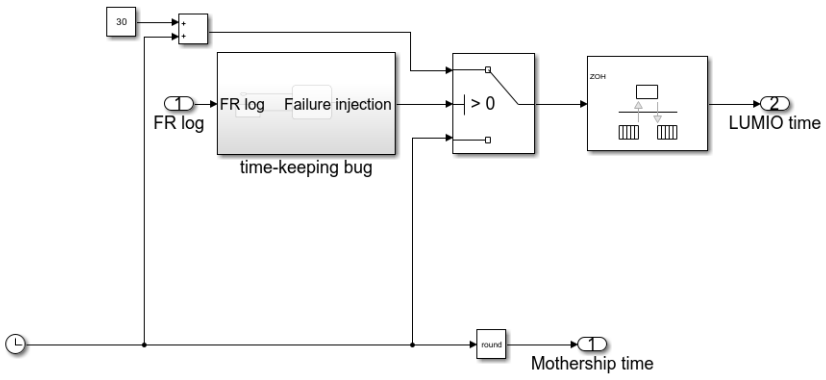

Figure F-29: logic used for the creation of the on-board time keeping and the Mothership time

F.10 User interface

In this Section, the relevant screenshots of the user interface developed to execute the FDI simulations, described in Section 5.12, are reported. Since the interface is divided into six tabs, one picture per tab is presented.

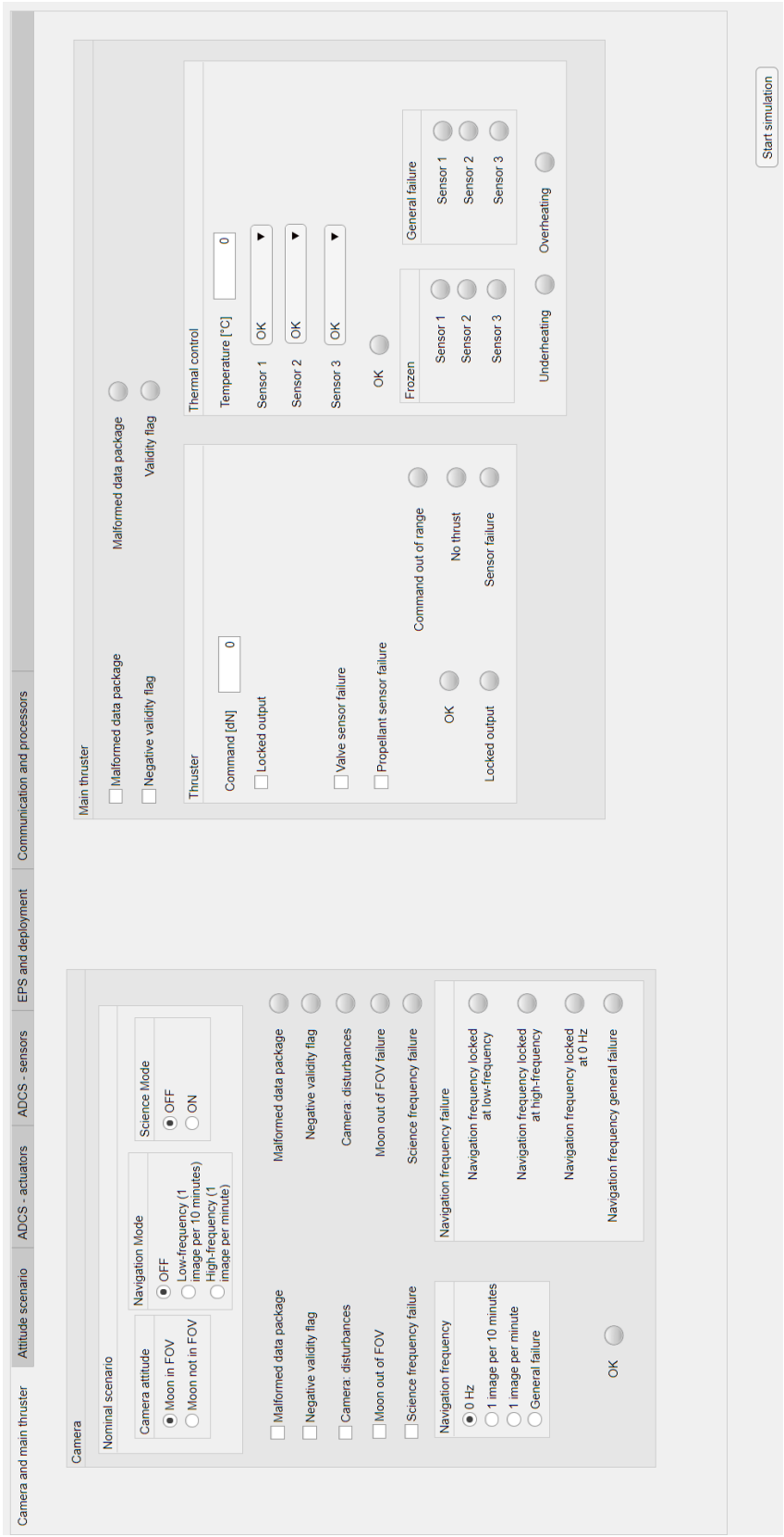

Figure F-30: first tab of the user interface, dedicated to failures of the camera and the main thruster

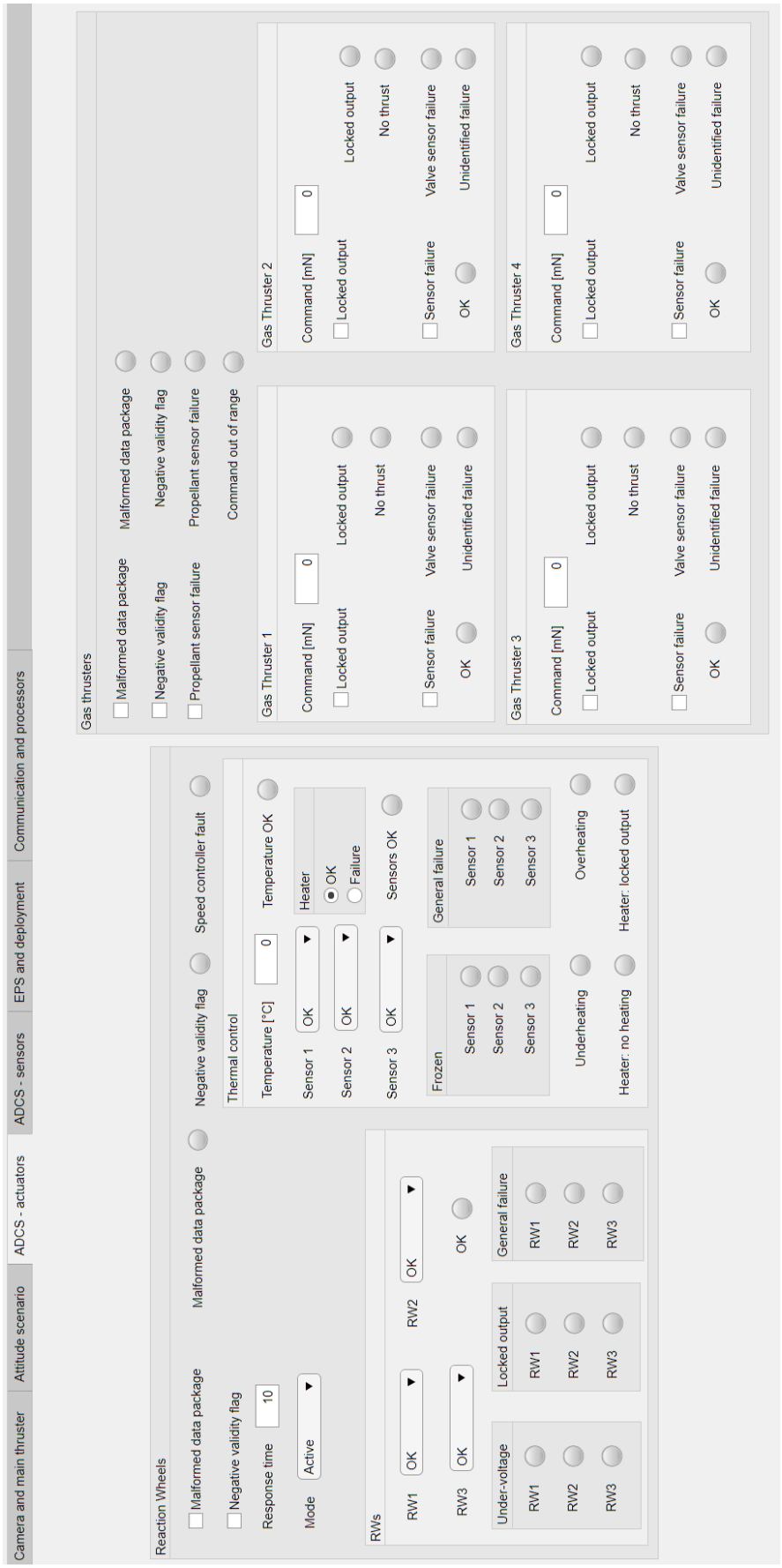

Figure F-31: third tab of the user interface, dedicated to failures of the ADCS actuators

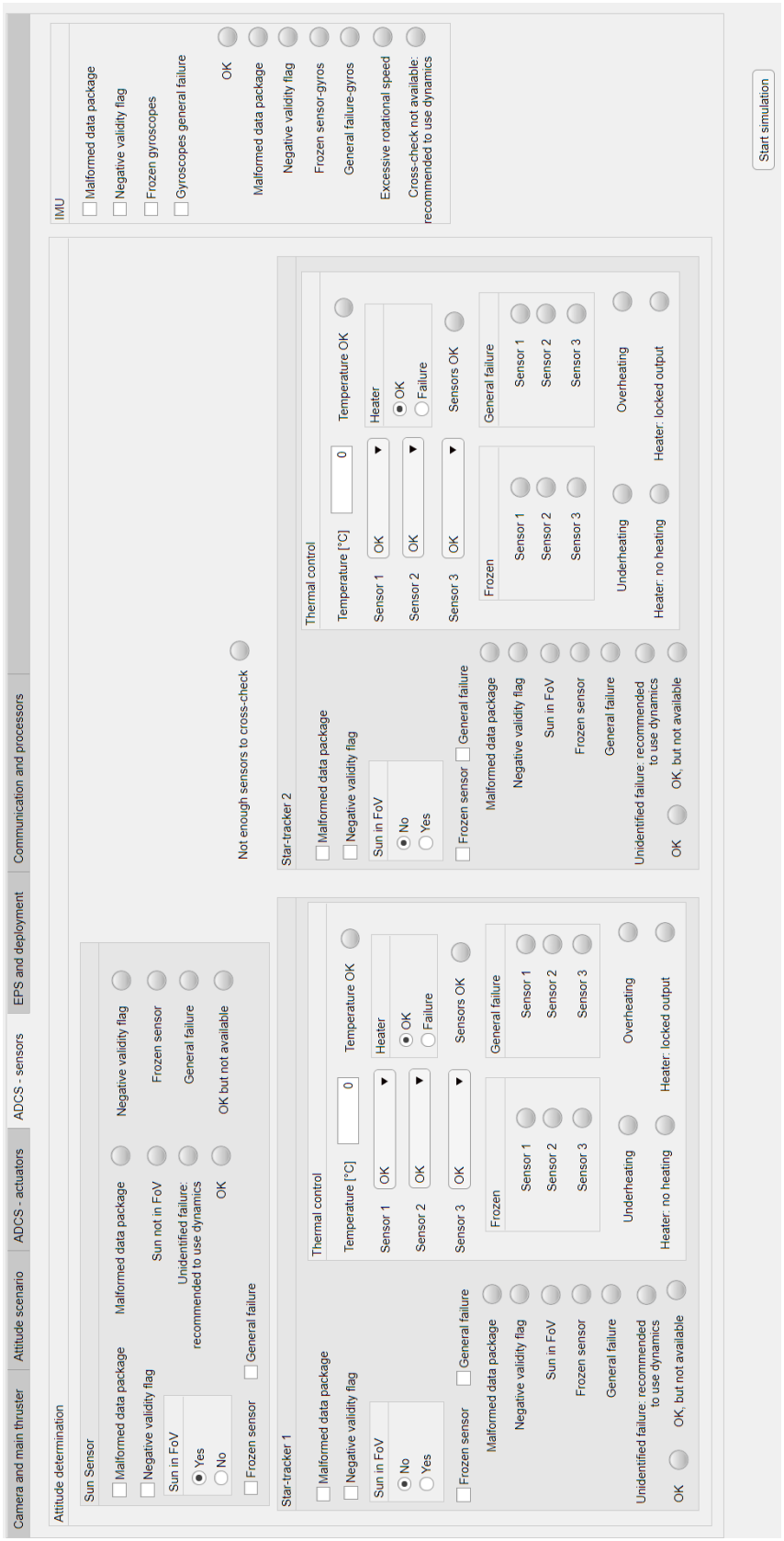

Figure F-32: fourth tab of the user interface, dedicated to failures of the ADCS sensors

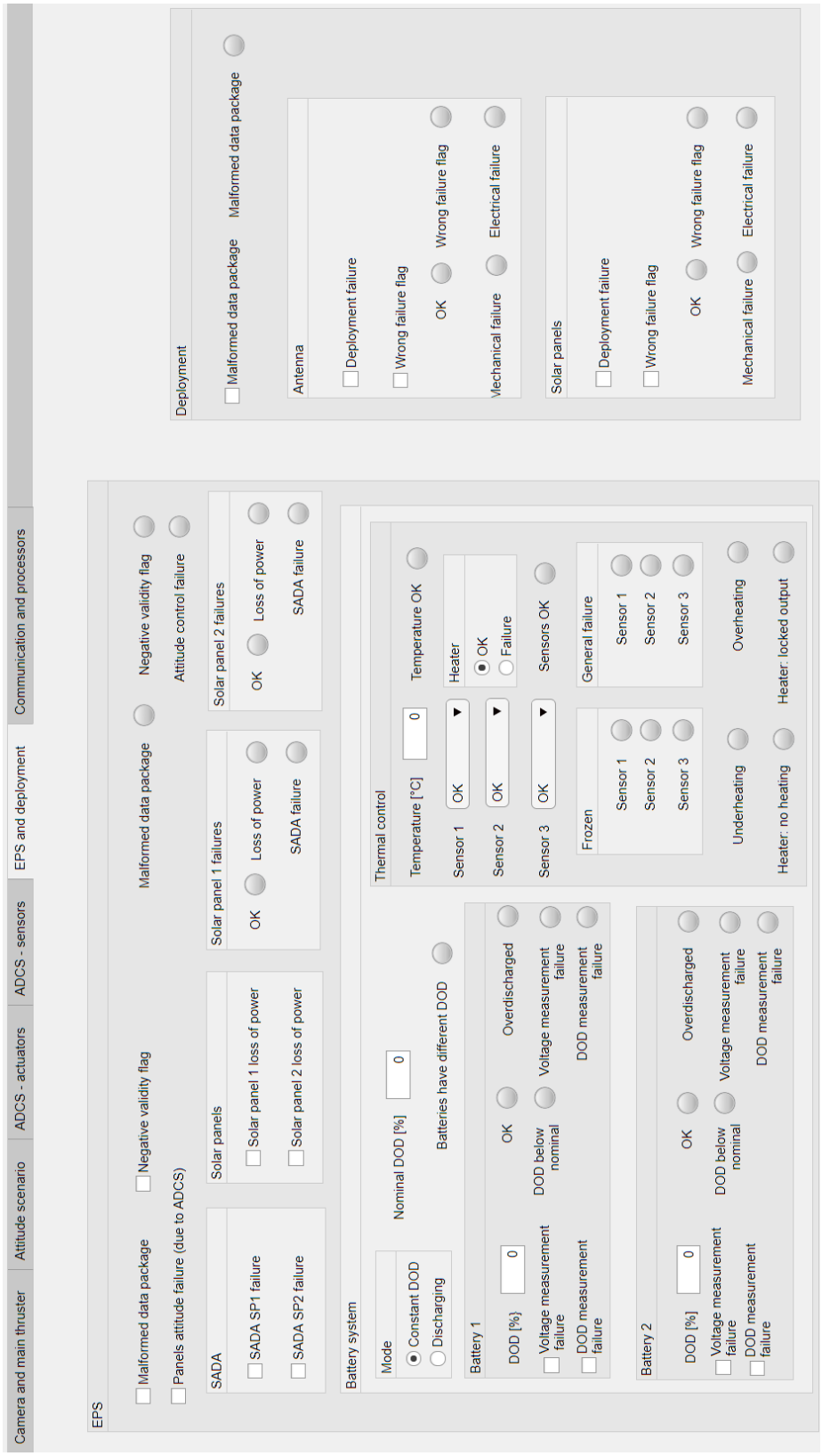

Figure F-33: fifth tab of the user interface, dedicated to failures of the Power module and the Deployment module

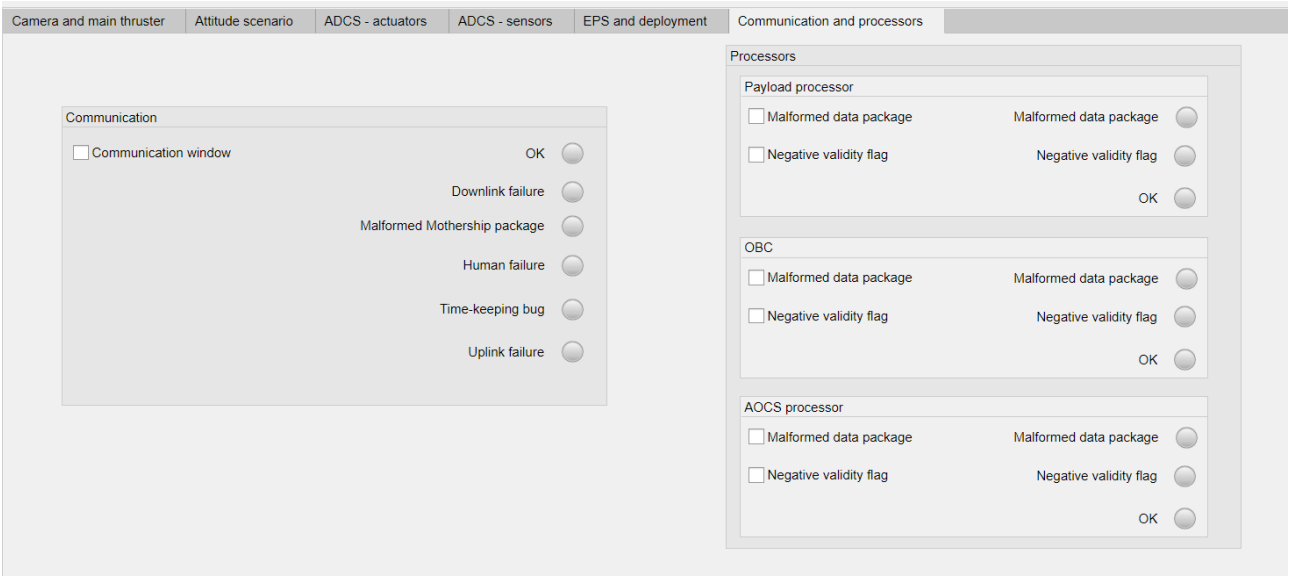

Figure F-34: sixth tab of the user interface, dedicated to failures of communication and processors

G List of simulations

In this Appendix, the list of simulations performed to verify the FDIR algorithm is documented. The model used to perform the simulations is described in Chapter 5, while the procedure and the success criteria for each test are explained in Chapter 6.

In Table G-1, the simulations are divided per test-case. For each test case, a brief description and the necessary scenario parameters to use are provided. Finally, the result of the test is presented, according to the criteria presented in Chapter 6.

Table G-1: list of simulations performed, divided per test-case

Test-case ID	Description	Scenario parameters	Result
CAM.1.1	Camera image processing disturbances	Disturbance failure CAM = 1	Positive
CAM.1.2	Camera image processing disturbances	Validity flag camera = 0	Positive
CAM.2	Moon outside Field of View	Moon FOV failure CAM = 1	Positive
CAM.3.1	Navigation frequency frozen at 0 Hz	Navigation frequency = 0	Positive
CAM.3.2	Navigation frequency frozen at low frequency	Navigation mode = 0 Navigation frequency = 2	Positive
CAM.3.3	Navigation frequency frozen at high frequency	Navigation frequency = 17	Positive
CAM.4	Navigation frequency – general failure	Navigation frequency ≠ 0, 2, 17	Positive
CAM.5	Low science frequency	Science mode = 0 Science frequency failure CAM = 1	Positive
CAM.6	High science frequency	Science frequency failure CAM = 1	Positive
CAM.7	Malformed data package	Malformed package CAM = 1	Positive
OBPDP.1	Hardware failure	Validity flag PD processor = 0	Positive
OBPDP.3	Malformed data package	Malformed data package PD processor = 1	Positive
OBC.1	Hardware failure	Validity flag OBC = 0	Positive
OBC.3	Malformed data package from Mothership	Malformed data package OBC = 1	Positive
OBC.4	Human failure	Ground command failure = 1	Positive
OBC.5	Malformed data package	Malformed data package Mothership = 1	Positive
OBC.6.1	Wrong failure flag during deployment – antennas	OBC wrong failure flag deployment antenna=1	Positive
OBC.6.2	Wrong failure flag during deployment – solar panels	OBC wrong failure flag deployment SP=1	Positive
OBC.7	Time-keeping bug	Time failure = 1	Positive
RW.1.1	Speed controller fault	Validity flag RW = 0	Positive
RW.1.2	Speed controller fault	Response time RW ≤ 5	Positive
RW.2	Undervoltage	RW undervoltage = 1 (for any RW)	Positive
RW.3	Locked speed	Locked speed RW =1 (for any RW)	Partially positive (10 s delay)
RW.4	General failure	General speed failure RW = 1 (for any RW)	Positive
RW.5.1	Overheating – active	Temperature RW > 50	Positive
RW.5.2	Overheating - idle	Temperature RW > 60	Positive
RW.6.1	Underheating – active	Temperature RW < 0	Positive
RW.6.2	Underheating - idle	Temperature RW < -10	Positive
RW.7	Malformed data package	Malformed data package RW = 1	Positive
GT.1.1	Locked output when GT are idle	Locked output GT = 1 (for any GT)	Partially positive (10 s delay)

Test-case ID	Description	Scenario parameters	Result
GT.1.2	Locked output when GTs are active	0<Command GT1<10 Locked output GT = 1 (for any GT except GT1)	Positive (failure detected but not isolated)
GT.2	Command out of range	Command GT > 10 (for any GT)	Positive
GT.3.1	No thrust when all GTs are active	0<Command GT<10 (all GTs) No thrust GT = 1 (for any GT)	Positive (detect but not isolated)
GT.3.2	No thrust when only one GT is active	0<Command GT<10 (for any GT) No thrust GT = 1 (for the same GT)	Positive
GT.4.1	Propellant sensors failure when all GTs are idle	Sensor failure propellant GT = 1	Partially positive (10 s delay)
GT.4.2	Propellant sensors failure when all GTs are active	0<Command GT<10 (for any GT) Sensor failure propellant GT = 1	Positive
GT.5.1	Valve sensor failure when all GTs are idle	Sensor failure valve GT = 1 (for any GT)	Positive
GT.5.2	Valve sensor failure when all GTs are active	0<Command GT<10 (for all GTs) Sensor failure valve GT1 = 1 (for any GT)	Positive (failure detected but not isolated)
GT.6	Malformed data package	Malformed package GT = 0	Positive
SS.1.1	Frozen Sun sensor – cross-check with two star-trackers	Angular speed mode = 0 SS frozen = 1	Positive
SS.1.2	Frozen Sun sensor – cross-check with one star-tracker	ST1 sun in FOV nominal = 1 ST1 sun in FOV = 1 Angular speed mode = 0 SS frozen = 1	Positive
SS.1.3	Frozen Sun sensor – no cross-check	ST1 sun in FOV nominal = 1 ST1 sun in FOV = 1 ST2 sun in FOV nominal = 1 ST2 sun in FOV = 1 Angular speed mode = 0 SS frozen = 1	Positive
SS.2.1	General failure Sun sensor – cross-check with two star-trackers	Angular speed mode = 0 SS general failure = 1	Positive
SS.2.2	General failure Sun sensor – cross-check with one star-tracker	ST1 sun in FOV nominal = 1 ST1 sun in FOV = 1 Angular speed mode = 0 SS general failure = 1	Positive
SS.2.3	General failure Sun sensor – no cross-check	ST1 sun in FOV nominal = 1 ST1 sun in FOV = 1 ST2 sun in FOV nominal = 1 ST2 sun in FOV = 1 Angular speed mode = 0 SS general failure = 1	Positive
SS.2.4	General failure Sun sensor	Validity SS = 0	Positive
SS.3	Sun not in FoV	SS sun in FOV = 0	Positive
SS.4	Malformed data package	Malformed package SS = 0	Positive
ST.1.1	Frozen star-tracker – cross-check with two units	Angular speed mode = 0 ST1 frozen = 1 (same for ST2)	Positive
ST.1.2	Frozen star-tracker – cross-check with one star-tracker	SS sun in FOV nominal = 0 SS sun in FOV = 0 Angular speed mode = 0 ST1 frozen = 1 (same for ST2)	Positive
ST.1.3	Frozen star-tracker – cross-check with one Sun sensor	ST2 sun in FOV nominal = 1 ST2 sun in FOV = 1	Positive

Test-case ID	Description	Scenario parameters	Result
		Angular speed mode = 0 ST1 frozen = 1 (same for ST2)	
ST.1.4	Frozen star-tracker – no cross-check	SS sun in FOV nominal = 0 SS sun in FOV = 0 ST2 sun in FOV nominal = 1 ST2 sun in FOV = 1 Angular speed mode = 0 ST1 frozen = 1 (same for ST2)	Positive
ST.2.1	General failure star-tracker – cross-check with two units	Angular speed mode = 0 ST1 general failure = 1 (same for ST2)	Positive
ST.2.2	General failure star-tracker – cross-check with one star-tracker	SS sun in FOV nominal = 0 SS sun in FOV = 0 Angular speed mode = 0 ST1 general failure = 1 (same for ST2)	Positive
ST.2.3	General failure star-tracker – cross-check with one Sun sensor	ST2 sun in FOV nominal = 1 ST2 sun in FOV = 1 Angular speed mode = 0 ST1 general failure = 1 (same for ST2)	Positive
ST.2.4	General failure star-tracker – no cross-check	SS sun in FOV nominal = 0 SS sun in FOV = 0 ST2 sun in FOV nominal = 1 ST2 sun in FOV = 1 Angular speed mode = 0 ST1 general failure = 1 (same for ST2)	Positive
ST.2.5	General failure star-tracker	Validity ST1 = 0 (same for ST2)	Positive
ST.3	Sun in FOV	ST1 Sun in FOV = 1 (same for ST2)	Positive
ST.4	Overheating	Temperature ST1 > 40 (same for ST2)	Positive
ST.5	Underheating	Temperature ST1 < -20 (same for ST2)	Positive
ST.6	Malformed data package	Malformed package ST1 = 1 (same for ST2)	Positive
IMU.1	Frozen gyroscope	Angular speed mode = 0 IMU gyro frozen = 1	Positive
IMU.2	General failure - gyroscopes	Angular speed mode = 0 IMU gyro general failure = 1	Positive
IMU.3	Malformed data package	Malformed package IMU = 1	Positive
AOCS.1	Hardware failure	Validity AOCS processor	Positive
AOCS.3	Malformed data package	Validity flag AOCS processor	Positive
DEP.1	Excessive tumbling rate	Omega > 30	Positive
MT.1	Command out of range	Command MT > 1	Positive
MT.2	No thrust	0<Command MT<1 No thrust MT = 1	Positive
MT.3	Locked output	Locked output MT =1	Partially positive (after 10 s)
MT.4.1	Propellant sensor failure - active	0<Command MT<1 Sensor failure propellant MT = 1	Positive
MT.4.2	Propellant sensor failure - idle	Sensor failure propellant MT = 1	Partially positive (10 s del)
MT.5.1	Valve sensor failure - active	0<Command MT<1 Sensor failure valve MT = 1	Partially positive (delay 10 s)
MT.5.2	Valve sensor failure - idle	Sensor failure valve MT = 1	Positive
MT.6.1	Overheating – active	0<Command MT<1 Temperature MT > 50	Positive
M.6.2	Overheating – idle	Temperature MT > 60	Positive
M.7.1	Underheating - active	0<Command MT<1 Temperature MT < 0	Positive
MT.7.2	Underheating – idle	Temperature MT < -10	Positive

Test-case ID	Description	Scenario parameters	Result
MT.8	Malformed data package	Malformed package MT = 1	Positive
SP.1	Solar panel – loss of power	Failure SP1 = 1 (same for SP2)	Positive
EPS.1	Malformed data package	Malformed package EPS = 1	Positive
EPS.2	Hardware failure	Validity EPS = 0	Positive
BT.1.1	Overheating – active	Battery mode = 1 Temperature battery > 70	Positive
BT.1.2	Overheating – idle	Temperature battery > 90	Positive
BT.2.1	Underheating – active	Battery mode = 1 Temperature battery < -30	Positive
BT.2.2	Underheating - idle	Temperature battery < - 50	Positive
BT.3	Over-discharged battery	DOD B1 > 80 (same for B2)	Positive
BT.4	Voltage measurement failure	Failure V measurement B1 = 0 (same for B2)	Positive
BT.5	DOD measurement failure	Failure DOD measurement B1 = 0 (same for B2)	Positive
COMM.1	No uplink to Mothership	Uplink failure = 1	Positive
COMM.2	No downlink from Mothership	Downlink failure = 1	Positive
DEP.2	No antenna deployment - electrical	Failure deployment antenna=1 DOD B1 > DOD nominal DOD B2 > DOD nominal	Positive
DEP.3	No antenna deployment - mechanical	Failure deployment antenna=1	Positive
DEP.4	No SP deployment - electrical	Failure deployment SP=1 DOD B1 > DOD nominal DOD B2 > DOD nominal	Positive
DEP.5	No SP deployment - mechanical	Failure deployment SP=1	Positive
SADA.1	Hardware failure	Failure SADA SP1 = 1 (same for SP2)	Positive
HEAT.1.1	No heating - active	Temperature RW < 0 RW heater failure = 1 (same for other units with heater)	Positive
HEAT.1.2	No heating – idle	RW mode = 1 Temperature RW < -10 RW heater failure = 1 (same for other units with heater)	Positive
HEAT.2.1	Locked output – active	Temperature RW > 50 RW heater failure = 1 (same for other units with heater)	Positive
HEAT.2.2	Locked output - idle	RW mode = 1 Temperature RW > 60 RW heater failure = 1 (same for other units with heater)	Positive
TSENS.1	Frozen temperature sensor	Frozen sensor RW1 =1 (same for other units with temperature sensors)	Positive
TSENS.2	General failure temperature sensor	General failure RW1=1 (same for other units with temperature sensors)	Positive