Politecnico di Torino

TU Delft

Master's Degree in Aerospace Engineering

Academic Year 2023/2024

Development and Verification of an Imaging System for PocketQube Satellites

Supervisors:
Prof. Sabrina Corpino, Politecnico di Torino
Res. Mehmet Şevket Uludağ, TU Delft

Co-supervisors:
Prof. Fabrizio Stesina, Politecnico di Torino
Prof. Stefano Speretta, TU Delft

Candidate:
Vincenzo Calabretta

To my family
and to each and every person who taught me something during these years.

Abstract

This thesis details the design, development, and verification of an imaging system for the TWIN-SAT satellite mission, currently being developed at TU-Delft. The mission incorporates two 3P PocketQube platforms in a connected configuration, with the imaging system tasked with capturing a sequence of images documenting their separation, serving both engineering analysis and outreach objectives. The project encompasses the entire system lifecycle: from conceptualization and design to development and verification. It builds upon lessons learned from previous missions, aiming to enhance acquisition and data handling capabilities within the limitations imposed by the PocketQube format. The initial phase involved a comprehensive analysis of the spacecrafts' separation dynamics, optimizing the imaging system's performance. This phase led to the selection of an appropriate COTS CMOS sensor and optics. Focus was placed on designing a payload data handling system capable of managing high image data throughput, ensuring continuous image acquisition, and facilitating real-time control and capture. The design and development process involves every aspect of the system including electronics and on-board firmware of the payload data handling system. A custom-developed board incorporates an ARM Cortex-M4 based MCU, a parallel CMOS sensor interface, a high-capacity and throughput SLC NAND memory subsystem, and interfaces for the satellite bus. The onboard firmware includes bare-metal drivers tailored for the payload hardware, the RTEMS real-time operating system, and application software. The drivers, along with a bootloader engineered for the custom board, facilitated the creation of a Board Support Package. This support package enables the use of the RTEMS RTOS with the STM32L4+ microcontroller family, which was previously unsupported. The thesis activities resulted in the production of a functioning Development Model that underwent a verification process to assess its performance and compliance with mission-derived system requirements. The process yielded positive outcomes, guaranteeing the achievement of mission objectives and facilitating the progression towards the development of a ProtoFlight Model.

Contents

List of Figures

List of Acronyms

ACS Attitude Control System

AFOV Angular Field of View

AHB Advanced High-performance Bus

AIV Assembly Integration and Verification

ASIC Application-Specific Integrated Circuit

BGA Ball Grid Array

BM Breadboard Model

BSP Board Support Package

CDH Command and Data Handling

CFE Core Flight Executive

CFS Core Flight System

CMOS Complementary Metal-Oxide-Semiconductor

CMSIS Common Microcontroller Software Interface Standard

CoM Center of Mass

COMM-SYS Communication system

COTS Commercial-Off-The-Shelves

CPU Central Processing Unit

DCMI Digital Camera Interface

DM Development Model

DMA Direct Memory Access

DRAM Dynamic Random Access Memory

ECI Earth Centered Inertial

ELF Executable and Linkable Format

eMMC Embedded MultiMediaCard

EPS Electric Power System

ESA European Space Agency

FBS Functional Breakdown Structure

FDIR Fault Detection Isolation and Recovery
FIFO First-In, First-Out
FFC Flexible Flat Cable
FOV Field of View
FOVB Field of View Budget
FMC Flexible Memory Controller
FRAM Ferroelectric Random Access Memory
GDB Gnu Debugger
GPIO General Purpose Input/Output
GPL Generic Public License
GNU GNU's Not Unix
HAL High Abstraction Layer
H-AFOV Horizontal Angular Field of View
HFOV Horizontal Field of View
HSL Horizontal Sensor Length
HSYNC Horizontal Synchronization
IAS Image acquisition Subsystem
ISR Interrupt Service Routine
JPEG Joint Photographic Experts Group
JUICE JUpiter ICy moons Explorer
LEO Low Earh Orbit
LL Low Level
LVLH Local Vertical Local Horizontal
LVDS Low-Voltage Differential Signaling
LQFP Low-profile Quad Flat Package
MCU Microcontroller Unit
MIPI-CSI Mobile Industry Processor Interface Camera Serial Interface
MCL Multi Level Cell
MTF Modulation Transfer Function
NASA National Aeroanutics and Space Administration
ODE Ordinary Differential Equation
PBS Product Breakdown structure
PCB Printed Circuit Board
PDP Power Distribution and Protection Subsytem

PCDH Payload Command and Data Handling

PL Payload

PLL Phase-Locked Loop

RAM Random Access Memory

RCC Reset and Clock Control

RF Radio Frequency

RTEMS Real-Time Executive for Multiprocessor System

RTOS Real-Time operating system

S/C Spacecraft

SEE Single Event Effect

SNR Signal to Noise Ratio

SoC System-on-Chip

SR Stakeholder Representative

SWD Serial Wire Debug

SLC Single Level Cell

SMS Static Memory Subsystem

SOP Small Outline Package

SRAM Static Random Access Memory

SSDV Slow Scan Digital Video

SVSH Sensor Horizontal Sensor Vertical

TIR Time In Range

TIV Time In View

TRL Technology Readiness Level

TSOP Thin Small Outline Package

UART Universal Asynchronous Receiver-Transmitter

V-AFOV Vertical Angular Field of View

VFOV Vertical Field of View

VSL Vertical Sensor Length

VSYNC Vertical Synchronization

WD Working Distance

Chapter 1

Introduction and scope of the project

This Master's thesis includes the design, development, assembly and verification of an imaging system compatible with the PocketQube satellite format, realized with the support of the Space Systems engineering group at TU-Delft.

The imaging system has been designed to satisfy the payload requirements of the Delfi-TWIN mission. The mission is going to be carried out by two PocketQube spacecrafts that are currently being Developed at TU-Delft, with contribution of students from Politechnic University of Turin.

1.1 Programmatic considerations and mission concept

The Delfi-Twin mission is an academic PocketQube mission that aims to achieve a set of research, education, and outreach goals.

- To test in the low earth orbit environment technologies for formation flight and autonomous operations.
- To produce images to be used for outreach purposes.
- To provide a project where students can acquire hands-on experience working on the project.

The high level mission concept consists in the deployment of two 3P PocketQube satellites in an attached 6P configuration. The S/C stack is going to be deployed from a 6P deployer and proceed to an initial commissioning phase. At the end of the S/C commissioning, the first mission phase will take place, that includes the separation of the two spacecrafts and the successive controlled distancing, until the formation flight configuration is reached. Successively, the second mission phase will consist of maintaining

the formation flight configuration via autonomous relative position control making use the use of differential drag.

Each of the two spacecraft will feature an imaging system to acquire an image sequence of the separation. The scope of this thesis is the conceptualization, design and development of the imaging system, inlcuding the development and verification of a Development Model.

Interfacing with the available stakeholders has been an essential step to determine the instrument characteristics and eventual system constraints. By interfacing with TU-Delft Space Systems engineering group, the following information about the project, and its stakeholders, has been gathered:

- TU-Delft university is the prime stakeholder of the project, it covers the launch costs and its interest is in the outreach and education outcomes of the mission.
- The TU-Delft Space Systems engineering research group is the S/C developer and is also a stakeholder of the project. The research group plans to utilize the mission to conduct research and validate technologies for a set of research topics: Formation flying, control of satellite attitude and position via differential drag, autonomous operations and more.
- Other research groups at TU-Delft plan to utilize the platform to test payloads in the low earth orbit environment.

Consequently, a mission concept has been developed that is able to achieve the mission goals of the stakeholders, and a set of payloads has been conceptualized to achieve each goal.

The development of the payload has overcome numerous challenges, some of which programmatic in their nature and derived from the mission development context.

To achieve the education goal, most of the work packages necessary for the realization of the mission are meant to be performed by students. This is an aspect that strongly influenced the organization of the project, since availability of students make a traditional project organization difficult to implement.

To enable parallelization of the development, the activities have been organized in two segments, a first segment dedicated to Mission Analysis and second involving System design and development.

Since the design process of the systems requires input from the Mission analysis function, the parallelized approach necessitates that the inputs of the sub-systems design process are known independendly from the mission analysis process.

This design assumption has been used to generate a plausible system budget allocation and high level requirements for the imaging system using as reference is the budget developed for the Delfi-PQ mission, due to the similarities in format, hardware and orbit of operation.

While an approach based on these assumptions can provide preliminary subsystems performance requirements, it cannot possibly provide instrument performance requirements, before the completion of the mission analysis phase.

This issue has been aggravated by the lack of a defined outreach strategy from the prime stakeholder.

Therefore, the lack of mission requirements created a situation where it has been not only necessary to conceptualize the observational strategy, but also to interface with

the prime stakeholder to understand and formalize the outreach needs.

Consequently, it has been agreed to include in the schedule of the activities an initial period of three weeks for a preliminary mission and instrument feasibility study, with the objectives:

- Formalization of the stakeholder outreach needs.
- Development of the observation requirements.
- Definition of the observational strategy.
- Development of the instrument performance requirements.
- Definition of the payload architecture.

The rest of the thesis timeframe is reserved for the design, development, production and verification of the payload subsystem. The key dates for the project top level schedule are shown in Table below. The schedule has been developed to avoid blocks in the development activities, with particular attention to the effect of procurement on the overall schedule. Specifically, the activities have been scheduled so that hardware and software development could have been executed while waiting for fruition of the successive procurement rounds.

Activity	Schedule
Preliminary mission and instrument feasibility study	1st week October - 3rd week October
Component selection	4th week October
small Initial procurement of test components for pre-development activities	4th week October
small Design of the software architecture and software development plan	1st week November
Hardware design for Development Model	2nd week November - 3rd week November
Hardware procurement for Development Model	3rd week November
Low level software development on Breadboard Model	4th st week November - 1st st week December
Development Model Assembly	2nd week December
Software development of RTEMS board support package	2nd week December - 1st week January
High level software development on Development Model	2nd week January - 4th week January
Development Model Verification	1st week February

Table 1.1: Schedule (indicative) for the project development schedule

1.2 The PocketQube format and derived systems constrains

The PocketQube satellite format is defined in a set of standards born from an initial conceptualization from professor Robert J. Twiggs, as a result of a collaboration between Morehead State University and Kentucky Space [28]. The standard has been defined in a further set of specifications by AlbaOrbital, TU-Delft and GAUSS Srl [2].

In general, PocketQube are inexpensive platforms developed making large use of COTS microelectronics components and composed of cubic format units of size 50 mm x 50 mm, and with a maximum mass per unit of 250g.

The PocketQube structure is characterized by the presence of a structural backplate, used as rail in the pod deployment system.

Amongst PocketQubes the most common form factor is the 3P configuration, consisting of three format units. The 3P factor is commonly used thanks to the ability to fit two triple junction solar cells over the length of the spacecraft.

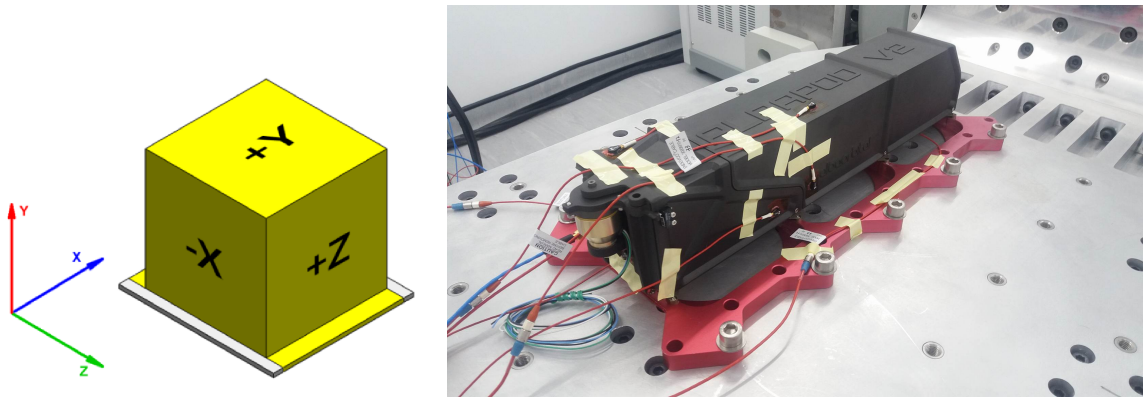


Figure 1.1: On the left, an Example of a PocketQube unit, with the inclusion of conventional axis specification. [2]. On the right, the AlbaPod 6P PocketQube deployer, from Alba Orbital [16]
(Image from the manufacturer's website).

The common solar cell placement configuration are body mounted solar cells like for the DelfiPQ spacecraft from TU-Delft [18], or deployable solar panels like for the Unicorn-2 satellite platform from AlbaOrbital (Figure 1.2).

Currently, power generation is the greatest factor that limits capabilities for most CubeSat or PocketQube satellites. This is because thanks to the great availability of small package electronics, it is possible to implement many of the S/C subsystems (EPS, C&DH, COMM-SYS) in a very compact format, making volume and mass less of a concern.

Sufficient power generation is essential to support performant payloads, that defines the capability of a given platform format. Communication and navigation payloads rely on the reception and transmission of RF signals and are limited in the duration and power of the transmission by the power generation. Earth observation systems are also relatively power hungry due to factors like:

- Power necessary for operation of the photodiode array (for CMOS sensors) and readout electronics.
- Power consumption of the sensor handling electronics. For COTS CMOS sensors this often includes an integrated MCU or ASIC that performs image correction, de-mosaicization, white balance, automatic exposure control and other image processing functions.
- High power consumption of high frequency MCU or SoC, required to support high speed serial interfaces used by most imaging sensors (Like SoC based on the ARM Cortex-A series of processor cores).

In regard to power generation, the Delfi-TWIN platforms is being developed with the same configuration of Delfi-PQ, a 3P PocketQube with body mounted triple-junction solar cells.

The power requirements assigned to the imaging payload are the following:

- The mean power consumption shall be equal or less than 150mW

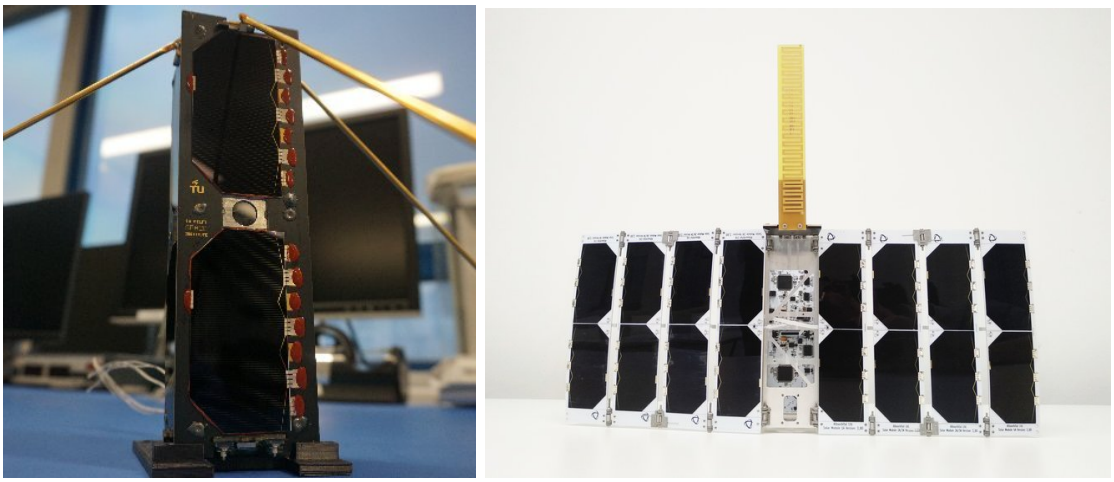


Figure 1.2: Example of two solar cell configurations: on the left Delfi-PQ that features body mounted solar cells, on the right the Unicorn-2 platform from AlbaOrbital[16](Image from the manufacturer's website), that uses deployable solar panels.

- The peak power consumption shall be equal or less than 1.5W

The mean power consumption takes in account the duty cycle of the payload and considers power consumption before the subsystem voltage regulators, therefore considering the power absorbed from the unregulated S/C power bus. The maximum power consumption is defined as a consequence of Delfi-PQ EPS, that has been sized for the activation of the hot-knives for antenna deployment.

At the beginning of the design process a configuration requirement has been received regards the utilization of a specific MCU as command and data handling hardware. The Delfi-TWIN spacecraft avionics make use of a distributed system architecture, similar to the one implemented in Delfi-PQ. The distributed C&DH system of Delfi-PQ makes use of a common MCU (MSP432P401R) [18] in every S/C subsystem, to facilitate the reuse of software and hardware between the subsystems. Each of the subsystems MCU communicate on a RS485 satellite bus and is responsible for subsystem operation and FDIR.

A SWD debug interface between the distributed elements of the system allows for firmware update of the elements of the system and needs to be exposed in the imaging system as well.

During previous activities in the scope of the Delfi-TWIN project, the STM32L496 microcontroller has been selected as the common MCU for the Delfi-TWIN subsystems. During the development of the system configurations candidates and the feasibility study, it has been determined that the utilization of the STM32L496 would lead to unsatisfactory performance due to a limited availability of peripherals, therefore an MCU of the L4+ family of the same product line from ST has been selected, featuring higher clock speed and a greater number of serial peripherals.

Chapter 2

Mission and Instrument feasibility study

The first step taken in the analysis activities has been to interface with the prime stakeholder to analyze and formalize its needs.

In particular the mission management team composed of Stefano Speretta and Mehmet Şevket Uludağ provided invaluable support in this regard, acting as interface between the autor and the prime stakeholder.

For simplicity in the rest of the document, the satellite hosting the imaging payload is referred as "Observer", while the observed satellite is referred as "Target". The separation and successive distancing of the satellites after the removal of the mechanical link between them is referred as the "separation event".

In the kickoff meeting it has been possible to identify the following high level functional requirement:

- The imaging system on the observer satellite shall acquire a sequence of images of the target satellite during the separation event.

The successive step has been to produce quantitative instrument performance requirements that would allow achievement of the outreach goal. The imaging system performance requirements are inherent to two areas: Quality of the captured image and acquisition frequency.

After a preliminary overview of the performance of the CMOS sensors that are both available in the market and compatible with the required MCU, a successive meeting has been organized with the stakeholder's representative (SR) to define the observation objectives. The discussion of the desired image quality has been guided making use of a test image to provide an empiric tool to quantify the desired target sampling performance. The test image is displayed in figure 2.1 and consists of a white canvas composed of 1900x1200 pixels containing a set of test boxes. The pixel count of the image is the same as the highest pixel count amongst the sensors available in the market (and specifically from component distributors) compatible with the STM32L496 DCMI interface.

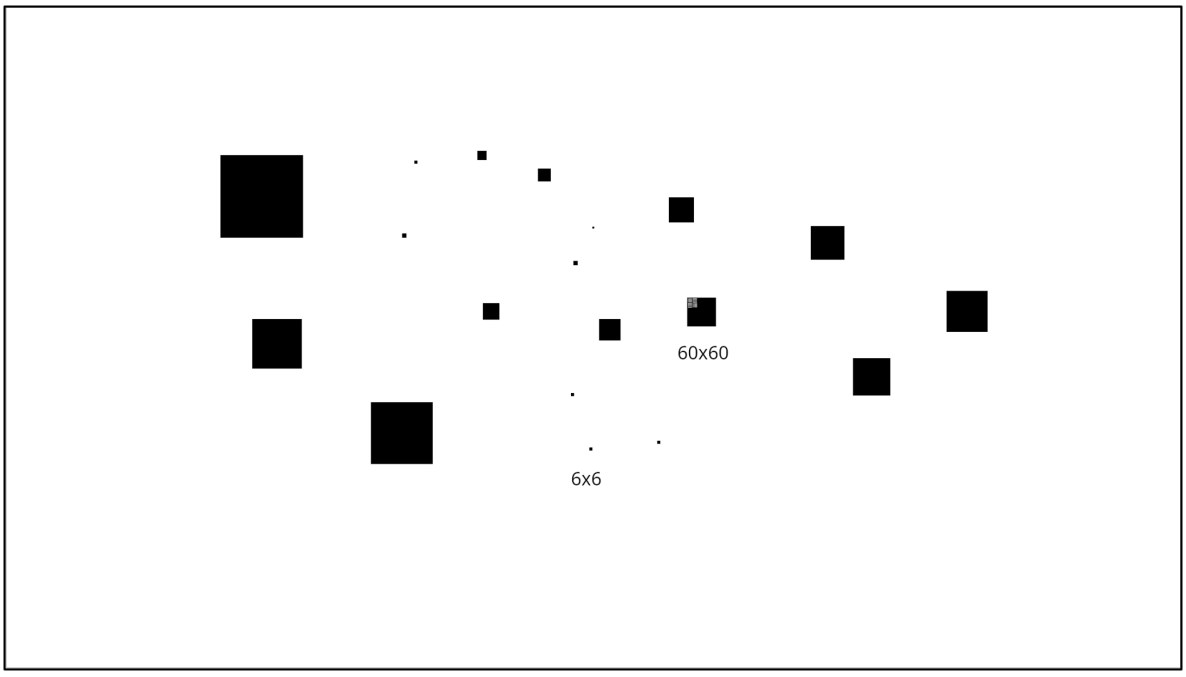


Figure 2.1: The test image utilized to discuss with the stakeholder representatives the desired characteristics of the images to be used for outreach.

The test image contains a set of dark squares that represent possible sizes of the target in the image. The SR has been asked to select the smallest box that could be useful for the outreach purpose. Each box contains a pattern that represents the smallest feature that could be sampled considering a spatial cutoff frequency that would only be limited by the sensor's sampling. Naturally, all the parties shared the understanding that the contrast of the imaged features would be less than the one depicted with solid colors in the test image and this information has been shared with the SR.

The SR selected a box of pixel size 60x60. From this information, it is possible to generate a set of quantitative criteria, so that it is possible to univocally determine if a captured frame is adequate for the outreach purpose. Therefore, the observability the following observation requirements are set of conditions:

- A valid image shall contain the entire target satellite in frame.
- A valid image shall sample the target satellite with a minimum of 3600 samples.
- A valid image shall contain resolved features of the target satellite of at least 2 mm of characteristic length.

2.1 The observation scenario

The separation event consists in the separation of the two S/C. When the mechanical link is released, a separation spring applies an impulse that results in a relative separation speed of $10 cm/s$. The separation velocity has been determined by the mission analyst to ensure adequate separation of the platforms and to avoid collisions in the

successive orbital periods.

The application of an impulse, in conjunction with the change in inertia of the two bodies will result in an expected tumbling motion. The ACS system of the spacecraft is not going to be sized to stabilize the tumbling motion before the two satellites will be too far to be imaged. Therefore, it is necessary to account the tumbling motion of the observer satellite during the design of the imaging system.

This imposes a set of constrains, specifically:

- It is necessary to determine the optimal combination between the horizontal and vertical AFOV and maximum working distance that would allow capturing of as many valid frames as possible of the target satellites.
- The imaging sensors selected shall be able to acquire images of the target satellite with a limited degradation of the SNR caused from the rotation of the observer satellite. In practice this means designing an imaging system able to acquire with rapid exposure time and using sensors with electronic global shutter.
- The requirement of being able to achieve the mission objectives even in case of failure of the ACS system implies that the attitude initial conditions cannot be assumed and an appropriate number of cases needs to be studied to be able to demonstrate the suitability of the design.

Moreover, the application point of the separation impulse is not yet determined. Another task of the preliminary observation analysis is to determine the best point of application of the separation impulse, to maximize the acquisition output.

To analyze the mission scenario, a simulator of the two satellites dynamics has been developed. The simulator has the purpose of simulating attitude and position of the two spacecrafts before and after the separation. A set of simulation cases have been studied, that include combinations of different initial conditions and different placement of the separation spring. The purpose is evaluating significant cases to gather information to guide the successive design steps.

The position-attitude-time state vectors obtained from the different simulation scenarios allows to evaluate the performance of a given imaging system in terms of valid frames that would be able to capture. By calculating the number of valid image frame captured by each valid combination of imaging system parameters, it is possible to determine the imaging system characteristics that maximize the number of valid images that it can capture.

Thanks to this method, it has been possible to produce instrument requirements that are able to maximize the outreach value of the mission, guiding the development of a payload that would be able to achieve the best possible satisfaction of the mission goals.

2.2 Satellite dynamics simulator

The satellite dynamics simulator is a computational software that has been developed ad-hoc for the project to determine position and attitude of the spacecrafts before and after the separation. The model has been defined in the Local Vertical Local Horizontal (LVLH) reference frame.

The LVLH reference frame is a quasi-inertial reference frame often used for proximity operations, thanks to the possibility to simplify in the LVLH reference frame the general 2-body position dynamics equation. The reference frame is defined as follows: the +Z axis is directed in the nadir direction, the +Y axis is perpendicular to the orbit plane with direction opposite to the angular momentum vector and +X axis is positioned to complete the right-handed triad so that it is horizontal in the orbit plane and in the direction of orbit travel.

When the frame is used to represent proximity operations, usually the LVLH frame is centered in the target spacecraft. In the scope of the analysis, the LVLH coordinate frame is centered in the CoM of the attached satellites at the initial condition. After separation, the two satellites will begin their separation relative to the origin of the reference frame.

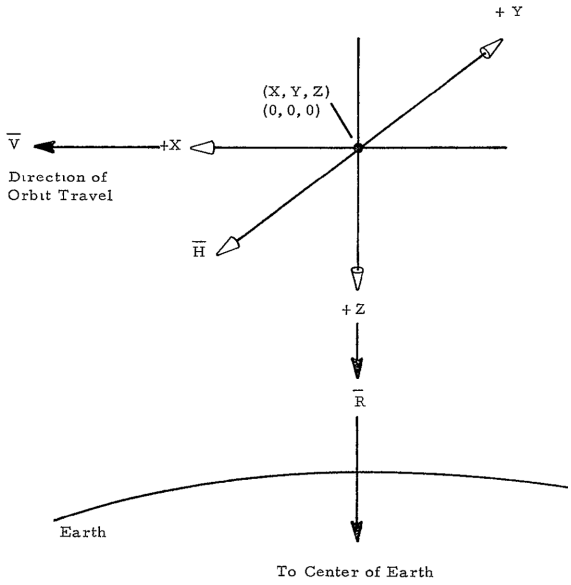


Figure 2.2: Diagram of the LVLH reference frame [10]

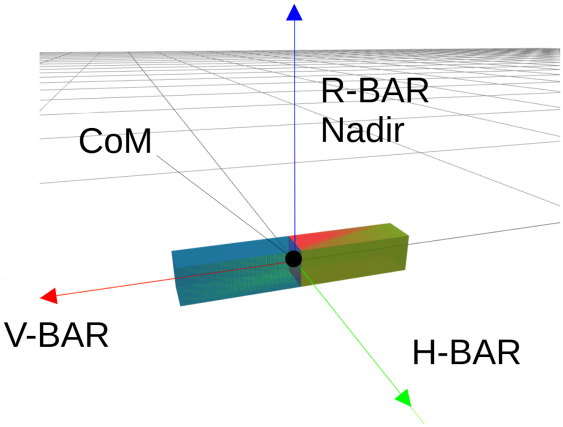


Figure 2.3: Placement of the origin of the LVLH reference frame in respect to the simulation bodies. Blue: observer satellite. Red target satellite. Superposed to the two bodies, a third green body represents the two satellites in the attached configuration.

2.2.1 Simulation subjects

The dynamic simulation scenario includes three bodies:

- **A body (Observer satellite):** This body represents the observer satellite, which is equipped with the imaging instrument. The inertial properties of the observer satellite have been assumed from the mass distribution of the Delfi-PQ satellite. Specifically, the PocketQube has been considered a parallelepiped with an homogeneous mass distribution and total mass equal to the design mass of Delfi-PQ. Consequently, the total mass the $m = 0.549kg$ and with dimensions $l = [0.15, 0.05, 0.05]m$ results in a diagonal tensor of inertia, with main diagonal $j_a = [0.0002287, 0.0011437, 0.0011437]kgm^2$.
- **B body (Target satellite):** This body represents the target satellite, it presents

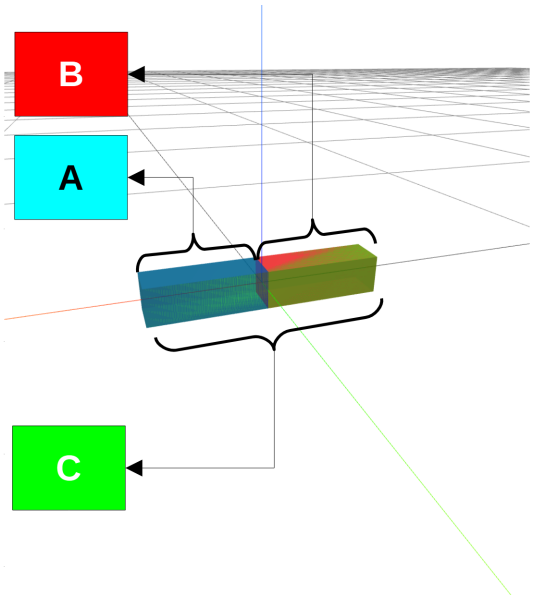


Figure 2.4: Simulated objects reference nomenclature.

the same inertial properties as the observer satellite,
$j_b = [0.0002287, 0.0011437, 0.0011437] kgm^2$

- **C body (Lead satellite):** This body represents the two PocketCubes in the attached configuration. It is referred to as the lead satellite. Its motion is defined at t_0 by the initial conditions and the simulation reference frame is centered on its initial CoM. The inertia tensor of the C body has been derived from the ones of the two constituent bodies (A and B) $j_c = [0.0004575, 0.0084637, 0.0084637]$.

2.2.2 Simulation phases

The simulation is composed of two phases, respectively before and after the separation of the two satellites:

Spin Phase During the spin phase, only the C body position and attitude dynamics are simulated, as it represents the A and B bodies linked together. The position and attitude of the A and B bodies are derived directly from the one of the C body, as they are represented by it. In this phase, the C body is free to rotate for a definite amount of time, after which the successive phase is simulated.

Separation Phase The separation phase consists in the simulation of the dynamics of the A and B bodies. The initial conditions of the simulation are determined from the last state of the spin phase simulation. The separation includes the unlinking of the two satellites and the application of the separation force generated by an elastic actuator. The applied total impulse can be applied in the direction of the CoM axis or in the off-axis interface between the two satellites, which is the backplate. The actuation impulse generates a variation of momentum and angular momentum, which is taken into account in the determination of the initial conditions of the distancing phase.

2.2.3 Dynamics model

The position of each spacecraft in the analysis is determined via the solution of the Hill equation, derived from the linearization of the complete nonlinear two body problem [20]. The Hill equation solved in the simulation domain is here reported:

$$\begin{aligned} \ddot{x} &= \frac{F_x}{m_s} + 2\omega\dot{z}, \\ \ddot{y} &= \frac{F_y}{m_s} - \omega^2 y, \\ \ddot{z} &= \frac{F_y}{m_s} - 2\omega\dot{x} + 3\omega^2 z, \end{aligned} \tag{2.1}$$

where ω is the mean orbital rate, $[x, y, z]$ is the position of the body in the Local Vertical Local Horizontal (LVLH) reference frame,$F = [F_x, F_y, F_z]$ is the forces applied on the spacecraft in the LVLH reference frame.

The attitude of each of the bodies is simulated via Euler equations and quaternion kinematics. The Euler equations are given by:

$$\dot{\omega_B} = I^{-1}[T_B - \omega_B^{\times} I \omega_B], \tag{2.2}$$

where T_B is the total torque applied to the spacecraft in the spacecraft body reference frame, ω_B is the angular velocity in the spacecraft and $\omega_B^{\times}$ is the cross product matrix of the ω_B vector.

At each time step, the attitude of the spacecraft is obtained from the angular velocity via the use of quaternion kinematics. Specifically, the time-derivative of the quaternion q_{ba} is computed such that it rotates a reference frame "a" into alignment to the reference frame "b" in which the angular velocity of "b" with respect to "a", and represented in "b", is $\omega_{ba,b}$. In the scope of the solution of this equation the "b" reference frame is the spacecraft reference frame, while the frame "a" is the LVLH reference frame.

$$\mathbf{q}_{ba} = \frac{1}{2}(\omega_{ba_b})\mathbf{q}_{ba} \tag{2.3}$$

$$\dot{\mathbf{q}}_{ba} = \frac{1}{2} \cdot \begin{bmatrix} 0 & -\omega_{ba,b,x} & -\omega_{ba,b,y} & -\omega_{ba,b,z} \\ \omega_{ba,b,x} & 0 & \omega_{ba,b,z} & -\omega_{ba,b,y} \\ \omega_{ba,b,y} & -\omega_{ba,b,z} & 0 & \omega_{ba,b,x} \\ \omega_{ba,b,z} & \omega_{ba,b,y} & -\omega_{ba,b,x} & 0 \end{bmatrix} \cdot \mathbf{q}_{ba} \tag{2.4}$$

To handle the different bodies the simulation is organized as follows: The attitude and position dynamics of the C body are calculated considering the simulation initial conditions. This first simulation, that constitute the first phase, lasts an arbitrary duration (time necessary for two rotations around the y_B axis). Successively, the final state vectors of the C body is used to determine the initial conditions for the second phase simulation, consisting in the solution of attitude and position ODE for the bodies A and B.

The initial angular velocity of the A and B bodies is calculated via conservation of angular momentum, that considers the change in inertial properties between the attached configuration and the detached, two bodies, configuration.

Due to the reduction of total inertia due to the different mass distribution, the result is an increase of spin rate for the two spacecrafts, independently of the angular momentum variation introduced by an offset separation spring.

The angular velocity of the observer spacecraft is an essential parameter in determining the feasibility of the observation. The presence of rotation introduces noise that is dependent on the integration time of the sensor (blur).

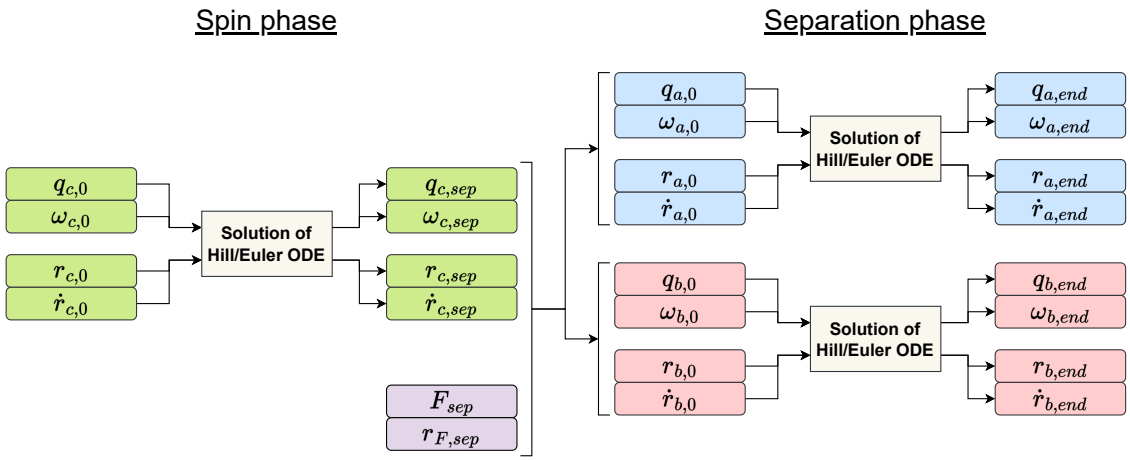


Figure 2.5: High level diagram of the simulation

2.2.4 Scenario characteristics

The C body is assumed to start in a circular orbit of arbitrary inclination with an altitude of 400Km. The characteristics of this initial condition are summarized in Table 2.1.

Symbol	Name	Quantity
z_c	Orbit altitude	400 Km
ω_{c0}	Orbit angular velocity (mean motion)	$\omega_{c0} = (\mu/(r_{earth} + z_c)^3)^{0.5}$
$\omega_{lvlh,0}$	Angular vel. of the LVLH rf. relative to ECI rf.	$\omega_{lvlh,0} = [0, -\omega_{c0}, 0]$

Table 2.1: Initial condition assumptions

2.2.5 Cases

For brevity, a subset of the simulation cases that have been studied is presented, consisting of the simulation cases in table 2.6. The simulation cases present different initial angular velocity for the satellite in the attached configuration and explore the possibility of having a separation spring in line with the axis that goes through the CoM

of the two satellites or positioned in the separation plate. In the first case there is no change in angular momentum, while in the second, the application of the spring impulse generates a change of angular momentum for each spacecraft.

Simulation cases		
Case ID	Initial conditions	separation spring offset
1	ω0 = [0,2,0] rad/s	yes
2	ω0 = [0,2,0] rad/s	no
3	ω0 = [0,5,1] rad/s	yes
4	ω0 = [0,5,1] rad/s	no
5	ω0 = [0,10,5] rad/s	yes
6	ω0 = [0,10,5] rad/s	no

Figure 2.6: Simulation cases discussed in this document

2.2.6 Simulator tech

The separation dynamics simulator has been developed using the Julia programming language [4], making use of a set of a set of open source packages. In particular the solver used is a Rosenbrock-Wanner method (Rodas5P) [22], implemented in the Differetialequations.jl Julia package [17]. The Differentialequations.jl package provides a set of interfaces towards ODE solvers.

A visualization engine has been built that provides real time 3D visualization of the results of the simulation. The visualization engine makes use of the Julia library Meshcat.jl [19] based on Three.js [5]

2.2.7 Analysis results

This section contains the results of the simulation cases. The graphs on the left show the position of the observer and target satellite and the ones on the right indicate each spacecraft attitude. These results are used successively to visibility of the target satellite from the observer as a function of the imaging system characteristics in a successive section.

The separation event is visible in both graphs. It is possible to observe how the rotation of the lead satellite (and therefore the initial conditions of the simulation) influence considerably the velocity of the observer and target satellites after the separation. This is something that needs to be accounted for in the mission analysis phase: The paramount necessity to avoid collisions between the two satellites requires that the separation scenario happens in a condition where the rotation of the lead satellite would lead to a safe release of the two S/C, without risk of successive collision. Since the release condition is so dependent on the initial condition of the separation, it is necessary that the attitude of the lead satellite is controlled, or at least precisely measured, to be able to assess if there are the conditions for a safe separation or not.

In regard to the attitude, it is possible to observe the change in angular velocity due to the separation of the two bodies, and of course the angular velocity is much higher when the separation spring is placed in line with the structural backplate.

In these simulations it has been considered that the separation spring would apply an angular impulse constructive in respect to the angular momentum possessed by the Observer satellite. This assumption is not necessarily true, since the initial angular velocity can be manifested in both directions and not necessarily in the one assumed in the simulation. Moreover, the lead S/C could be tumbling with an angular veloity on both of the major axis of inertia (Y_B,Z_B).

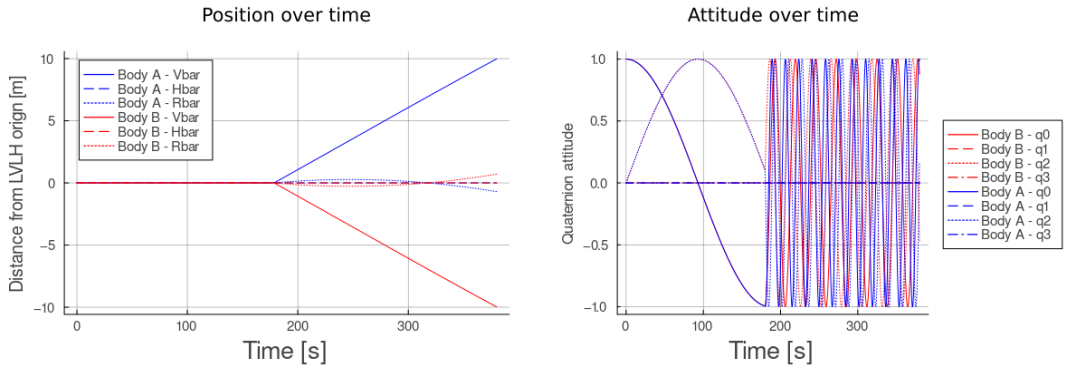


Figure 2.7: Simulation case 1: $[\omega_0]_B = [0, 2, 0]deg/s$, offset separation spring

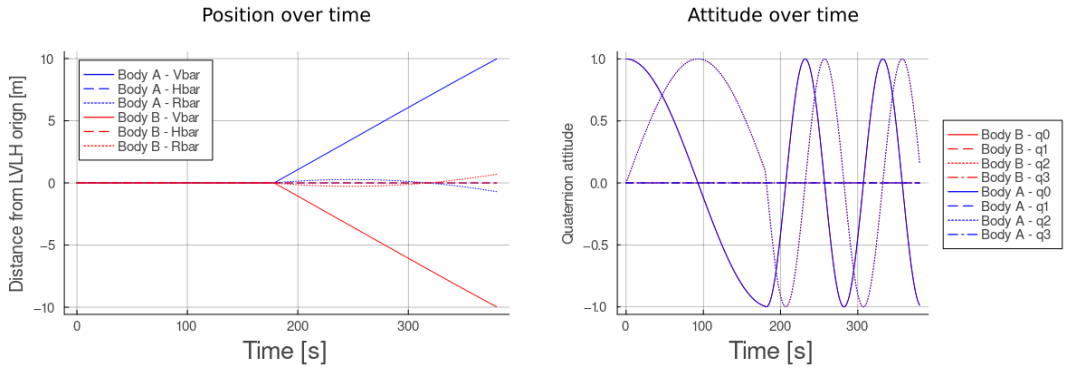


Figure 2.8: Simulation case 2: $[\omega_0]_B = [0, 2, 0]deg/s$, separation spring in line with CoM.

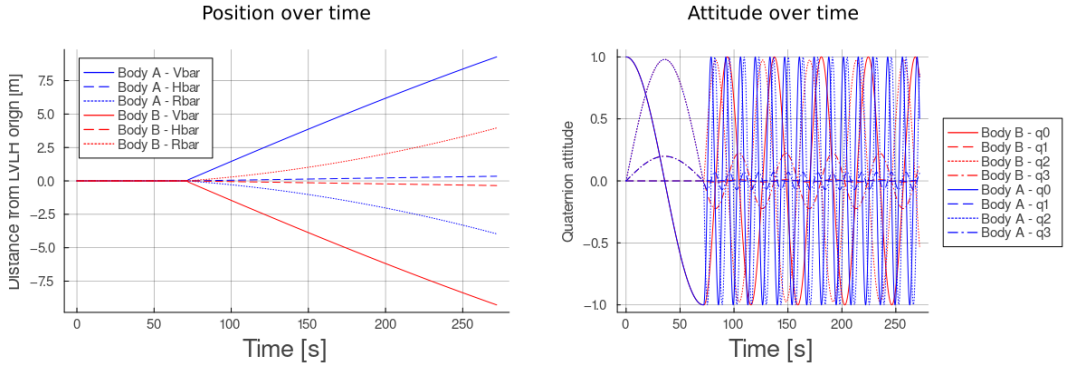


Figure 2.9: Simulation case 3: $[\omega_0]_B = [0, 5, 1]deg/s$, offset separation spring.

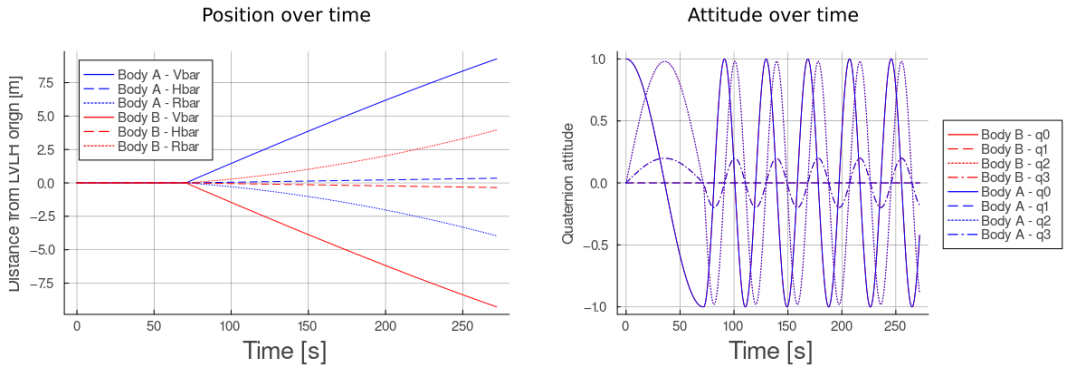


Figure 2.10: Simulation case 4: $[\omega_0]_B = [0, 5, 1]deg/s$, separation spring in line with CoM.

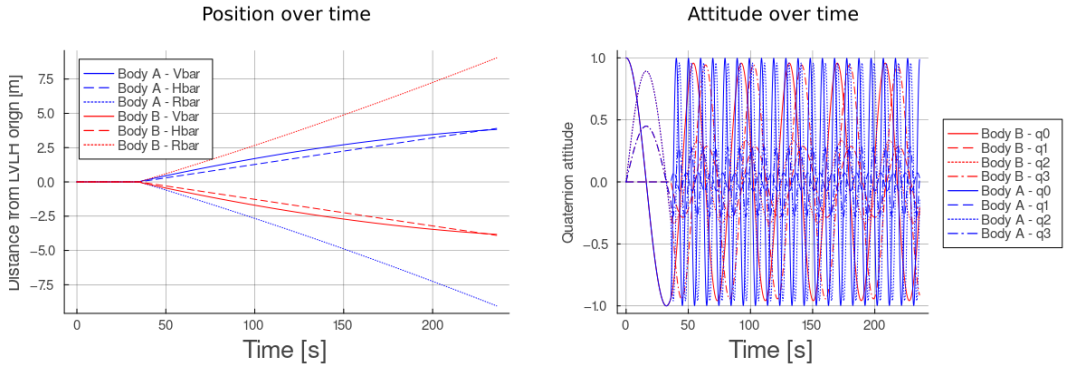


Figure 2.11: Simulation case 5: $[\omega_0]_B = [0, 5, 1]deg/s$, offset separation spring.

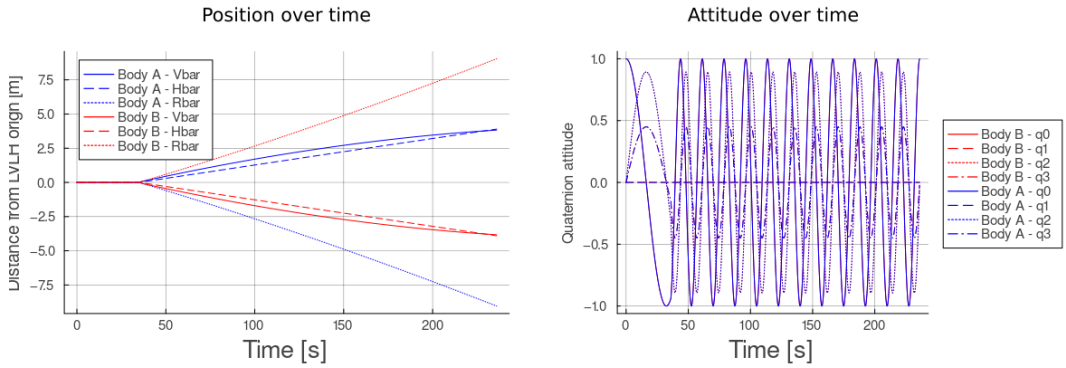
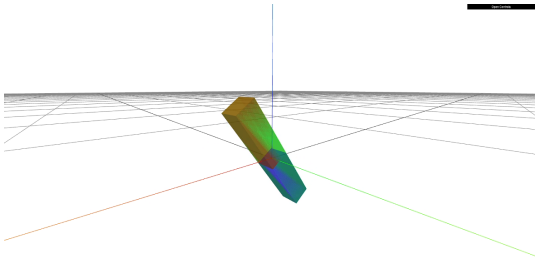
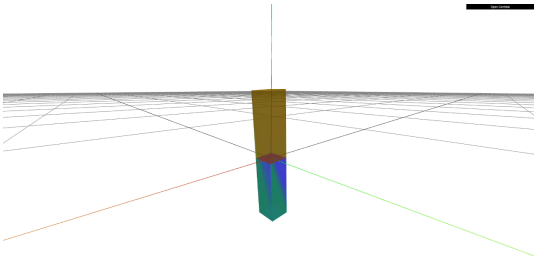


Figure 2.12: Simulation case 6: $[\omega_0]_B = [0, 5, 1]deg/s$, separation spring in line with CoM.

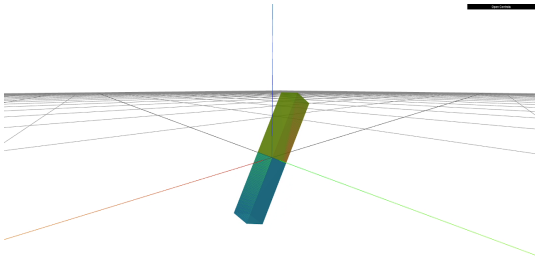
Visualization of case 5



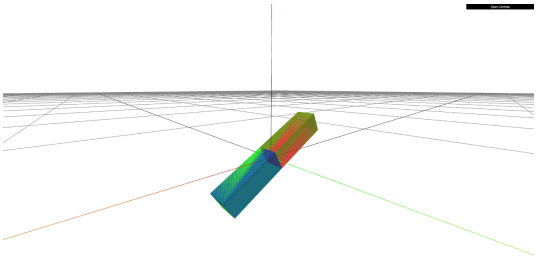
(a) Visualization of simulation case 5, frame 80



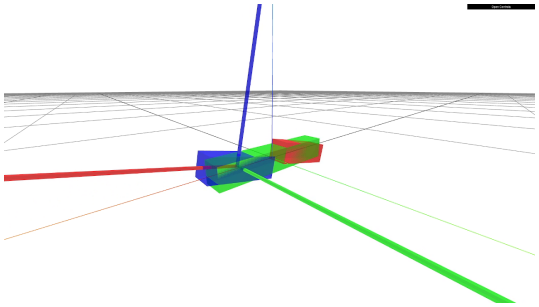
(b) Visualization of simulation case 5, frame 90



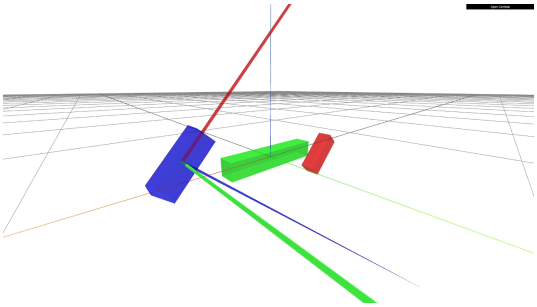
(c) Visualization of simulation case 5, frame 100



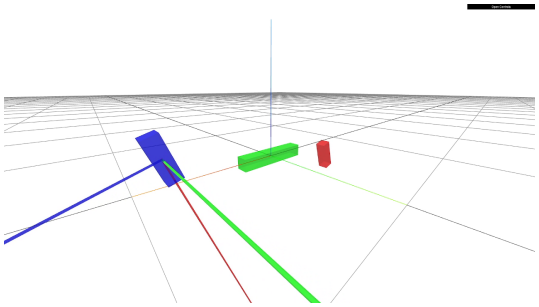
(d) Visualization of simulation case 5, frame 110



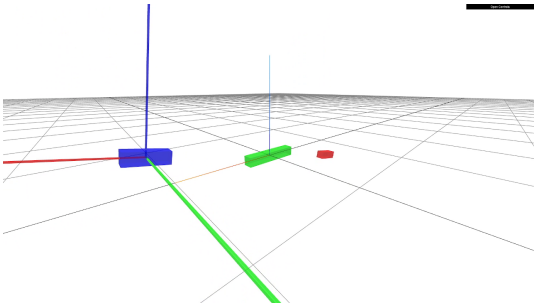
(e) Visualization of simulation case 5, frame 120



(f) Visualization of simulation case 5, frame 130



(g) Visualization of simulation case 5, frame 140



(h) Visualization of simulation case 5, frame 150

2.3 Overview of the image acquisition chain

To analyze the number of valid frames that can be captured, the whole acquisition chain needs to be considered. The acquisition chain is defined as the set of processes and relative elements that allow for the generation of the data product.

The properties of the images captured, and therefore the perceived quality, are solely dependent on the acquisition segment of the system, while the maximum acquisition frequency depends both on the sensor and the data handling subsystem.

When evaluating the performance of an imaging system, both the optics and the sensor characteristics concur to the properties of the captured image. It is necessary to analyze both elements of the acquisition system holistically.

2.3.1 Image formation

The image formation process consists in the focusing of the incoming radiation on the sensor array. In the observation scenario that is being considered, the radiation gathered by the optics is assumed to originate from Lambertian Scattering of the solar radiation on the surface of the target satellite. This is a simplification and the validity of the assumption depends on the surface of the target satellite.

The selection of a specific test box in the test image from the SR allows characterizing the target in terms of pixel coverage and feature resolution in the object space. It is assumed that the test box represent the +x face of the target PocketQube, since it is the smallest profile of the tumbling target PocketQube that can be imaged and therefore the worst case scenario. The pattern that identifies the smallest detail of interest is composed of alternating white and black stripes with a spatial period of 1/30 of the side of the square. Since the assumed size of the +X PocketQube panel in the object space is 45mm, this means that the spatial frequency of the significant feature in the object space is:

$$k_{x,obj} = k_{y,obj} = \frac{30lp}{45mm} = 0.75lp/mm \tag{2.5}$$

and would be equivalent to a feature size of $1.33mm$.

The target in the object space is imaged by the camera lenses, forming the image in the image space.

The primary magnification is defined as ratio of image height to object height and describes the ratio between the imaged scene size in the image space and the size of the target in the object space [27].

$$m = h_{img}/h_{obj} \tag{2.6}$$

The magnification is considered to be linear, and is applied to the complete surface of the sensor.

Assuming that the entire surface of the sensor is illuminated, it is possible to define the

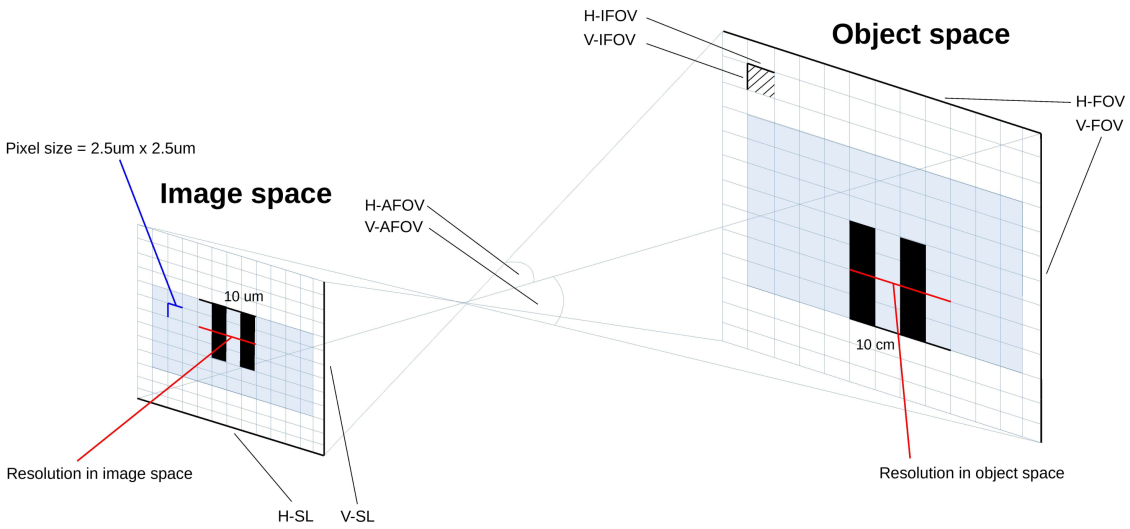


Figure 2.13: Representation of the relation between object space and image space

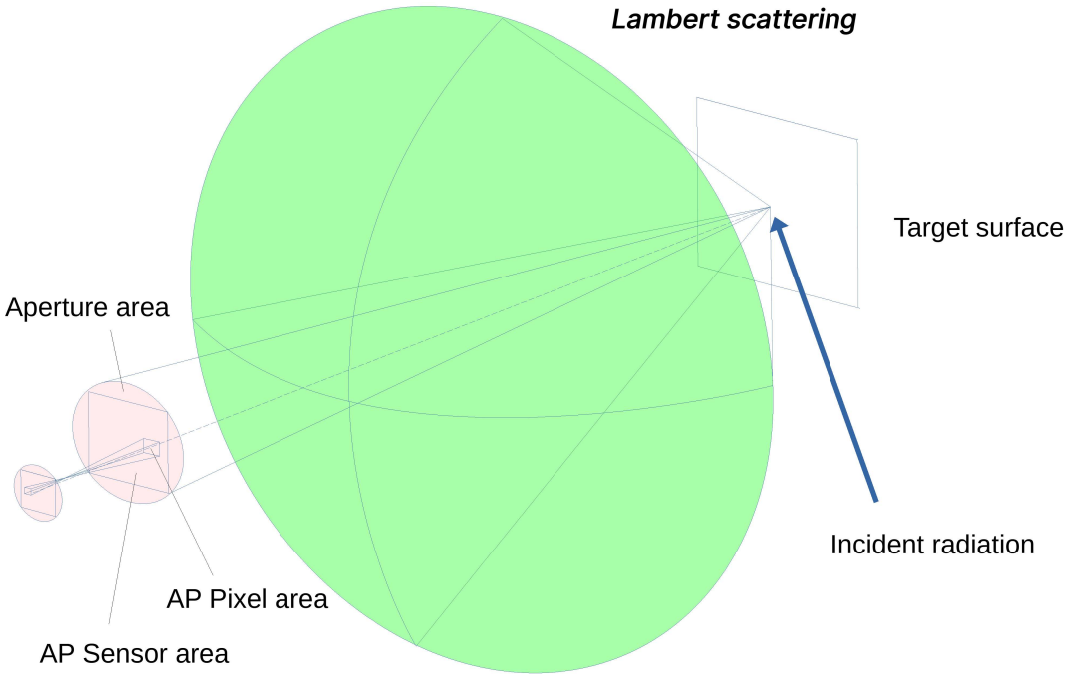


Figure 2.14: Qualitative illustration of Lambertian scattering

magnification of the lens assembly as the ratio between sensor size and FOV [9].

$$m = \frac{HSL}{HFOV} = \frac{VSL}{VFOV} \tag{2.7}$$

Where HSL is the horizontal sensor length, VSL the vertical sensor length, HFOV the horizontal FOV and VFOV the vertical FOV.

The FOV is a significant parameter in the evaluation of the acquisition for this particular application. A FOV can be realized from combination of Angular FOV (AFOV) and Working Distance (WD), that is the distance at which the target is located relative to the aperture of the imaging system. The combination between the AFOV and the maximum WD determines the geometry of the observable region, defined as the region of space in which the target satellite is considered observable. The geometry of the observable region and the dynamics of the satellites determine the total time for which the target satellite is observable.

$$FOV = 2 \cdot WD \cdot \tan\left(\frac{AFOV}{2}\right) \tag{2.8}$$

Therefore, due to the tumbling motion of the two spacecrafts during separation, it is necessary to determine the optimal balance between the working distance and the angular field of view. Qualitatively:

- An increase in the Angular Field of View (AFOV), considering the tumbling motion of the spacecraft, would extend the time interval during which the target satellite remains in view. However, this would simultaneously decrease the maximum working distance, thereby reducing the number of observation windows before the target satellite becomes too distant to be resolved.
- A high AFOV and maximum Working Distance (WD) can only be achieved thanks to a very high Field of View (FOV). Consequently, the FOV assumes the role of a budget variable.
- The maximum FOV for each sensor that allows for the satisfaction of the observational requirements is defined as the Field of View Budget (FOVB).

To calculate the FOVB for an arbitrary sensor, it is necessary to calculate the spatial frequency of the feature in the image space as a function of the FOV and to determine the highest FOV that still allows for sampling of said frequency.

$$FOVB = 2 \cdot WD_{max} \cdot \tan\left(\frac{AFOV}{2}\right) \tag{2.9}$$

The highest frequency resolvable by the sensor is the Nyquist spatial frequency and is half of the spatial sampling frequency of the sensor detector elements (pixels):

$$R_n = \frac{1}{2}\frac{N}{HSL} \tag{2.10}$$

The feature spatial frequency in the object space can be referred to the image space via the magnification, itself is function of the FOV (either horizontal or vertical):

$$k_{x,img} = k_{x,obj}/m = k_{x,obj}\frac{HFOV}{HSL} \tag{2.11}$$

Since the resolution limit is given by the sensor sampling cutoff frequency (Nyquist frequency) and cosidering a design that is going to be limited in resolution by the sensor and not the optics, it results tha the maximum FOV that would allow imaging of the spatial frequency of the feature is:

$$FOVB = R_n\frac{HSL}{k_{x,obj}} \tag{2.12}$$

To select the best sensor for the application, a database of global shutter imaging sensors compatible with the requred MCU has been built. Equation 2.12 is applied for each sensor of the database to obtain the FOV budget. Sensor with higher FOVB would be able to produce a lager detection volume and therefore increase the number of frames that the imaging system would be able to capture during the separation event.

Lastly, after the sensor for the application is selected, and therefore the sensor cutoff spatial frequency is defined, it is necessary to select a lens assembly such that its MTF would not degrade the contrast at the sensor cutoff frequency. This means selecting appropriate optics so that the system is effectively limited by the sensor and not the optics, satisfying the previous assumptions.

2.4 CMOS sensor database

2.4.1 Configuration requirements and compatible camera interfaces

Due to power constraints, the configuration requirements mandated the use of an ARM Cortex M4 processor. The STM32L4 family of microcontrollers, as demonstrated by their successful operation in missions such as SINGER payload computer for the SPEI-SATELLES mission [7], have proven their suitability for operation in the Low Earth Orbit (LEO) environment. As stated before, The configuration requirement has been adjusted to include the STM32L4+ family of microcontrollers as acceptable MCU for the imaging system.

Both the STM32L4 and STM32L4+ families support DCMI camera interface. DCMI is ST's nomenclature for a parallel interface utilized for raw or jpeg image data, commonly utilized for low performance imaging sensors. The interface is composed of 8-12 parallel data lines for the pixel data and a set of lines for signal synchronization (VSYNC/H-SYNC, PixelClock).

The DCMI peripheral implemented by ST presents a throughput limitation related to the AHB ($DCMI_{CLK} < AHB_{CLK}/2$) bus clock of the MCU. Since the Arm Core-M4

can be clocked up to 80Mhz for the STM32L4 family and 120Mhz for STM32L4+, the specified maximum throughput is respectively limited to 32MB/s [25] and 54MB/s [24].

The DCMI peripheral throughput imposes a limit on the combination of resolution and frame rate.

The system has been designed to work with a pixel sampling of 8 bit, but can optionally work with a 10 bit sampling. It has been decided to utilize a 8bit sampling is because it involves a reduced data volume amongst the other sampling modes and presents no throughput loss when packed in a 32bit word. The latter aspect is particularly important because the utilization of the MCU bus, as we will see, is one of the aspects that are critical for the acquisition frequency of the Development Model.

To gather information on the available automotive-grade COTS CMOS sensors, the product portfolios of the most prominent CMOS manufacturers were analysed, respectively from OnSemi, Omnivision, AMS-OSRAM, Sony, STMicroelectronics.

The analysis of the products available from the manufacturers highlighted a segmentation of the market in legacy low-end sensors with parallel interfaces and high end, more recent sensors with serial interfaces.

2.4.2 CMOS sensors industry shift from parallel to High-Speed Serial Interfaces

The research of compatible CMOS sensors highlighted a set of trends from CMOS sensor manufacturers and integrators, that are interesting to analyze in the context of the current development work.

The most prominent trend consists in the shift from the use of parallel camera interfaces towards high-speed serial interfaces. Serial interfaces offer superior throughput and facilitate system integration due to the lack of necessity of parallel line syncronism and reduction in the number of data lines.

Some of the high-speed serial interfaces currently used to interface with CMOS sensors include:

- Mobile Industry Processor Interface - Camera Serial Interface 2 (MIPI-CSI 2)
- Low-Voltage Differential Signaling (LVDS)
- subLVDS, a reduced voltage version of the LVDS electrical specification, predominantly used by CMOS sensors produced by Sony.

But there are challenges that are associated to the utilization of newer interfaces. An high speed serial interface requires an adequately high clock peripheral, are usually available only on relatively power hungry MCU or SoC.

Being limited to sensor working with parallel interfaces means to look to (and to rely on) legacy products, that consequently are characterized by inferior performance, uncertainty about the components longevity and future availability, scarce availability of technical documentation and lack of support from the manufacturers due to the lack of interest in supporting products close to being discontinued.

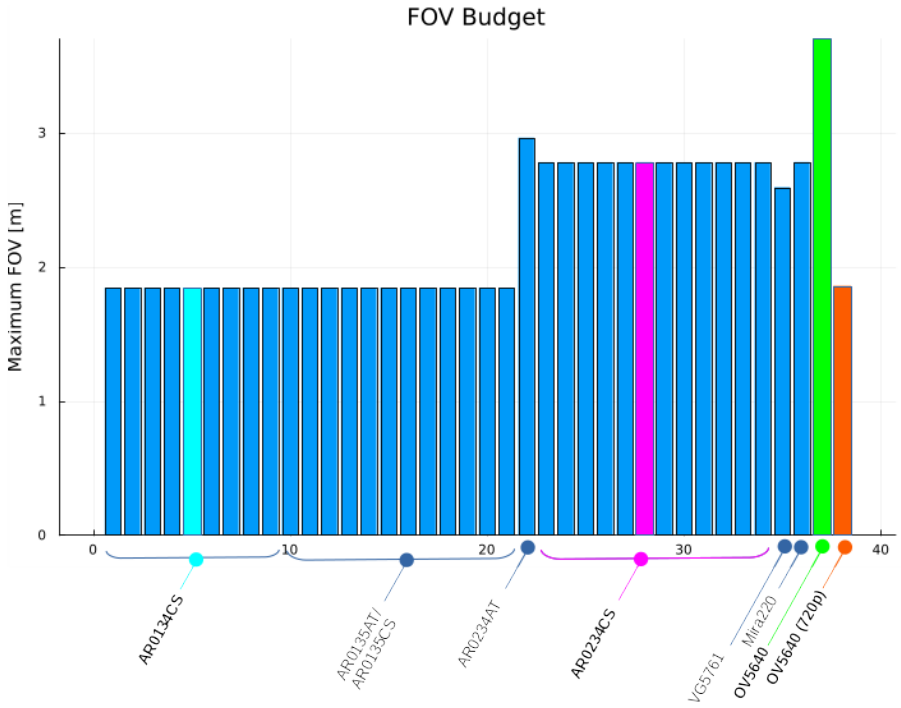


Figure 2.15: Evaluation of FOV budget for the sensors in the compatible sensors database

2.4.3 Database

The database includes a set of global shutter sensors from ONSEMI, STMicroelectronics, ams OSRAM, Univision. The database presents the additional inclusion of the OV5640 rolling shutter sensor in two different sampling (5Mpx and 720p) configurations.

The OV5640 sensor is a successive addition in a second iteration of the sensor evaluation analysis. The first analysis resulted in the selection of the AR0234CS sensor from ONSEMI. Unfortunately, the manufactured disagreed to provide technical documentation necessary to work with the sensor, therefore it has been necessary to select a different sensor at least for the development model activities. The OV5640 sensor makes use of a rolling shutter, that is unsuitable for the application due to rolling-shutter distortion induced by the tumbling of the observer and the target satellite. That said, it is an inexpensive, easy to procure sensor that has been used as a backup to conduct the initial development activities.

The graph displayed in figure 2.15 shows the FOV budget calculated for each of the sensors in the database.

2.5 OV5640 sensor performance analysis

In this section, the performance of the OV5640 sensor in the 720P configuration are shown and discussed. It has been decided to present the results of the analysis for this particular configuration, since it is the one utilized in the Development Model that

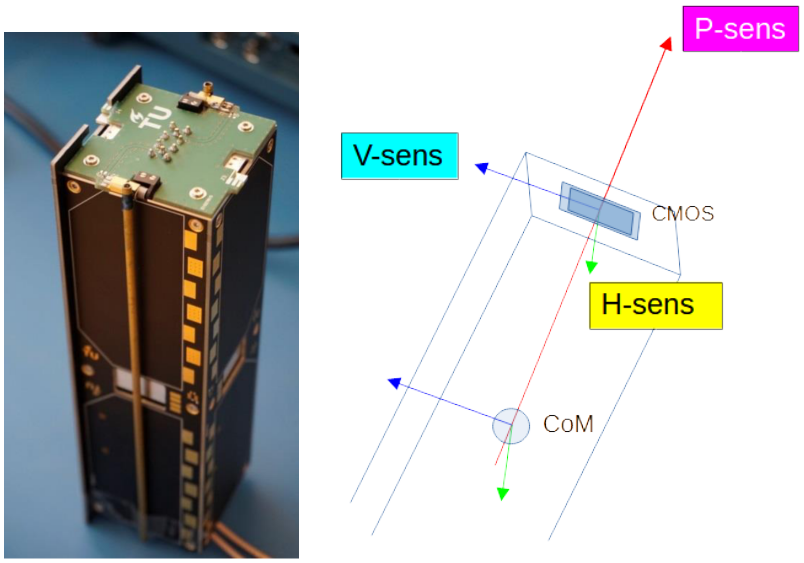


Figure 2.16: Delfi-PQ (Left), Body and Sensor reference frames (right)

has been assembled and verified. Moreover, it features a FOVB equal to $1.848m$, very similar to the one of the AR0134CS sensor, that is the most performant global shutter sensor which technical documentation was available.

To evaluate the presence of the target satellite in the imaging sensor frame for each instant of the simulation, it is necessary to define an instrument reference frame and a criterion that defines if the target satellite can be imaged.

2.5.1 Instrument reference frame

The position of the origin of the instrument reference frame (SHSV) is $O_i, b = [75, 0, 0]mm$, relative to the body reference frame and corresponds to the center of the imaging sensor.

The instrument reference frame axis are aligned with the ones of the Body axis. The instrument axis are respectively defined as sensor-perpendicular (P-Sens), sensor-horizontal (H-sens) and sensor-vertical (V-sens).

2.5.2 Observability criterion

The target satellite is represented in the instrument reference frame as a position vector $\mathbf{r}$ with its origin in the sensor reference frame. The vector is within the observable region if it satisfies the following conditions:

1. The vector can be represented via a $[2-3]$ intrinsic rotation in the SHSV frame of the generator vector $\mathbf{r}^*$, that is if

$$\mathbf{r}^* = R_{y,\text{shsv}}(\Theta)^{-1} R_{z,\text{shsv}}(\Psi)^{-1} \mathbf{r} \tag{2.13}$$

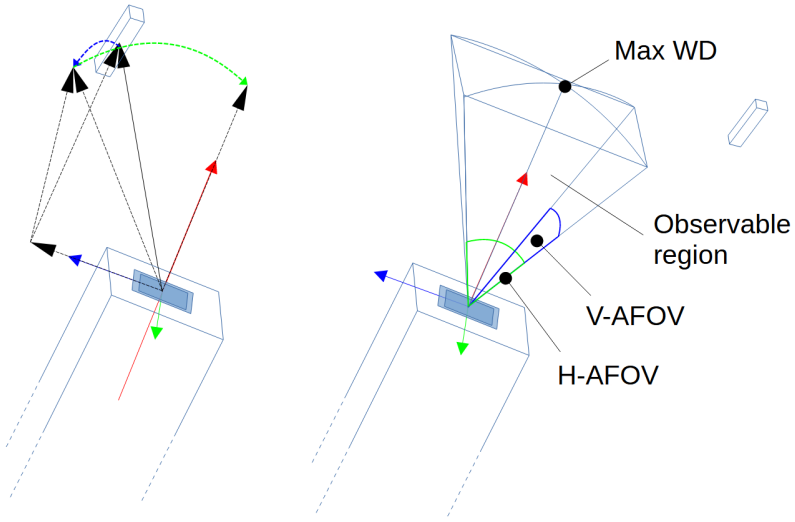


Figure 2.17: Visualization of the imaging system observable region

where R_a represents the rotation matrix for the a axis and the angles of rotation are within the respective intervals:

$$\Theta \in \left[-\frac{AFOV_v}{2}, -\frac{AFOV_v}{2}\right], \ \Psi \in \left[-\frac{AFOV_h}{2}, -\frac{AFOV_h}{2}\right] \tag{2.14}$$

2. The projection of the position vector, r, on x_{shsv} should be positive. This can be mathematically represented as:

$$r \cdot x_{shsv} > 0 \tag{2.15}$$

3. The 2-norm of the position vector, denoted as $||r||_2$, should be within the minimum and maximum working distance limits. This can be expressed as:

$$||r||_2 \in [min(WD), max(WD)] \tag{2.16}$$

Here, $min(WD)$ is the minimum working distance derived from the optical system specifications. On the other hand, $max(WD)$ is the maximum distance for which it is possible to satisfy the detection requirements. The value of $max(WD)$ is a function of the Field of View at the Base (FOVB) of the sensor in use and the Apparent Field of View (AFOV).

2.5.3 Time In Range (TIR) and Time In View (TIV)

The total time in range is the time that is necessary for the relative distance between the Observer and the target to be greater than the maximum working distance. For a

fixed FOVB, the maximum working distance is:

$$WD_{max} = \frac{HFOVB}{HAFOV}. \tag{2.17}$$

The simulation software determines the time in range as the last timestamp for which the target satellite is in frame. Therefore, in the 5th simulation case displayed in figure 2.18 for $HFOV <$ 23 deg the target has never been imaged (by the definition of the observability criterion) and for the rest of the HFOV dominion, the TIR follows the function above, as expected.

The TIR parameter does not consider the attitude of the spacecraft, but it is indicative of the ability to image the target at range as a function of the horizontal AFOV.

The Time In View is the total time for which the target satellite resides in the observable area. It is obtained by the sum of every simulation time step for which the observability criterion are satisfied.

The TIV is the ultimate product of the analysis process. Given a system acquisition frequency, it allows to determine the number of valid image frames that a sensor configuration can capture for a set of initial conditions.

Given a specific dynamics simulation case, the TIV is calculated for an angular horizontal FOV in the span of [20,130] degrees. For each AHFOV the TIV is determined and plotted in the graphs below, that include each of the dynamics simulation cases presented in the previous sections. The results consider a variation of the Horizontal angular FOV. The same process can be applied to the vertical FOV. It is important to point out that the ratio between the Horizontal and Vertical FOV is dependent on the aspect ratio of the sensor, that is the ratio in horizontal an vertical dimensions of the sensor.

Below the TIR and TIV graphs, there are a set of graph that are meant to display the process of determination of the total time in view. An example case is presented, that is relative to a sensor with HAFOV = 70 deg and dynamics simulation case 5 (offset separation spring, $\omega_0 = [0, 10, 5]$ deg/s).

Each of the graphs 2.22,2.24,2.24, represent respectively the comparison of distance to the sensor, and angular coordinates, in respect to the limits of the observation region. In a simulation time step, the target is considered in view only if all the three criteria are concurrently satisfied. In this simulation case, the relatively fast rotation motion applied by the separation spring allows the observer satellite to execute two rotations before the range of the target satellite becomes greater than the maximum working distance for the considered HAFOV. But, due to the angular velocity, the duration of the observation views are limited.

2.5.4 Analysis of the simulation results

In general, from the results of the different dynamics cases, it is possible to observe how the highest time in view is achieved by the simulation case 2, and specifically for a HAFOV of 40 deg. This happens because of the low rotation speed that allows for a specific HAFOV to conduct a single, high duration observation after the satellite performs a $180 - HAFOV/2$ deg rotation after the separation. While this scenario would provide a high total time in view, is considered to be too dependent on the

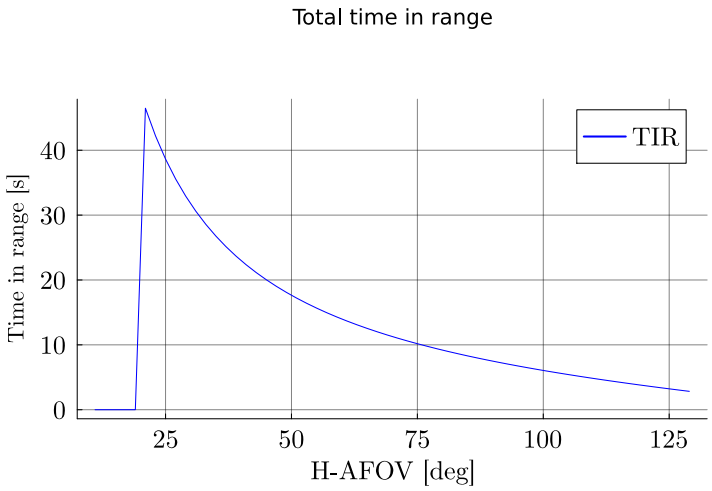


Figure 2.18: Total time in range as a function of HFOV and, therefore, maximum WD for simulation case 5 (offset separation spring, $\omega_0 = [0, 10, 5]$ deg/s)

initial conditions to be considered. Specifically, the initial conditions of the simulation could be achieved only with attitude control before the separation, something that goes against the requirement of being able to achieve the mission goals in case of failure of the ACS subsystem.

Much more promising are the simulation cases in which the separation spring is offset in line with the backplate (cases 1,3,5). The presence of the separation spring makes the resulting motion of the spacecraft a lot less dependent on the initial conditions. The higher rotation speed enables to image the target satellite reliably over several passes. The HAFOV that determines the lengths of the passes is balanced by the number of passes, since smaller HAFOV allows for greater maximum working distance and therefore a greater number of passages. This creates a condition where the observation results are more independent on both the initial conditions and the HAFOV of the imaging system, in respect to the cases where the separation spring is in the axis that passes between the two spacecrafts FoV. Moreover, the possibility of applying a determined momentum in the specific Y_B rotation axis generates a situation where rotation in the specific y_B axis is induced by the separation spring. This enables the effective utilization of a sensor with a high aspect ratio, placing the imager greater AFOV in the direction of Y_B rotation, since this rotation axis is necessary going to be the primary axis of rotation. Given the above considerations, it is advised to place the separation spring in line with the backplate. To achieve reliably the best performance independently of the initial attitude before separation, the imaging system shall feature a HAFOV in the range of [30-50] deg, that would yield a total time in view greater than 1.5 s.

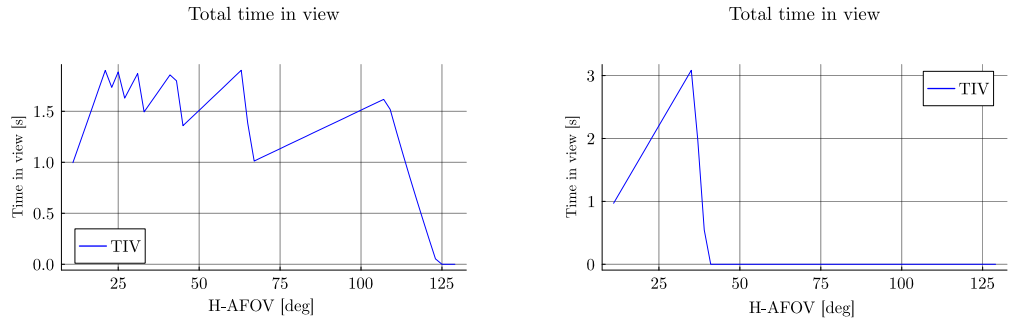


Figure 2.19: Results of TIV evaluation for cases 1 and 2 ($\omega_0 = [0, 2, 0]$ deg/s) On the left simulation case 1 (offset separation spring), on the right simulation case 2 (separation spring aligned with CoM axis of both S/C);

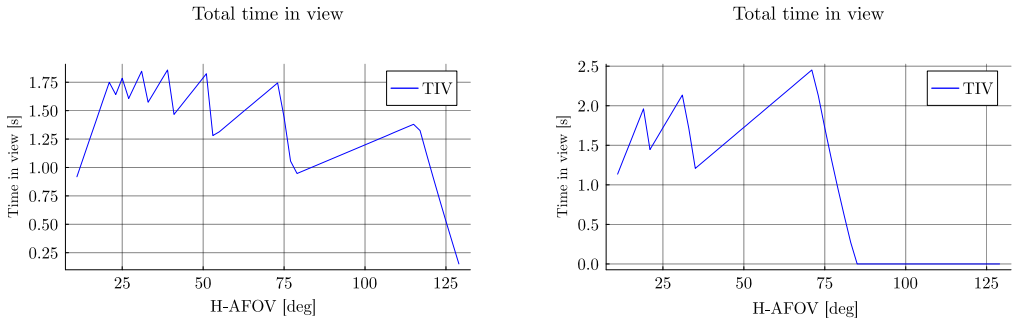


Figure 2.20: Results of TIV evaluation for cases 3 and 4 ($\omega_0 = [0, 5, 1]$ deg/s) On the left simulation case 3 (offset separation spring), on the right simulation case 4 (separation spring aligned with CoM axis of both S/C);

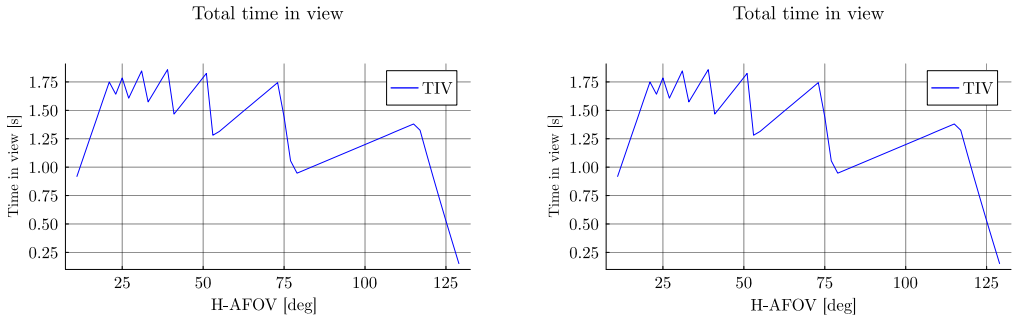


Figure 2.21: Results of TIV evaluation for cases 5 and 6 ($\omega_0 = [0, 10, 5]$ deg/s) On the left simulation case 5 (offset separation spring), on the right simulation case 6 (separation spring aligned with CoM axis of both S/C);

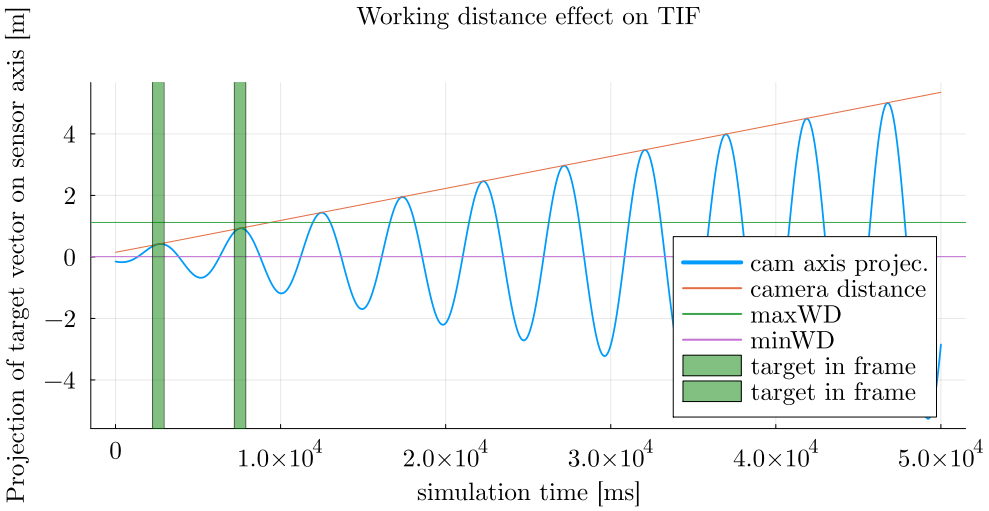


Figure 2.22: This graph shows the value of the projection of the target satellite position vector on the P-sens (perpendicular to sensor surface) axis of the instrument reference frame. The Working distance limits are highlighted.

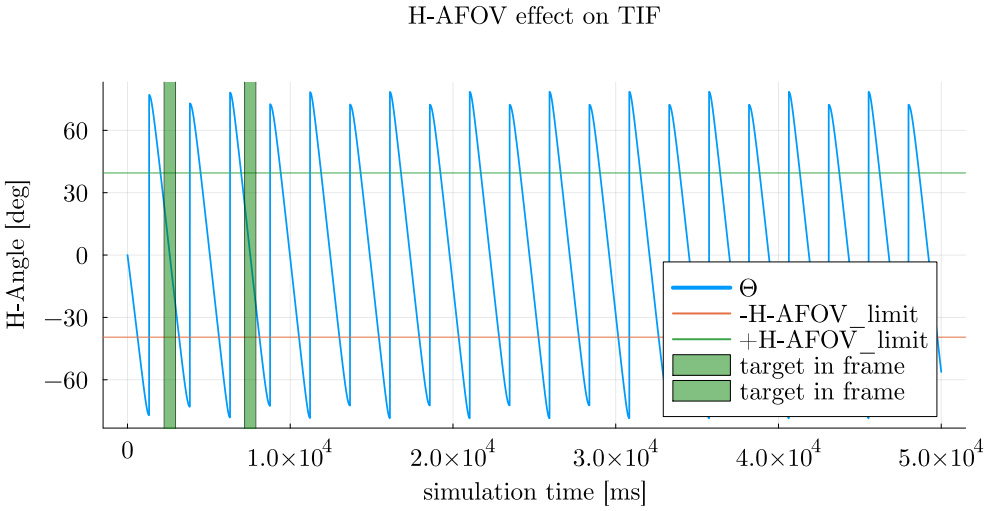


Figure 2.23: The graph shows the horizontal relative angular position of the Target satellite relative to the instrument reference frame and the vertical angular limits to the observability region.

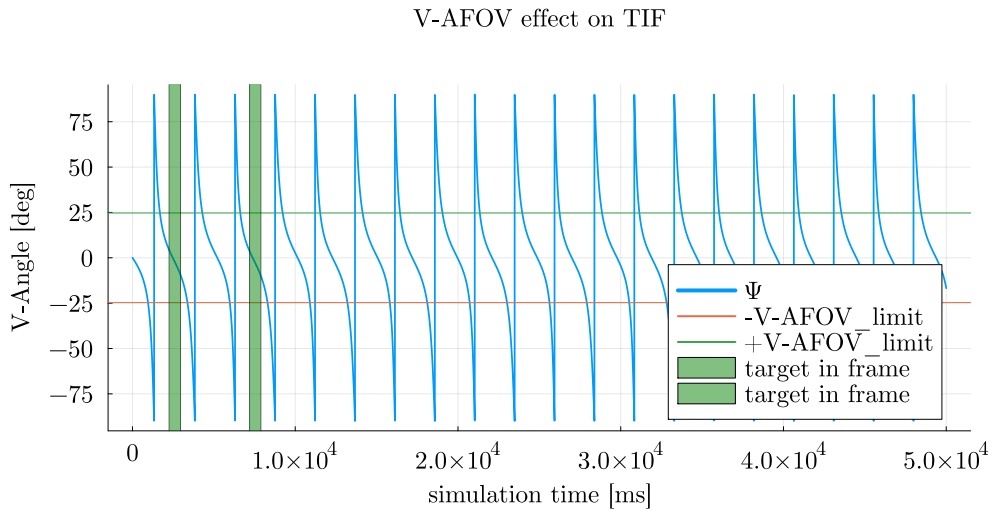


Figure 2.24: The graph shows the vertical relative angular position of the Target satellite relative to the instrument reference frame and the vertical angular limits to the observability region.

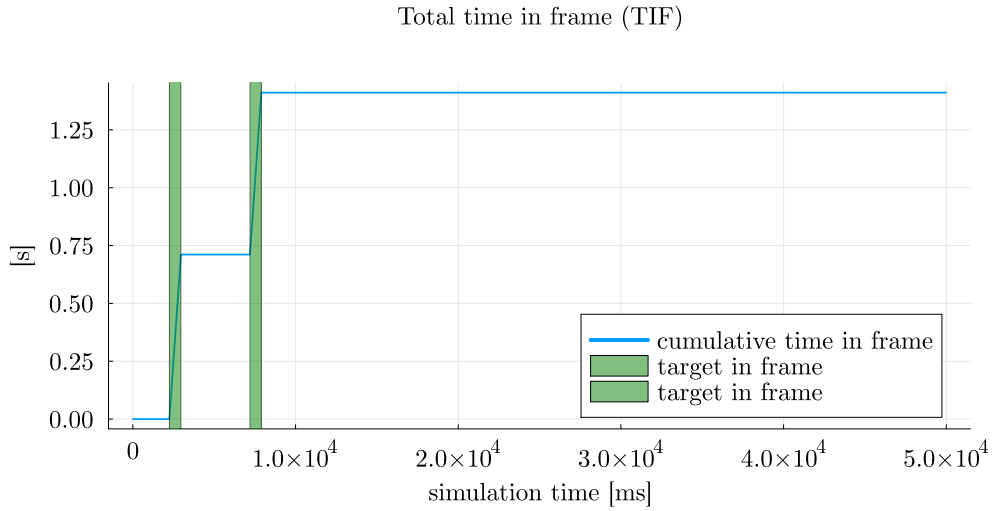


Figure 2.25: This graph shows the total time in view as a function of simulation time

2.5.5 Selection of COTS lens assembly

The lens assembly considered for the future system implementation is the 8mm FL, Red Series M12 Lens#57-909 from EdmundOptics. These lenses feature an 8 mm focal length(f) and an aperture of f/2.5, resulting in a primary magnification of 0.020X.

Given the primary magnification, and the feature object space spatial frequency $k_{x,obj} = k_{y,obj}0.75lp/mm$ it results in an image space spatial frequency of 37.5 lp/mm.

$$k_{x,img} = \frac{k_{x,obj}}{m} = 37.5lp/mm \tag{2.18}$$

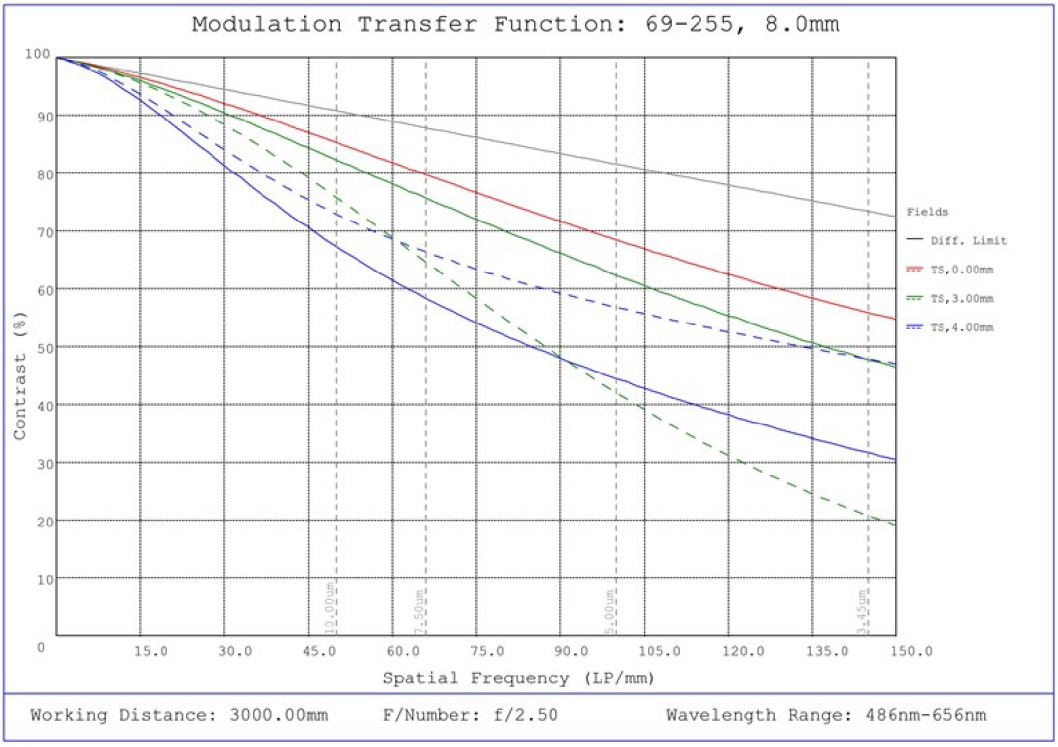
The selection of the lenses is permissive, due to the low image space spatial frequency that is necessary to ensure.

It is possible to compare the image space spatial frequency with the lenses Modulation Transfer Function specifications for these lenses provided by the manufacturer.

The lenses selected provide a MTF equal to 0.7 for the required image spatial frequency, that is sufficient with great margin to satisfy the contrast requirement.

Due to the nature of sensor-constrained imaging system, the choice of the lenses is flexible. In particular these lenses have been selected being optimized for infinite conjugate applications and are therefore suitable to gather images of earth once concluded the separation phase of the mission, in order to further increase the outreach outcome.

Lastly, thanks to the modular design and utilization of standard M12 threaded COTS lenses, it is possible to switch in the future the mounted lenses, to evaluate different configuration or to adapt the imaging system for another mission or application.



#69-255, 8.0mm FL, Red Series M12 Lens, Modulated Transfer Function (MTF) Plot, 3000mm Working Distance, f2.5

#69-255, 8.0mm FL, Red Series M12 Lens, Depth of Field Plot, 3000mm Working Distance, f2.5

Figure 2.26: Modulation Transfer Function and Depth of Field graphs for the Red Series M12 Lens#57-909 from EdmundOptics

Chapter 3

System design

The process utilized to achieve the system design consists in the definition of a set of candidate system architectures utilizing a functional decomposition approach. Successively a tradeoff study has been conducted to select the system architecture that would best suit the mission objectives and the development, assembly and verification context.

The input of the systems design process are the high level instrument requirements derived from the program objectives and stakeholders expectations, analyzed in the application analysis phase. The process is defined by its intermediary products, respectively the following System Breakdown structures:

- Functional Breakdown Structure (FBS).
- Product Breakdown Structure (PBS).

And the System Definition products:

- N^2 Matrix.
- System Block diagram.

The process follows the steps: The system high level functional requirement has been decomposed in lower level functions and organized in a function tree. Hardware and software products are identified and functions are assigned to each of the elements of the product tree via the use of a F/E matrix. Connections and interfaces between elements of the system are identified via the use of an N2 Matrix, and successively the system is laid out in a System Block Diagram.

An essential input to the system design process are the interfaces of the system with the rest of the S/C. Thanks to the commonalities of the Delfi-TWIN bus with Delfi-PQ, most of the interfaces specifications have been derived from the latter. Respectively:

Mechanical interfaces with S/C bus The Delfi-PQ spacecraft makes use of four rods that support an internal PCB stack. The four rods are held in place thanks to two aluminum frames at both ends of the stack and additional anchor points at the middle [14].

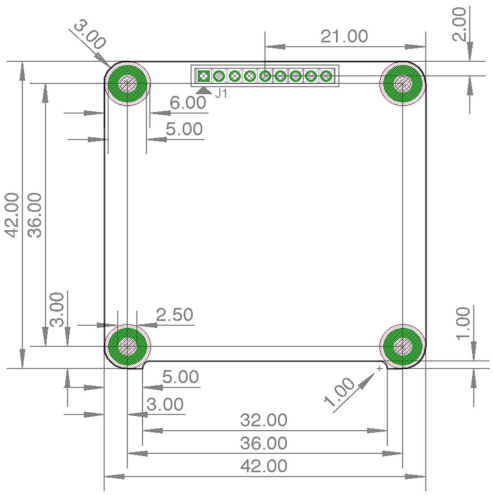


Figure 3.1: Drawing of the Delfi-PQ PCB form factor [14]. (Dimensions in mm)

Each PCB of the stack makes use of a standardized PCB format that allows for integration with the structural rods and the rest of the stack. The drawing of the PCB format displayed in figure 3.1 highlights the pass-through holes necessary for the structural rods. The drawing includes the Delfi-PQ inter-board PQ-9 connector that represents the physical layer of the satellite electrical and data bus

Consequently, the imaging system PCB shall adhere to the standardized PCB format to ensure compatibility of the mechanical interface.

The development of the Development Model has been considered as useful opportunity to test and evaluate the a new physical interface, that substitutes the PQ-9 connector with the SAMTEC-FSI-105 10 pin board to board connector.

Electrical ad Data interfaces with S/C bus The Electrical and data bus for the Delfi-PQ spacecraft makes use of the PQ-9 inter-board connector and allocates to the pins the functions in table 3.3.

The changes applied with the new version of the satellite bus involve power distribution and regulation. It has been decided to decentralize power regulation, including the necessary dc-dc regulators in each of the satellite subsystems, therefore the regulated power lines are substituted with an unregulated bus (SW0). This change allows to free pin 6-7-8 that can be assigned to other functions. In the scope of the Development model, the pins have been allocated to power the sensor voltage domains. The new functional configuration is shown in table 3.4.

Another interface that needs to be accounted for is the debug interface. Its physical layer is a Molex 8 pin connector allowing for the connection of a debug probe via the SWD debug protocol, a debug UART and a signal line to inhibit the on-board watchdog.

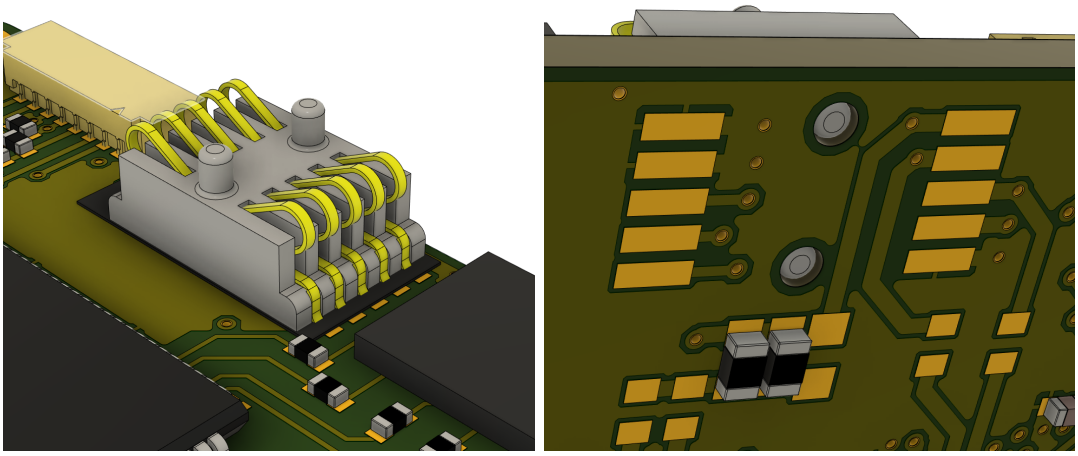


Figure 3.2: New electrical and data bus interface for the Delfi-TWIN satellite

Pin	Signal	Function
1	RESET	System reset
2	D-	RS-485 inverting signal
3	D+	RS-485 non-inverting signal
4	GND	Ground
5	V_BUS1	Power
6	V_BUS2	Power
7	V_BUS3	Power
8	V_BUS4	Power
9	GND	Ground

Figure 3.3: Table containing the function-pins assignment of the Delfi-PQ spacecraft

Pin	Signal	Function
1	GND	Ground
2	D-	RS-485 inverting signal
3	V_BUS2	Power
4	D+	RS-485 non-inverting signal
5	V_BUS3	Power
6	RESET	System reset
7	V_BUS3	Power
8	V_BUS1	Power
9	GND	Ground
10	GND	Ground

Figure 3.4: Table containing the function-pins assignment of the physical layer for the Delfi-TWIN spacecraft

3.1 Functional Breakdown Structure

The functional breakdown structure contains presents a hierarchical representation of the functional decomposition of the high level functional requirement of the system. Provides an overview of the functions that the system needs to perform and how the different low level functions concur to the realization of higher level functions.

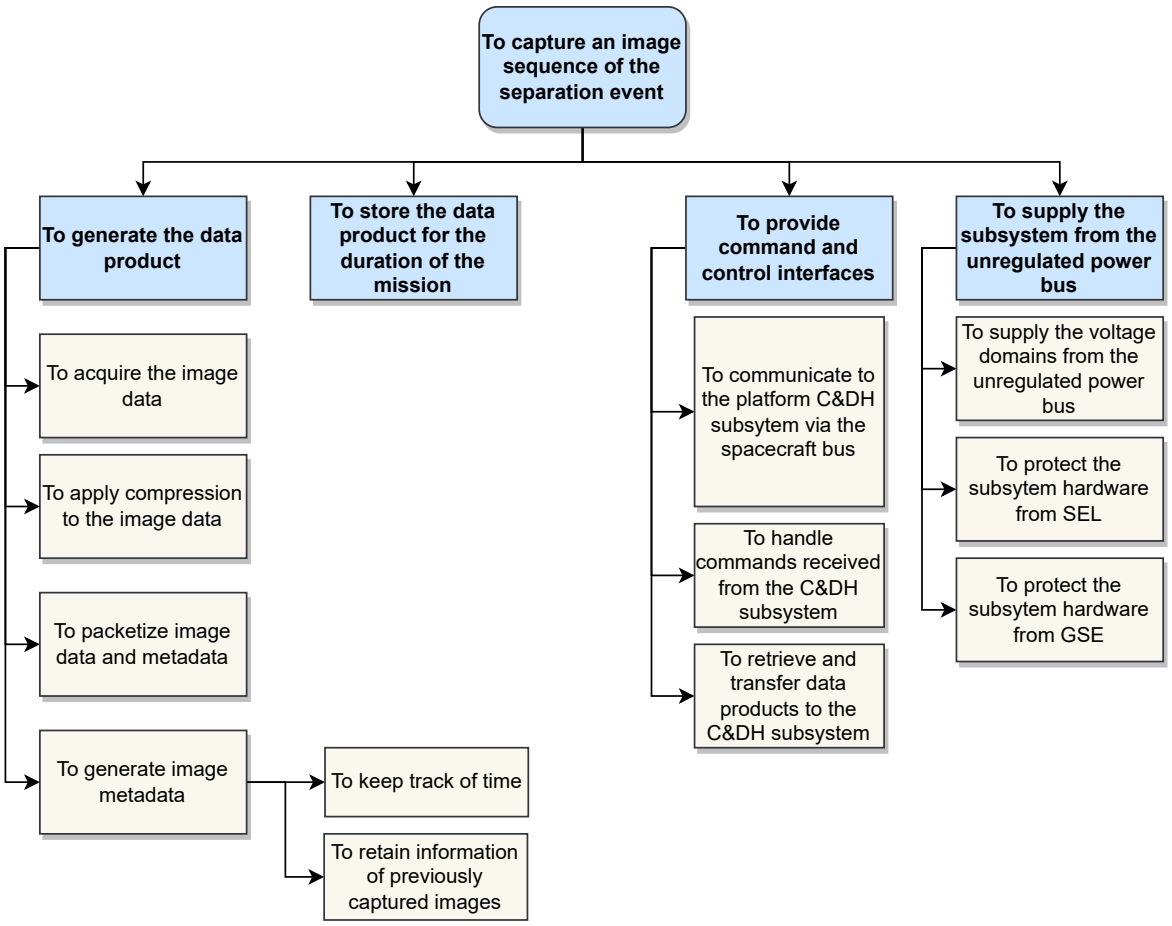


Figure 3.5: Functional breakdown structure of the imaging system.

3.2 Product Breakdown Structure

The PBS has been produced after successive iterations and represents the products that constitute the final system architecture of the imaging system. Therefore, it includes information on the design that were not defined in the initial draft. Namely: the selected memory technology (NAND SLC), the utilization of the RTEMS real time operating system.

This product tree contains two layers of decomposition containing both hardware and software products. The system has been dived in four subsystems: The Payload Command and Data Handling subsystem (PCDH), the Image Acquisition Subsystem (IAS),

the Static Memory Subsystem (SMS) and the Power Distribution and Protection Subsystem (PDP).

This organization of the system products has been devised to promote modularity of each of the subsystems. The subsystems are conceived so that they can be implemented as separate modules for the development model. The modular approach presents the following advantages:

- In case of the necessity of redesign of a portion of the system, it is possible to apply the changes only in the relevant module and perform a second round of procurement only for the affected module.
- It is possible to switch a specific module in case of failure with one in working condition, and use a second module to continue the activities while the first undergoes maintenance.
- It is possible to perform independent verification on each of the assembled modules, before the successive integration. This enables the development of a AIV plan that more easily discerns subsystem related issues with integration and interfaces issues.
- It is possible to perform environmental testing on a specific portion of the system.
- The utilization of a standard interface between the PCDH and IAS allows to substitute the sensor module, enabling future upgrades of the system or enabling performance evaluation testing with different sensor modules.

In addition, the product tree is a useful tool to illustrate which hardware products have been developed from the ground up for the project, while which others have been adapted from the previous Delfi-PQ designs. In general, every hardware device that interfaces with the software requires a low level driver. The vast majority of the hardware products are integrated with the software via bare metal drivers developed ad hoc during the thesis activities, while a minor number of functions make use of drivers derived from ST High Abstraction Layer (HAL) drivers. This topic is expanded upon in the software development section.

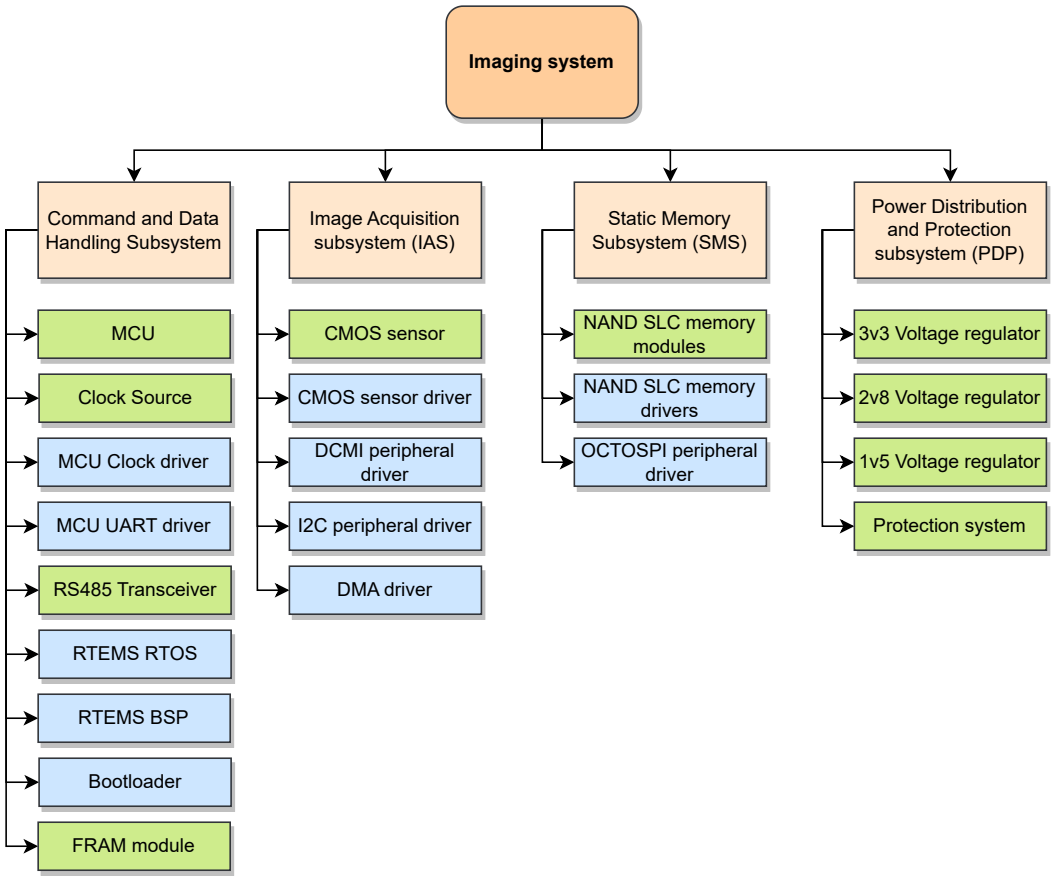


Figure 3.6: Product breakdown structure for the imaging system.

3.3 System block diagram

The System Block diagram here displayed offers an overview of the system at the highest level for the latest design iteration. The different subsystems can be easily identified as well as the interfaces previously specified.

The IAS includes the lens assembly and the image sensor.

3.3.1 State diagram

The state diagram offers an overview of the operative states of the instrument.

In the context of the development of the DM, operative constrains were not available, therefore the simplest operative state diagram has been devised from the high level functional decomposition.

The imaging system state diagram includes four non-transitory operating states: Stand-by, Video Capture, Data processing, Data transfer. Two additional states are defined and are used in transitory, automatically handled conditions: initialization state, recovery state.

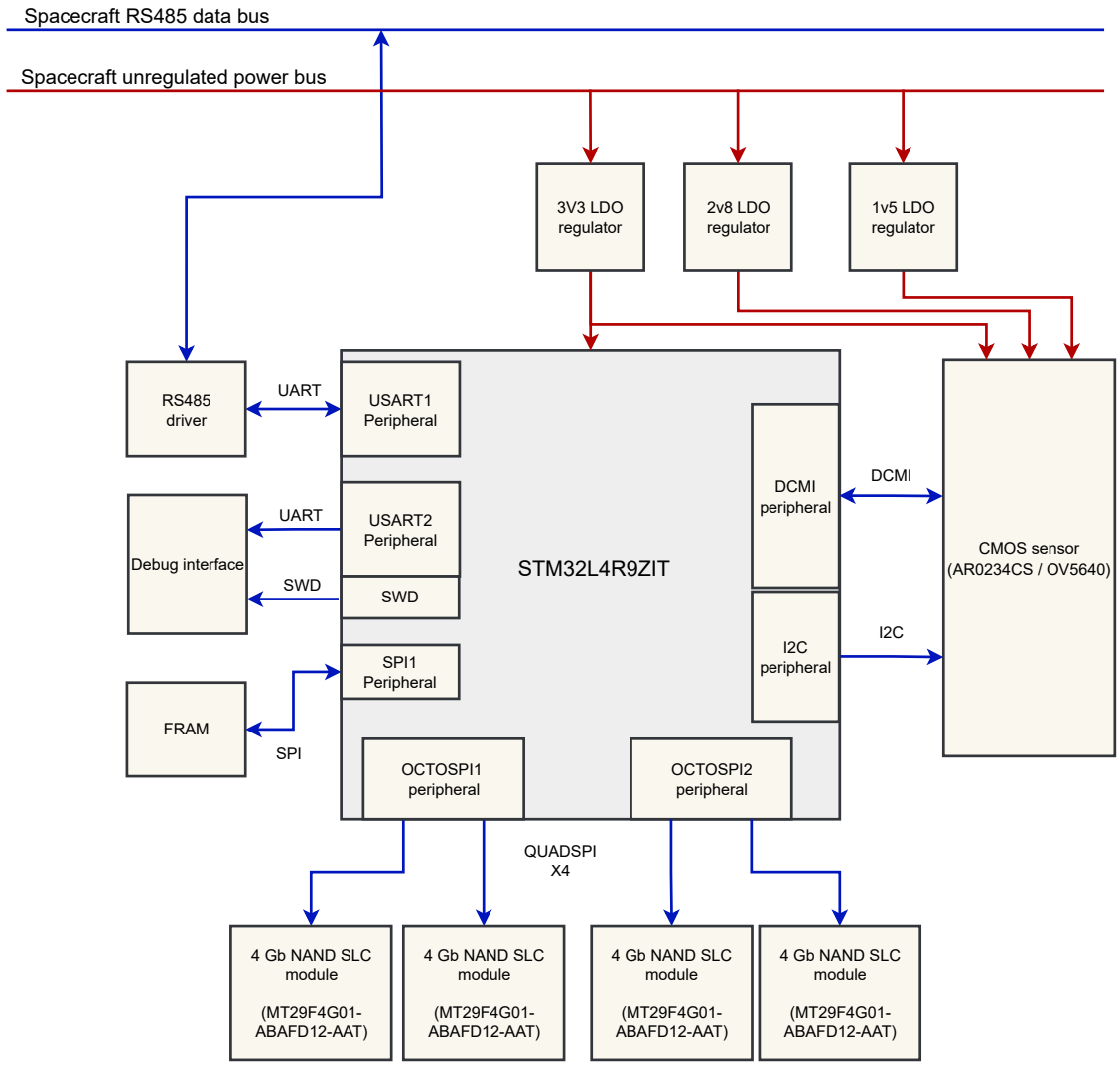


Figure 3.7: Imaging system block diagram.

The organization of the system states follows a standard architecture for simple embedded applications. The system nominally operates following instructions stored in volatile memory, loaded initially from an image in static memory during an initialization process. In case of non-nominal condition, the simplest recovery option is to save critical information in a static buffer or static memory and then execute a power cycle or a system reset, to allow a re-initialization from the correct initial state saved in static memory.

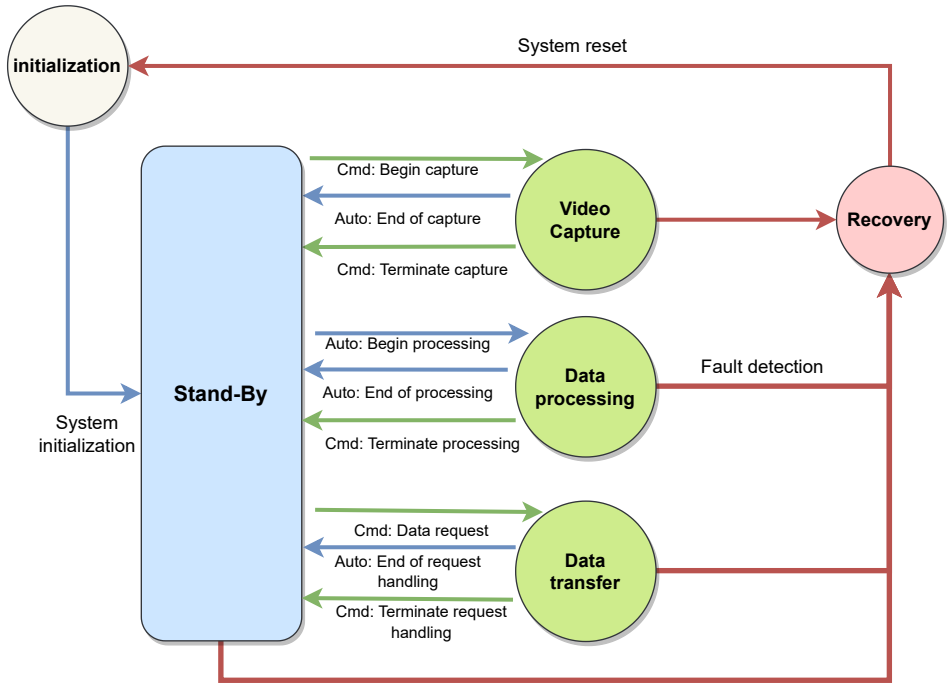


Figure 3.8: Imaging system state diagram.

3.4 Design considerations for the data Handling System

3.4.1 Image acquisition pipeline

The image acquisition pipeline is defined as the chain of time-sensitive processes executed by the system on the image data. It includes the sequential functional steps that the system executes on the captured image data to achieve its handling. It is critical to define an acquisition pipeline that performs image data handling with sufficient performance to satisfy the performance requirements of the system in terms of acquisition frequency.

In other words, the instrument frame acquisition rate depends on the period (and therefore the frequency) of the output of the image acquisition pipeline.

In a pipelined process, the process with the inferior throughput defines the throughput of the pipeline. Therefore, it is essential to devise a set of processes that does not create a bottleneck. The data handling pipeline consists of the following steps:

- Output image data from the sensor module via parallel interface.
- Transfer of data from the sensor to the DMA FIFO used by the DCMI interface.
- Transfer of data from DMA FIFO to RAM circular buffer.
- Generate and copy metadata to frame data in the RAM circular buffer.

- Transfer of image data and metadata to flash modules page buffer, util all data is transferred.
- Program the image data in static memory array.

The throughput of these processes are a consequence of the components used responsible for each process. Except for the programming in static memory process, the other steps of the pipeline are executed by elements of the system that have been already defined due to other systems constrains, respectively:

- The STM32L4R9ZIT MCU has been selected directly being the most performant MCU satisfying the configuration requirement and featuring a non BGA package (LQFP-144).
- The AR0234CS and the development counterpart OV5640 CMOS sensors has been selected due to its performance characteristics and component availability.

Therefore, it is necessary to design a SMS that would not hinder the performance of the selected components. The following table contains the maximum throughput of each element of the pipeline.

Process	Maximum throughput	Notes
Sensor data output on DCMI interface (1080p, 15 fps)	32 Mbps	Dependent on sensor capture mode and clock settings
STM32L4R9ZIT DCMi peripheral	54 MBps	
STM32L4R9ZIT DMA transfer (1/4 bus time allocation)	20 Mbps (Assumed)	Assumiing 6 clock ticks per transfer, high uncertainty
STM32L4R9ZIT DMA transfer (complete allocation)	80 Mbps (Assumed)	Assumiing 6 clock ticks per transfer, high uncertainty
STM32L4R9ZIT Core AHB transfer	40 MBps (Assumed)	Assumiing 6 clock ticks per transfer, high uncertainty
SMS memory programming throughput requirement	40 MBps	

Figure 3.9: Table containing the throughput of the elements of the pipeline

The process that governs the throughput of the pipeline is primarily the acquisition of image data from the sensor, which is capped at a rate of 32MBps. To maintain a margin of 20%, a system requirement has been established to set the minimum static memory throughput to 40MBps.

The successive section is dedicated to the description of memory configurations architecture candidates and the description of the rationale followed in the selection of the SMS architecture to be developed.

3.4.2 Tradeoff between memory technologies

The objective of this section is to design a memory configuration that would be able to handle the maximum data throughput that the sensor can produce.

The following aspects have been taken into account when choosing a memory module for the application:

- Memory technology write time performance.

- Memory technology suitability to the operative environment.
- Memory technology typical capacity.
- Package characteristics and component rating.
- Memory module interface.

The memory technology can impact several factors like capacity, write/read performance, availability, packages, interfaces, write/read operation endurance. The most common high capacity memory technologies are NOR flash and NAND flash.

The first step of the selection process has been an analysis of the available MCU interfaces that can be used to integrate memory modules and the maximum performance that each interface allows, as a consequence of the compatible memory modules categories.

The STM32L4R9 provides dedicated memory interfaces like the Flexible Memory Controller (FMC), the eMMC interface, and the general purpose OCTO-SPI interface.

Respectively, the FMC is used mostly with dynamic memories (SRAM, DRAM). It also allows for read and write in memory mapped mode. The limitation of this interface is the maximum addressable capacity, that is limited to 256 MB. The high data throughput generated by the imaging system (40MBps for RAW at maximum capture rate) would rapidly fill the maximum addressable capacity. Also, most of the SLC memories in the range parallel utilize TSOP packages that very bulky and would occupy a great portion of the floor plan of the PCB. Therefore, utilization of the FMC has not been considered a suitable option.

The MCU eMMC interface allows for adequate throughput and supports eMMC memories, which feature the highest capacity between the available interface/memories configurations (32 GB). The disadvantage of eMMC memories lies in their use of an internal controller to manage the interface and the NAND memory array. The presence of the controller has advantages, in particular it simplifies the logic of the interface and writes to the memory using wear leveling procedures, but the presence of a controller increases the subjectivity to SEE.

Another factor that went against the use of the eMMC memories is the package. The most common packages are fine pitch 153-pin BGA package, the integration of a memory module with said package would need a verification step of the solder joints that requires X-ray inspection instruments that are not available at the facilities in support of the project.

Another point is the fact that high capacity eMMC modules typically utilize MLC or 3D NAND configurations, that typically have inferior write/erase cycles longevity and inferior tolerance to the radiation effects when compared to SLC NAND cells. In conclusion, the uncertainties related to the suitability to the space environment of the controller and the reliability and integration issues due to the packages made the use of eMMC memories a powerful option, with drawbacks that would be difficult to address in the timeframe of the activities.

The OCTOSPI interface allows for the integration of serial memories. It can interface both SLC NAND or NOR modules and to access serial DRAM or SRAM modules.

To increase the flexibility of the system, it has been considered to use memories in the standard 24-bga NOR memory package. This package is common for serial ram and

flash nor memories, but there also serial NAND flash module that are compatible with the pinout configuration.

From these considerations, the following SMS architectures have been designed and compared, to determine the static memory architecture for the imaging system

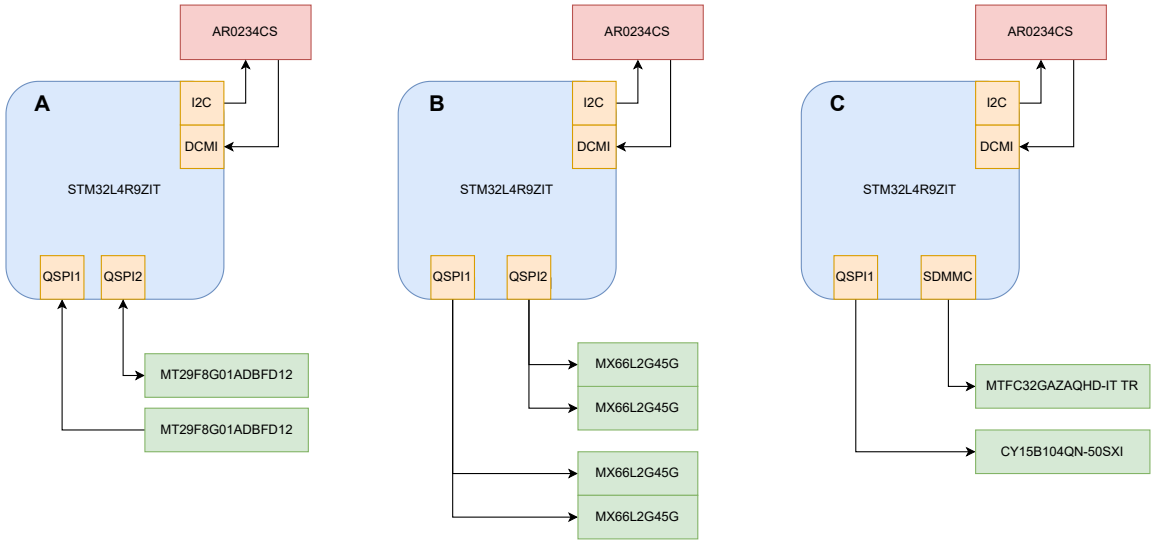


Figure 3.10: SMS architecture candidates.

Architecture candidate A candidate This architecture makes use of two SLC NAND module of the MT29F family. The MT29F is a family of NAND devices produced by Micron that has been developed to be package-compatible with NOR memory modules. The commonality of the package (and most of the command interface) enables eventual substitution of serial SRAM, NOR static memories and MT29F memories by replacing the component itself on the same PCB and uploading a new device driver. This feature is very convenient in the context of the rapid development schedule as it would allow substituting a set of NAND memory modules with serial SRAM modules that could be used to increase the volatile memory that could be allocated as frame buffer if necessary.

Architecture candidate B This architecture tries to implement a solution making use of NOR memory modules instead of the NAND modules used in candidate A. Specifically, it makes use of four Macronix MX66L2G45G memory modules. The MX66L2G45G module is a 2Gb NOR flash memory that features a pages size of 256 bytes. Its programming performance characteristics include a typical page program time of $25us$ and a maximum program time of $60us$. The resulting programming performance is 10.2 MBps (typical) and 4.7 MBps (worst case), therefore four modules are necessary to satisfy the average programming speed requirement of 40MBps.

The memory module can be interfaced with a multiple I/O SPI, up to QUAD-SPI and is available both in a standard NOR memory 24-Ball BGA format or a 16 pin SOP. Due to the limited programming throughput, it is more pin efficient to interface with the module via DUAL-SPI (resulting in an I/O throughput for 30 MBps) and consequently being able to integrate four modules in a single OCTO-SPI interface. Memory module programming power consumption is particularly relevant, since the

	Memory module		
	MT29F4G01ABAFD12-AAT	MX66L2G45G	MTFC32GAZAQHD-IT TR
Memory technology	NAND SLC	NOR	NAND MLC
Integrated controller	-	-	MMC
Module capacity	512 MB	256 MB	32 GB
Module page size	4352 B	256 B	N/A (controller)
Module page program time (TYP)	200 us	25 us	N/A (controller)
Module page program time (MAX)	UNKN	60 us	N/A (controller)
Module program throughput (TYP)	21.8 MB/s	10.2 MB/s	100 MB/s
Module program throughput (WORST)	UNKN	4.27 MB/s	UNKN
Module I/O interface	QUAD-SPI	QUAD-SPI	eMMC
Module I/O speed (MAX)	60 MB/s	60 MB/s	200 MB/s
Module Page I/O time	72.5 us	4.27 us	N/A (controller)
Module memory longevity	1E5 cycles	1E5 cycles	N/A (controller)
Module idle power consumption	15 uA	140 uA	200 uA
Module page program power consumption (TYP)	20 mA	35 mA	80 mA
Module page program power consumption (MAX)	25 mA	45 mA	110 mA
Module package	24-Ball BGA	24-Ball BGA / 160-SOP	153-ball VFBGA
Temperature range	-40°C to 105°C	-40°C to 85°C	-40°C to 95°C
Suceptibility to radiaiton environment	Good	UNKN	UNKN (controller)
Assemply module number	2	4	1
Assembly program throughput	43.8 MB/s	40.8 MB/s	100 MB/s
Module page program power consumption (TYP)	40 mA	140 mA	80 mA

Figure 3.11: Memory configuration architecture comparison.

architecture makes use of four memory modules. The total programming operating current would be 140mA, equivalent to 462mW, that represents almost a third of the peak power consumption allowed by the EPS requirements. This aspect makes this architecture undesirable in comparison to the lower power consumption of the others.

Lastly, satisfying the programming requirement in the worst case condition would necessitate the utilization of 8 memory modules, that would use significant space in the PCB footprint and an increase of poser consumption, that becomes significant considering the high number of components.

Architecture candidate C This architecture makes use of an eMMC memory module. The considerations made about the utilization of eMMC related to the package and the susceptibility are applicable to this architecture.

Potentially it is the architecture that would be best for more performant systems, and It would be the choice of preference for imaging systems that work with high performance MIPI-CSI CMOS sensors. That said, the development context and the scope of the project does not allow for the verification steps that would be necessary for this architecture.

Final architecture The final architecture is an adaptation of candidate architecture A. It makes use of both the two OCTO-SPI interfaces available in the STM32L4R9 MCU to utilize two pairs of MT29F4G01ABAFD12-AAT. Each pair of QUAD-SPI memories are interfaces with the OCTO-SPI peripheral in a dual QUAD-SPI configuration. In this configuration, the chip select and clock lines are shared between the

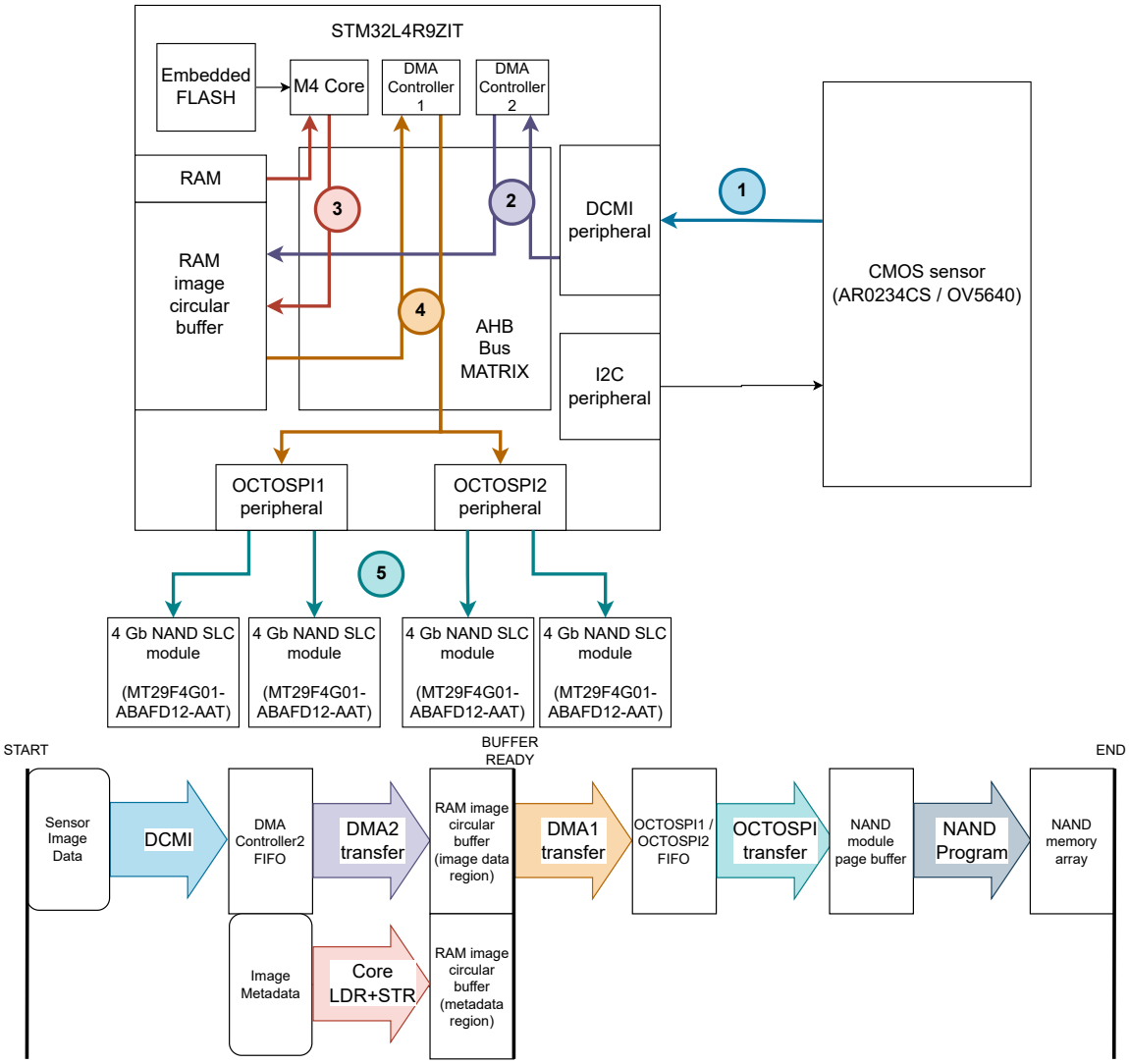


Figure 3.12: Overview of the image frame handling pipeline.

two memories and each of the memory stores respectively the most significant and the least significant part of each byte. The two sets of memories can be used in a buffered ping-pong configuration, further increasing the SMS throughput and capacity. The final architecture data handling pipeline follows the block diagram in figure 3.12.

3.5 Model Philosophy

The objective of a model verification strategy is to achieve product verification while minimizing cost, schedule and managing engineering risk.

The model philosophy that is presented in this document has been developed to guide the development and verification activities for the thesis project and therefore, is limited the scope to the thesis activities themselves. It encompasses the following:

- Development and successive verification of functional and performance requirements assigned to two Breadboard Models (BM). The breadboard models have been developed from COTS components for initial functional verification of hardware proof of concepts and a subset of software bare metal drivers.
- The development and verification of a Development Model (DM) for functional and performance verification of the instrument subsystems and the integrated system.

The thesis timeframe has been considered unsuitable for the development and production of successive models necessary to complete the life cycle of the system. An Engineering Model and successive Protoflight model are considered future steps for the successive activities.

An important aspect that needs to be highlighted is that the model philosophy devised has been designed with the intentional absence of performance verification at the subsystem component level.

This choice has been motivated by the insufficient time to execute performance verification at every system level and brings with it the risk of discovering component level or element level lack of performance in a late stage of the project development.

This choice followed the following rationale:

- Performance verification at the system element level would have required development overhead due to the differences in hardware between the breadboard models and the development model. Investing the required temporal resources would have yielded a considerable risk of being unable to deliver a functional system by the end of the thesis activity due to the already limited schedule.
- Due to the mission outreach objectives, an inferior performance of the system would lead to a lesser satisfaction of the mission goals, but would not jeopardize the achievement of said goals and therefore, the success of the mission. On the other end, the inability to deliver a functional product would cause great pressure on the successive project schedule and a probable reevaluation of the mission goals and scope.

Therefore, the risk of nonconformity to performance requirements for subsystem components has been considered acceptable.

An issue of this kinf presented itself for a performance parameter of the static memory subsystem. In particular inferior throughput for the transfer of image data from the RAM circular buffer to the OCTOSPI MCU interface FIFO has been observed in tests conducted on the DM.

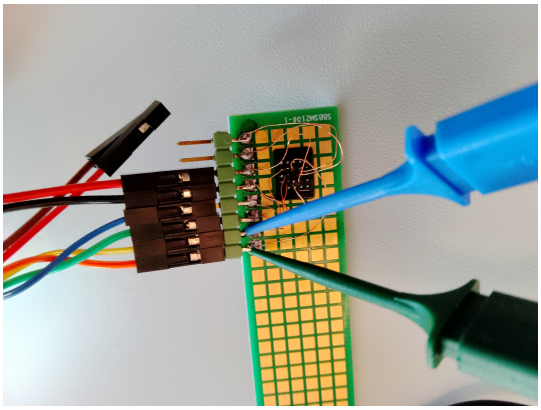
The effect of inferior performance in this particular step of the data handling chain has been compounded by the consequent inability to configure the data handling processes as a pipeline due to the implementation of circular buffer handling routines not respecting the design timings. The consequence has been the necessity of reconfiguring the pipeline process in a sequential process, further diminishing the data throughput.

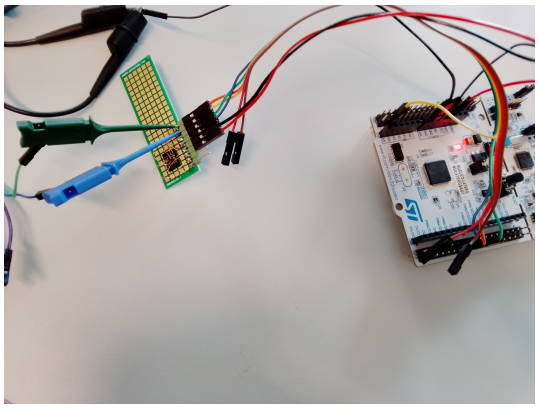
It is still worth defending the model philosophy since it still allowed for the successful development of a first iteration of the functional product that can be further iterated upon, with the purpose of solving the engineering issues already identified (and the ones at are going to be discovered via the development model itself).

In summary, a tradeoff between engineering risk and programmatic risk led to a decision that prioritized programmatic risk, while increasing engineering risk. The engineering

risks taken caused a late discovery of a design bottleneck that still allowed the production and verification of a functional Development Model with inferior performance to the one expected from the design. The DM is still satisfies the model objective and represents a starting point for successive design iterations.

3.5.1 BM1 - NUCLEO-L476RG + MT29F2G01ABAGD12





NUCLEO-L476RG + MT29F2G01ABAGD12 Model objectives
To familiarize with STM QuadSPI peripheral.
To develop and test bare metal drivers for the QuadsSPI interface
To develop and test bare metal drivers for the MT29 memory family
To develop and test bare metal driver for the USART interface
To develop and test bare metal driver for MCU clock sources
To validate debug tools and procedures
To support the development of the DM static memory subsystem schematic.

The Breadboard Model 1 is the fist model developed. Its setup has been constructed in the earliest days of development using hardware readily available in Tu-Delft workshop with the exception of the NAND module. It includes a nucleo development board featuring an STM32L476 MCU, connected via jumper wires to a development NAND memory module. The MT29F2G01 SLC NAND module is part of Micron MT29F memory family and has been selected being a lower grade and more available alternative to the MT29F4G01ABAFD12-AAT memory used for the DM. Thanks to its availability has been easily and rapidly procured at the beginning of activities.

While the hardware utilized in this model is similar to the one used in the DM, there are differences that required a later rework and adaptation of the software products that have been developed using this model.

32L4R9IDISCOVERY Model objectives
To support development of the RTEMS BSP. To support integration of bare metal drivers with the RTEMS BSP. To develop and test the bootloader for the RTEMS+Application software image. To verify functionality of the RTEMS RTOS. To identify low level software issues related to hardware-software interaction and develop solutions and workarounds. To update the UART peripheral driver for the STM32L4S9AI MCU.

3.5.2 BM2 - 32L4R9IDISCOVERY demonstration platform

The models consist of the 32L4R9IDISCOVERY demonstration board from ST, and has been used predominantly to develop low level software products.

It originates from the necessity to have a backup development setup that could have been used while solving eventual hardware issues with the development model.

The model allowed for the identification of a software issue related to a misconfiguration of the OCTOSPI Multiplexer peripheral and allowed for decoupling of the initial development of hardware and software products.

3.5.3 Development Model

The Development Model is implemented in three modules:

- Compute Module (CM)
- Daugherboard (DB)
- Sensor Module (SM)

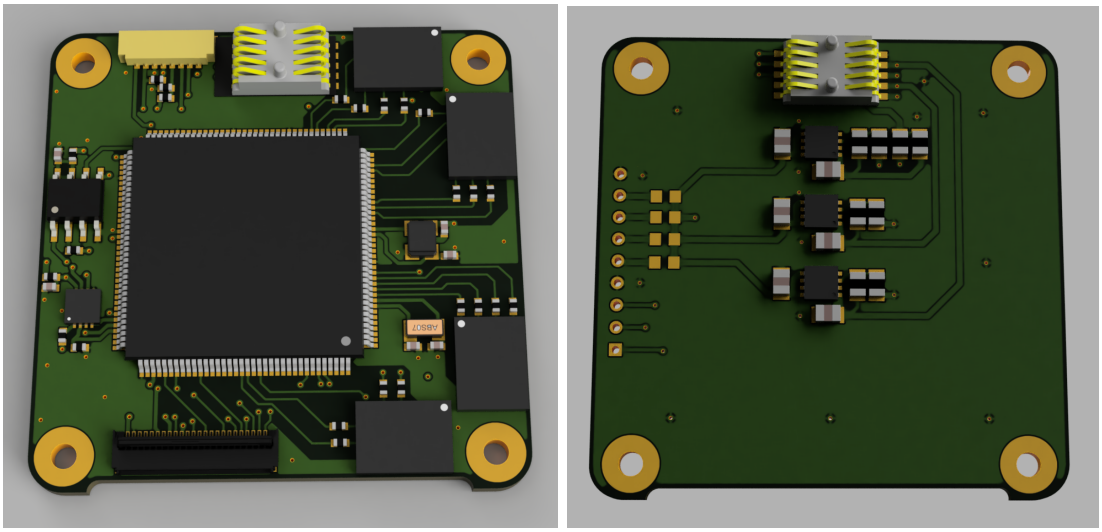


Figure 3.13: Compute Module and daughterboard render.

The compute module is stacked on top of the daugheterboard and connected to it via a 6 pin bard to board connector, that is the physical level of the satellite board connector.

The daugheterboard has been designed to be compatible with the Delfi-PQ satellite bus physical layer (PQ-9), while the inter board connection between the DB and the CM is realized via the SAMTEC-FSI-105 10 pin board to board connector.

This configuration allows the Development Model via the daughterboard to be compatible with the test equipment developed for Delfi-PQ, while utilizing the new bus interface for the Compute Module.

Chapter 4

Hardware design and development

4.0.1 Compute Module schematic

The electronics design of the compute module hardware has been performed via the use of the EAGLE software and laid out in a hierarchical schematic.

The high level schematic of the compute module contains the various subsystems, the interfaces of the boards and the connections between the different subsystems, specifically:

- A control unit consisting of the STM32L4R9ZIT MCU.
- An FRAM SPI module for radiation tolerant storage of the software image.
- An RS485 transceiver used to drive the RS485 line and interfaced via UART to the MCU.
- The SMS subsystem, consisting of four MT29F4G01ABAFD12-AAT SLC NAND memory modules.
- The CMOS sensor interface.
- Satellite Bus interface.
- Debug and programming interface.

The voltage regulator necessary for the 3.3V domain that supplies the devices on board s present on the daugterboard, as well as the 2.8V and 1.5V regulators necessary to supply the CMOS sensor via the sensor interface.

The protection electronics have not been included since satellite-standard anti latch-up and static protection systems are currently being developed as part of a different project at TU-Delft. The only measures that have been included for protection at a board supply level is a current limiting resistor and a ferrite bead on 3.3V supply line right after the connector. Each of the subsystems includes as well an appropriately sized current limiting resistor.

4.0.2 MCU component selection

The STM32L4+ family of microcontrollers have been selected during the system design. Th MCU model selected is the STM32L4R9ZIT, featuring the highest memory capacity segment with a 2 MB embedded flash and a 640 KB RAM and a LQFP-144 package. The LQFP package has been selected for being easier to inspect the quality of the solder joints in respect to the BGA package.

The STM32L4R9 MCU has been preferred to the STM32L496 MCU for the superior availability of serial peripheral interfaces (2xOctoSPI vs 1xQuadSPI) and superior MCU maximum clock (120Mhz vs 80Mhz).

The MCU makes use of two external clock sources, respectively a 32.7kHz oscillator and a 48Mhz oscillator. Stability of the supply for the MCU is granted via the use of two decoupling networks, for the digital and the analog supply domain. The decoupling networks are made of a set of $100nf$ capacitors and a bulk decoupling capacitor with a value of $4.7uf$.

4.0.3 NAND memory subsystem

Amongst the NAND SLC modules the MT29F4G01ABAFD12 modules have been selected being the QUAD-SPI modules that were available during design and development with the highest capacity and programming throughput.

A literature research has been conducted to estimate the radiation hardness of this memory component. The most relevant result is a radiation hardness evaluation campaign performed by M. Bagatin, S. Gerardin, A. Paccagnella at University of Padova [3] under ESA contract as part of the JUICE program [29]. The test campaign determined a total ionization dose (TID) of 65krad before the manifestation of retention errors and block erase failures. The specific model that has been tested is no longer manufactured, but newer models in the same MT29F family are now available, amongst witch there is the MT29F4G01ABAFD12 that has been used in the imaging system design. Of course, it is not possible to take for granted these results, since they refer to a different component and possibly to a different manufacturing node and process. The memory analyzed in the cited research makes use of a 25-nm process, while technical documentation for the MT29F4G01ABAFD12 does not specify the process node. That said this information can be still be useful as a reference for a future environmental verification campaign.

The NAND SLC modules are organized in two pairs, each pair connected to an OCTOSPI interface on the STM32L4R9ZIT MCU. On the logic and signal level, the modules are configured as DUAL-QUADSPI: Each pair shares the clock and the chip-select and the first chip of the pair receives the first four data lines, while the second the latter four. As a consequence, every 8 bit transfer performed to the OCTOSPI interface results in the data being split between the memory pair.

A write operation consist of transferring the data to a page buffer on each of the modules and successively execute a program command. Due to the DUAL-QUADSPI configuration the total page buffer appears to be double the size of each module page buffer and total capacity is doubled in respect to a single module.

Due to the perceived resilience to the radiation environment and the automotive extended temperature range (-40C, 105C) the reliability of the memory modules is considered high, therefore it has been considered acceptable to have two memory components

in series in a reliability block diagram of the subsystem. Of course reliability data is not available, therefore it is not possible to calculate the reliability of the static memory subsystem as a whole.

Moreover, in case of failure of one or several memory modules, the OCTOSPI-Multiplexer available in the STM32L4+ family provides options for in flight reconfiguration. The event of failure of up to two modules would produce the loss of the data already stored, but if at least two modules are in functional condition, it is always possible to configure the system to have a working pair, therefore being able to meet the acquisition performance requirements necessary for operation.

Lastly, it could be conceivable to use the two pairs, and the ability to multiplex the interface, to have a system where the same information is saved on both pairs, necessitating a failure on both symmetric modules to generate a data loss. This feature has not been implemented in software due to the perceived already high reliability of the subsystem, but is a possibility for future developments.

4.0.4 Sensor interface

The physical sensor interface is a 25 pin FFC (Flexible Flat Cable) designed to match the pinout of an OV5640 camera module. The rationale is that it enables easy test of the data handling subsystem thanks to the connection of a test OV5640 camera module. The logic interfaces (DCMI,I2C) are common amongst CMOS sensors with parallel interfaces, therefore the interface can be used with an arbitrary sensor (including the sensor initially selected, AR0234CS), with an appropriate breakout board. In the future it is going to be possible to switch the OV5640 with one of the more suitable global shutter sensors discussed in the previous chapters, simply developing the sensor host board matching the same pinout.

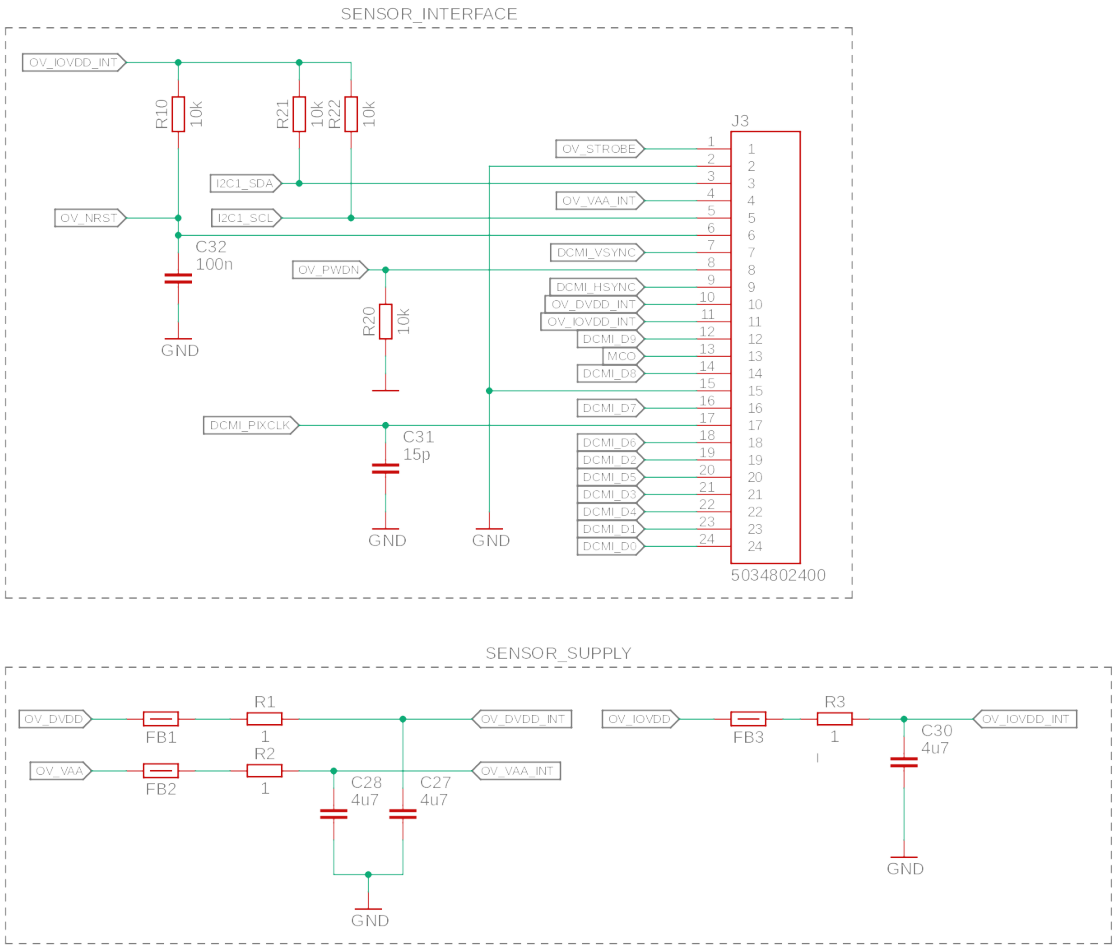
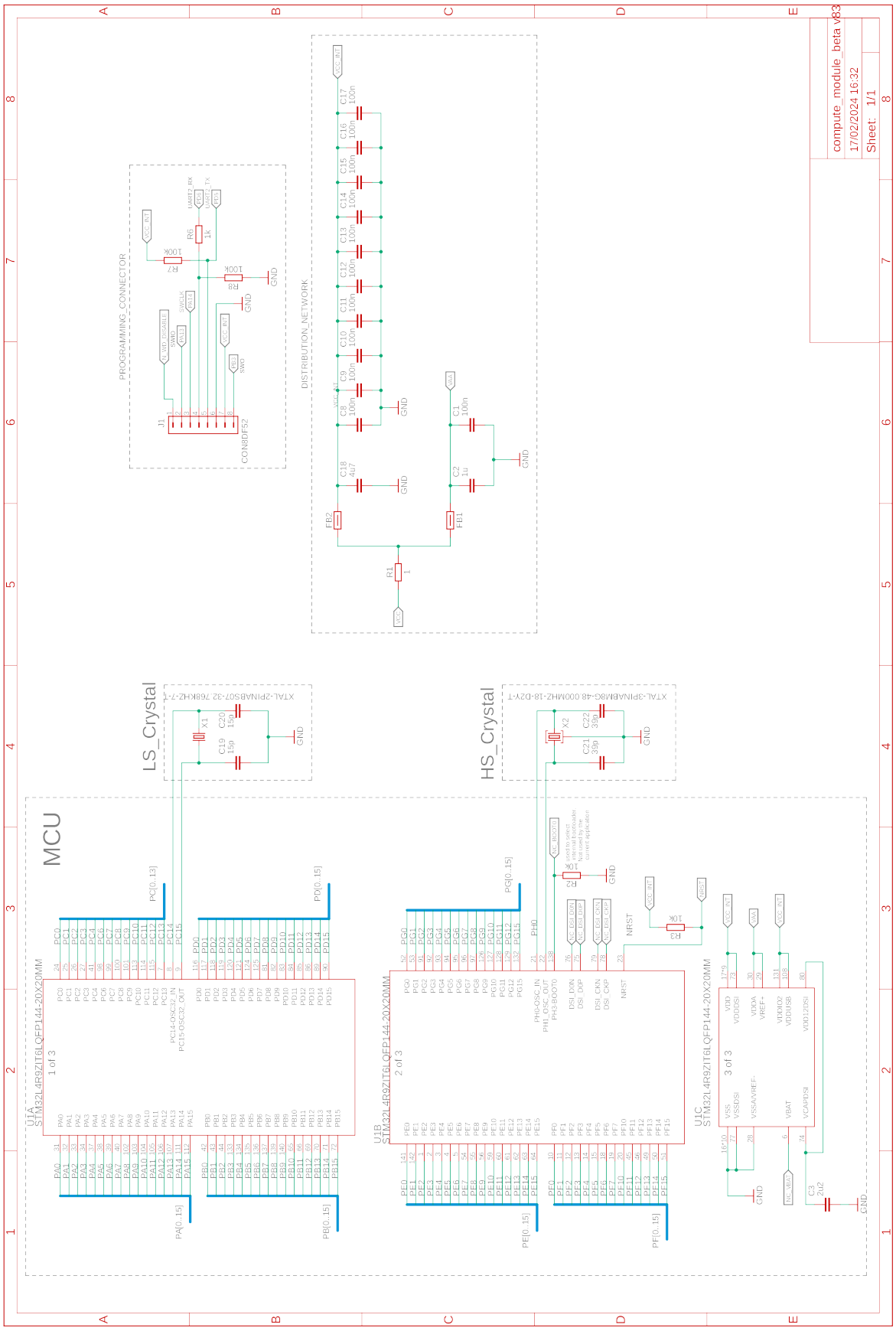


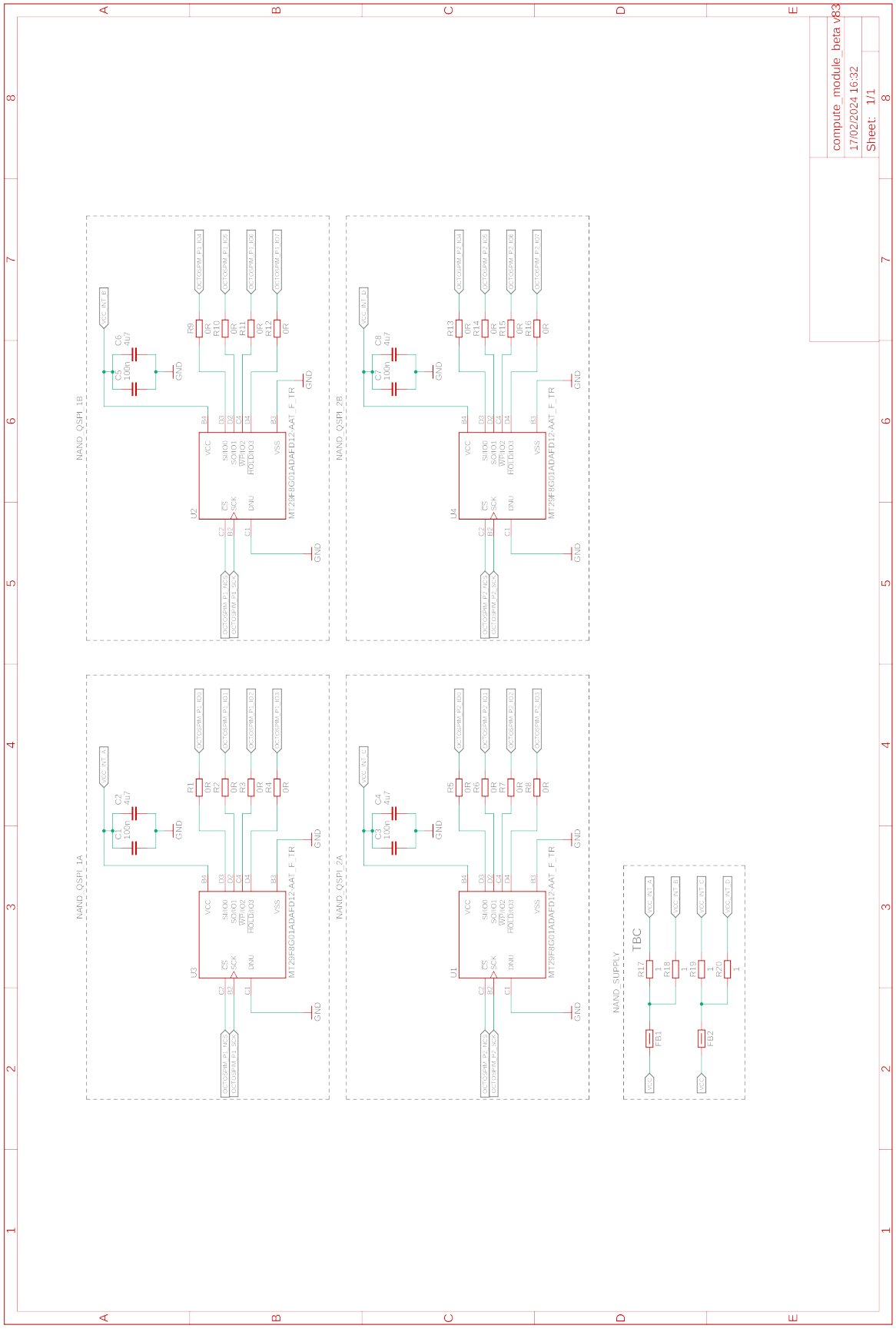
Figure 4.1: Schematic detail of the sensor data and power interface.

Compute module high level hierarchical schematic

MCU schematic



NAND memory subsystem - Schematic



4.1 Compute Module PCB design and production

The PCB for the development model has been developed keeping in consideration the following aspects:

- The design shall keep into the account the future presence of the power regulation and protection system, therefore reserving adequate space in the floor plan of the PCB.
- The PCB design for the DM shall make use of EuroCircuits design rules, and shall prioritize speed of procurement and affordability. This it to enable rapid iteration of the design. The initial prototype is bound to have issues that are expected to require a successive iteration to be fixed. This is especially valid in the development context of the project, where functions like integration with the sensor (and therefore the MCU DCMI interface) could only be tested with the development model, due to absence of accessible relevant physical interfaces on the breadboard models.
- Lastly, it has been necessary to procure, assembly and perform initial functional verification of the DM before the New Years break, in order to have functional hardware to work on during the break.

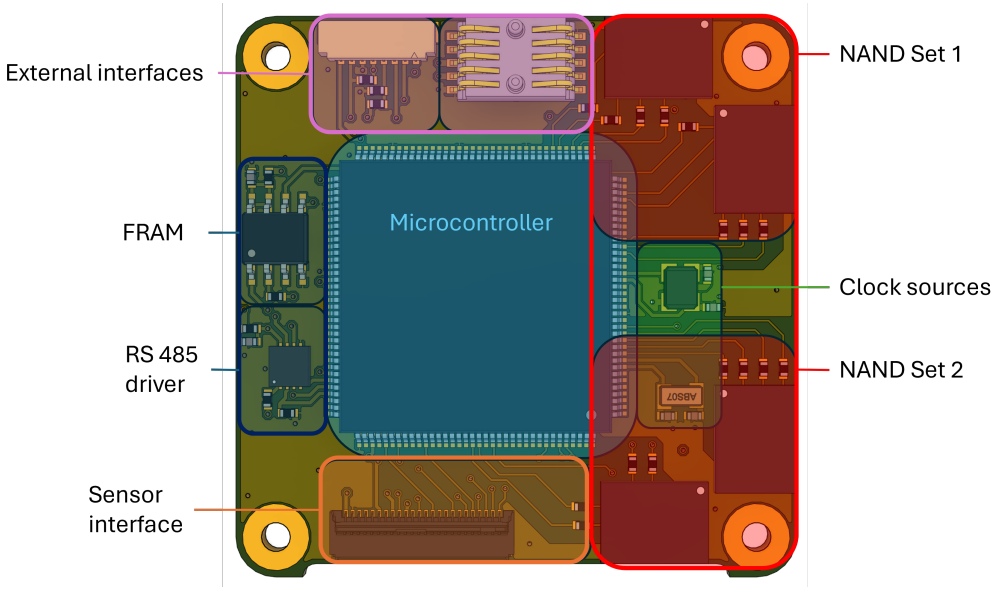


Figure 4.2: Compute Module floor plan (Top side).

PCB floor plan The PCB floor plan presents at its center the MCU, due to its role as the central interface node. The right side of the PCB is reserved for the two memory pairs and the clock sources, while the external interfaces are placed both on the top side, which in the satellite configuration hosts a removable panel that can be opened to provide access to the interfaces. The image sensor interface on the other end, is situated in the bottom portion of the board, to make use of the PCB cutout to route more easily the flexible flat sensor cable.

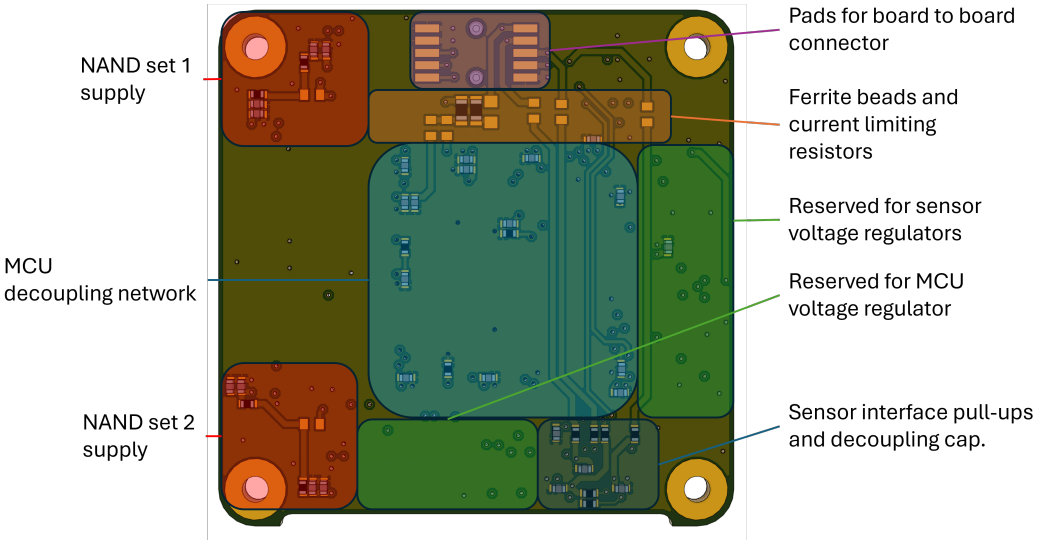


Figure 4.3: Compute Module floor plan (Bottom side).

PCB stack-up The board layout has been developed for an 8-layer FR4 PCB. The PCB buildup used is from the standard pool from Eurocircuits to minimize procurement time and presents the following characteristics:

PCB Layer stack			
#	Name	Type	Thickness
1	Signal Top	Signal	Top Copper 0.018 mm
*	dielectric - 1	Prepreg	PR1080 0.07 mm + PR2116 0.12 mm
2	[2] GND	Signal	Inner Copper 0.035 mm
*	dielectric - 2	Core	FR4 0.2 mm
3	[3] Signal	Signal	Inner Copper 0.035 mm
*	dielectric -3	Prepreg	PR2116 0.12 mm + PR2116 0.12 mm
4	[4] GND	Signal	Inner Copper 0.035 mm
*	dielectric - 4	Core	FR4 0.2 mm
5	[5] Signal	Signal	Inner Copper 0.035 mm
*	dielectric - 5	Prepreg	PR2116 0.12 mm + PR2116 0.12 mm
6	[6] GND	Signal	Inner Copper 0.035 mm
*	dielectric - 6	Core	FR4 0.2 mm
7	[7] VCC	Signal	Inner Copper 0.035 mm
*	dielectric - 7	Prepreg	PR2116 0.12 mm + PR1080 0.07 mm
8	Signal Bottom	Signal	Bottom Copper 0.018 mm

Figure 4.4: Table containing the layer stackup of the Compute Module PCB.

Only a pass-through via pair has been defined and used in the design. The PCB has been designed without the use of blind via to further reduce the manufacturing time

and cost.

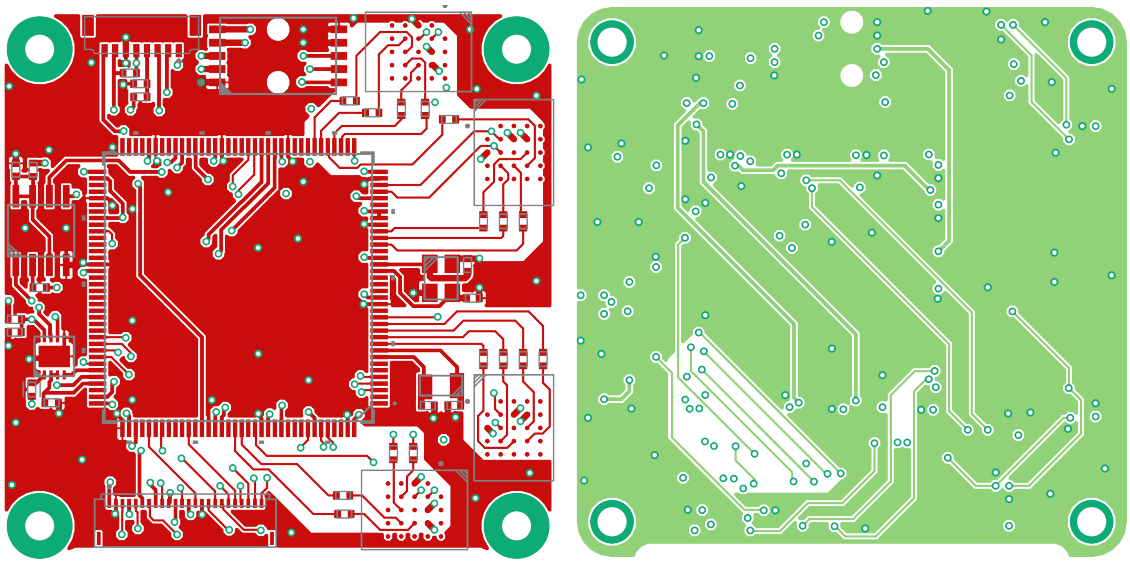


Figure 4.5: PCB signal top layer (left) and signal layer 3 (right).

PCB layers and routing The top layer is used predominantly to route signals, with a good portion of the layer assigned to the routing of the OCTOSPI interface for the SMS. Particular attention has been used in the routing this interface, due to the relatively high interface speed 120Mhz, and the mission-critical aspect of image data transmission to the static memories for storage.

The OCTOSPI routes were appropriately separated and sized to implement a set of impedance controlled microstrips to improve signal integrity.

The specification from the microcontroller indicates a set of rising times speeds that each GPIO can be configured to use. The GPIO pins repurposed for the OCTOSPI interface makes use of the fastest speed setting, that results in a slew time of $3.3ns$ ([24]). Consequently, the width of the microstrips has been calculated to be 0.18 mm of trace width. The manufacturer calculator has been used for speed of use, since it contains already the information about materials used for manufacturing of the boards.

It has been decided to place the high speed memory signals on the surface (rather than in an inner stripline) due to the ease of accessibility of said lines for probing and debugging the signals during driver development.

Moreover, a set of 402 SMD bridges have been placed on the memory data lines to provide a physical interface where test wires could have been soldered to analyze the signal. The pads provide the additional benefit of enabling the addition of in series resistors to the data lines in case it would have been necessary to counter ringing. Luckily, this measure was superfluous, and the successive verification process proved that the signals did not present abnormal oscillations during operation of the memories at the specified 120Mhz frequency.

Other signals that are routed on the top layer are some of the DCMI signals, the clocks form the two clock sources, and most of the signals towards the RS485 and the FRAM memory modules.

The layer 3 and 5 are used for other signals that were not routed in the first layer. The signals in these layers were intentionally spaced out and sandwitched between two ground layers to improve singnal integrity for the high speed interfaces like DCMI.

The Layer 7, also referred as VCC layer, contains the supply domains used for the devices on the board and the MCU. Specifically, the layer surface is filled with two domains: the inner voltage domain is responsible for supplying the MCU and is derived from the other external domain through a ferrite bead and current limiting resistor. The outward domain is a 3.3V domain that originates from the board to board connector and is used to supply all the others devices. Every device present on the board interfaces to this voltage domain via a dedicated ferrite bead and current limiting resistors.

The purpose of the current limiting resistor is to reduce the maximum current in case of a short circuit in the device, limiting the heat and the damage generated in the time between the event and the activation of the protection systems.

Lastly, the bottom layer hosts most of the power distribution components and the power supply lines necessary for the image sensor. In a successive iteration, this layer will host the DC-DC switching voltage regulators for both the sensor and the MCU and the latch-up protection system.

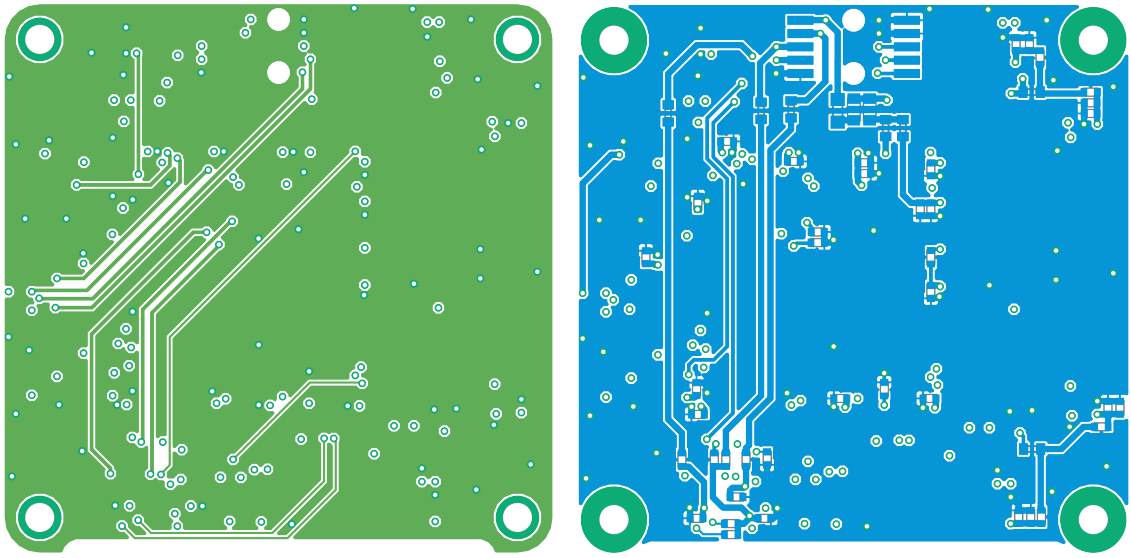


Figure 4.6: PCB signal layer 5 (left) and bottom signal layer (right).

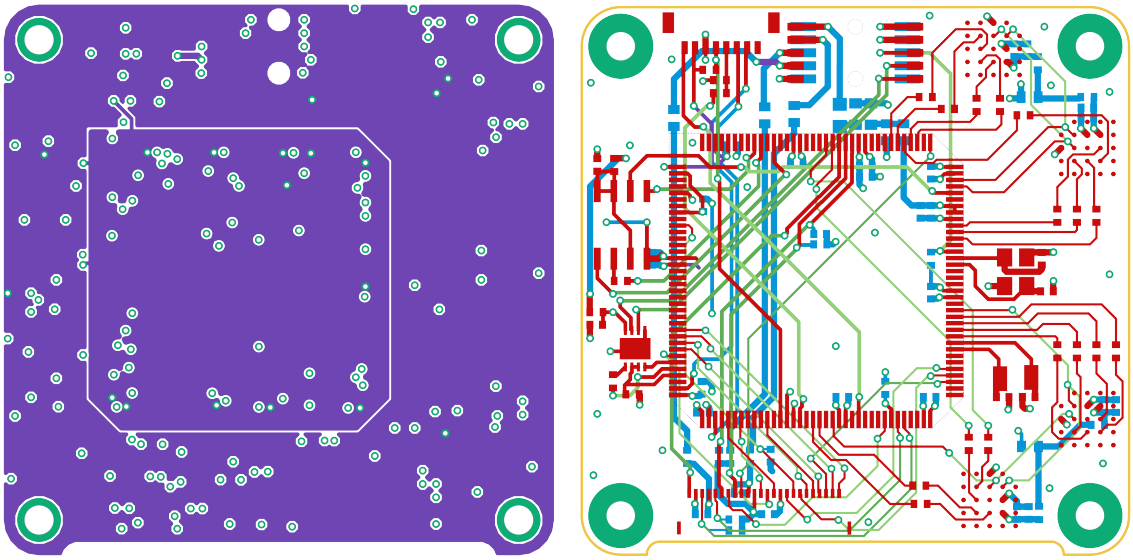


Figure 4.7: VCC distribution layer (left) and overview of signal layers without filling (right).

4.1.1 PCB manufacturing

After an initial inspection of the PCB, the boards have been assembled by the author in a TU-Delft workshop. Most of the top layer of the compute module has been assembled making use of a stencil and a reflow oven. During the first assembly, it has been discovered that the stencil cutouts for the MCU pins were oversized, therefore an excessive amount of paste were being deposited. During the reflow process, this caused the excess solder to be sucked by the vias that are in proximity to the MCU pads (most of which used to connect decoupling capacitors). This created shorts that required the MCU to be removed and successively hand-soldered.

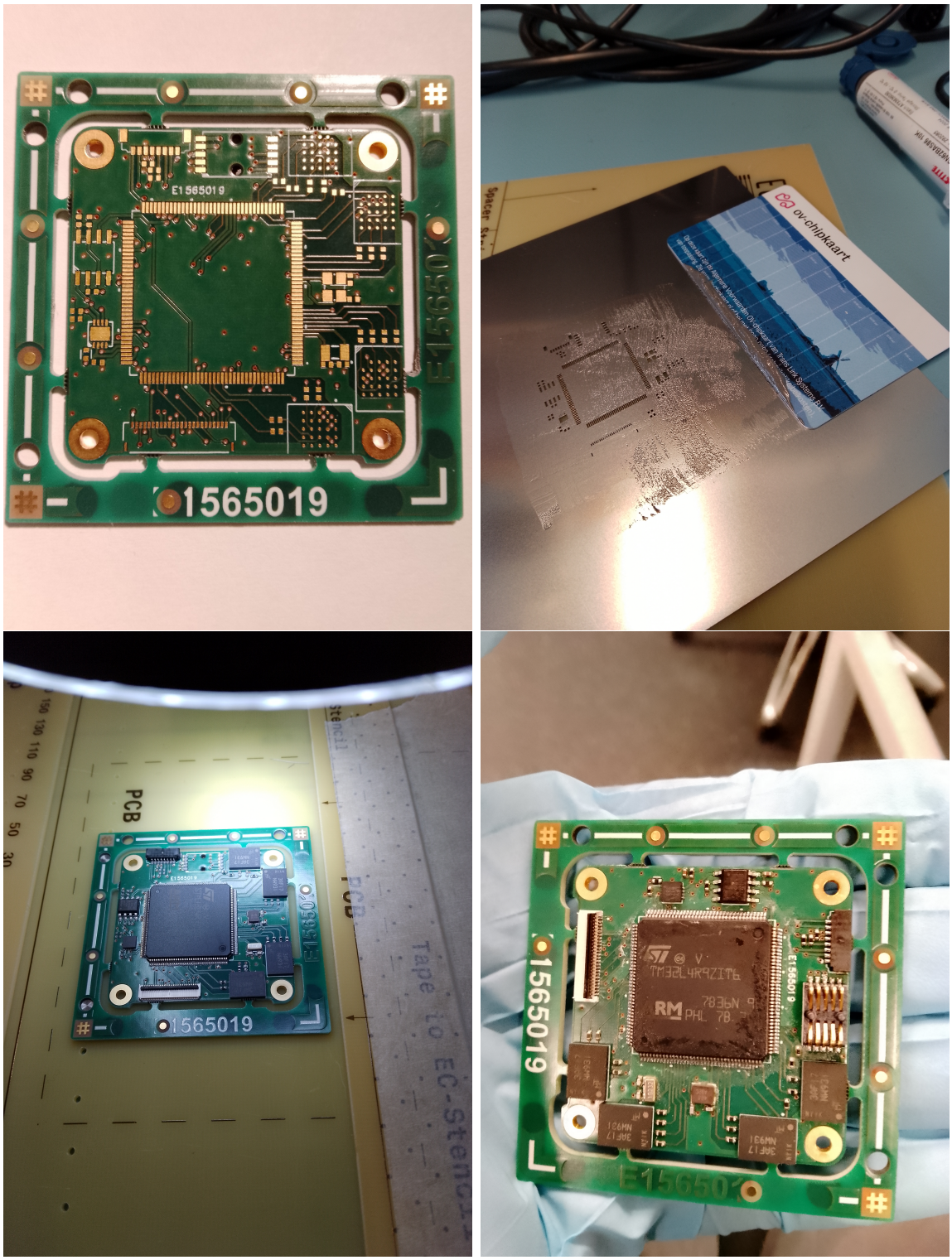


Figure 4.8: Images of the Compute Module assembly: on the top right an image of the PCB as delivered from the manufactured. On the top right, stencil utilized for the solder paste distribution. Bottom left includes the board with all the components placed, before placement in the reflow oven. Bottom right shows the assembled board after the reflow process.

Chapter 5

Software Design

5.1 General Software architecture

The on-board software consists in a layered architecture, consisting of the following layers:

- Driver layer.
- Board support package (BSP) layer.
- RTEMS RTOS layer.
- Application layer.

The lowest layer consists of bare metal drivers that utilize the hardware functions directly, by interfacing with the hardware registers. The successive layer, the Board Support Package (BSP) integrates the drivers and the hardware specific software elements and configurations that enable the execution of the upper layers and allows them to interface with the hardware. The BSP includes a bootloader and the necessary software to support the execution of the RTEMS (Real-Time Executive for Multiprocessor Systems) Real Time Operating System (RTOS).

The layer right above the BSP is the RTEMS real time operating system itself. RTEMS provides a set of features and services to the application, such as task management, inter-task communication and synchronization, interrupt management and time management.

The application makes use of the underlying layers to implement the high level functions assigned to the onboard software.

Figure 5.1: Hierarchical software architecture (left), RTEMS logo (right)

5.2 The RTEMS real time operating system

RTEMS (Real-Time Executive for Multiprocessor Systems) is an open source RTOS (Real time operating system) widely used in space applications.

In the past 20 years, RTEMS has been utilized as RTOS in flight computers and payload controllers for a multitude of space missions organized by space agencies around the world [15], including the missions:

Developed by the European Space Agency (ESA):

- Herschel
- Planck
- BepiColumbo
- ExoMars rover
- Galileo GNSS constellation

Developed by the National Aeronautics and Space Administration (NASA):

- Mars Reconnaissance Orbiter
- Dawn
- Fermi
- Magnetospheric Multiscale
- Solar Dynamics Observatory
- Parker Solar Probe
- Juno
- Curiosity rover

The RTEMS software is an open source project that is distributed under the terms of the GNU GPL2 license and supports the execution of the Core Flight System (CFS) and Core Flight Executive (CFE) middlewares.

Execution of the RTEMS RTOS is supported for supports most radiation hardened processors, including in the European context GR712RC, the newer GR740 and many more processors and processor core IPs. The supported hardware includes a specific set of processors from several architectures like ARM, PowerPC, Intel, SPARC, MISP, RISC-V and more.

An important portion of the software development work has been the development of a Board support package that would allow RTEMS to be executed on the custom hardware developed during the thesis activities.

Thanks to the work of this thesis, RTEMS now supports the STM32L4+ family of microcontrollers, enabling its utilization on accessible, inexpensive hardware that is suitable for PocketQube and CubeSat applications.

5.3 Software Development Process

The software products have been developed iteratively, focusing initially on the lowest levels, proceeding to the highest, that is the application software. The development process follows and derives from the model philosophy. An overview of the assignments between the software products and the models is displayed in figure 5.3.

The methodology employed for the software products development is a test-based approach, where each software component is developed, tests are generated, and the suitability of the software produced is verified.

Due to schedule considerations, the development process has been focused on the generation of earliest functioning software solution, providing rapid results that can demonstrate early the feasibility of the design and possibly provide useful performance figures. After simple functional implementations are developed, it becomes easier to identify issues and potentially useful abstractions, that can be addressed in a successive rework. The rework phases allow for managing the technical debt and to reduce the explorative code to manageable level, while avoiding investing time in abstractions that are than not necessarily used.

The resulting iterative work pattern consists in phases of exploratory and experimental development and consolidation phases, during which abstraction are constructed, and the code is harmonized. The method aims to achieve a balance between feature implementation and stability.

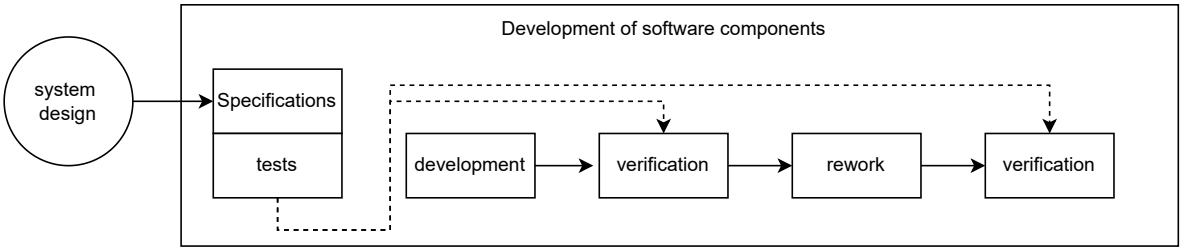


Figure 5.2: Development process of each software component

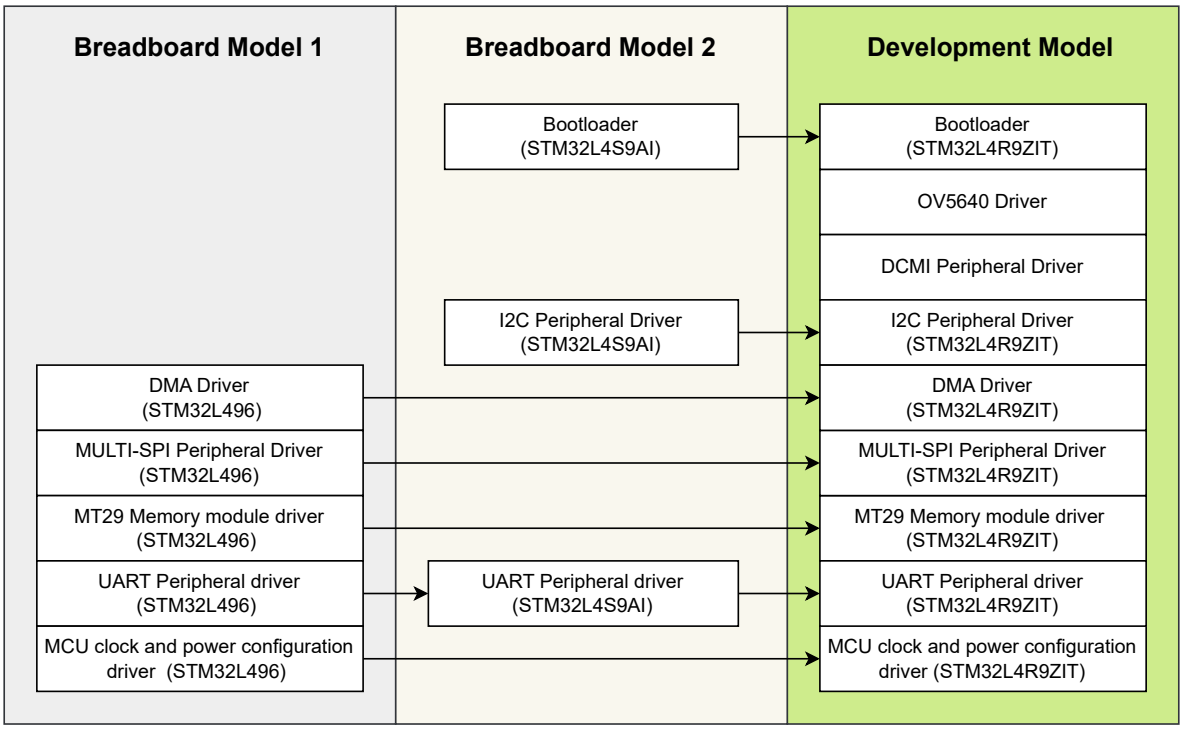


Figure 5.3: Model assignment of software products.

5.4 Board support package

The RTEMS Board Support Package (BSP) is a collection of software and configuration files used to support RTEMS on a specific hardware platform, providing the necessary low level software components necessary to allow RTEMS and the application to interface with the hardware.

At the time The RTEMS documentation provides limited support in relation to the development of board support packages. Therefore, the board support package has been developed using as reference the STM32F4 and STM32H7 BSP developed by Embedded Brains Gbh [8] and available in the RTEMS sources [12]. The BSP has been analyzed to determine undocumented practices and software interface in regard to:

- BSP provided RTEMS initialization sequence.
- Shared armV7m linker script template.
- Integration with STM HAL and LL libraries.
- WAF BSP build specification.

RTEMS supports a layered source structure each BSP so that each BSP can be built reusing previously developed compatible layers. The layers of each BSP are respectively divided in CPU independent, CPU Architecture dependent, CPU dependent, Board dependent, Peripheral dependent.

As specified in [26] the minimal set of components necessary for a BSP are Low-level initialization code, a console driver and a clock driver. The board support package

moreover contains all the software products necessary for the application software to fulfill the assigned functions.

Amongst the drivers developed and integrated in the BSP, a set of software products have been developed from scratch as bare-metal drivers, while a set has been integrated from ST Libraries. Including ST Libraries is something that was initially avoided. Successively a limited number of drivers from the HAL libraries have been included in the BSP. The motivation behind this change was due to the necessity of having on hand functional drivers to debug hardware issues, since it is time consuming to debug new hardware and software concurrently. Therefore, the inclusion of the HAL drivers has been an useful addition in the scope oh hardware verification, but the inclusion brought with it a set of drawbacks in terms of software test coverage, perceived reliability and firmware image size. These aspects considered, the HAL drivers are a development support and their removal from the codebase is recommended for future development of the engineering model.

The next table presents a summary of the software product's origin and the specific verification model to which the verification process has been assigned.

Software product	Origin	Verification model
Bootloader	Bare metal Dev.	Pre-Development B
OV5640 Driver	Bare metal Dev.	Development Model
DCMI Peripheral Driver	STM HAL	Development Model
I2C Peripheral Driver	STM HAL	Pre-Development B
DMA Driver	Bare metal Dev.	Pre-Development A
OCTOSPI Peripheral Driver	Bare metal Dev.	Pre-Development A + adaptation for DM
MT29 Memory module driver	Bare metal Dev.	Pre-Development A + adaptation for DM
UART Peripheral driver	Bare metal Dev.	Pre-Development A
MCU clock and power configuration driver	BMD & STM HAL	Pre-Development A + adaptation for DM

5.5 Driver development overview

Most of the drivers developed in the project are bare-metal drivers. Bare metal drivers are designed to operate directly on the hardware without an operating system or any other software layers, having direct access to the M4-Core or MCU peripheral registers.

In the specific case of this BSP, it has been decided to decouple driver initialization and the BSP initialization due to the following considerations:

- The utilization of RTEMS has been considered as an exploratory endeavor. The ability to develop a functional Board Support Package in the time frame of the activities was not certain. Therefore, to minimize technical risk, the drivers have been developed in a less decoupled configuration, so that they could be used (or reused in the future) in a bare metal context with minimal rework effort.
- The MCU and peripheral registers reset condition presents functional defaults that can be used right away for the initialization of the BSP. Therefore, clock and peripheral configuration can be performed after RTOS initialization during application initialization.

Consequently, the system initialization is performed via a custom initialization system executed in the application initialization task.

Each driver has been tested independently on the pre-development models (BM1 and BM2). The atomic verification of software components has been supported by the bare metal approach, that allowed for:

- Independent execution and testing of each software product
- Direct utilization of the hardware features via access of the hardware registers.
- Ease of assessment of the resources utilized specifically by the driver (stack utilization, size of the executable sections (.DATA .TEXT .BSS).

Furthermore, the tests for the STM and the DCMI interface have been reworked in automated tests that can be executed on the Development Model and during the rest of the Product Lifecycle to rapidly verify function and performance of system elements (For example during hardware acquisition for the Engineering Model, or for diagnostics on successive models).

5.6 STM32L4R9ZIT BSP Development

5.6.1 Analysis of the Initialization sequence provided by the STM32H7 BSP

This section presents the results of the analysis of the initialization sequence of the STM32H7 BSP, with the purpose of using these finds as a starting point to introduce the initialization process utilized in the STM32L4R9 BSP that has been developed.

The initial start file contains the entry symbol of the executable, it is the first file presented to the linker [26] and starts the process to link the executable. The initial assembly code performs the minimal actions to allow execution of the C code and

successively calls sequentially two low level initialization functions: *bsp_start_hook_0*() and *bsp_start_hook_1*().

The initialization assembly code is architecture specific: since the ARMv7 processor architecture is supported, the relative source file has been included as an architecture shared component.

The BSP start hooks provide the earliest interface with the BSP. Typically, the first hook is used for initial configuration of the MCU, including the configuration of essential systems and peripherals. In the case of the STM32H7 BSP, the initialization process is extended by the initialization of all the peripherals, specifically

- MCU power settings configuration: for stm32 microcontrollers this is done by setting the power control registers, configuring this way the power state of the MCU. This often involves setting the values for the internal voltage regulators.

  The clock regime of the MCU is often dependent on the power state. For instance, during development for the STM32L4R9ZIT it has been necessary to change the power-state to a boost mode before configuring the PLL to clock at 120Mhz. This is the reason why power configuration is done before clock configuration.

- MCU clock configuration: This involves enabling the clock sources and configuring the PLLs. The configuration is usually not performed in a single step but rather via a step increase of the MCU clock, with successive checks of clock stability at each step.

- Peripheral clock configuration: In stm32 MCU in the reset state the peripherals and the MCU AHB and APB buses are not clocked in order to save power. It is necessary to clock the peripheral that are meant to be used during system initialization. This operation is performed by setting the appropriate fields in the Reset and Clock Control (RCC) register.

- Peripheral configuration: at this point it is possible to configure the MCU peripherals as for necessity. The HAL_init function is a wrapper for initialization procedures of the HAL driver. This includes not only the configuration of the peripherals and the relative GPIO pins, but also the creation of the necessary objects in memory used for future operation.

- Configuration of the Flexible Memory Controller (FMC) peripheral. The FMC can be used to expand the available RAM by memory mapping a connected SRAM memory device. The peripheral is configured early, to have this resource available in the successive phases.

In the case of the STM32H7 BSP functions above are performed via the use of functions provided by the HAL driver, that has been integrated in the BSP.

The second BSP start hook is typically used for the preparation of the memory area. This includes copying the .DATA, .FAST_TEXT, .FAST_DATA from the load to the runtime area, located in RAM. Successively, the RAM region reserved for the .BSS section is cleared. Lastly, the .TEXT region is copied in the runtime area and execution is branched to the runtime area. At this point the execution is performed exclusively from the RAM runtime area, and the successive step is the execution of the boot_card() function, that is the function that starts the high level initialization of the RTEMS system.

The rtems_initialize_executive() makes use of a system initialization linker set [11] that initializes only the RTEMS features that are necessary by the specific application.

Successively RTEMS starts multitasking. At this point the ready task with the highest priority gets executed, in this case the application initialization task.

Figure 5.4: Low level BSP initialization (first initialization phase)

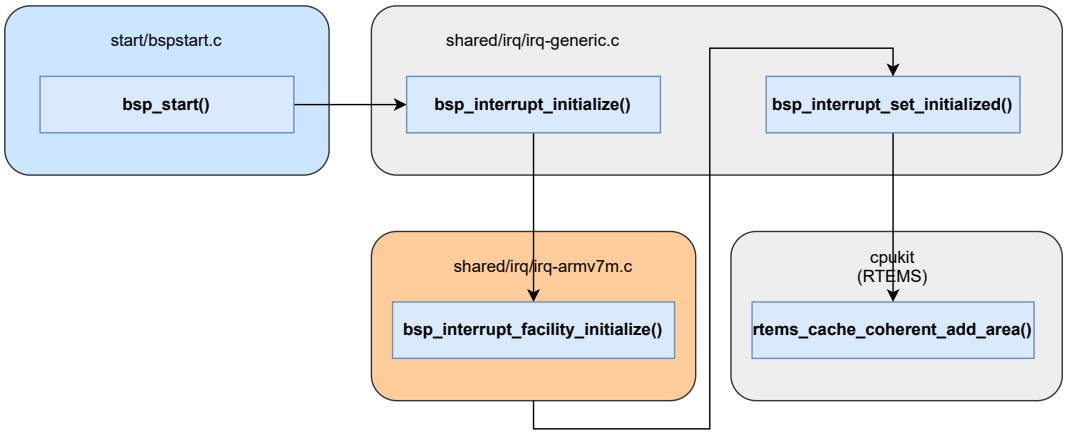


Figure 5.5: BSP initialization (second initialization phase)

5.6.2 Initialization manager for imaging system software components

The initialization of the peripherals for the STM32H7 BSP makes use of the HAL_init() procedure. This procedure consists in a sequential execution of the peripheral drivers that are enabled via a set of defines. The usual workflow relies on the generation of the system configuration, via the utilization of the CubeMX software from ST, which generates the project files, including the necessary defines to allow of the configuration of the behavior of the drivers and initialization directive.

In the context of development for the RTEMS system, this approach necessitates to integrate newly generated project files in the RTEMS BSP source and successively recompile the BSP. This process is time-consuming due to the high amount of project files, the necessity of updating the build specifications, and the necessity of an automated software verification chain.

The STM32L4R9 BSP does not utilize a static sequential peripheral driver initialization like the STM32H7 BSP, but rather makes use of a dynamic initialization system that has been specifically developed to satisfy specific needs.

Often drivers needs to be initialized in a specific order, due to the necessity of having some peripheral already initialized as a requirement: For instance it is necessary that the clock sources and clock configuration to be configured before initialization of the UART driver. Likewise, the clock configuration requires itself that the power state of the MCU has been already configured.

A particular example is the one displayed in figure 5.6, that displays the software drivers and configurations that are prerequisites to the DCMI peripheral driver initialization. Albeit all the elements are dependencies of the DCMI driver, it is necessary that the initialization steps of the various components are executed in a specific order.

A similar scenario presents itself for the initialization of the other peripheral drivers, like for instance the driver for the OCTOSPI interface.

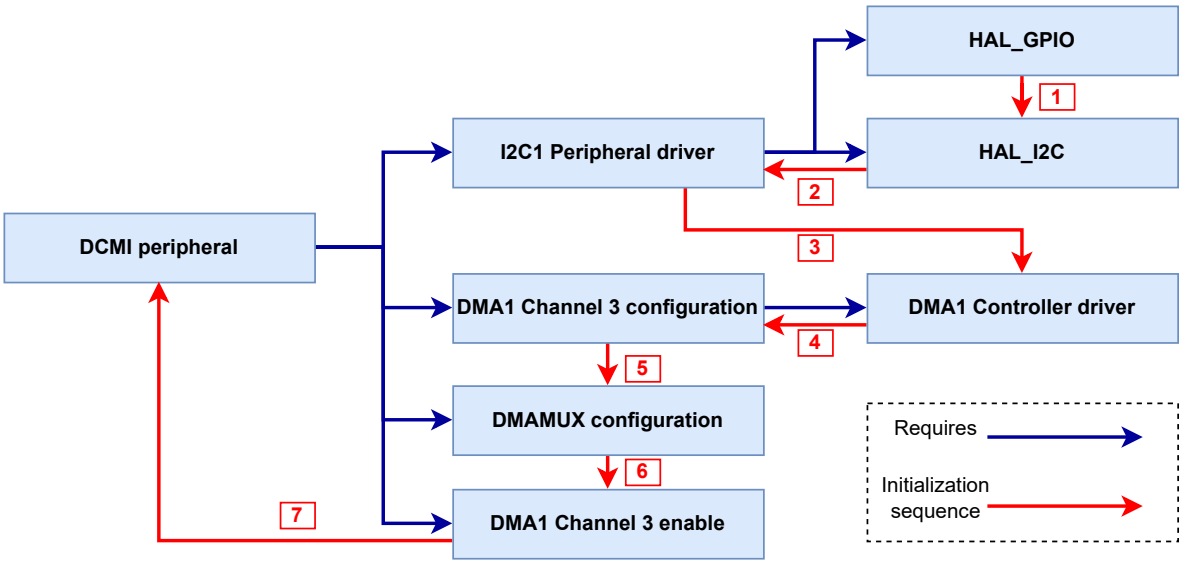


Figure 5.6: Initialization order necessary for correct initialization of the DCMI peripheral

The development of an initialization system has been therefore been considered necessary and the following set of requirements has been defined, to guide its development.

- The initialization system shall be independent of the BSP
- The initialization system shall be portable to a bare-metal software architecture.
- The initialization system shall perform an initialization sequence that is minimal in respect to hardware features utilized in the executable.
- The initialization system shall avoid re-initialization of hardware resources.
- The initialization system shall detect resources conflicts between utilized resources.
- The initialization system shall be able to initialize, de-initizlize, reinitialize peripheral drivers dynamically.
- The initialization system shall provide information about initialization sequence.

The architecture of the initialization system follows the idea of using a pooled linked list to keep track and build the initialization sequence of the drivers progressively, while the software is executed.

The objective is to edit the sequence stored in the linked list as soon as a resource is used, and perform the initialization right after.

Initially, a pool of nodes of fixed sizes is statically reserved in memory for the linked list elements. Whenever a software element requires a specific feature, in invokes the *hwlist_require* method, with as arguments the initialization function handler required and an optional argument that identifies a "parent" function handler. If the parent handler has been provided (i.e. is not NULL) the hardware node is inserted in the hardware list right after the parent node. If the parent handler is NULL, the hardware node is added as head of the hardware list.

Before exiting, the *hwlist_require* calls the *hwlist_iterate* method. The *hwlist_iterate* methiod executes the initialization functions pointed by the initialization handler registered in each node, iterating from the tail of the list up to the head.

During the execution of a initialization function, it is possible that uninitialized hardware resources are requested and therefore the process repeats iteratively, until all the dependency layers are explored.

The result is a process that keeps track of the dependencies between the different drivers and initializes the requirements as needed. It also provides the following benefits:

- The resources are available at the earliest moment, since the initialization actions are performed as soon as the underlying requirements are satisfied.
- It allows keeping track of the order of initialization of the whole system.
- It allows determining if a resource has been initialized and avoids conflicts between resources that share the same components.

Moreover, in the future it would be possible to expand on this system, implementing new feautures. For example, in case of failure of a system element, it would be possible to deinitialize the resource. A monitoring task could keep track of the changes to the hardware list and address issues by issuing reinitialization or other recovery actions (like a system reset).

The system respects the requirements formulated for it: since it does not depend on any software components and works directly with arbitrary functions' handler derived directly from the source, it is easily portable to another system architecture, may it be bare metal or another RTOS.

The figure 5.7 shows the set of operations and the respective order for the nested initialization of two components. Instead, figure 5.8 shows the initialization process that has been implemented for the DCMI interface.

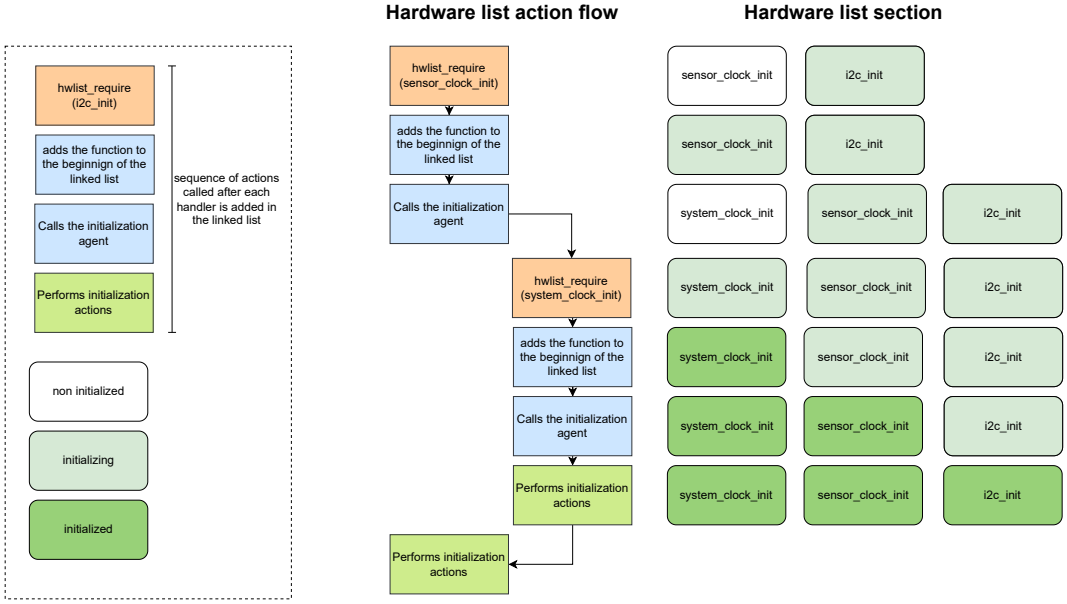


Figure 5.7: Actions performed by the initialization system when a resource is required

Figure 5.8: Initialization flow for the DCMI peripheral

5.6.3 Initialization process for the STM32L4R9ZIT BSP

This brief section describes the initialization process for the STM32L4R9ZIT, to highlight the differences in comparison to the STM32H7 BSP. It is a minimal version of the BSP initialization process, and has been presented because it provides a good example of the minimal directives that are necessary to implement a BSP initialization.

The most relevant change is the removal of the hardware initialization during the BSP, that has been rather executed via the hardware list manager in the application initialization task.

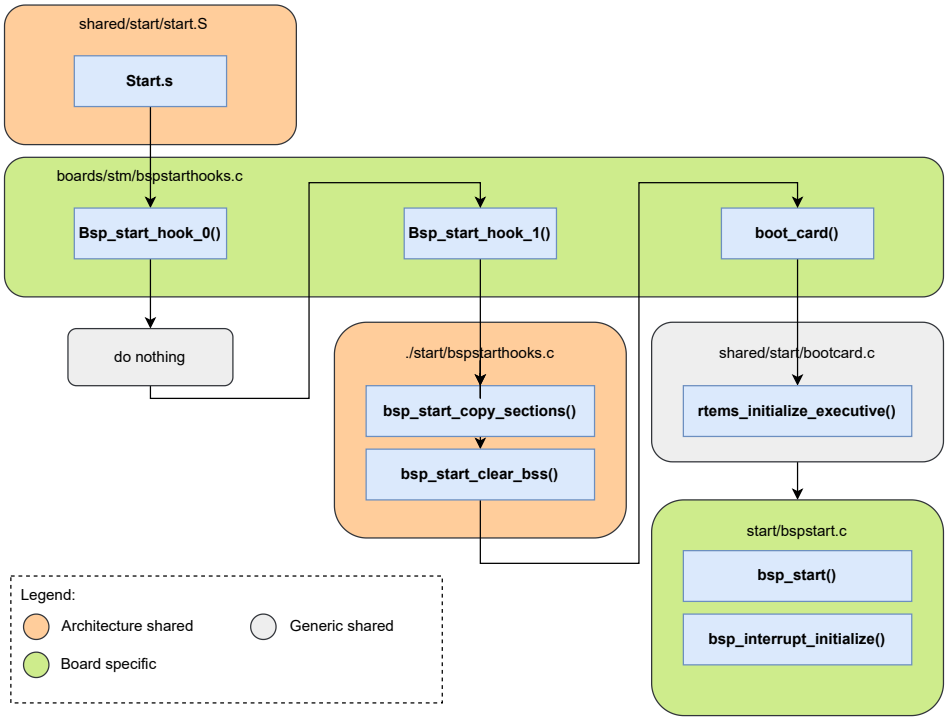


Figure 5.9: Initialization flow of the STM32L4R9ZIT BSP

5.6.4 CMSIS Header libary

The Common Microcontroller Software Interface Standard (CMSIS) (cite) is a set of software interfaces for Cortex-M and entry-level Cortex-A processors. CMSIS provide functional defines for MCU registers and hardware specific implementation of low level operations. It allows to easily work with MCU registers, referring to them via the use of pre-defined and standardized macros, rather than the specific memory address. The standardization of these interfaces allows for software to be portable between MCU

that support the standard and that feature the same high level registers (and therefore peripherals).

The CMSIS source is composed of a set of header files. ARM (The processor core IP designer) provides CMSIS sources for each processor core, in this case for the Cortex-M4 core. ST Microconductors provides instead header files that provide interfaces to the MCU peripherals.

The utilization of CMSIS standard has been particularly useful in the development of the low level software, since most of the drivers developed are bare metal and interface directly with the registers via the use of this interface. In particular, the drivers developed for the Breadboard model A (STM32L476) have been easily ported to the Breadboard Model B (STM32L4S9AI) and successively on the development model (STM32L4R9ZIT) seamlessly, thanks to the utilization of this interface. The steps necessary to port the firmware have been the substitution of the ST's CMSIS headers with the ones for the appropriate processor (since all the MCU here listed make use of the Coretex M4 core). The verification of the drivers previously resulted right away in positive results for each model. Only minimal changes were required for the QUADSPI/OCTOSPI drivers since the similar peripherals are not actually the same and therefore the interfaces makes use of slightly different register organization.

5.6.5 Console driver

As specified in [26], the termios software layer works as interface between the application and the Low-Level Device Driver.

The [26] specifies how in the default application configuration RTEMS opens during system initialization a /dev/console device file to create the file descriptors 0, 1 and 2 used for standard input, output and error, respectively.

The BSP and Driver Guides provides a description of the three elements that are necessary for a serial device driver:

- A section in the build specification: addition of the shared source `../../shared/dev/serial/console-termios.c` and the bsp source `console/console.c`
- A low level driver providing:
  - handler table

  - device context specialization

- A BSP-Specific initialization routine `console_initialize()` that calls `rtems_termios_device_install()` providing a low-level driver context for each installed device.

The console driver implemented for the STM32L4R9ZIT BSP registers the UART2 interface (that is the UART routed to the debug interface) to the termios file descriptor tty2, used for error reporting. This allows for the RTEMS error handling functions an interface to print useful information in case of errors or non-nominal behavior. The application software is also meant to use the tty2 for debug prints and can do so via the fprint() function (using the termios console driver currently configured with the HAL driver) or via direct access to the bare metal driver function `write_char` and `write_buff`.

5.7 Application Software

The Application software consists in the implementation and orchestration of the processes necessary to achieve the functional requirements. The application software realized for the development model covers only the core functionalities of image acquisition and data handling. A simple command interface via the debug adapter has also been developed, but other aspects like communication to the spacecraft bus, command handling and FDIR features have not been implemented.

The application software is organized in two different tasks: the application initialization task and the frame handler task. Figure 5.10 displays the execution flow for the DM software when debug adapter is not attached or specific commands are not issued via the debug interface.

The debug command interface works in implementing a set of GDB scripts developed via the python extension that ar able to perform a set of functions, useful to operate the system. Respectively:

- Download of the images stored in static memory to the host PC.
- Call of a handler directive to clear a specific region of the static memory.
- Call of a handler directive to download initialization log and system status.

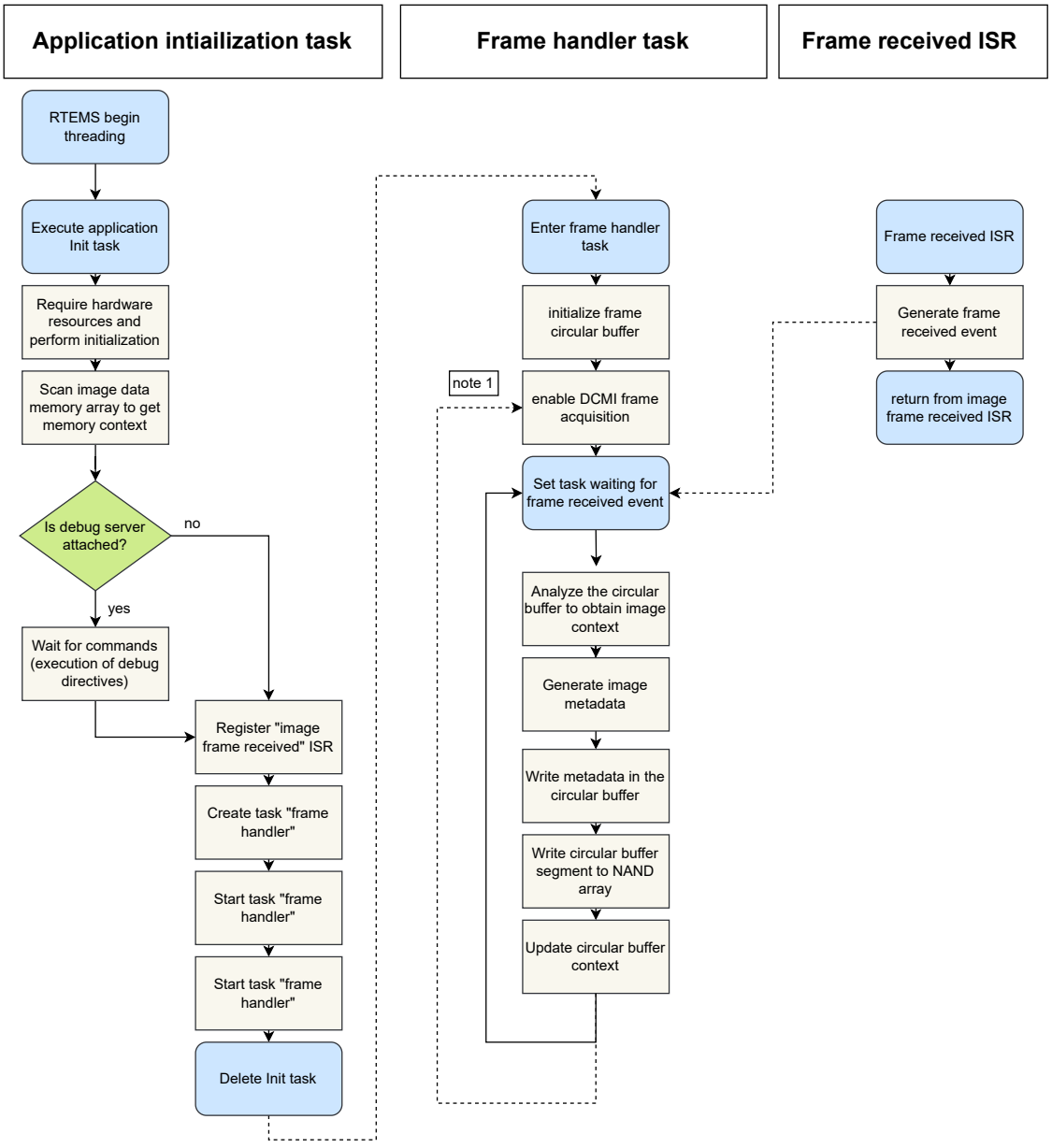


Figure 5.10: Application software flow diagram.

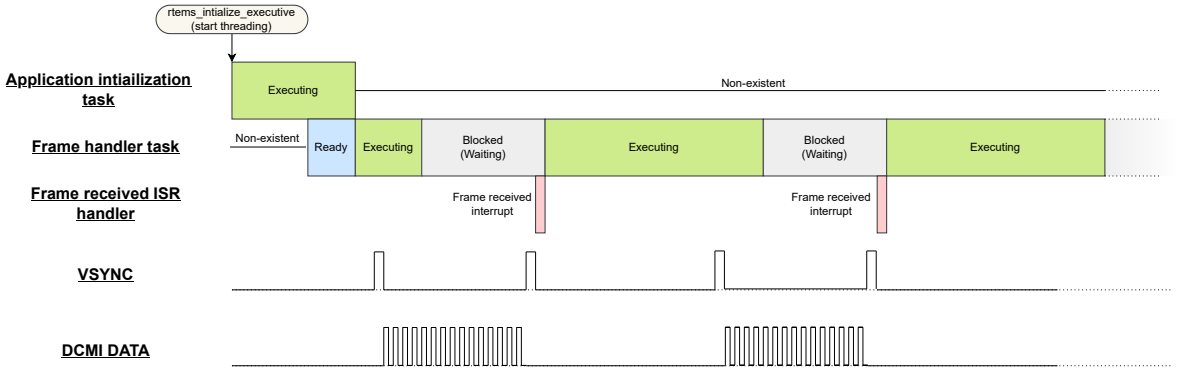


Figure 5.11: Task scheduling diagram.

5.7.1 Initialization task

The application initialization task is registered right away and is executed after RTEMS begins multithreading, that is after the final steps of low level initialization, consisting of the actions defined in bsp_start(). The first steps executed by the initialization task are to require the hardware necessary for the execution of the functionalities via the use of the hardware list initialization system. The requested software elements are:

- Debug Uart interface.
- DCMI inteface.
- NAND memory subsystem (that includes OCTOSPI peripheral, MT29F memory driver and many more).

In total, the initialization system from these three requests initializes independently 27 software elements that are requirements to the selected items. Some of these elements are for instance change in power state to boost, applying an overvoltage to the core, increasing system clock from the reset state of 4Mhz to 120Mhz, and many more.

To operate the NAND array, the system requires contextual information on its state: Such information includes the areas in the address space where image data is stored, the amount and location of bad blocks.

The system has been designed to never rely memory state information held in volatile memory, due to the possibility of losing the information in system resets. It has been decided against storing state information on the SMS. This is because for performance reasons a file system has not been implemented on the NAND array and therefore state information would need to be written in specific pages/block of memory, since

the system would need to know from a reset state where to get such information. But writing frequently in the same blocks (like it would be necessary in this case) would easily lead to depleting the write endurance of the NAND blocks in question, creating serious issues for the longevity of the system.

Instead, the system reads the totality of the memory array during initialization, logging in a structure statically allocated in the RAM .BSS section, containing information about utilization of the memory. This information is essential, since it indicate in the NAND address space the presence of images. The information is later used to identify where to write new acquired images and for retrieval of the images already captured.

Due to the high total capacity this operation requires quite a bit of time (a few minutes). This can be managed operatively, by defaulting in searching only a minor region of the array and to disregard the data contained in the rest of the section, or to postpone this "instrument commissioning operation" to a later, command driven operation, rather than an automated execution at system initialization.

Once the NAND memory context has been built, the software execution is either stopped automatically if a debug interface is attached, or successively, a set of actions are performed to register and enable the Frame Handler Task and the Frame Received ISR.

Once these actions are performed, the task Frame Handler is set to Ready and the initialization tasks deletes itself. Being the frame handler task the only task in the ready state, the task is dispatched and its execution begins. Given the conditions specified, this happens independently of the scheduler that has been used.

In the scope of the Development Model, there has not been a necessity to analyze and conduct a tradeoff amongst scheduling algorithms. It is intended that once command handling, watchdog and communication tasks are introduced, and also the prospect of re-uising the software configuration on other systems is more thoughtfully analyzed in the future, more constrains will be identified. At this point it will be beneficial to determine the best scheduling algorithm for the application via a trade-off study.

The current software image configures RTEMS to use a Deterministic Priority Scheduler, that is a preemptive deterministic scheduling algorithm that considers a priority assigned for each task. Therefore, the task executing on the processor at any given time is the task with highest priority, and the algorithm provides round-robin access

to tasks with the same priority level. The ready chain makes use of a FIFO, therefore in a group of equal priority tasks, tasks will be executed in the order as they become ready or the FIFO order.

5.7.2 Frame Handler Task

The frame handler task at its beginning initializes and clears the frame circular buffer.

The purpose of a frame buffer is to provide a memory region where data can be rapidly stored by the image acquisition peripheral, via the use of a DMA transfer.

The system has been designed to work with a great variety of CMOS sensors featuring a parallel interface. Most CMOS sensors output either raw pixel values data or JPEG compressed data.

Raw image data is transferred sequentially in a raster pattern. The image data is often the output of de-mosaicing the pixel readout and applying image correction processing, like white balance. For raw image data, the data transfer is synchronized via the use of the HSYNC and VSYNC signals, that work as a data valid mask and indicates respectively the end of a frame line and the end of the frame. The polarity of these signals is dependent on the sensor and can be configured in the sensor driver.

JPEG data is the output of an additional compression process applied on the raw data and performed by the CMOS sensor SoC. A JPEG image is a bytestream of variable size identified by a header(0xFFD8) and a closer (0xFFD9). The size of the compressed image depends on the homogeity of the initial raw image and the loss factor of the compression algorithm utilized. Therefore, the transfer is synchronized via a different use of the synchronization signals: HSYNC signal is used as a data valid signal and VSYNC indicates the start/end of the image transfer. Figure 5.12 obtained from the MCU reference manual [24] shows the transmission signal timings for a JPEG image.

The ram buffer has been sized to contain the maximum size that can be transferred via a single-buffered DMA transfer, therefore $262.14KB$ (or 65535 4 byte words). The frame buffer is statically allocated in the MCU volatile memory and represent a notable size (40.1%) of the total RAM capacity of $640KB$.

Via experimentation, it has been observed that typical 720p JPEG images captured via the use of the test sensor (OV5640) utilize a size in memory of up to 100KB and up

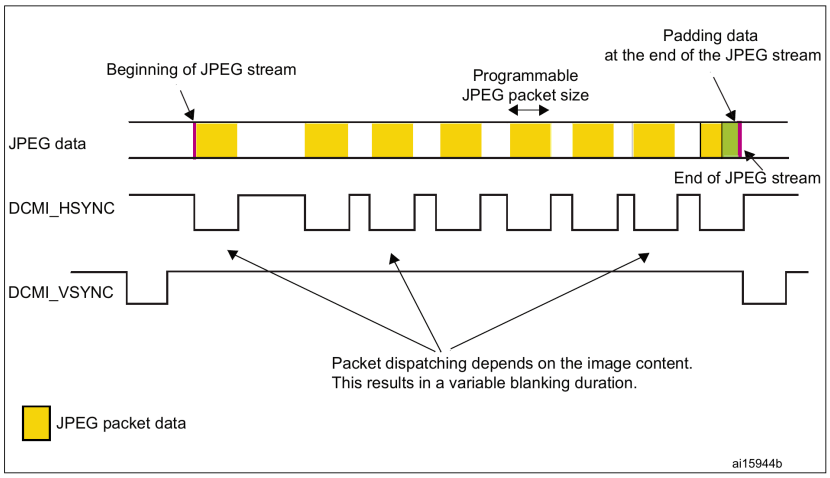


Figure 5.12: Timing diagram for JPEG transfer over DCMI interface. From ST STM32L4R9 reference manual.

to 200KB for 1080p images. Therefore, the current frame buffer size would be able to support capture of 1080p (2 Mpx) JPEG compressed images.

In the execution flow of the Frame handler task, the next step is enabling the image capture for the DCMI peripheral. The peripheral can be configured to capture frames in a single or continuous frame acquisition. In both cases a Frame Ready interrupt is generated when a complete frame has been received, with the only difference that the peripheral "capture enable" flag is reset after the acquisition of a valid frame for the single acquisition mode and needs to be re-set by the software to enable a successive acquisition.

Once the peripheral is enabled for capture, it waits for the next frame start sequence to acquire and transfer the image data.

At this point the frame handler task enters an infinite loop, at the beginning of which changes its state from Executing to Blocked, waiting for an event that is generated by the Frame Captured ISR. The Frame Captured ISR has been registered during the application initialization. The associated interrupt is generated only if a complete valid frame has been received by the DCMI interface, and therefore it can execute only after the interface has been enabled for acquisition.

The only purpose of the ISR is to generate the event that readies the Frame Handlder once a frame is ready to be handled.

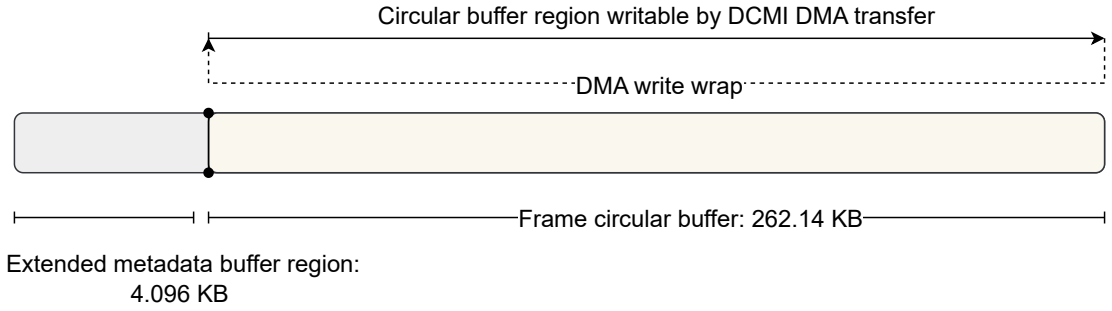


Figure 5.13: Structure of the circular buffer.

When the frame handler task continues its execution, the image data is present in the frame circular buffer. The nature of a circular buffer results in the placement of the image data in an undetermined section of the buffer. Therefore the first step of the current software iteration scans the circular buffer to find the header and the closer of the newly captured image. It has to be noted that once an image in the buffer has been completely handled, its header and closer in the circular buffer are overwritten, such that previous images cannot be found during the scanning operation. Contextual information about the buffer are determined and stored at each iteration, consisting of:

- Image JPEG header pointer
- Image JPEG closer pointer
- Size of the captured image
- current position of the scanning pointer

To reduce the scanning overhead, the scanning operation resumes in the circular buffer from the last position scanned in the previous iteration (that usually is the position of the closest of the previous image).

Usually, in case of a correct acquisition, the header of the next image is placed by the DMA right after the closest of the previous image. To avoid readings that could be caused by the acquisition of bytes from the peripheral in case of spurious activation of the synchronization signals, the header of the image is still searched, instead of copying directly the data right after the closest of the previous image capture.

The successive actions performed by the frame handler task are described in figure 5.14 making use of an example of the capture of two consecutive frames.

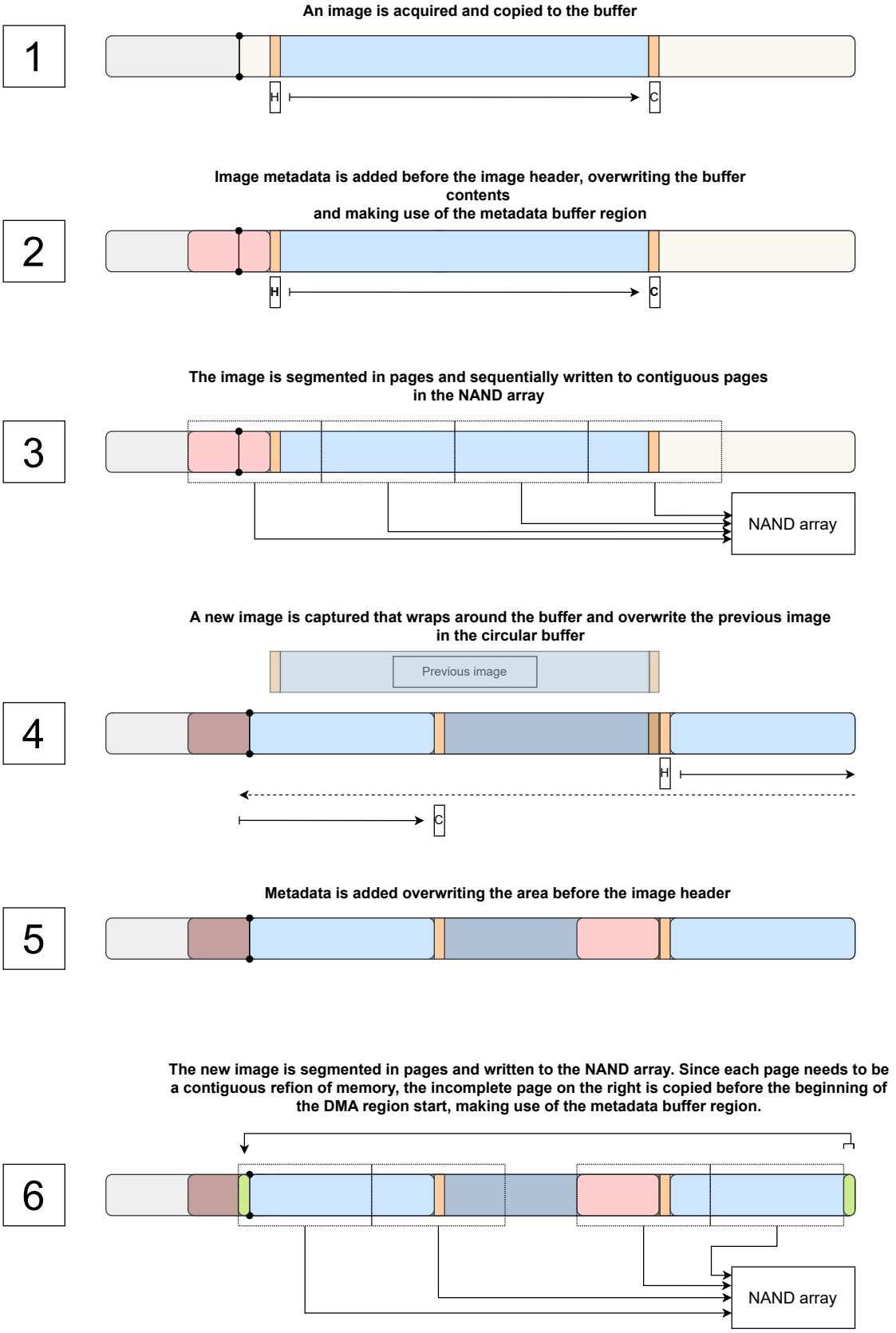


Figure 5.14: Summary of operation performed on the frame buffer.

Chapter 6

System Verification Campaign

A preliminary AIV plan has been devised for the imaging system lifecycle, in relation with the one of the spacecraft.

Due to the limited time for the integration and verification activities that have been conducted are finalized exclusively to achieve functional and performance verification of the integrated system. Interface verification has been performed only at the imaging system level, therefore interfaces with the S/C bus have not been verified. Thermal environmental verification has not been performed on the Development Model but it is advised, since it would allow to detect and identify potential issues related to component performance near the edge of the declared temperature operative range.

Model	BM		DM		PFM	
Level	PL	S/C	PL	S/C	PL	S/C
General						
Functional	**P**		**P**	X	X	X
Performances			**P**	X	X	X
Mission			X			X
Mechanical interfaces			**P**	X		X
Optical alignment					X	(X)
Mechanical						
Acoustic						
Random vibration					(X)	X
Sinusoidal vibration					X	X
Thermal						
Thermal cycling	X				X	X
Thermal vacuum					(X)	X
Electrical/RF						
EMC (conducted & radiated)			X	X		
ESD				X		
Electromagnetic auto-compatibility						X

P: test performed
X: test to be performed
(X): test may be performed, depending on the AIV programme

Figure 6.1: PL and S/C test matrix. Tests at subsystem level are not shown.

6.1 Capture of test images

The first test is a DCMI interface functional test meant to verify the functionality of the sensor and the image acquisition interface, after integration of the sensor module with Command Module and Daugherboard stack.

The test consists of:

- Verification of electrical and logic level of the DCMI and sensor configuration I2C interface via the use of an oscilloscope and a logic analyzer.
- Configuration of the sensor's registers to output on the DCMI interface a test pattern
- Capture of the test pattern via the DCMI peripheral and storage of the image in volatile memory.
- Download of the test image from the imaging system to a support workstation for visualization.

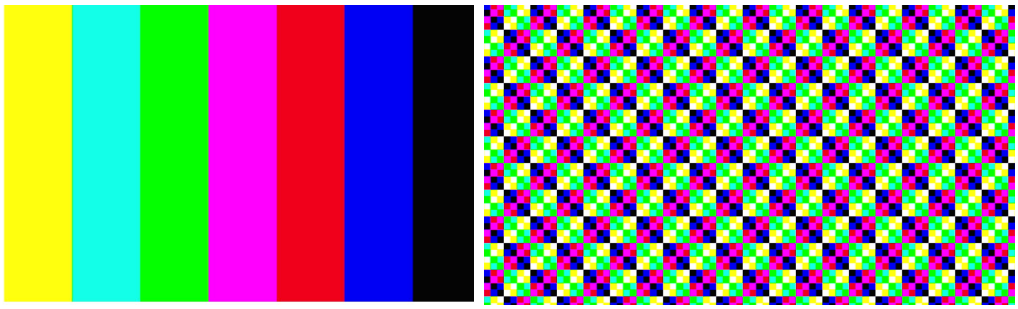


Figure 6.2: Test images generated by the OV5640 sensor and captured by the data handling module

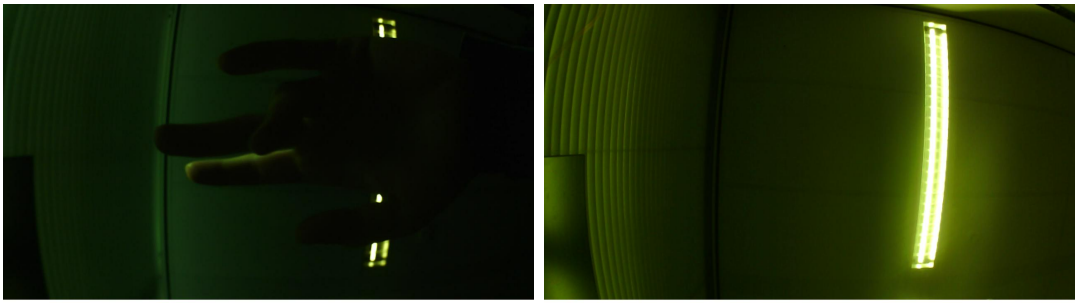


Figure 6.3: First captured frames, before configuration of the sensor automatic white balance correction

The result of the test has been successful, and the test images captured from the sensor are shown in figure 6.2.

The successive test has been performed to verify the ability to configure the sensor for nominal capture, and to calibrate the sensor to output usable images, obtaining sensor configuration that can be suitable for the successive operations.

The test consists in configuring the sensor mode of capture and additional parameters relative to exposure control and white balance. A set of frames are successively captured and downloaded to assess the image quality.

The first two amongst the initial frames captured in this test are displayed in figure 6.2. In these first frame it is visible how the automatic exposure control of the sensor is working, but not the white balance correction. After a successful white balance calibration, the successful capture of well-balanced images allowed to determine a set of configurations that can be used for future.

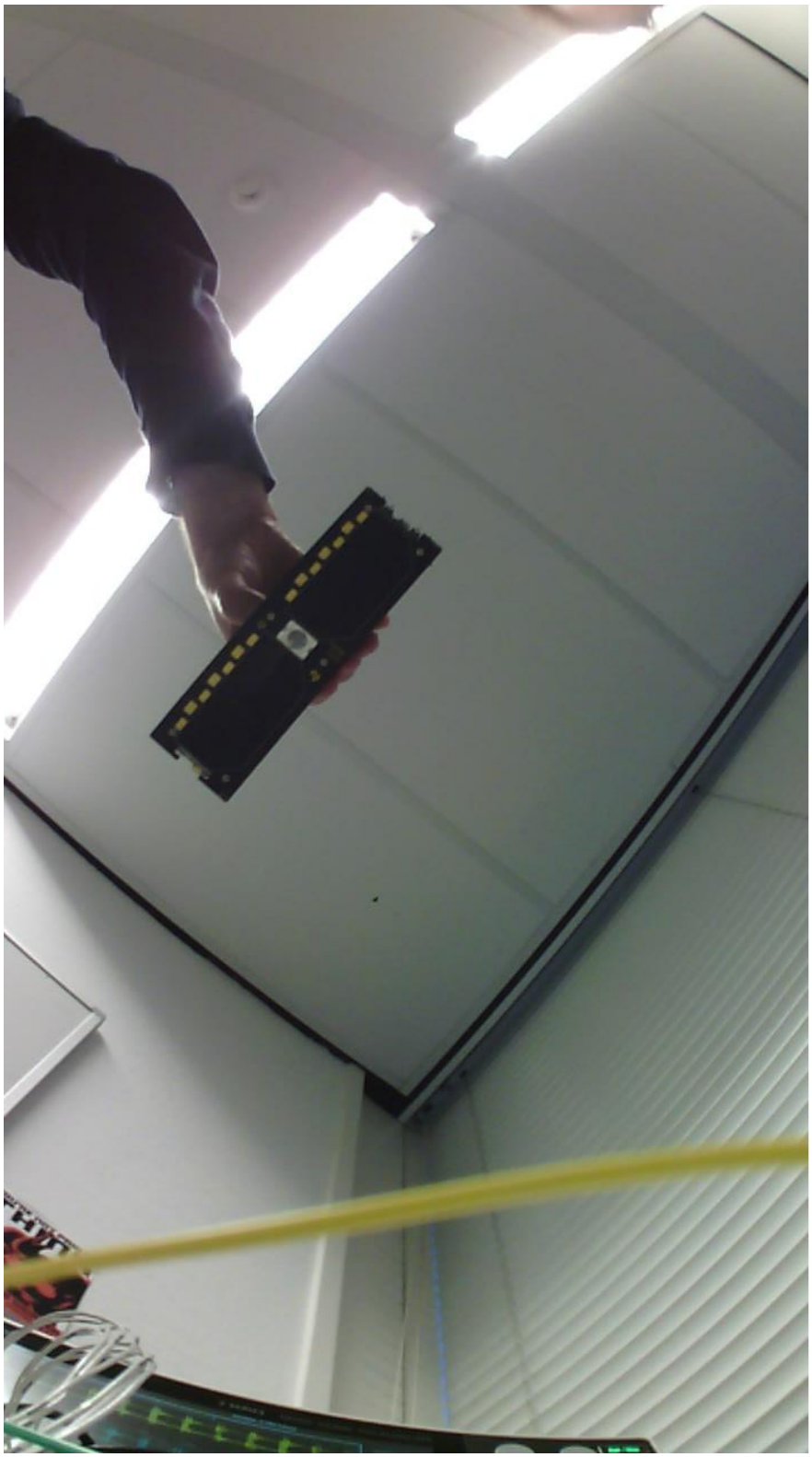
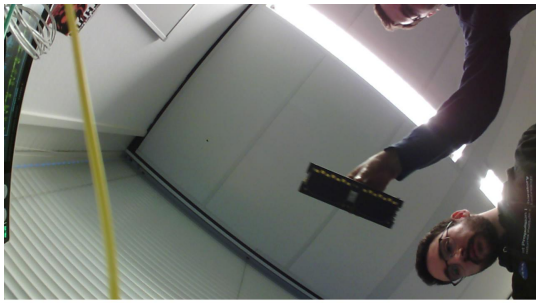


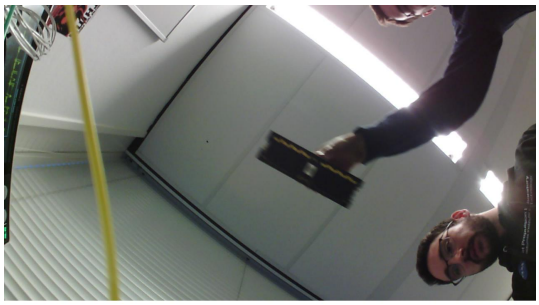
Figure 6.4: fig:Frame after white balance and exposure calibration (90 deg rotation)

(a) Frame 13

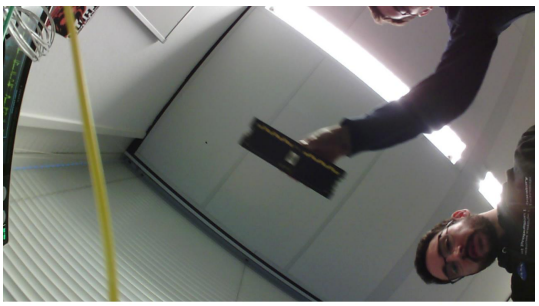


(b) Frame 14

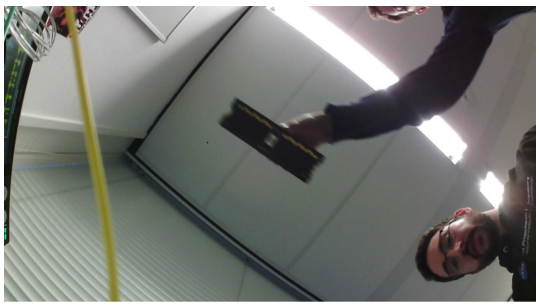
(c) Frame 15



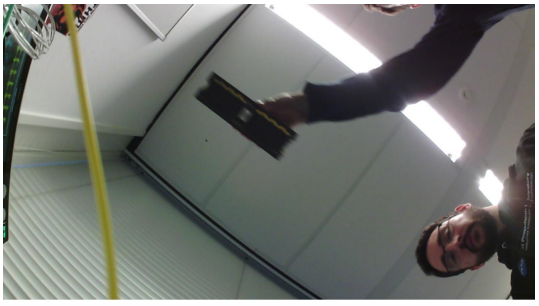
(d) Frame 16



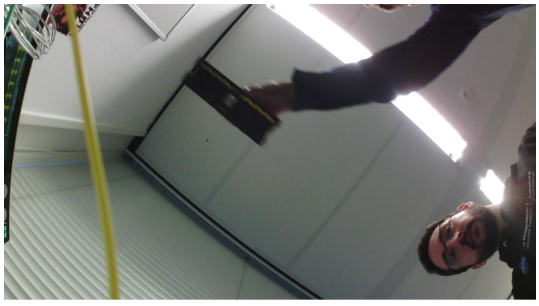
(e) Frame 17



(f) Frame 18



(g) Frame 19



(h) Frame 20

Table 6.1: Demonstration of Camera System

6.2 Evaluation of acquisition performance

The next test included the capture of a sequence of 1500 image frames, with the purpose of determining the performance of the data handling system, characterizing the time necessary to handle an image frame.

In the context of the system operation, the acquisition time is the time necessary to the data handling system to execute the data handling processes starting from the reception of the image in the frame buffer (that triggers the frame received ISR) up to the end of the transfer to the NAND memory array. Since initially the acquisition time proved to be superior to the capture period of the sensor, it has been necessary to configure the DCMI peripheral in blocking mode. This means that after the peripheral has been enabled to capture a new frame, it will begin to capture only at the beginning of the successive frame outputted by the sensor. The time between starting to wait for the next frame and the frame ready in the frame buffer is defined as waiting time. The total time necessary to obtain a frame, starting from enabling the peripheral, up to the end of the handling process, is defined as frame time. The frame time defines the necessary time to obtain a frame, and it defines the period of the acquisition process and, therefore, the acquisition frequency.

The acquisition time distribution for this test shows a mean acquisition time of $avg(T_a) = 0.1096$s, with a mode of $mode(T_a) = 0.0717$s.

It is possible to observe how a 90% confidence level centered on the mode of the distribution has bounds [0.06981, 0.07359] s , while the rest of the distribution shows an equal distribution of acquisition time amongst the frames, in the interval of [0.054167,0.225783] s.

The distribution of waiting time presents a mean of $avg(T_w) = 0.1096$s, with a mode of $mode(T_w) = 0.0717$. For this values as well the distribution is constant over the domain of [0.054167,0.225783] s, with the exception of the interval in proximity to the mode value.

The frame time distribution is obtained by the sum for each frame of the acquisition time and the relative frame time. It shows the total time necessary to capture the frame. The average frame time of the imaging system is $avg(T) = 0.1916$, while the mode is $mode(T) = 0.1577$s. Considering the average frame time, it results that the imaging system in this configuration presents an average acquisition frequency of 5.22

fps.

The PPLs of the sensor modules are configured to output 30fps at 720p when supplied with a 24Mhz input clock. The sensor clock is provided from an MCU pin, configured as a clock output. The clock is produced by the AHB clock of 120Mhz applying a divider of value 8, obtaining a 15Mhz output clock. Proportionally, the image acquisition frequency of the sensor is 18.75 fps, equal to a sensor frame period of 0.05333 s.

It is easy to observe how the sensor frame period is reflected on the frame time distribution of the imaging system: It is evident that for most of the frames, the frame time is equal three times the sensor frame period, therefore while a frame is handled by the system, the sensor produces three frames, resulting in the loss of two frames each frame captured.

In some cases the performance is inferior, and the imaging system is able to handle one frame in 4,5 or even 6 frames generated by the sensor.

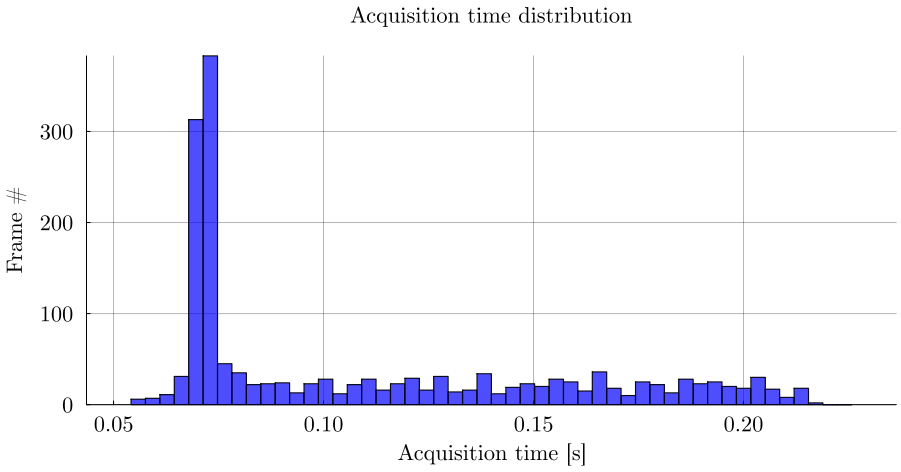
The distribution of the acquisition time shows that the 0.59 % of T_a is less than two times the sensor frame period (0.1066 s), while the rest of the time is wasted waiting for the next frame, an operation that always takes more than a frame period. This is because the waiting time includes the time to effectively fill the frame buffer: a fraction of a frame period is wasted waiting, while a complete frame period is necessary to complete the data transfer from the sensor to the frame buffer. In fact the waiting time distribution is never less than the sensor frame time.

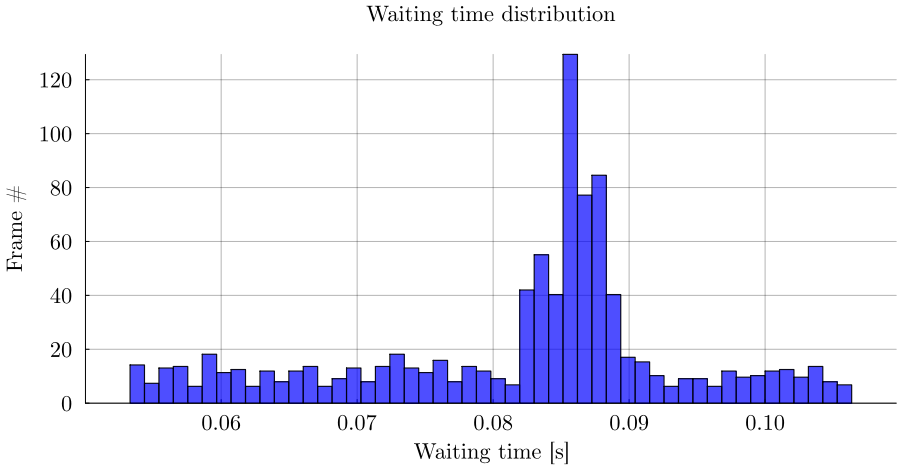
The system throughput can be estimated considering the size of the images that are being transferred. Figure 6.5 shows a distribution of the size of the JPEG images captured during the test. The sample size of the images captured is just 660 out of the 1500 images of the test. This is because in this test the utilization of the memory has been limited to a specific set of blocks and the other images have been overwritten during the capture. The average image size is 71.967 KB, with a minimum/maximum range of [54.435, 97.353] KB.

In terms of throughput, considering the average acquisition time and the average JPEG image size, the resulting average data handling throughput is 656.640 KB/s, that is way less than the throughput that the systems has been designed to handle. Considering the total frame time, the system throughput is necessarely worse and equal to 456.358 KB/s

The evaluation process allowed for the identification of two issues:

- Since the process is not configured as a pipeline, the best achievable performance is half of the sensor acquisition rate, due to the time that is necessary to fill the frame buffer being a sensor time period in addition to the successive acquisition time. Even if the acquisition time would be less than the sensor capture time, the peripheral would still need to wait for the beginning of the successive frame.
- The acquisition time is way more than what expected in the design phase. This results in a considerably inferior data handling throughput than expected.





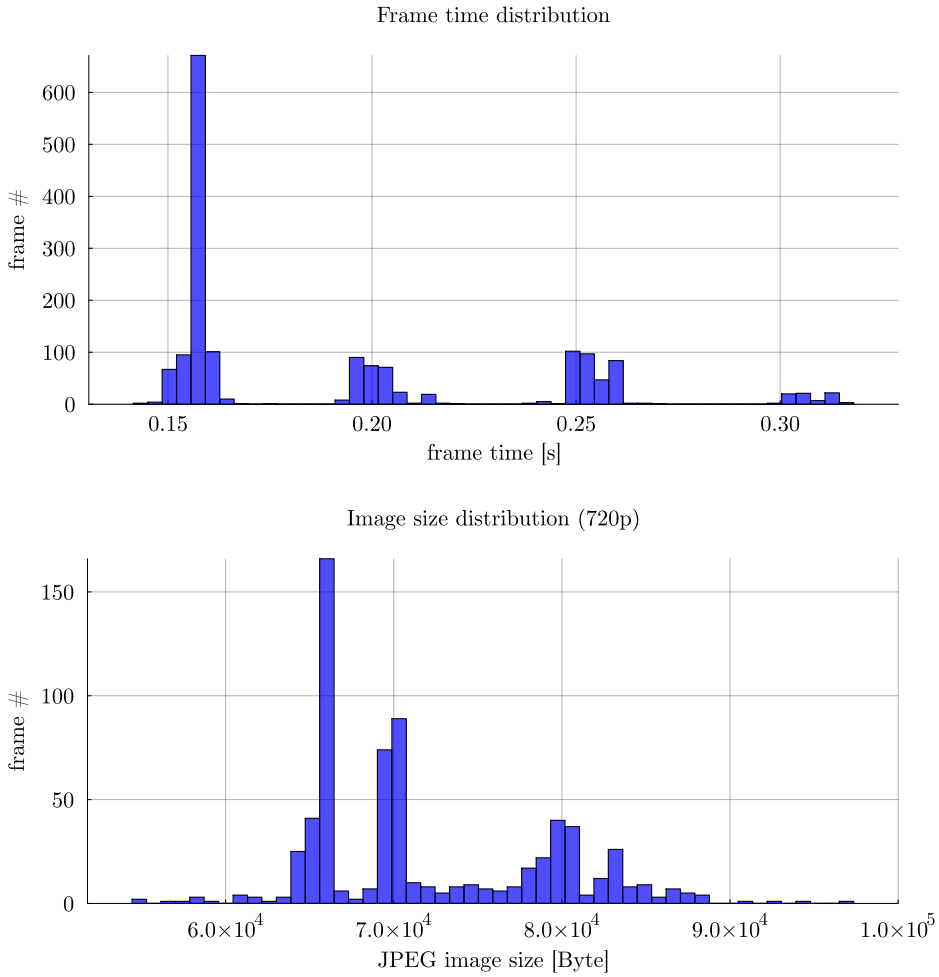


Figure 6.5: Distribution of the JPEG image size determined from the images acquired during the test.

6.3 Analysis of the acquisition process

The system is already designed to operate as a pipeline, therefore, to restore the process pipeline, it is necessary to apply modifications to allow the system to satisfy the timing requirements.

Moreover, the circular buffer shall have sufficient capacity to hold two frames concurrently, to avoid that a continuous capture would overwrite the buffer region from where the current frame is being handled. The acquisition time shall be less than the frame capture time.

This condition is respected for compressed 720p images, since the 262.14 KB circular

buffer can contain 2.7 images in the worst case.

To address the timings issue it has been necessary to investigate the cause of the performance reduction of the data handling process.

To analyze the performance of the whole process, it has been decided to measure the timing of each of the operations that compose the acquisition process. In order to do so, a set of probe wires has been soldered to the Compute Module, to have easy access to signals that could be measured with an oscilloscope. A simplified diagram of the pipeline process id displayed in the figure below.

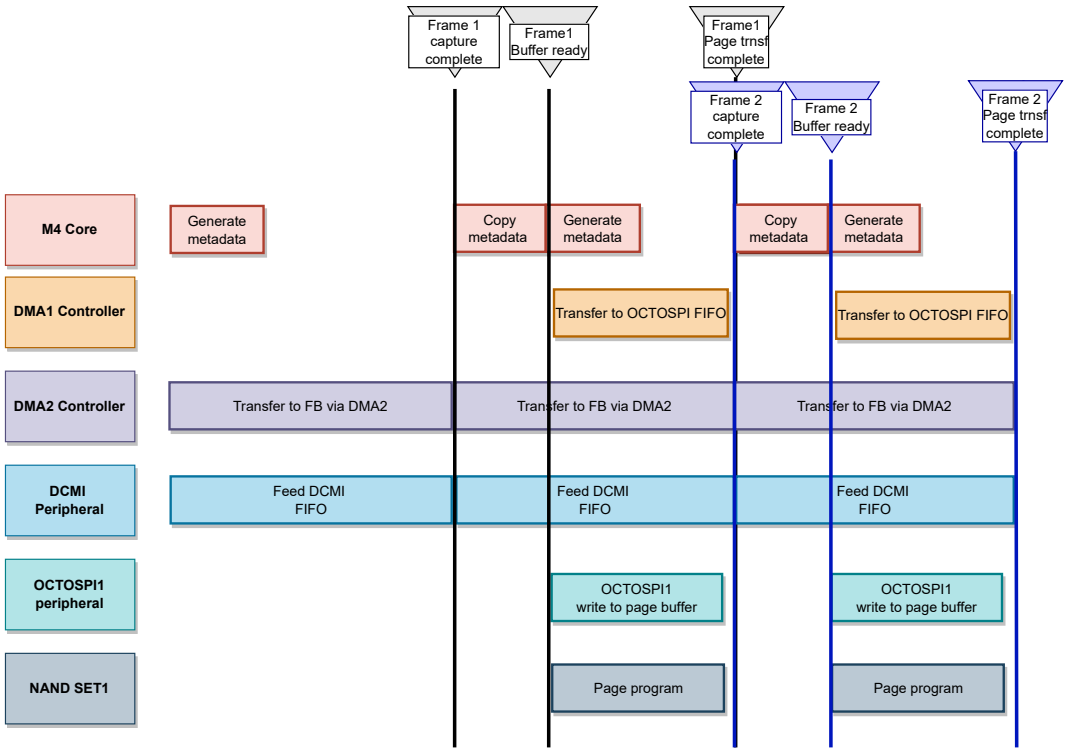


Figure 6.6: Data handling process.

Starting from the tail of the pipeline, the first process measured is the transfer and programming of the page-sized chunks from the frame buffer to the NAND modules. Saving an image in the SMS requires to transfer and program a number of pages that is dependent on the size of the image frame. From the characterization of the previous test, the average frame size is 71.967 KB, while the size of the page buffer is 8.182 KB in a dual-quadspi configuration and 4.096 KB for a single memory module. In this characterization test the memories are used in a dual-quadspi configuration, but the

driver has been configured to still consider a page size of 4.096 KB, to allow for the measurement of the single module throughput, resulting in an average of 18 pages per image. This estimation does not consider metadata, that is stored in the first page of the image frame.

Saving a page in the memory modules consists in the following steps:

- Transfer of the data from the frame buffer to the octospi-fifo.
- Transfer of the date via the octospi-interface to the NAND module page buffer.
- Programming of the data from the page buffer to the nand memory.

Figure 6.7 shows the transfer of an image frame in the octospi interfaces. The total duration of the transfer is 8 ms, and represents a transfer of 14 pages. This operation contains both the time necessary to transfer data to the memory modules and the programming time, because a page buffer needs to be programmed before it can be overwritten by the successive page. The throughput of the transfer results to be $4.096KB * 14/0.008s = 7.168MB/s$. The time necessary to execute the operation is minimal in respect to the acquisition time, therefore this step in the process is not the bottleneck.

Figure 6.8 shows a detail of the transfer and programming operation. The first segment is the transfer of data to the page buffer. Between the first and the second segment, a "page program" command is sent to the memory modules and the second segment is a continuous polling of the NAND modules to check when the programming operation is finished. The figure allows obtaining the timing of the data transfer and programming operation. Respectively, the data transfer operation lasted 300 us, while the programming operation required 200 us. The transfer throughput results to be $4.0096KB/0.0003s = 13.65$ MB/s, which is the 22.75% of the maximum data rate of the peripheral advertised by the manufacturer [23] (60MB/s), potentially suggesting that the octospi FIFO is being supplied with an insufficient throughput, but this aspect has not been further investigated. The programming performance of the NAND modules are instead exactly as for the component datasheet.

The next step has been to measure the time necessary to analyze the frame buffer. As described in the software design chapter, once a frame has been transferred via DMA to the frame buffer, it is necessary to search in the buffer for the JPEG header and closer, since the size of the image is variable.

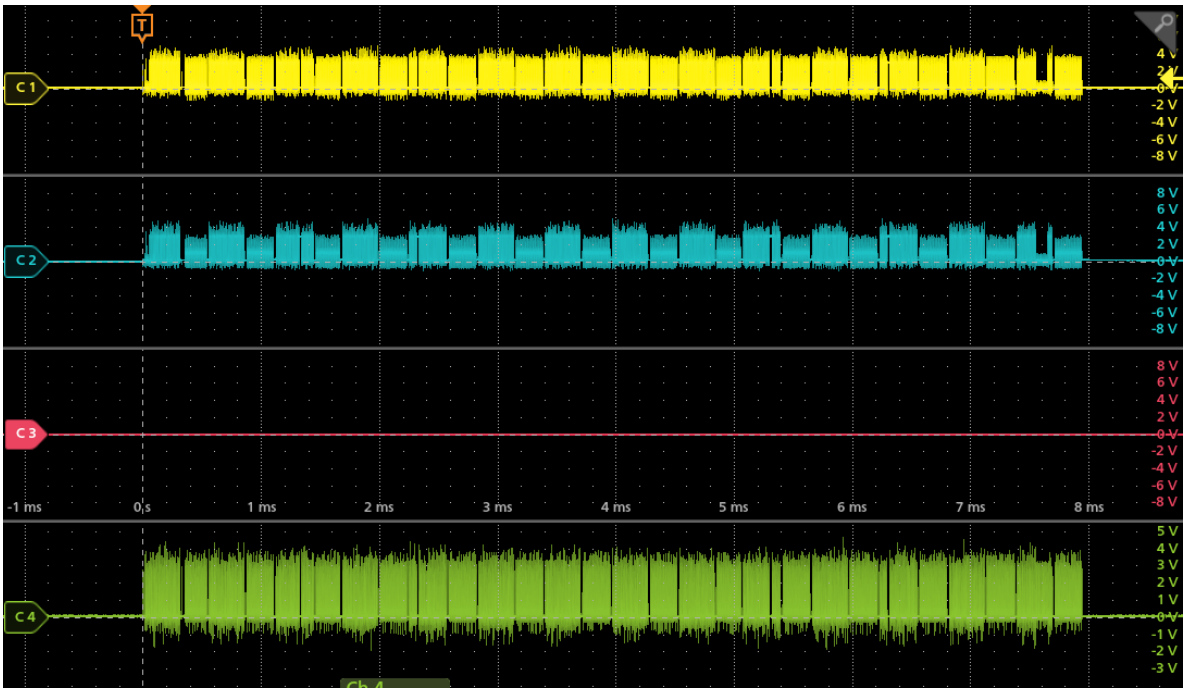


Figure 6.7: Capture of the transfer of an image frame over OCTOSPI. In the figure the yellow and blue signals are two octospi data lines, while green is the interface clock.

The time necessary to complete this operation is measured via the use of a GPIO signal, that is set high during the buffer analysis operation. Figure 6.9 shows (in two different timescales) the duration of the buffer analysis process (red) in comparison with the time necessary to complete the transfer to memory, indicated by the octospi clock (green).

It is visible how the buffer analysis takes most of the time necessary to handle a frame: in most cases is more than half of the total frame time. Even if the search process in the buffer begins from the end of the previous frame, it still requires several hundreds of milliseconds to complete.

Lastly, the time before the buffer analysis is the waiting time. It has been measured similarly to the previous case and is displayed in figure 6.10. Waiting time can be minimized once the frame acquisition time is below the sensor capture time.

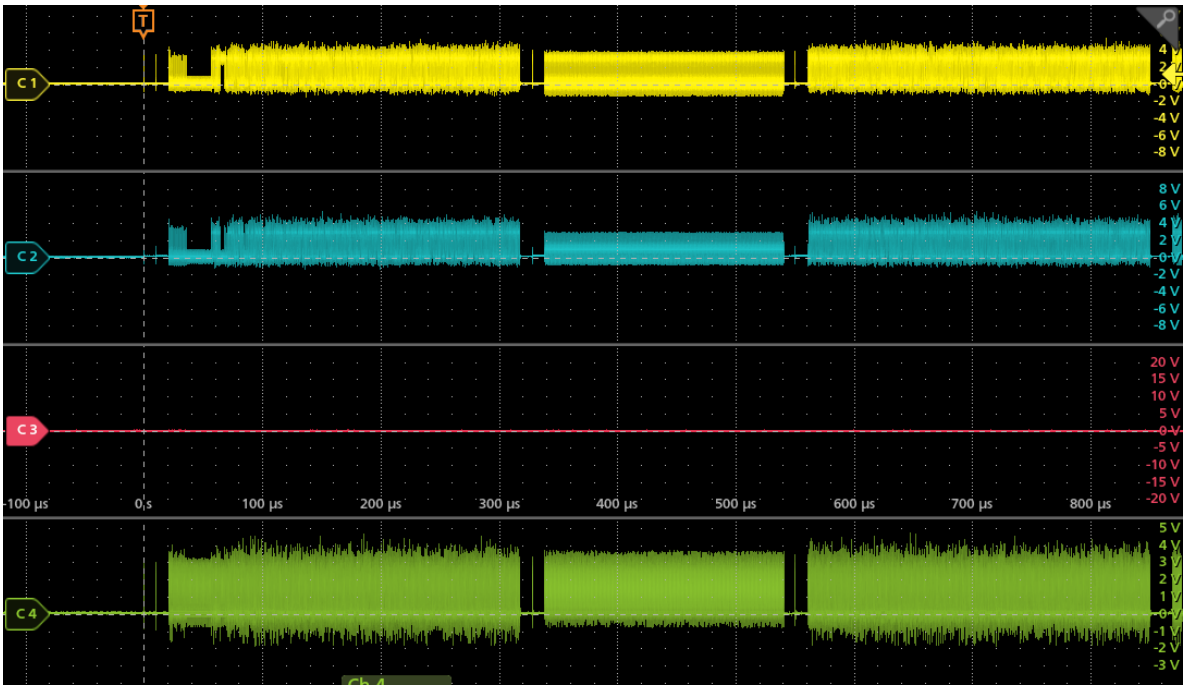


Figure 6.8: Transfer of a page to the page buffer of the memory modules and successive programming of the page buffer in the NAND memory.

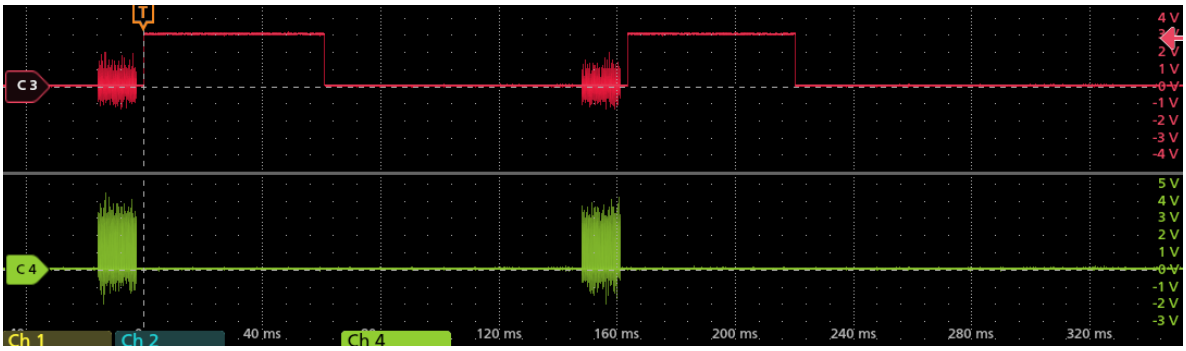


Figure 6.10: Measurement of the waiting time (red), in comparison with the transfer of the image in memory (green)

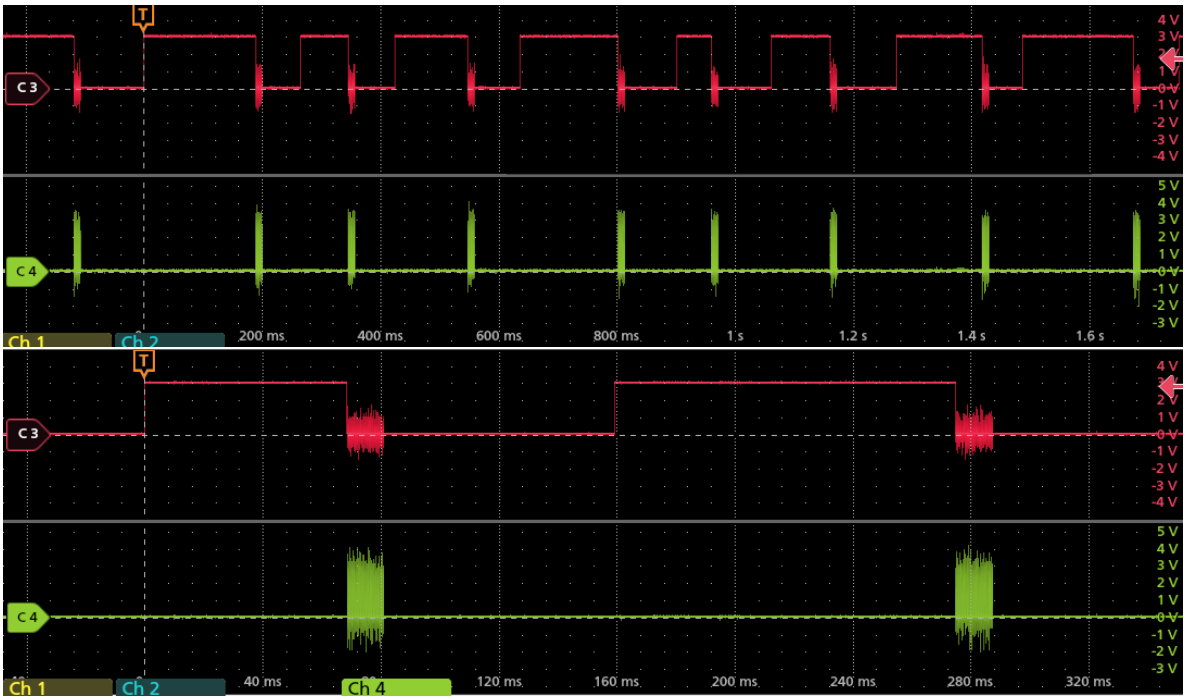


Figure 6.9: Measurement of the time necessary to find JPEG header and closer in the frame buffer. In red: buffer analysis process. In green: clock of the octospi interface

6.4 Addressing the identified issues

Considering the relative speed of transfer of the image frame to the SMS in respect to the time necessary to search for the header and closer of the imge.

Consequently, it has been decided to rather write the whole buffer directly to the SMS, rather than only the image frame. This would imply that the analysis of the buffer would be performed when the buffer itself is retrieved from the SMS rather than when the image is stored.

The consequence is that the memory capacity is used inefficiently, sice the buffer is being written rather than the image. Considering the writing throughput of a single module determined in the previous section, writing the complete buffer would require $262.14KB/7168KB/s = 0.03657s$. Considering the throughput of two modules that operate seamlessly in a dual-quad configuration, the resulting time would be $262.14KB/14336KB/s = 0.01828s$.

These changes have been implemented, and a new test has been performed, to characterize the system after the changes.

6.5 Improved acquisition system performance

Both of the successive tests consists in the capture of a sample of 2500 frames. The first modification that has been applied consists in writing directly the whole buffer in the SMS, for each image frame that is received.

The mode of the acquisition time distribution is 0.0402 seconds, similar to the expected value. The actual measured value is slightly greater than the expected value, since the previous image data size estimation did not consider the metadata.

The measurements show that frame acquisition has been performed in a time interval that is inferior to the frame period. Since the process still relies on the DCMI peripheral in single capture mode, the waiting time is necessarily more than a frame period, and this results in a system frame time that is essentially always equal to double the sensor capture time. This means that in this configuration, the data handling system can handle the sensor acquisition frequency of 18.75 fps, but due to the DCMI peripheral limitation, one in each two frames is lost, limiting the frame rate to 9.375 fps. In any case this test demonstrated that the timing requirements to achieve a pipeline acquisition process are satisfied.

Therefore, the peripheral has been successively configured to work in continuous mode and the SMS has been configured to work in dual-quadspi mode. The results of a successive test respect the predictions, the average acquisition time is $average(T_a) = 0.01927$ and the mode of the acquisition time distribution is $mode(T_a) = 0.01923$. This means that the time necessary to the data handling system to manage a frame is 36.13% of the frame time of the sensor, and therefore the frame rate of the system is limited by the sensor, rather than the data handling system at 18.75 fps. If the average acquisition time is considered, the maximum acquisition rate for a 720p JPEG image would be 50.69 fps. By comparing this acquisition rate with the average image size, it is possible to determine the theoretical maximum average throughput of the imaging system, that is $71.967KB * 50.69fps = 3.648MBps$.

If the imaging system throughput is applied to a 1080p raw image produced by the initial selected sensor (AR0234CS), it would result in a maximum average frame rate of $3.684Mbps/2.3MB = 1.60fps$.

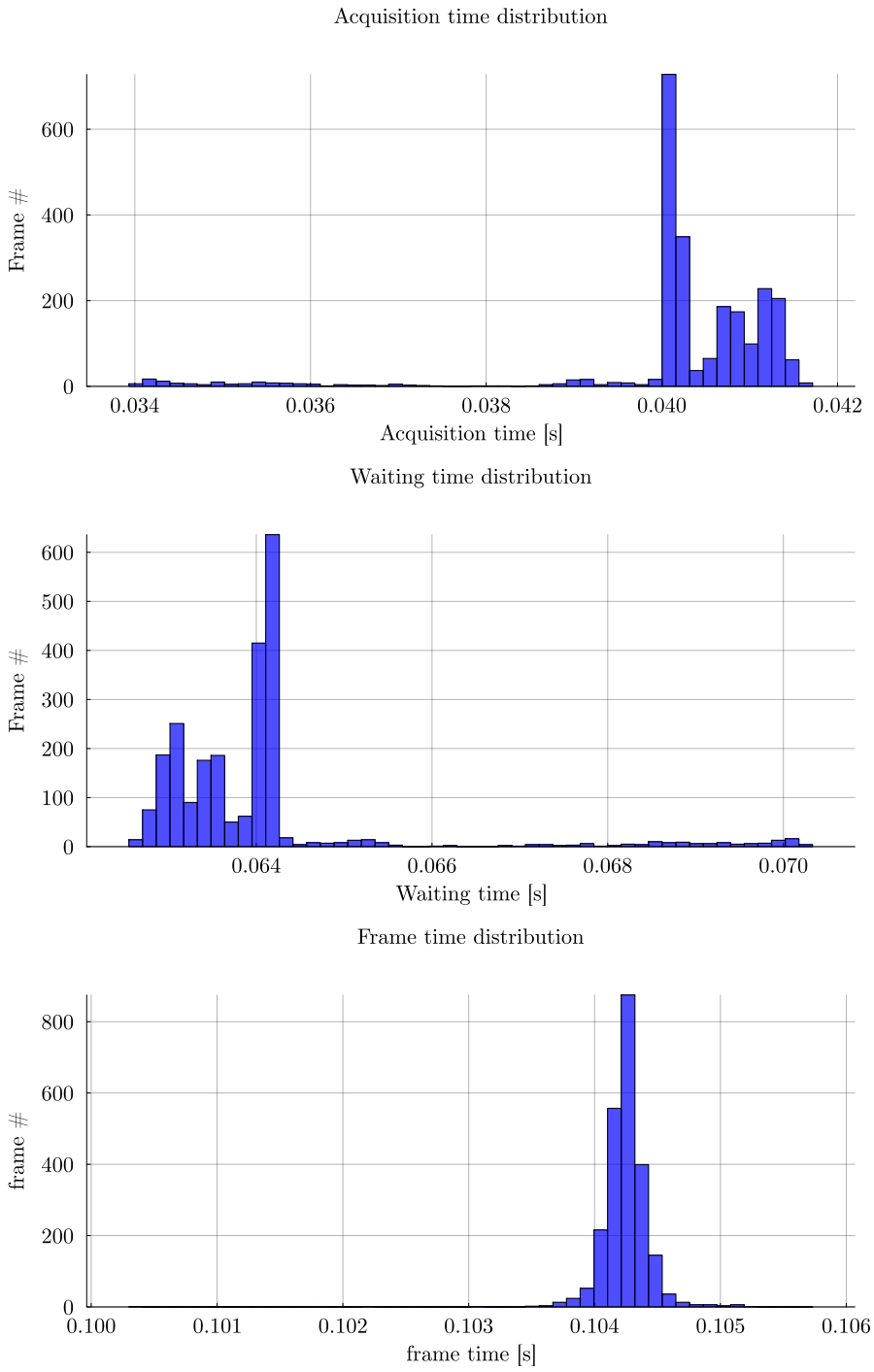


Figure 6.11: Timing distribution of the imaging system with the addition of frame buffer direct writing

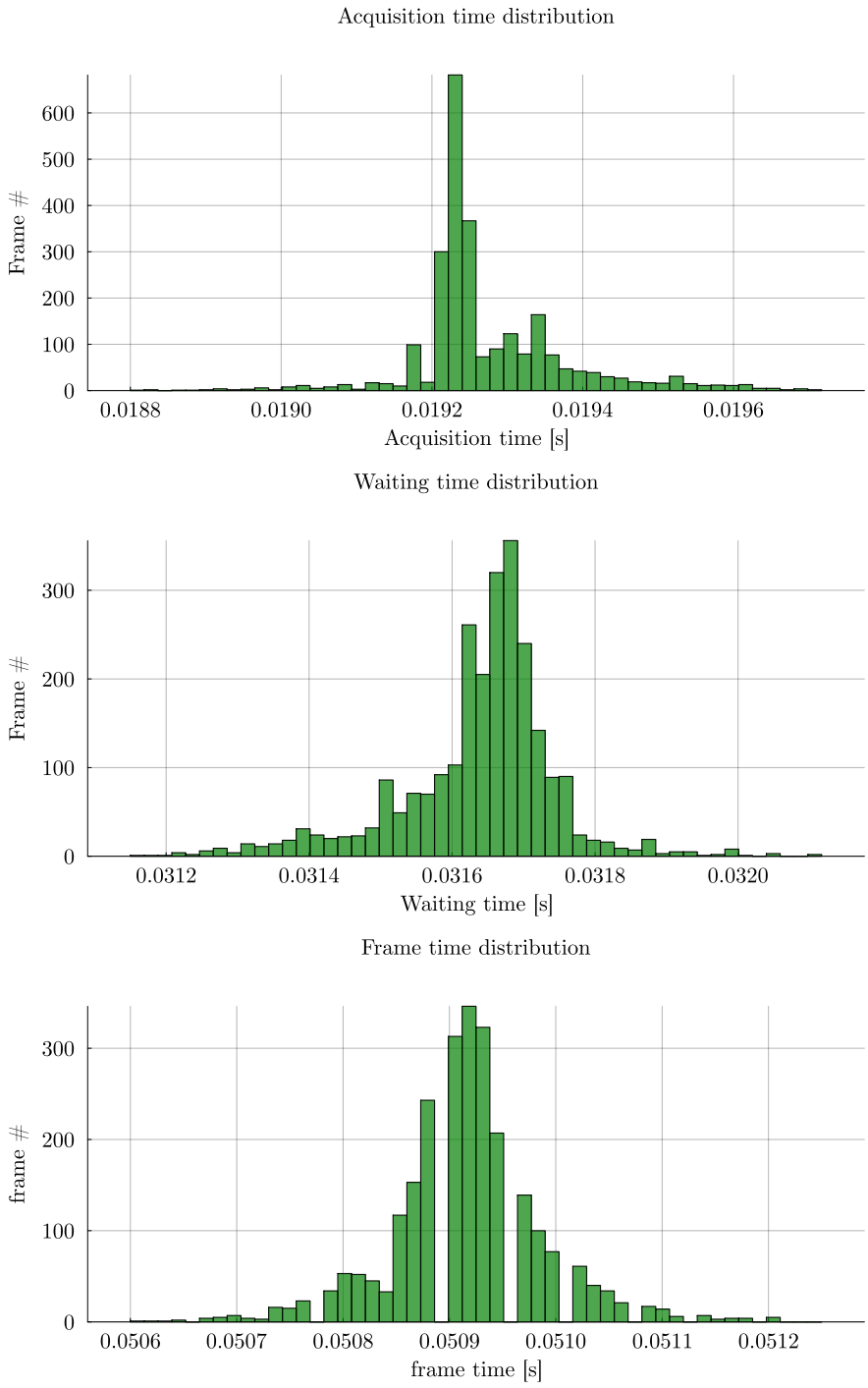


Figure 6.12: Timing distribution of the imaging system with direct writing of the frame buffer, dual-quadspi memory modules and DCMI peripheral configured in continuous capture mode.

Projection of the results on the imaging system with ping-pong SMS As stated before, the SMS is formed of four memory modules, configured in two dual-quadspi configurations. Up to this point only a single dual-quadspi has been considered, since the second set of memories was initially inteded to increase the total capacity rather than the throughput. That said, since the writing operation consists of alternating transfer/programming phases, it is possible to utilize the two set of memories in a ping-pong configuration. This would mean that while a set of memories is programming its page buffer, the MCU can transfer data to the page buffer of the other set. This feauture has not yet been implemented, but the implementation itself is straightforward, since the bare metal drivers that have been developed provide complete access to the hardware at its lowest level.

The measurements of the octospi interface displayed in figure 6.8 show that there is a difference between the time needed to fill the page buffer(300 us) and the time necessary to program it in the NAND memory (200 us). In order to implement a ping-pong configuration, ideally both phases would have a duration of 300 us, resulting in a total operation time of 600 us, resulting in a reduction of the throughput for a single dual-quad set of 16.7%.

The total throughput of the imaging system with a SMS in a ping pong configuration can be estimated as

$$THR_{PP} = THR_S(1-\alpha) + THR_S * \alpha * (1-0.167) * 2 \tag{6.1}$$

Where α is a participation factor in the total throughput of the data storage process and THR_S indicates the throughput of the imaging system for a single memory module set. Considering the process diagram in figure 6.6, it is the ratio between the [buffer ready, page transfer complete] time interval and the [capture complete, page transfer complete] time interval.

The measurement of the timings of the "copy metadata" process resulted in a negligible duration, therefore α is arbitrarily set to 0.95.

The resulting estimated maximum throughput for a ping-pong configuration would be 5.956 MB/s, resulting in a 1080p raw image acquisition rate of 2.5896 fps. If a ping-pong SMC configuration would be utilized to capture JPEG 720p images, the maximum average frame rate would be $5.956MB/s/71.967KB = 82.76fps$.

Sensor	Is verified	image format	SMS configuration	framerate [fps]	# captured frames	Total frame capacity
OV5640 (15Mhz)	YES	JPEG 720P	Single Dual-Quadspi	18.75	28	20000
OV5640	NO	JPEG 720P	Single Dual-Quadspi	50.69	76	20000
OV5640	NO	JPEG 720P	Double Dual-Quadspi	60	90	20000
AR0234CS	NO	RAW 720P	Single Dual-Quadspi	3.2	4	2170
AR0234CS	NO	RAW 720P	Double Dual-Quadspi	5.18	7	2170
AR0234CS	NO	RAW 1080P	Single Dual-Quadspi	1.6	2	1085
AR0234CS	NO	RAW 1080P	Double Dual-Quadspi	2.59	3	1085

Figure 6.13: Performance of the different sensor configuration applied to the mission scenario.

Determination of data product generation From the performance results of the final verification test and the expected total time in view obtained from the simulation of the separation phase, it is possible to predict the data products that would be produced during the mission, presented in the table above.

6.6 Comparison with COTS components

To assess the performance of the Delfi-TWIN imaging system, it has been compared to other imaging systems for CubeSat platforms that are compatible with the PocketQube format and that provided accessible datasheets or specifications. Respectively:

- Pikocamera from Orion Space,[21].
- KissCAM V2 from MVP aerospace [1].
- Leo2MP from Infinityavionics [13].
- CAM1U KIKAS CubeSat Camera from Crystalspace [6]

To analyze the performance of the different imaging sensors, a set of performance parameters has been identified and utilized to compare the different components.

Image format Integrated imaging sensors can output data in raw pixel value format, or in a compressed data format (usually JPEG 2000). Raw data is utilized for scientific applications and as input for any task that involve image processing, like navigation. Compressed images are typically utilized for outreach and engineering purposes, since

Figure 6.14: On the left the Leo2MP engineering camera from Infinityavionics [13],while on the right the KIKAS imaging system from Crystalspace (Images from each the manufacturer's website)

Figure 6.15: Image of the Development Model of the Delfi-TWIN imaging system during a performance test. The OV5640 test sensor is connected to the sensor control and data interface.

image fidelity can be sacrificed in order to reduce image size and, therefore, the time necessary for downlink. Typically processing algorithms make use of raw images, therefore compressed images needs to be decompressed to be used for such purpose.

The imaging system (IMS) for the Delfi-Twin can potentially capture either JPEG or RAW images, depending on the sensor that has been integrated with the Compute Module. The OV5640 sensor, utilized during the verification tests described in the previous sections, is able to output JPEG images, while every global shutter sensor amongst the ones considered in the initial sensor selection can produce RAW images, including the global shutter sensor selected (AR0234CS).

Out of the set of COTS imaging systems that the Delfi-TWIN payload is going to be compared against, the Leo2MP and Pikocamera IMS produce JPEG compressed images, while the KIKAS and the KissCAM V2 output RAW image data. Therefore, the two sensor classes are analyzed separately.

The IMS producing JPEG images are compared to a 1MP and a 2MP configuration of the Delfi-TWIN imaging system, both of which utilizing the OV5640. Out of these two configurations, performance parameters have been experimentally measured only for the 1MP configuration, while the performance of the 2MP configuration is derived from the data handling throughput measured for the 1MP configuration. Both of the IMS configurations utilize a single set NAND module memory configuration.

On the other end, two Delfi-TWIN IMS configurations are compared to other IMS producing RAW images. Both configurations utilize the AR0234CS sensor, the first makes use of a single set NAND modules, while the second utilizes the complete SMS in a ping-pong configuration.

Comparison of imaging systems that produce JPEG images The Delfi-TWIN IMS provides better performance for every performance parameter in respect to other COTS imaging systems featuring JPEG data output. The table considers the Delfi-Twin 1080p configuration as the comparison baseline. The closest commercial component in terms of performance is the Leo2MP, that is able to produce around half of the frame rate at almost double the power consumption during capture. The pikocamera presents an inferior power consumption (-33.33%) at the cost of not being able to capture an image sequence, but just single images. Moreover, the Delfi-TWIN IMS can provide a storage capacity up to 50 times the capacity of the Leo2MP imaging system.

Imaging system comparison (Absolute values)	Delfi-TWIN IMAGER 720p JPEG	Delfi-TWIN IMAGER 1080p JPEG	Leo2MP*	Pikocamera
Image format	JPEG	JPEG	JPEG	JPEG
Resolution [Mpx]	1	2	2	2
Frame rate [fps]	50	25	12	SINGLE
Acquisition data throughput** [MB/s]	3.648	3.648	2.4	N/A
On board storage [MB]	2000	2000	400	64
Number of storable images	20000	10000	200	32
Idle power consumption [mW]	108.9	108.9	N/A	176
Capture power consumption [mW]	511.5	511.5	1000	341

*: The Leo2MP reported framerate is specified to be framerate dependent. It is likely that the indicated framerate is not achieved at the 2MPx resolution

**: Acquisition data throughput approximated to resolution/10 x framerate for JPEG images

Figure 6.16: Table containing absolute parameters for IMS producing JPEG output

imaging system comparison (Delta %)	Delfi-TWIN IMAGER 720p JPEG	Delfi-TWIN IMAGER 1080p JPEG	Leo2MP*	Pikocamera
Image format	JPEG	JPEG	JPEG	JPEG
Resolution [Mpx]	-50.00%	0.00%	0.00%	0.00%
Frame rate [fps]	100.00%	0.00%	-52.00%	SINGLE
Acquisition data throughput** [MB/s]	0.00%	0.00%	-34.21%	N/A
On board storage [MB]	0.00%	0.00%	-80.00%	-96.80%
Number of storable images	100.00%	0.00%	-98.00%	-99.68%
Idle power consumption [mW]	0.00%	0.00%	#VALUE!	61.62%
Capture power consumption [mW]	0.00%	0.00%	95.50%	-33.33%

Figure 6.17: Table containing relative comparison of the Delfi-TWIN IMS with OV5640 sensor operating at 1080p with other IMS producing JPEG output

Therefore, the imaging system developed, when integrated with a JPEG sensor, can achieve more than double the performance of possible competitors at a fraction of power consumption.

Comparison of imaging systems that produce RAW images The framerate performance of the KIKAS system falls between that of the Delfi-TWIN imaging system in a single set memory configuration and the same Delfi-TWIN system in a ping-pong SMS configuration. However, the KIKAS system exhibits a slightly lower framerate than the latter, albeit with a reduced power consumption.

On the other hand, the KissCAMV2 system demonstrates inferior performance across all parameters when compared to both the KIKAS and Delfi-TWIN imaging systems.

imaging system comparison	Delfi-TWIN IMAGER 1080p RAW	Delfi-TWIN IMAGER PP-1080p RAW	KIKAS (Crystal space)	KissCAM V2
Image format	RAW	RAW	RAW	RAW
Resolution [Mpx]	2	2	2	1.2
Frame rate [fps]	1	2.59	2	SINGLE
Acquisition data throughput** [MB/s]	3.648	5.956	4	N/A
On board storage [MB]	2000	2000	750	4
Number of storable images	1000	1000	375	3
Idle power consumption [mW]	108.9	108.9	75	165
Capture power consumption [mW]	1023	1023	750	396

**: Acquisition data throughput equal to resolution x frame rate for RAW images

Figure 6.18: Table containing absolute parameters for IMS producing RAW output

imaging system comparison	Delfi-TWIN IMAGER 1080p RAW	Delfi-TWIN IMAGER PP-1080p RAW	KIKAS (Crystal space)	KissCAM V2
Image format	RAW	RAW	RAW	RAW
Resolution [Mpx]	0.00%	0.00%	0.00%	-40.00%
Frame rate [fps]	-61.39%	0.00%	-22.78%	SINGLE
Acquisition data throughput** [MB/s]	-38.75%	0.00%	-32.84%	N/A
On board storage [MB]	0.00%	0.00%	-62.50%	-99.80%
Number of storable images	0.00%	0.00%	-62.50%	-99.70%
Idle power consumption [mW]	0.00%	0.00%	-31.13%	51.52%
Capture power consumption [mW]	0.00%	0.00%	-26.69%	-61.29%

Figure 6.19: Table containing relative comparison of the Delfi-TWIN IMS with AR0234CS sensor and SMS in ping-pong configuration with other IMS producing RAW output

Therefore, the Delfi-TWIN imaging system provides competitive performance with other imaging systems of similar format currently available.

Chapter 7

Conclusion

This document contained a high level overview of the activities that have been conducted during the thesis project. Many details relative to implementation had to be left out to allow for an high level perspective on the scope of the project.

In conclusion, the conceptualization, design, development and verification activities encompassed the system from the lowest element level up to the system in its entirety. The resulting imaging system is an improvement over current COTS imaging systems compatible with the PocketQube format and has been specifically tailored to achieve the outreach mission goal of the Delfi-TWIN mission.

The short 5-month timeframe did not impede the production of value in a set of areas:

- A methodology to evaluate the results of proximity observation of the spacecraft has been developed. Supported by a set of computational tools, provides a tested framework to evaluate the capability of arbitrary imaging systems to image a target in the S/C proximity.
- A tradeoff study between components for the acquisition system has been carried out, resulting in the selection of an optimal combination of COTS optics and CMOS sensor.
- A complete baseline avionics system has been developed, including software and hardware products. The avionics system is going to be used as technological reference for future implementations of subsystems for the Delfi-TWIN S/C. Furthermore, the Compute Module consists in a payload data handling system that

can be repurposed as OBC for future PocketQube or CubeSat platforms, featuring an RTOS, high availability of I/O, extended temperature range and radiation tolerant components.

- A component symbols library has been developed to be utilized during the production of the Compute Module and Daughterboard. The library includes symbols, footprints and 3D models for the components utilized in the hardware and will be reused for the development of other subsystems for Delfi-TWIN spacecraft.
- The bare metal drivers developed during the thesis allows supporting high level utilization of the hardware peripheral and components from current and future application software. The drivers functionality has been verified, and automated tests have been developed for the Static Memory Subsystem (SMS) and the DCMI sensor interface. In particular, the Multi-SPI driver (Single, Dual, Quad, Octo-SPI) and the memory driver for the MT29F04 module, enables future systems to utilize high capacity, high throughput NAND SLC modules, featuring great temperature operative range and radiation tolerance.
- The RTEMS Board Support Package that has been developed allows the execution of the RTEMS RTOS on the previously unsupported STM32L4R9 MCU. The BSP includes a bootloader, the RTEMS software initialization procedure and integration with the low level drivers.
- The specification-based initialization system that has been developed allows for initialization and monitoring of the software components of the imaging system. In the context of the Delfi-TWIN project, it enables future definition of board variants for the different subsystems, enabling the generation of the lower software layers from the specifications of each of the subsystems. The monitoring features in the future can be expanded in a FDIR system for the software components.

The development of the RTEMS Board support package for the STM32L4R9 MCU is a particularly important achievement: It allows future students to familiarize and work with the RTEMS RTOS without expensive hardware, but rather utilizing inexpensive COTS development boards and the Development Model itself.

The modularity of the system allows for integration of a wide set of imaging sensors, including future automotive global shutter CMOS sensors, allowing for repurposing of the Compute Module for other applications or mission requirements.

Overall, the thesis activity successfully achieved all of the initially agreed-upon objectives. The Development Model represents a significant technical milestone in the advancement of the Delfi-TWIN mission, paving the way for future enhancements and improvements to the already market-competitive Imaging System.

Appendix A

N2 matrixes

This appendix contains the N2 matrix of the Imaging System, developed during the system definition phase.

			Imaging system											
			CDH						IAS	SMS	PDF			
			MCU	High speed clock source	Low speed reference clock source	RS-495 transceiver	FRAM memory module	Debug interface	CMOS sensor	NAND SLC memory modules	3v3 Regulator	2v8 Regulator	1v5 Regulator	Protection system
Imaging system	CDH	MCU		x	x	X	X	X	X	x				x
		High speed clock source	x											
		Reference clock source	x											
		RS-495 transceiver	x											
		FRAM memory module	x											
		Debug interface	x											
	IAS	CMOS sensor	x											
	SMS	NAND SLC memory modules	x											
	PDF	3v3 Regulator	x			x	x	x		x				x
		2v8 Regulator							x					x
		1v5 Regulator							x					x
		Protection system	x						x					

			Imaging system											
			CDH						IAS				SMS	
			MCU clock driver	MCU UART peripheral driver	RTEMS RTOS	RTEMS MCU BSP	Application software	Bootloader	CMOS sensor driver	DCMI MCU peripheral driver	I2C MCU peripheral driver	MCU DMA driver	OCTOSPI Peripheral driver	NAND module driver
Imaging system	CDH	MCU clock driver		X	X	X								
		MCU UART peripheral driver			X	X								
		RTEMS RTOS	X	X			X	X						
		RTEMS MCU BSP	X	X			X	X	X	X	X	X	X	X
		Application software			X	X								
		Bootloader			X	X								
	IAS	CMOS sensor driver	X			X								
		DCMI MCU peripheral driver				X								
		I2C MCU peripheral driver				X								
		MCU DMA driver				X								
	SMS	OCTOSPI Peripheral driver	X			X								
		NAND module driver				X								

Figure A.1: N2 matrixes for hardware and software products of the imaging system

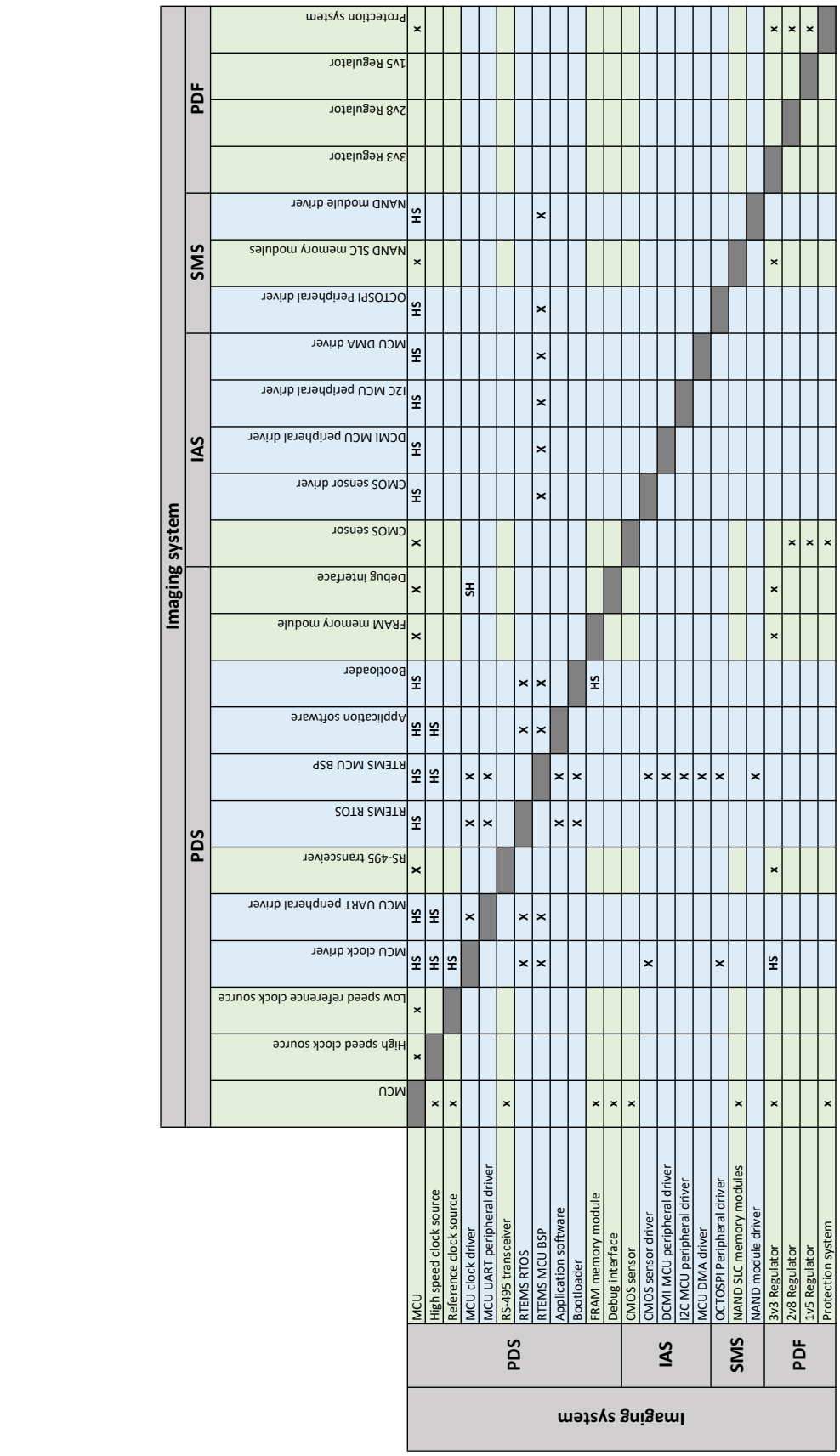


Figure A.2: N2 matrix for the complete set of the imaging system products

Appendix B

MCU Breakout

MCU GPIO BREAKOUT	
GPIO BANK A	
PA1.Mode=Full_Duplex_Master	PA1.Signal=SPI1_SCK
PA2.Mode=OCTOSPI1_Port1_NCS	PA2.Signal=OCTOSPIM_P1_NCS
PA3.Mode=O1_P1_CLK	PA3.Signal=OCTOSPIM_P1_CLK
PA4.Mode=Slave_10_bits_External_Synchro	PA4.Signal=DCMI_HSYNC
PA6.Mode=OCTOSPI1_IOL_Port1L	PA6.Signal=OCTOSPIM_P1_IO3
PA7.Mode=OCTOSPI1_IOL_Port1L	PA7.Signal=OCTOSPIM_P1_IO2
PA13\ (JTMS/SWDIO).Mode=Trace_Asynchronous_SW	PA13\ (JTMS/SWDIO).Signal=SYS_JTMS-SWDIO
PA14\ (JTCK/SWCLK).Mode=Trace_Asynchronous_SW	PA14\ (JTCK/SWCLK).Signal=SYS_JTCK-SWCLK
GPIO BANK B	
PB0.Mode=OCTOSPI1_IOL_Port1L	PB0.Signal=OCTOSPIM_P1_IO1
PB1.Mode=OCTOSPI1_IOL_Port1L	PB1.Signal=OCTOSPIM_P1_IO0
PB3\ (JTDO/TRACESWO).Mode=Trace_Asynchronous_SW	PB3\ (JTDO/TRACESWO).Signal=SYS_JTDO-SWO
PB7.Mode=Slave_10_bits_External_Synchro	PB7.Signal=DCMI_VSYNC
PB8.Mode=I2C	PB8.Signal=I2C1_SCL
PB9.Mode=I2C	PB9.Signal=I2C1_SDA
PB10.Mode=I2C	PB10.Signal=I2C2_SCL
PB11.Mode=I2C	PB11.Signal=I2C2_SDA
GPIO BANK C	
PC1.Mode=OCTOSPI1_IOH_Port1H	PC1.Signal=OCTOSPIM_P1_IO4
PC2.Mode=OCTOSPI1_IOH_Port1H	PC2.Signal=OCTOSPIM_P1_IO5
PC3.Mode=OCTOSPI1_IOH_Port1H	PC3.Signal=OCTOSPIM_P1_IO6
PC4.Mode=OCTOSPI1_IOH_Port1H	PC4.Signal=OCTOSPIM_P1_IO7
PC6.Mode=Slave_10_bits_External_Synchro	PC6.Signal=DCMI_D0
PC7.Mode=Slave_10_bits_External_Synchro	PC7.Signal=DCMI_D1
PC8.Mode=Slave_10_bits_External_Synchro	PC8.Signal=DCMI_D2
PC9.Mode=Slave_10_bits_External_Synchro	PC9.Signal=DCMI_D3
PC10.Mode=Slave_10_bits_External_Synchro	PC10.Signal=DCMI_D8
PC12.Mode=Slave_10_bits_External_Synchro	PC12.Signal=DCMI_D9
GPIO BANK D	
PD3.Mode=Slave_10_bits_External_Synchro	PD3.Signal=DCMI_D5
PD9.Mode=Slave_10_bits_External_Synchro	PD9.Signal=DCMI_PIXCLK
GPIO BANK E	
PE4.Mode=Slave_10_bits_External_Synchro	PE4.Signal=DCMI_D4
PE5.Mode=Slave_10_bits_External_Synchro	PE5.Signal=DCMI_D6
PE6.Mode=Slave_10_bits_External_Synchro	PE6.Signal=DCMI_D7
PE12.Mode=NSS_Signal_Hard_Output	PE12.Signal=SPI1_NSS
PE14.Mode=Full_Duplex_Master	PE14.Signal=SPI1_MISO
PE15.Mode=Full_Duplex_Master	PE15.Signal=SPI1_MOSI
GPIO BANK F	
PF0.Mode=OCTOSPI2_IOL_Port2L	PF0.Signal=OCTOSPIM_P2_IO0
PF1.Mode=OCTOSPI2_IOL_Port2L	PF1.Signal=OCTOSPIM_P2_IO1
PF2.Mode=OCTOSPI2_IOL_Port2L	PF2.Signal=OCTOSPIM_P2_IO2
PF3.Mode=OCTOSPI2_IOL_Port2L	PF3.Signal=OCTOSPIM_P2_IO3
PF4.Mode=O2_P2_CLK	PF4.Signal=OCTOSPIM_P2_CLK
GPIO BANK G	
PG0.Mode=OCTOSPI2_IOH_Port2H	PG0.Signal=OCTOSPIM_P2_IO4
PG1.Mode=OCTOSPI2_IOH_Port2H	PG1.Signal=OCTOSPIM_P2_IO5
PG9.Mode=OCTOSPI2_IOH_Port2H	PG9.Signal=OCTOSPIM_P2_IO6
PG10.Mode=OCTOSPI2_IOH_Port2H	PG10.Signal=OCTOSPIM_P2_IO7
PG12.Mode=OCTOSPI2_Port2_NCS	PG12.Signal=OCTOSPIM_P2_NCS

Figure B.1: Pin breakout for the STM32L4R9 MCU utilized in the Compute Module

Bibliography

[1] mvp aerospace. *KissCam V2 datasheet*. URL: https://satsearch.co/products/mvp-aerospace-kisscam-v2.

[2] Gauss S.r.l AlbaOrbital TU-Delft. *The PocketQube Standard,Issue 1*. PDF. URL: https://static1.squarespace.com/static/53d7dcdce4b07a1cdbbc08a4/t/5b34c395352f5303fcec6f45/1530184648111/PocketQube+Standard+issue+1+-+Published.pdf.

[3] Marta Bagatin Alessandro Paccagnella Simone Gerardin. *Studies of radiation effects in new generations of non-volatile memories*. PDF. URL: https://escies.org/download/webDocumentFile?id=60380.

[4] Jeff Bezanson et al. "Julia: A fresh approach to numerical computing". In: *SIAM review* 59.1 (2017), pp. 65–98. URL: https://doi.org/10.1137/141000671.

[5] Ricardo Cabello. *Three.js*. https://github.com/mrdoob/three.js. Accessed: March 19, 2024. 2009.

[6] crystalspace. *c1u cubesat camera*. URL: https://crystalspace.eu/products/crystalspace-c1u-cubesat-camera/.

[7] *Development of a versatile on board computer for small satellites*. https://webthesis.biblio.polito.it/27928/1/tesi.pdf. Accessed: March 23, 2024.

[8] Embedded Brains. *Homepage*. n.d. URL: https://embedded-brains.de/en/homepage/.

[9] Nicholas James Gregory Hollows. *Object Space Resolution*. https://www.edmundoptics.com/knowledge-center/application-notes/imaging/object-space-resolution/. Accessed: March 23, 2024.

[10] William M. Hall. *An introduction to shuttle/LDEF retrieval operations: the R-BAR approach option*. PDF. URL: https://ntrs.nasa.gov/api/citations/19780010162/downloads/19780010162.pdf.

[11] Sebastian Huber. *RTEMS linker set based initialization*. URL: https://devel.rtems.org/ticket/2408.

[12] Sebastian Huber. *RTEMS STM32H7 BSP*. URL: https://devel.rtems.org/ticket/3910#comment:16.

[13] infinityavionics. *leo2mp CubeSat camera*. URL: https://infinityavionics.com/products/leo2mp/.

[14] J. Bouwmeesterm E Gill M. Ş. Uludag S. Speretta and T.PerezSoriano. *A NEW ELECTRICAL POWER SYSTEM ARCHITECTURE FOR DELFI-PQ*. PDF. URL: https://www.gaussteam.com/wordpress/wp-content/uploads/2018/02/IAA-AAS-CU-17-07-03-M.%C5%9E.Uludag.pdf.

[15] OAR. *About RTEMS*. URL: https://www.rtems.com/aboutrtems.

[16] Alba Orbital. *Alba Orbital launch services*. 2024. URL: http://www.albaorbital.com/launch (visited on 03/19/2024).

[17] Christopher Rackauckas and Qing Nie. "DifferentialEquations.jl–a performant and feature-rich ecosystem for solving differential equations in Julia". In: *Journal of Open Research Software* 5.1 (2017).

[18] Sorin Radu et al. "Delfi-PQ: The first pocketqube of Delft University of Technology". In: *Proceedings of 69th International Astronautical Congress*. Article IAC-18-B4.6B. International Astronautical Federation, IAF. Bremen, Germany, 2018.

[19] Robin Rdeits. *MeshCat.jl*. https://github.com/rdeits/MeshCat.jl. Accessed: March 19, 2024. 2024.

[20] A.H. de Ruiter, C. Damaren, and J.R. Forbes. *Spacecraft Dynamics and Control: An Introduction*. Wiley, 2012. ISBN: 9781118403327. URL: https://books.google.it/books?id=mGSYHJIlDxEC.

[21] Orion Space. *pikocamera shop webpage*. URL: https://www.tindie.com/products/orionspace/pikocamera-camera-for-pocketqube/.

[22] G. Steinebach. "Construction of Rosenbrock–Wanner method Rodas5P and numerical benchmarks within the Julia Differential Equations package". In: *Bit Numer Math* 63.27 (2023). URL: https://doi.org/10.1007/s10543-023-00967-x.

[23] STMicroelectronics. *AN5050 Octal-SPI interface (OctoSPI) on STM32L4+ Series*. Accessed: March 23, 2024. 2018. URL: file:///C:/Users/v/Downloads/STM32L4%20Octo-SPI%20Application%20note.pdf.

[24] STMicroelectronics. *STM32L4 Series Advanced ARM-Based 32-Bit MCUs*. Accessed: March 23, 2024. 2021. URL: file:///C:/Users/v/Downloads/rm0432-

stm32l4-series-advanced-armbased-32bit-mcus-stmicroelectronics.pdf.

[25] STMicroelectronics. *STM32L47xxx, STM32L48xxx, STM32L49xxx and STM32L4Axxx Advanced ARM-Based 32-Bit MCUs*. Accessed: March 23, 2024. 2024. URL: https://www.st.com/resource/en/reference_manual/rm0351-stm32l47xxx-stm32l48xxx-stm32l49xxx-and-stm32l4axxx-advanced-armbased-32bit-mcus-stmicroelectronics.pdf.

[26] RTEMS Documentation Team. *Initialization Code - BSP How-To*. https://docs.rtems.org/branches/master/bsp-howto/initilization_code.html. Accessed: 2024-02-14. 2024.

[27] U. Teubner and H.J. Brückner. *Optical Imaging and Photography: Introduction to Science and Technology of Optics, Sensors and Systems*. De Gruyter STEM. De Gruyter, 2019. ISBN: 9783110472943. URL: https://books.google.it/books?id=e3WcDwAAQBAJ.

[28] Robert J. Twiggs. *Making it Small*. PDF. Archived from the original (PDF) on 3 March 2016. Retrieved 7 September 2013. URL: https://web.archive.org/web/20160303185449/http://mstl.atl.calpoly.edu/~bklofas/Presentations/DevelopersWorkshop2009/7_CubeSat_Alt/1_Twiggs-PocketQub.pdf.

[29] TEC-QEC Véronique Ferlet-Cavrois ESA / ESTEC. *Electronic radiation hardening – Radiation Hardness Assurance and Technology Demonstration Activities*. PDF. URL: https://sci.esa.int/documents/34530/36042/1567259051875-JUICE_instrument_Workshop_9_11_2011_Ferlet-Cavrois.pdf.