

MOSAR

Deliverable Reference	: D3.6
Title	: Detailed Design Document
Confidentiality Level	: PU
Lead Partner	: SpaceApps
Abstract	: This report is the output of the detailed design activities of the project MOSAR. The document provides the overall system architecture, the description of the ground and space system operations in the context of the project demonstration and the detailed design of the different components and sub-systems.
EC Grant N°	: 821996
Project Officer EC	: Christos Ampatzis (REA)



MOSAR is co-funded by the Horizon 2020
Framework Programme of the European Union



Detailed Design Document

DOCUMENT CHANGE RECORD				
Version	Date	Author	Changed Sections / Pages	Reason for Change / RID No
1.0.0	12/06/2020	Consortium	All	First release for CDR milestone review
1.1.0	19/08/2020	Consortium	2.1 2.2 3.1.2.5 4.2.6 5.2.2 5.4.2.2	Added note on reconfiguration operation control as feedback to RID-OG09-146 Corrected bookmark, as feedback to RID-OG09-145 Add information about timing of network discovery, as feedback to RID-OG09-147 Clarify requirement of initial WM position conditions, as feedback to RID-OG09-121 Elaborate on SpW communication latency with WM, as feedback to RID-OG09-122 Clarify that mass budget includes internal harness (also applied to the other modules), as feedback to RID-OG09-124
1.2.0	24/11/2020	SpaceApps SITAEL	5.4 5.4.1.2 5.5.1	Update SM Mass budget tables, following detailed design updates Update picture of SM with last version of the detailed design Update spacecraft buses pictures with last version of the detailed design



Contents

1	Introduction	10
1.1	Purpose and Scope	10
1.2	Document Structure	10
1.3	Applicable Documents.....	10
1.4	Reference Documents.....	10
1.5	Acronyms.....	12
2	MOSAR System Overview	14
2.1	MOSAR Mission Overview	14
2.2	From Mission Overview to MOSAR Demonstration	16
2.3	Demonstrator Components and Architecture	20
2.3.1	Ground Segment	20
2.3.2	Space Segment	21
2.3.3	Generic Data and Power Architecture	22
2.4	MOSAR Product Tree	26
3	Spacecraft Reconfiguration	28
3.1	Hardware Reconfiguration.....	28
3.1.1	Mechanical Reconfiguration	28
3.1.2	Data Reconfiguration.....	28
3.1.3	Power Reconfiguration	33
3.2	Software Reconfiguration	33
3.2.1	General software reconfiguration strategy	33
3.2.2	Introducing timing analysis of software reconfigurations.....	35
3.2.3	Overview of the schedulability analysis method.....	35
3.2.4	Assessing schedulability with SCM	36
3.2.5	Integration into the TASTE toolchain.....	40
3.2.6	References	41
4	System Operations.....	42
4.1	Ground Segment Operations	42
4.2	Space Segment Operations	45
4.2.1	CLT/SVC Reconfiguration	45
4.2.2	CLT Nominal Operations	46
4.2.3	Elementary Actions.....	46
4.2.4	Routines Sequences	49
4.2.5	System Start-Up – Initial Configuration	50
4.2.6	WM Re-Localization	51



Detailed Design Document

4.2.7	SM Re-Localization	53
5	Components Detailed Design	56
5.1	Design and Simulation Tool	56
5.2	Autonomy Agent and Planning.....	58
5.2.1	Reactors Architecture	59
5.2.2	Robotic Arm.....	61
5.3	Telemetry and Telecommand Service	65
5.4	Spacecraft Modules and Payloads.....	68
5.4.1	Spacecraft Modules Detailed Design	68
5.4.2	SM1-DMS	71
5.4.3	SM2-PWS.....	74
5.4.4	SM3-BAT	80
5.4.5	SM4-THS.....	84
5.4.6	SM5-OSP1.....	87
5.4.7	SM6-OSP2.....	90
5.5	SVC/CLT Buses	92
5.5.1	Mechanical Design	92
5.5.2	SVC Data and Power Architecture	93
5.5.3	CLT Data and Power Architecture.....	94
5.5.4	SVC/CLT Power Budget.....	96
5.6	Walking Manipulator	97
5.6.1	Description.....	97
5.6.2	EtherCAT route to space	98
5.7	HOTDOCK Standard Interface.....	100
5.7.1	General Overview.....	100
5.7.2	HOTDOCK Declinations	101
5.7.3	HOTDOCK Implementation in MOSAR	102
5.7.4	HOTDOCK Detailed Design	103
5.7.5	HOTDOCK Software Interface	103
5.8	Centralized Power Distribution Unit	106
5.8.1	Description.....	106
5.8.2	cPDU Software Interface	107
5.9	R-ICU and FMC Board	109
5.9.1	SpaceWire and CAN Interface FMC Card.....	109
5.9.2	R-ICU Hardware Design	109
5.9.3	R-ICU Software Architecture	111



Detailed Design Document

5.9.4	SpaceWire PnP Support.....	113
5.10	Visual Processing System.....	115
5.10.1	Main Processing Pipeline	115
5.10.2	Processing Line Variants.....	116
5.10.3	Software Architecture	119
5.10.4	Visual Processing System Setup	119
6	Annexes	123
6.1	MOSAR Setup Position References.....	123



List of Figures

Figure 2-1: MOSAR roadmap to space application.....	14
Figure 2-2: Spacecraft design MOSAR-like mission (client and servicer)	15
Figure 2-3: Life-cycle of the MOSAR mission concept.....	15
Figure 2-4: Spacecraft with different mission configurations.....	16
Figure 2-5: On-orbit reconfiguration with substitution and addition of modules	16
Figure 2-6: Representation of MOSAR mission	17
Figure 2-7 – Different use cases of MOSAR mission concept.	18
Figure 2-8: MOSAR Demonstrator Overview	20
Figure 2-9: Main components of the MOSAR space segment.....	22
Figure 2-10: MOSAR generic data and power architecture	25
Figure 2-11: MOSAR demonstrator product tree	26
Figure 3-1 MOSAR SpW Network Reconfiguration Flowchart.....	30
Figure 3-2 MOSAR Network Discovery (All SMs Powered On)	32
Figure 3-3: Software reconfiguration manager.....	34
Figure 3-4: Considered MOSAR architecture for software reconfiguration.....	35
Figure 3-5: Worst-case reconfiguration delay computation process	36
Figure 3-6: Example of applying SCM on a TASTE model.....	39
Figure 3-7: Integration of SCM into the TASTE toolchain	40
Figure 4-1: Design, simulation and planning stages	44
Figure 4-2: Re-configuration operation architecture.....	45
Figure 4-3: WM transfer Motion Sequence	47
Figure 4-4: WM Approach Motion Sequence	48
Figure 4-5: SI Connection Sequence	49
Figure 4-6: Routine 1 – WM re-localization between initial SI (SI-W) and Target SI (SI-X).....	52
Figure 4-7: Routine 2 – SM re-localization	53
Figure 5-1: Visualization of the fixed FES in SimVIS	56
Figure 5-2: Overview of the ERGO Agent tailoring for MOSAR.....	58
Figure 5-3: PGCI deployment.....	59
Figure 5-4: Robotic arm main components	62
Figure 5-5: RARM Manager relevant methods.....	63
Figure 5-6: Driver specialization	64
Figure 5-7: Collision Checker relevant methods	64
Figure 5-8: Ground and PUS Services instances	65



Detailed Design Document

Figure 5-9: PUS Console (left: TM/TC log, right: housekeeping parameters)	66
Figure 5-10: 1U SM - 2U SM	68
Figure 5-11: 4U SM - 8U SM	69
Figure 5-12: Connection beam	69
Figure 5-13: Sandwich panels internal Frame.....	70
Figure 5-14: Ground demonstrator structure.....	71
Figure 5-15: OBC-C NUC board	72
Figure 5-16: SM1-DMS data architecture.....	72
Figure 5-17: SM1-DMS power architecture.....	73
Figure 5-18: SM2-PWS data architecture	74
Figure 5-19: SM2-PWS power architecture.....	75
Figure 5-20: CAD model of Thermal IF	75
Figure 5-21: Thermal subsystem located inside SM2-PWS and SM4-THS.....	76
Figure 5-22: SM3-BAT data architecture.....	80
Figure 5-23: SM3-BAT power architecture.....	81
Figure 5-24: Battery power architecture	82
Figure 5-25: SM4-THS data architecture	85
Figure 5-26: SM4-THS power architecture.....	85
Figure 5-27: ZED stereo camera.....	87
Figure 5-28: SM5-OSP1 data architecture	88
Figure 5-29: SM5-OSP1 power architecture	88
Figure 5-30: SM6-OSP2 data architecture	90
Figure 5-31: SM6-OSP2 power architecture	90
Figure 5-32 SVC structure view	93
Figure 5-33: HOTDOCK interface plates.....	93
Figure 5-34: SVC/CLT data architecture (including fixed SM1 and SM2).....	95
Figure 5-35: SVC/CLT power architecture (including fixed SM1 and SM2).....	95
Figure 5-36: MOSAR walking manipulator	97
Figure 5-37: Walking Manipulator avionics / data architecture	98
Figure 5-38: HOTDOCK Standard Interface (in Active and Passive configurations)	100
Figure 5-39: cPDU power architecture	106
Figure 5-40 SpW and CAN FMC interface card installed on R-ICU carrier board.....	109
Figure 5-41 Hardware Design and Zynq MPSoC Components for the MOSAR R-ICU.....	110
Figure 5-42 General overview of R-ICU software architecture	112
Figure 5-43 R-ICU detailed software architecture	113
Figure 5-44: Surface Anomaly Detection Pipeline. This is the basic structure used for the implementation of all the other features.....	116



Detailed Design Document

Figure 5-45: Module Anomaly Detection	117
Figure 5-46: Interface Anomaly Detection.....	117
Figure 5-47: Reconfiguration Anomaly Detection.	118
Figure 5-48: DFN and DFPC architecture	119
Figure 5-49: Hardware configuration of vision subsystem	120
Figure 5-50: Software architecture of the vision subsystem	121
Figure 6-1: References of the HOTDOCK standard interfaces spot on SVC and CLT.....	123
Figure 6-2: References of the spacecraft module faces (in the initial MOSAR setup configuration) ..	123



Detailed Design Document

List of Tables

Table 2-1 – Declination of ground demonstrators covering main functionalities identified in real mission scenarios	19
Table 2-2: Components design responsibilities	27
Table 2-3: MOSAR software main responsibilities	27
Table 3-1: Assumptions for the SpaceWire network architecture	36
Table 4-1: System characteristics relevant for configuration	42
Table 5-1: Selection of PUS services for MOSAR	66
Table 5-2: SM1-DMS mass budget	73
Table 5-3: SM1-DMS power budget	73
Table 5-4: Main characteristics of Thermal IF	76
Table 5-5: Thermal subsystem TM HK1 – Status Data	77
Table 5-6: Thermal subsystem TM HK2 – Temperature Data	77
Table 5-7: Thermal subsystem TM HK3 – Power and Pressure Data	77
Table 5-8: Thermal subsystem TM HK4 – Flow and Motor Data	78
Table 5-9: Battery TMC Codes – Packet Types List	78
Table 5-10: SM2-PWS mass budget	79
Table 5-11: SM2-PWS power budget	79
Table 5-12: Battery TM HK1 – Status Data	82
Table 5-13: Battery TM HK2 – Power Data	83
Table 5-14: Battery TMC Codes – Packet Types List	83
Table 5-15: SM3-BAT mass budget	84
Table 5-16: SM3-BAT power budget	84
Table 5-17: SM3-BAT mass budget	86
Table 5-18: SM3-BAT power budget	86
Table 5-19: SM5-OSP1 mass budget	89
Table 5-20: SM5-OSP1 power budget	89
Table 5-21: SM6-OSP2 mass budget	91
Table 5-22: SM6-OSP2 power budget	91
Table 5-23: SVC/CLT power budget	96
Table 5-24: Features of the different declination of HOTDOCK	101
Table 5-25: Standard interface selection for MOSAR components (O=optional)	102
Table 5-26: HOTDOCK TM HK1 – Fast Polling Data	103
Table 5-27: HOTDOCK TM HK2 – Status Data	103
Table 5-28: HOTDOCK TM HK3 – Temperatures Data 1	104
Table 5-29: HOTDOCK TM HK4 – Temperatures Data 2	104



Detailed Design Document

Table 5-30: HOTDOCK TM HK5 – Motor Data (Debug Only).....	104
Table 5-31: HOTDOCK TMC Codes – Packet Types List	105
Table 5-32: cPDU TM HK1 – Fast Polling Data	107
Table 5-33: cPDU TM HK2 – Status Data.....	107
Table 5-34: cPDU TMC Codes – Packet Types List.....	108
Table 5-35: DFPCs Variants.....	116
Table 5-36: DFPCs Instances	118
Table 5-37: Hardware Selection for Visual Processing System.....	121



1 Introduction

1.1 Purpose and Scope

This report is the output of the detailed design activities of the project MOSAR. The document provides the overall system architecture, the description of the ground and space system operations in the context of the project demonstration and the detailed design of the different components and sub-systems.

This document updates the information provided at PDR in [RD1] (Preliminary Design Document), for CDR at the detailed design level. It is supported by the description of the demonstration and tests procedures in [RD2].

1.2 Document Structure

In brief, the document is structured as follows:

- Chapter 1 Introduction:** this section.
- Chapter 2 MOSAR System Overview:** gives an introduction about the MOSAR operational mission concept and the demonstration purpose, with the definition of the MOSAR setup components and global architecture.
- Chapter 3 Spacecraft Reconfiguration:** describes the concept of hardware and software reconfiguration, addressed in the MOSAR project.
- Chapter 4 System Operations** introduces the ground and space segment operations, as they will be illustrated with the MOSAR demonstrator.
- Chapter 5 Components Detailed Design:** describes each MOSAR setup component at the detailed design stage.

1.3 Applicable Documents

- AD1 Strategic Research Cluster “Space Robotics Technologies” – Collaboration Agreement
- AD2 MOSAR Consortium Agreement, version 1.0 (7-Nov-2018)
- AD3 MOSAR Grant Agreement (821996) (18-Jan-2019)
- AD4 MOSAR Grant Agreement (821996) AMENDMENT Reference No AMD-821996-1
- AD5 MOSAR Grant Agreement (821996) AMENDMENT Reference No AMD-821996-3
- AD6 MOSAR D1.4 System Requirements Document, MOSAR-WP1-D1.4-SA issue 1.0.0 (1-Sep-2019)

1.4 Reference Documents

- RD1 MOSAR D2.4 Preliminary Design Document, MOSAR-WP2-D2.4-SA issue 1.1.0
- RD2 MOSAR Demonstration Procedures, MOSAR-WP3-D3.5-DLR, issue, 1.0.0



Detailed Design Document

- RD3 MOSAR D3.1 OG1-5 Adaptations and Extensions Detailed Design, MOSAR-WP3-D3.1-GMV, issue 1.0.0
- RD4 MOSAR-TASU-SYS-MAN-001_1.0_TAS-UK_R-ICU User Manual
- RD5 MOSAR-TASU-FMC-TEST-001_1.0_TAS_UK_FMC_Hardware_Verification
- RD6 European Cooperation for Space Standardization. Space Engineering (2016) — Telemetry and telecommand packet utilization (PUS) Services. ECSS-E-ST-70-41C.
- RD7 MOSAR-WP3-D3.7-SA_1.0.0 (HOTDOCK Design Definition File)
- RD8 MOSAR-WP3-D3.8-SA_1.0.0 (HOTDOCK Design Justification File)
- RD9 ISECG Technology Working Group, Telerobotic Control of Systems with Time Delay - Gap Assessment Report, 2018
- RD10 Beckhoff EtherCAT-G: <https://www.ethercat.org/en/ethercat-g.html>; date of link 10. June 2020)
- RD11 EtherCAT Technology Group; Press Release, https://www.ethercat.org/download/press/etg_102015_e.pdf; date of link 10. June 2020)
- RD12 Alexander Beyer et.al., CAESAR: Space Robotics Technology for Assembly, Maintenance, and Repair, 69th International Astronautical Congress (IAC), Bremen, Germany, 1-5 October 2018
- RD13 Nathan Britton et.al., KRAKEN: Compliant Manipulator for Small Spacecraft, ICRA 2019 Workshop on Robot Technology for In-Space Assembly, 2019



Detailed Design Document

1.5 Acronyms

AD	Applicable Documents
API	Application Programming Interface
APID	Application Process Identifier
ASN.1	Abstract Syntax Notation One
CAD	Computer-Aided Design
CAN	Controller Area Network
CDR	Critical Design Review
CLT	Client Satellite
CMD	Command Dispatcher
COTS	Commercial Off-the-Shelf
cPDU	central Power Distribution Unit
CPU	Central Processing Unit
DFN	Data Fusion Node
DFPC	Data Fusion Processing Compound
DLR	Deutsches Zentrum für Luft- und Raumfahrt
DMA	Direct Memory Access
DMS	Data Management System
EC	European Commission
ERGO	European Robotic Goal-Oriented Autonomous Controller
FES	Functional Engineering Simulator
FMC	FPGA Mezzanine Card
FPGA	Field Programmable Gate Array
GCI	Ground Control Interface
GPIO	General Purpose Input/Output
GUI	Graphical User Interface
HD	HOTDOCK
I3DS	Integrated 3D Sensors
ICU	Instrument Control Unit
ID	Identifier
IEEE	Institute of Electrical and Electronics Engineers
IF	Interface
IP	Intellectual Property or Internet Protocol
JSON	JavaScript Object Notation
LVDS	Low Voltage Differential Signaling
LVTTL	Low Voltage Transistor Transistor Logic
MCC	Mission Control Centre
MOSAR	Modular Spacecraft Assembly and Reconfiguration
MPSoC	Multiprocessor System on Chip
OBC	On-Board Controller
OBC-C	OBC of the Client Spacecraft
OBC-S	OBC of the Servicer Spacecraft
OBCP	On-Board Control Procedures
OBSW	On-Board Software
OG	Operational Grant
OMPL	Open Motion Planning Library
OS	Operating System
OSP	Optical Sensor Payload
PC	Personal Computer-Aided
PCB	Printed Circuit Board
PCI	Peripheral Component Interconnect
PDDL	Planning Domain Description Language
PDU	Power Distribution Unit
PGCI	PUS Ground Control Interface



Detailed Design Document

PICU	Payload Interface Control Unit
PUS	Packet Utilization Standard
PWS	Power Subsystem
QoS	Quality of Service
R-ICU	Reduced Instrument Control Unit
RARM	Robotic Arm
RCOS	Robot Control Operating System
RD	Reference Document
RID	Review Item Discrepancy
RMAP	Remote Memory Access Protocol
SI	Standard Interface
SM	Spacecraft Module
SoC	System on Chip
SOD	SM Operations Decomposer
SpW	SpaceWire
SVC	Servicer Spacecraft
SW	Software
TASTE	The ASSERT Set of Tools for Engineering
TBC	To Be Confirmed
TBD	To Be Determined
TC	Telecommand
TCP	Transmission Control Protocol
THS	Thermal Subsystem
TM	Telemetry
TSP	Time and Space Partitioning
UART	Universal Asynchronous Receiver-Transmitter
UDP	User Datagram Protocol
URDF	Universal Robot Description Format
USB	Universal Serial Bus
WAN	Wide Area Network
WM	Walking Manipulator
WMC	Walking Manipulator Controller



2 MOSAR System Overview

2.1 MOSAR Mission Overview

The technologies developed in MOSAR aim to support the development of fully modular and on-orbit reconfigurable spacecraft. This ambitious objective is declined into a demonstrative mission concept that allows the deployment of MOSAR technologies in the near/middle future.

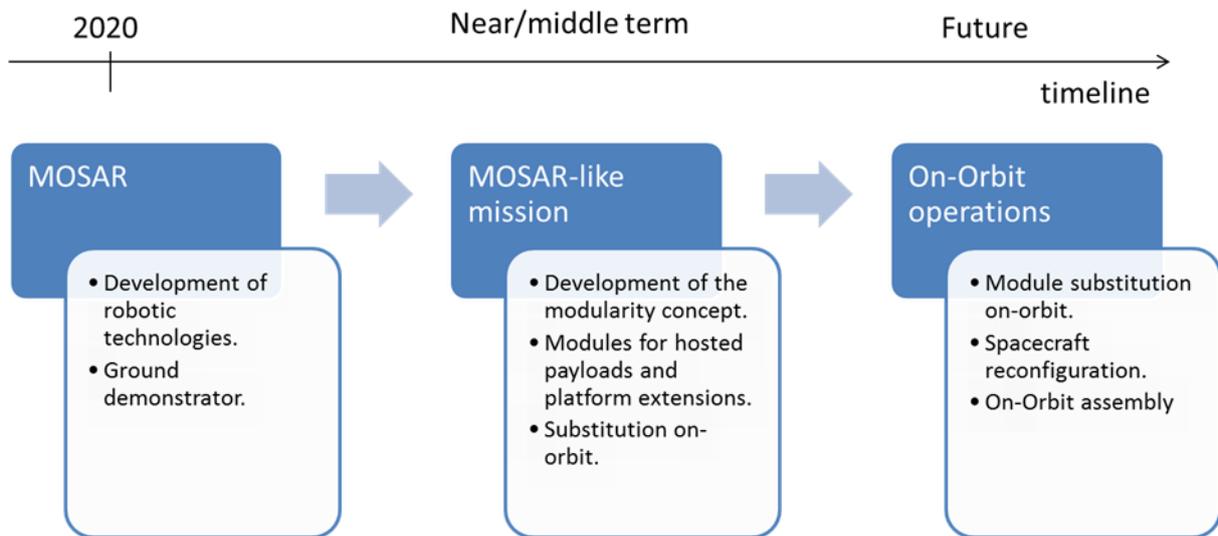


Figure 2-1: MOSAR roadmap to space application

The proposed demonstrative mission concept and spacecraft design are represented in the next figure. The baseline scenario is the one of a Servicer Spacecraft (SVC) transporting a cargo of Spacecraft Modules (SM) and a dedicated Walking Manipulator (WM). This last enables a number of operations with the transfer of SM from and to the Client Spacecraft (CLT), with the purpose of adding, replacing or enhancing client platform or payload functionalities. In order to allow the servicing operations CLT and SVC control authority shall be regulated; in fact, in order to replace or add SMs to the Client, the Servicer shall be able to manage the CLT operations. This could be achieved by providing the CLT with a dedicated operative mode (e.g. a Servicing Mode), similar to a standard satellite Safe Mode, allowing the SVC to control only the essential resources needed for the servicing operations and enabling the CLT to re-take control if necessary.

The client satellite is built around a standardized common platform (product line), with all the required main functionalities. The platform is designed to cover an envelope of payloads and functions. It can be initially specialized, on-ground, with a primary payload and services modules. Both the platform and the payload are equipped with standard interfaces enabling the connection of the standard modules, for initial configuration on-ground or for addition or replacement of units during on-orbit mission.



Detailed Design Document

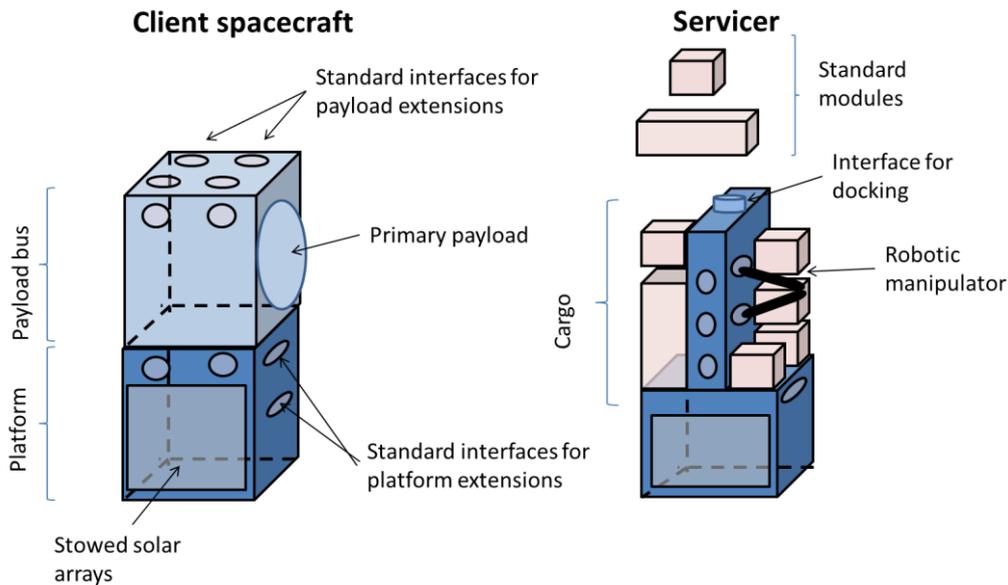


Figure 2-2: Spacecraft design MOSAR-like mission (client and servicer)

This mission concept aims at making use of MOSAR technologies to introduce enhanced functionalities on space missions, mainly on two topics:

- Standardization of design thanks to modularity approach: a common design of the product line allows multiple configurations to meet different mission objectives thanks to the standardized interfaces that could be used to plug additional payloads and service modules. In this way the customization needed to meet specific mission objectives does not need a redesign of the main elements of the spacecraft.
- The spacecraft, equipped with multiple standard interfaces and some modules, performs its original missions for some years with the option of introducing new elements or reconfiguring the mission by replacing or adding new modules brought by a servicer satellite equipped with a robotic manipulator.

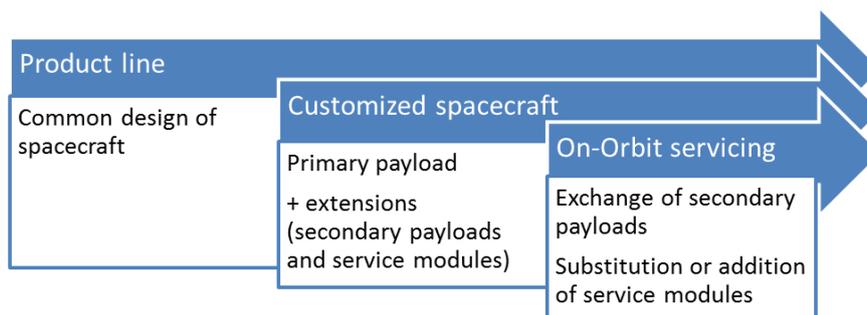


Figure 2-3: Life-cycle of the MOSAR mission concept

Figure 2-4 illustrates examples of different client spacecraft mission configurations.



Detailed Design Document

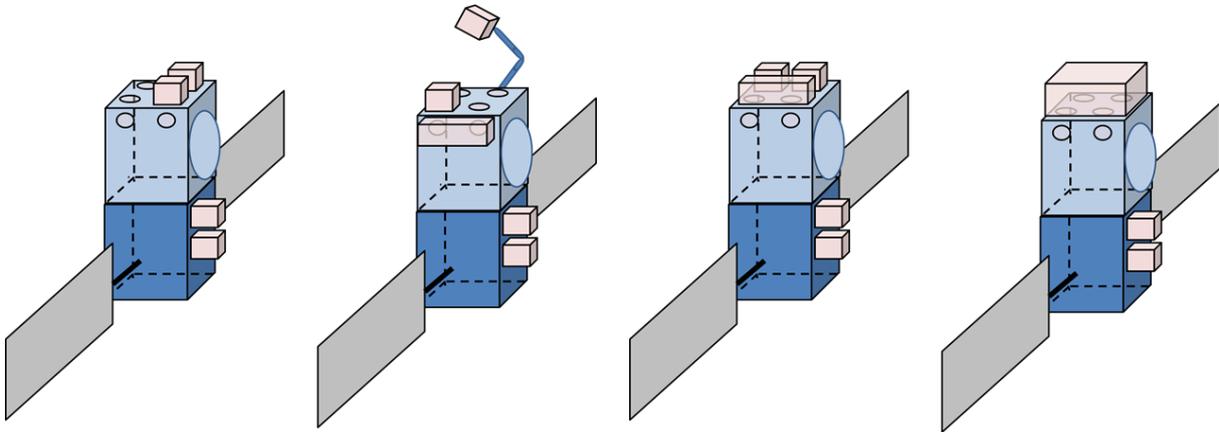


Figure 2-4: Spacecraft with different mission configurations

2.2 From Mission Overview to MOSAR Demonstration

MOSAR demonstrator aims at reproducing the main functionalities needed for on-orbit module assembly and reconfiguration of spacecraft, as described in the mission section above. Figure 2-5 illustrates the concept for the spatial mission, involving a client and a servicer satellite, already docked. The scope of the project covers the phases immediately after on-orbit rendez-vous of the space vehicles and docking; in particular:

- Exchange of modules between servicer and client spacecraft.
- Assembly of modules interacting with other modules or with the spacecraft bus
- Connection of a new payload.

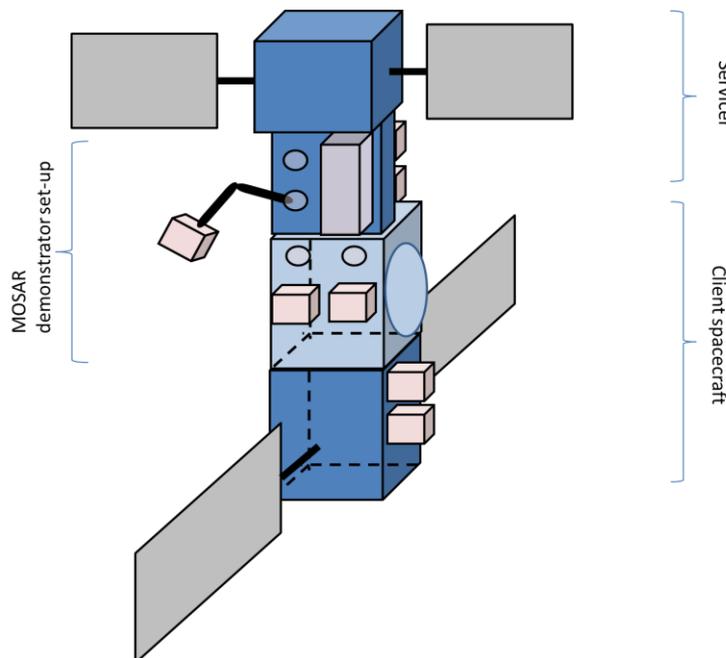


Figure 2-5: On-orbit reconfiguration with substitution and addition of modules



Detailed Design Document

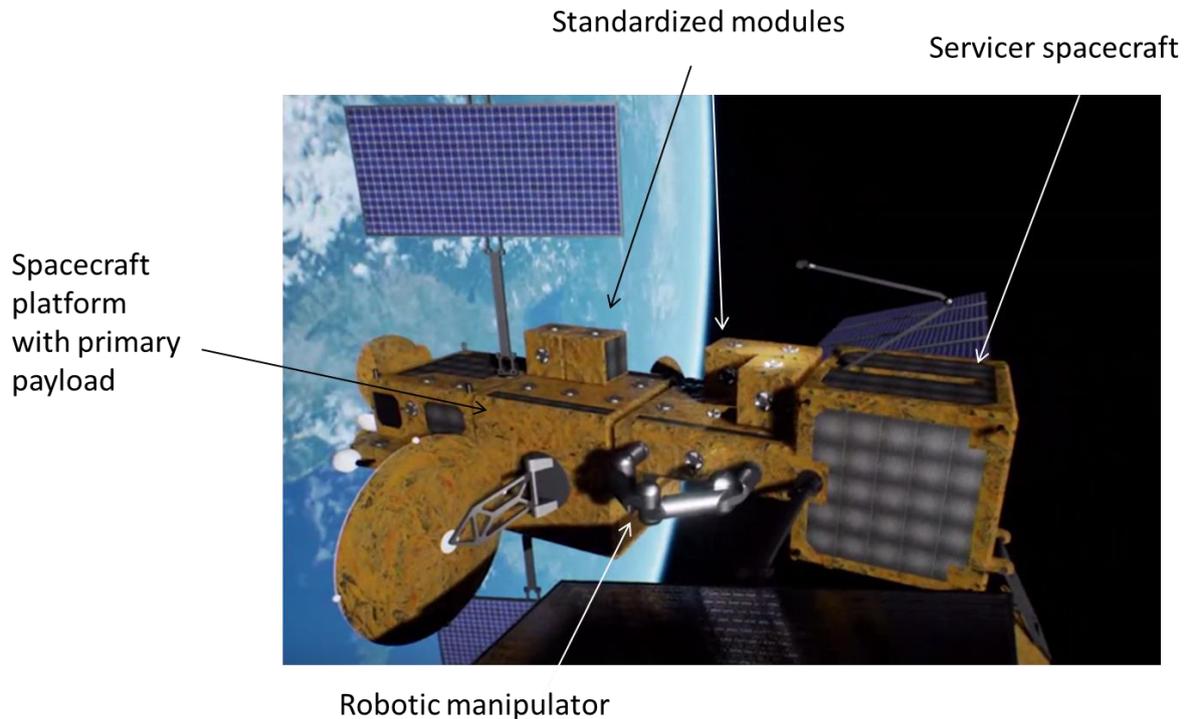


Figure 2-6: Representation of MOSAR mission

The ground demonstrator will be focusing in the validation of the sequence of operations needed for the preparation of the operations, the manipulation of the standard modules and the operations of the robotic manipulator. It will address the following topics related to modular and re-configurable spacecraft:

- **Hardware:** with the possibility to re-configure the physical arrangement of the spacecraft and/or providing means to add/replace/upgrade specific functions.
- **Software:** with the possibility to re-configure components responsibilities and support the re-configuration operations
- **Data:** with the possibility to re-route TM/TC and data transmission along the different elements
- **Power:** with the possibility to re-route and control the power transmission along the elements.
- **Thermal:** possibility to manage thermal loads, to increase dissipation capabilities or maintain thermal stability of modules.

Based on the mission concept presented in section 2.1, different use cases could be imagined representative of real mission operations. Following an incremental approach, the following scenarios are considered:

- **Basic scenario:** assembly of a secondary payload. The secondary payload is not very demanding in terms of resources, and only needs data connection and power interface to be provided by the client spacecraft.
 - As an example, let's consider an optical camera.
- **Enhanced payload:** in this case the secondary payload set-up is more complex. In the first scenario the payload only needed the spacecraft bus to provide the resources needed. In this case, the payload is more demanding, and the capacity available on the client spacecraft is not enough to support its operation. In this case the payload will be added together with some additional modules that provide power, thermal or data capabilities.



Detailed Design Document

- An example of this could be an infrared camera, usually very demanding in terms of thermal management.
- Full functionality: following the incremental approach, in this case a payload is added to a satellite already on-orbit, but in this case the payload needs much more resources that are obtained through the addition of payload and platform extensions. Complexity is increased, having numerous modules interconnected and connected to the spacecraft bus. In terms of data and power routing, several paths are available, requiring smart management by the client OBC in order to optimize the system combined resources (spacecraft bus plus extensions).
 - An example of this scenario is a radar payload requiring high power that is provided by service modules that are brought by the servicer satellite.
- Substitution of modules: in case of failure of a certain module, or following a preventive maintenance strategy; some modules of the spacecraft could be substituted. This supposes that the spacecraft implements some equipment in a modular shape, enabling on-orbit substitution.
 - For instance battery packs or radiator surfaces.

The following figure presents a schematic view of the different scenarios. The modules are represented by its functionality; each module implementing a basic function:

- SPL: secondary payload.
- PICU: payload interface control unit, module managing power and data distribution to the payload.
- THS: thermal subsystem: module implementing some dissipative surface and a fluidic circuit enabling thermal management of a neighbour module.
- BAT: for battery.
- SA: for solar array.

Note that the figure does not intend to be representative in terms of architecture or shape of the spacecraft or the module, it only presents each element by its main functionality.

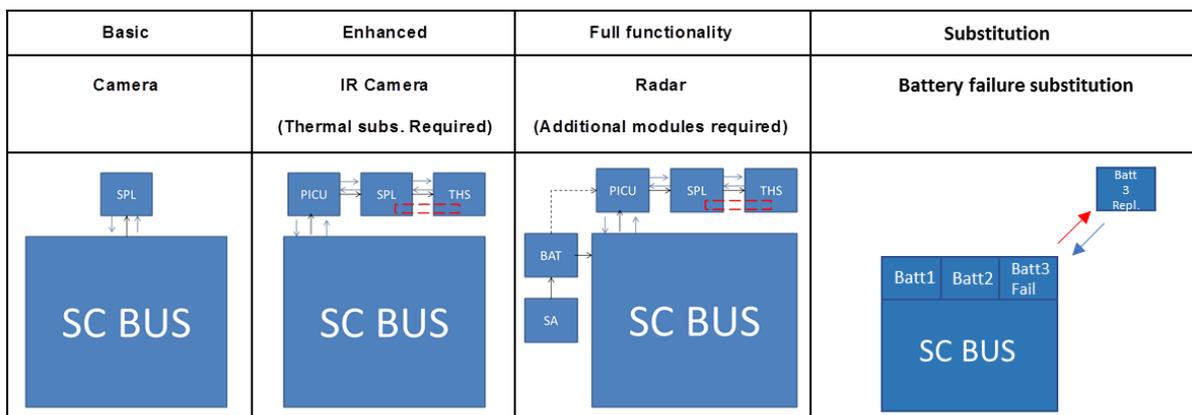


Figure 2-7 – Different use cases of MOSAR mission concept.

As the final goal of the project, the purpose of the demonstration is to illustrate the full sequence of operations representative of the different use-cases described above:

- Scenario 1: Assembly of the CLT with new SM from the servicer
- Scenario 2: Detection and replacement of a damaged SM
- Scenario 3: Re-routing of power/data bus through the SM
- Scenario 4: Thermal transfer between SM
- Scenario 5: Client software reconfiguration, as function of available spacecraft modules



Detailed Design Document

Real mission scenarios		Basic	Module substitution	Enhanced payload	Full functionality
		Optical camera	Battery pack	IR camera	Radar payload + service modules
Main functionalities needed	Assembly of the CLT with new SM from the servicer	X	X	X	X
	Detection and replacement of damaged SM		X		(X)
	Re-routing of power/data bus through the SM				X
	Thermal transfer between SM			X	(X)
Demonstrators		S1 – Initial assembly of SMs	S2 – Replacement of failed SM	S3 – Thermal transfer between two SMs	S4 – Automatic CLT network reconfiguration

Table 2-1 – Declination of ground demonstrators covering main functionalities identified in real mission scenarios

The MOSAR demonstrator shall allow verifying and validating the following functionalities relevant for future modular spacecraft missions (with reference to the MOSAR mission's requirements [AD6]):

- Design and creation of a re-configuration execution plan (FuncR_S105)
- Simulation of the execution plan (FuncR_S106)
- Manipulation and repositioning of SM (FuncR_S101)
- Control and re-location of the WM (FuncR_S104, FuncR_S107)
- Update/upgrade of satellite functionalities (FuncR_S102)
- Data and power transfer between SM
- Resources re-allocation, data and power routing (FuncR_S110)
- Heat management between SM (FuncR_S115)
- Failure detection and handling (FuncR_S111), with automatic client network reconfiguration (without the support of the servicer or ground segment)

These functionalities will be demonstrated in MOSAR through the operational scenarios described above, and detailed in [RD2].



2.3 Demonstrator Components and Architecture

Figure 2-8 represents the main components of the MOSAR demonstrator that is split between the space and the ground segment. The space segment is representative of the central part of the assembly in Figure 2-5.

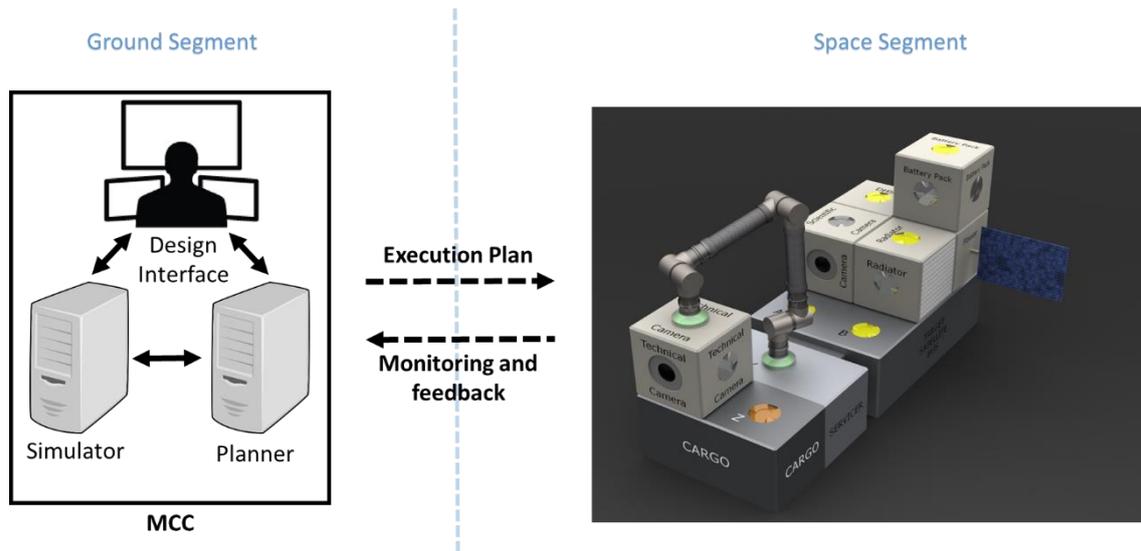


Figure 2-8: MOSAR Demonstrator Overview

2.3.1 Ground Segment

The autonomous re-configuration, between the servicer spacecraft and the client satellite, follow an execution plan prepared and validated off-line, in the Monitoring and Control Centre (MCC), on the ground segment. The MCC includes a satellite design, modelling and validation tool, specifically targeting modular satellites applications. It also allows the automatic planning of the assembly or reconfiguration sequence that can be verified with a multi-physics simulator. All these elements are working iteratively together to prepare a valid execution plan that is finally uploaded to the spacecraft for execution. Based on the monitoring and feedback information received from the spacecraft during the operations (e.g. detected failed module), the MCC can update the execution plan. The MCC finally includes visualisation front-end to support the design, verification and monitoring activities during sequence execution.

In the context of the MOSAR demonstrator scenario, the ground segment consists of a design tool, the ERGO Agent, including the planner and Functional Layer, the MOSAR simulator and the Monitoring and Control interface. Specific adaptations are required in the planner algorithms and in the libraries of the simulator models to consider the gravity conditions in the ground lab and the actually available MOSAR components in the demonstrator setup. However, these adaptations have no impact on the general system architecture and demonstrator mission procedure.

The simulator replaces the demonstrator space segment, introduced in 2.3.2, on a functional level. More specifically, the simulator is a virtual demonstrator that is

- using the same input and output signal interface as used by the demonstrator space segment,
- using the same system components as used by the demonstrator space segment to represent the system topology and its reconfigurations,
- providing the same manipulator skills and degrees of freedom for spacecraft module assembly as done by the demonstrator space segment,



Detailed Design Document

- providing the same spacecraft module and HOTDOCK functionalities as the demonstrator space segment,
- providing power, thermal, and data network functionalities and logics to let the user analyse the system as done with the demonstrator space segment.

In terms of system architecture, apart from differences in the selected communication protocols, the simulator is fully transparent for the ERGO planner. The goal is to let the planner deliver the same results on algorithm level, independent from the attached system appearance.

2.3.2 Space Segment

The MOSAR demonstrator space segment and its main components are illustrated in Figure 2-9. It includes:

- The servicer spacecraft (SVC) has the role to bring the building blocks and tools for the operations of (re)-configuration of the client satellite. It has its own on-board computer (OBC) that will manage all the operations of configurations (following the plan prepared by the ground segment). It also manage the communication with the monitoring and control centre (MCC), on ground, during the assembly operations.
- The client satellite bus (CLT) is the mechanical structure that supports the assembly of the client satellite. During (re)-configuration operations, the CLT is docked to the SVC and has a data interface with it, such that the SVC OBC can operate components on the CLT.
- The spacecraft modules (SM) are individual units proposing a specific function of the satellite, either as sub-system (e.g. OBC, power generator, batteries, thermal management) or as payload (e.g. camera). In the demonstration of MOSAR, they take the shape of 1U cube elements. For future space applications, different shape and size can be considered, as illustrated in section 5.4.1.
- The walking manipulator (WM) is a symmetric robotic arm (both extremities can play the role of the robot base or the end-effector). It allows manipulating the SM between the SVC and CLT during assembly operations. It has also the capability to “walk” along the structures to be able to reach the different components.

Standard interfaces connectors, HOTDOCK, are used to inter-connect all the above components. They can provide mechanical, data, power and thermal transfer, respectively, between the modules, the spacecraft/bus and the walking manipulator.

The MOSAR demonstrator will also include an external visual 3D processing system that is used to analyse and demonstrate algorithms and technics, which will support future autonomous re-configuration operations. This will mainly address detection of unexpected configuration, spacecraft deterioration and discrepancies, based on validation of modules, interfaces or manipulator shapes and features detection. At this stage, this feature will not be directly interfaced with the modular spacecraft operations.



Detailed Design Document

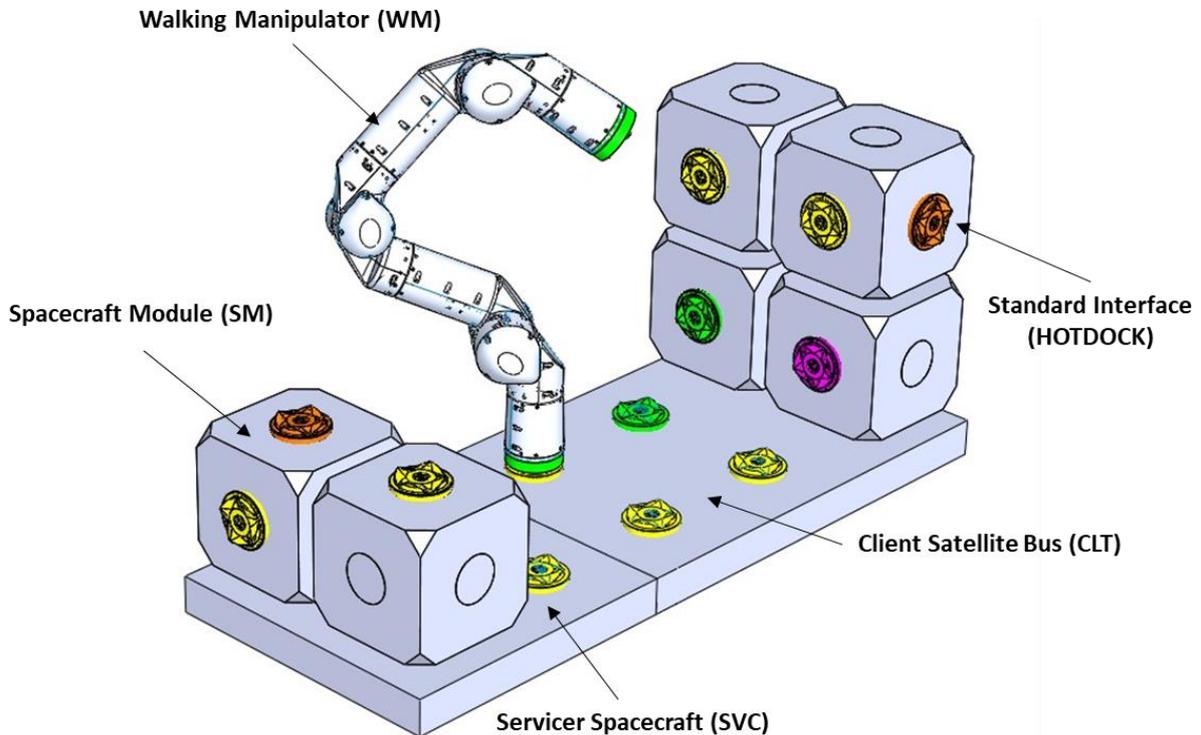


Figure 2-9: Main components of the MOSAR space segment

2.3.3 Generic Data and Power Architecture

The components of the system (SM, WM, SVC and CLT) present a similar architecture for the powering and control of their elements (SI, payload or PDU). The corresponding data and power architecture is illustrated in Figure 2-10. We can identify the following elements.

2.3.3.1 On-Board Computers (OBC)

The OBC runs the software that will manage the high-level functions of the spacecraft during assembly/reconfiguration or during nominal operations. The setup will include two OBCs, the OBC-S located in the SVC (mainly dedicated to the reconfiguration operations), and the OBC-C as payload of one of the SM on the CLT (more dedicated to the spacecraft nominal operations).

The OBC-S has the following software components:

- The Autonomy Agent that manages the execution of the other components according to the plan execution.
- The Telemetry and Telecommand (TCC) Service that provides the capability to command the Agent from the MCC and sent back telemetry of the operations. The TTC Service also allows for commanding at individual component level, for testing purposes and anomaly resolution.
- The HOTDOCK Management that provides high level control and telemetry of the SI during reconfiguration (with the purpose to provide more intelligence in the local components, as described below).
- The Data and Power Management to support, respectively, data (by commanding R-ICU) and power (by commanding cPDU) re-configuration during the assembly process.



Detailed Design Document

- The Client Management to manage the states of the CLT during reconfiguration. Before starting the robotic operation, the CLT must transition to a safe mode and the OBC-C will hand over control of the functions needed for by reconfiguration to the OBC-S. At the end of the operation, the process is reversed and the OBC-C restarts nominal operations in the new configuration. The Client Management component arbitrates these transitions.
- The Walking Manipulator Management for the high-level control and telemetry of the WM during the reconfiguration operations (the WM is always considered as part of the SVC, although it can be re-localized physically on the CLT).

The OBC-C, more oriented to the CLT nominal operations, has the following software components:

- The Telemetry and Telecommand (TCC) Service for commanding and monitoring of the SMs during nominal operations.
- The Reconfiguration Management that manages the modes of the software, according to which SMs are available.
- The Data and Power Management to support data and power routing during nominal operations (e.g. for FDIR management, as faulty node isolation).
- The Payload Management for the operations of the SMs payloads (e.g. thermal, battery,...)
- The Payload relay to support the transfer of large size data (e.g. video images).

2.3.3.2 R-ICU

The R-ICU is derived from the ICU developed in I3DS. Located in each SM and in the two spacecraft (SVC and CLT), it has mainly two roles:

- The R-ICU controller will interface and control the different elements of the SM or spacecraft bus. That includes the HOTDOCK SI, the Power Distribution Unit (see below) and the SM payloads. The R-ICU receives high-level commands from the OBCs and translate them in low level control sequence. Having this local intelligence in the SM reduces the number of data transfer and allows the OBC to stay at a higher abstraction level in regards to the command control, improving the independence from the SM possible evolutions.
- The R-ICU SpaceWire Router manages the SpW routing and data communication functions (see below for the description of the SpW bus).

It has to be noted that the Walking Manipulator doesn't include an R-ICU due to volume constraints. The R-ICU is then replaced by a local controller associated with a SpW conversion unit (from USB to SpW), to be able to connect to the MOSAR SpW bus.

2.3.3.3 cPDU

The central power distribution unit (cPDU) is interfaced with the main power bus of the spacecraft through the HOTDOCK. It has the function of routing of the power bus with the other HOTDOCK of the component. It also converts and provides the required power/voltage to the other elements (R-ICU, payload).

2.3.3.4 HOTDOCK Standard Interface

Each component has a set of HOTDOCK interfaces that provide a pass-through for the main power bus and the SpW data bus, respectively connected to the cPDU and the R-ICU. On top of that the SI ensures the mechanical connections between the different components.



Detailed Design Document

2.3.3.5 Payloads

The payload is specific to each component and is interfaced with the R-ICU for command, telemetry and data transfer. In the case of the WM, this corresponds to the Joint Controllers. For the SM, in MOSAR battery, thermal and camera payloads will be integrated.

2.3.3.6 Communication Links

The MOSAR demonstrator implements a SpaceWire network between the different components (modules, WM, spacecraft), passing by the HOTDOCK interfaces. The communication is enabled by the SpaceWire routers in the R-ICUs. They direct the SpW packets according to the topology of modules as defined by the OBC and the planner. The Figure 2-10 highlights the SpW network between the OBCs and the R-ICU modules. In the case of the WM, it is not, by nature, permanently attached to one element, but will always connect through one of the R-ICU router to be integrated inside the network (the WM himself can play the role of a SpW relay between two SM). The two OBCs are directly connected through SpW to represent the data link of the docking interface.

In order to enable the communication between the SM and the OBC, the RMAP protocol is proposed. RMAP enables memory access over a SpaceWire network with a defined packet structure. The commands sent by the OBC through RMAP are directly written to the R-ICU memory, and the data to be sent back will be placed in memory and retrieved by the OBC. The control software, running on the R-ICU will get access to the memory to enable the control and telemetry of the connected elements (SI, cPDU and payloads).

The CAN protocol is proposed as the standard connection between the R-ICU and the elements, unless the connected item has specific interface (e.g. with a camera). This is the nominal interface for the control of HOTDOCK. For the other elements developed in the course of the project, this approach will simplify the development and integration of them, as a common CAN compatible controller can be developed (also required for local control of the element).

The communication between the Monitoring PC (Ground) and the OBC uses generic mechanisms:

- For general monitoring and command purposes, the TASTE middleware (PolyORB-HI) is used. The TM/TC services of the Client spacecraft (including the SMs) are based in the ESROCOS PUS Services library.
- For large data transfer, a specific mechanism is needed. The ZeroMQ protocol is selected for this purpose. The ZeroMQ library is selected for commonality with the I3DS framework. I3DS uses ZeroMQ for communication with the ICU and transport of the data (both raw sensor data and metadata encoded in ASN.1). The deployment of ZeroMQ in the ICU and the encoding of the data from the optical sensors (which are selected from the I3DS sensor suite) have therefore been tested in the previous project.

2.3.3.7 Main Power Bus

The main power bus voltage has been selected at 48V, as it is the closest ECSS standard voltage (for systems up to 8kW), while considering the demonstrator commercial components and the limitation of the current rating along the system (compared to the initial envisaged 28V). Most of the component being compatible, it is also possible to go for 50V (ECSS Standard), after a final check of all constraints.



Detailed Design Document

2.3.3.8 Generic Data and Power Deployment Diagrams

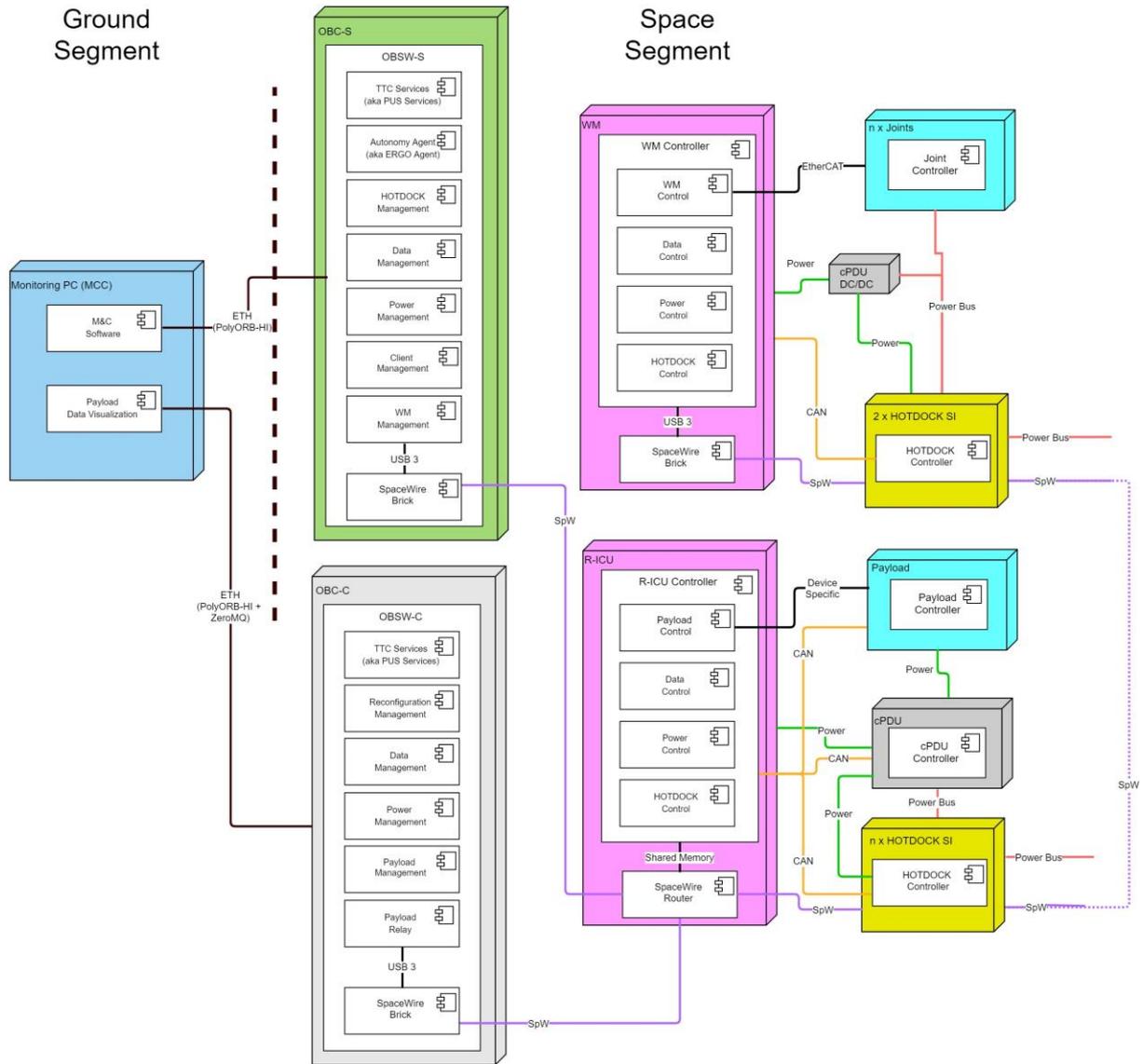


Figure 2-10: MOSAR generic data and power architecture



Detailed Design Document

2.4 MOSAR Product Tree

Figure 2-11 illustrates the global MOSAR demonstrator product tree, highlighting the main components of the setup. Table 2-2 provides for the different components the design responsibilities among partners. Table 2-3 defines the main software responsibilities. Specific contributions from other partners can be required but are not displayed in the table.

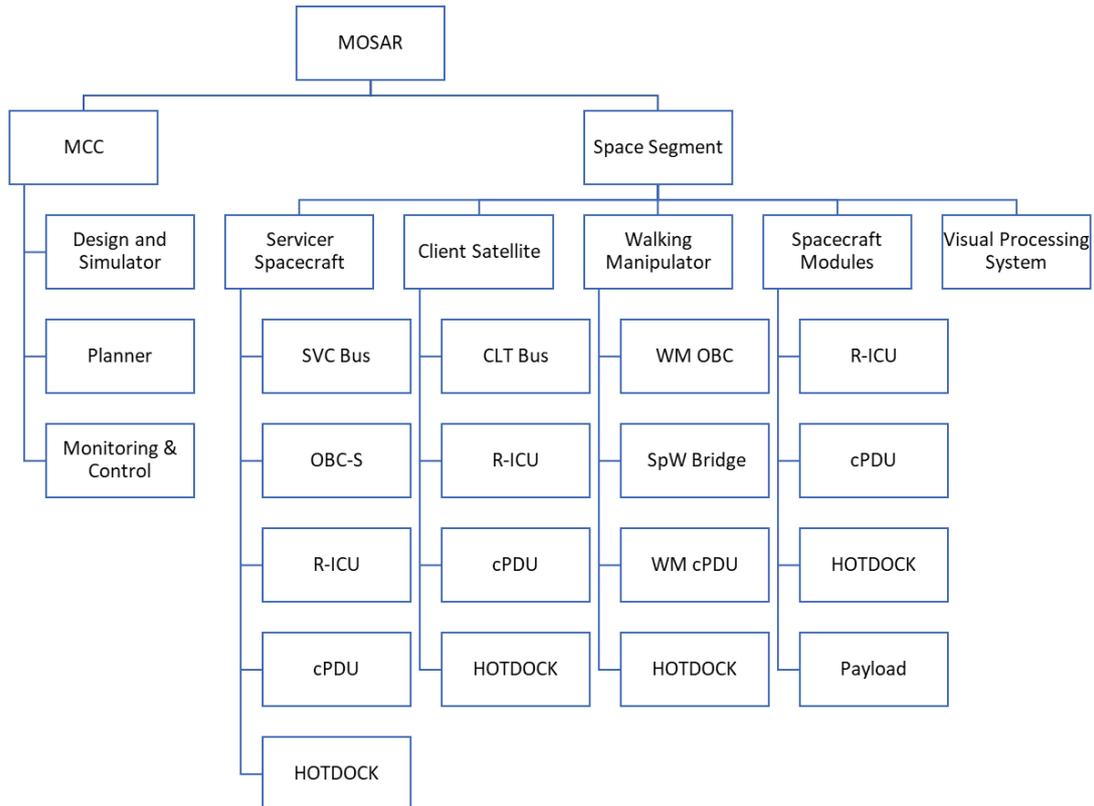


Figure 2-11: MOSAR demonstrator product tree



Detailed Design Document

Table 2-2: Components design responsibilities

Component	Responsible Partner
MCC Design and Simulator Tool	DLR
MCC Planner	GMV
Monitoring and Control	GMV / DLR
SVC and CLT buses	SITAEL
OBC-S	SPACEAPPS
R-ICU	TAS-UK
cPDU	SPACEAPPS
HOTDOCK	SPACEAPPS + MAGSOAR (thermal interface)
Walking Manipulator	SPACEAPPS
Spacecraft Modules	SITAEL
Spacecraft Modules Payloads	DMS – OBC: SPACEAPPS SM2 – PWS: MAGSOAR (thermal subsystem) SM3 – Battery: SPACEAPPS SM4 – THS: MAGSOAR (thermal subsystem) SM5/SM6 – Optical: TAS-F
Visual Processing System	USTRATH

Table 2-3: MOSAR software main responsibilities

Software	Responsible Partner
MCC Design and Simulator Software	DLR
MCC Planner Software	GMV
Monitoring and Control Software	DLR + GMV (TTC service)
OBC Software	GMV + ELLIDISS (TTC service, Autonomy Agent, Component Management)
R-ICU Software	TAS-UK + SPACEAPPS (support for components control)
cPDU Controller software	SPACEAPPS
HOTDOCK Controller software	SPACEAPPS
Walking Manipulator High-Level software	DLR
Walking Manipulator Low-Level software	SPACEAPPS
Spacecraft Modules Payloads software	See above
Visual Processing System software	USTRATH



3 Spacecraft Reconfiguration

3.1 Hardware Reconfiguration

3.1.1 Mechanical Reconfiguration

The standard interfaces, embedded in each spacecraft module, ensure the mechanical re-configuration capability of the spacecraft. This includes the mechanical connection as well as the physical harnessing for data, power and thermal transfer.

Mechanical re-configuration allows placing, theoretically, each module in different positions and orientations, as function of the need of the mission, as payload orientation, thermal management.... This is of course limited to module constraints, including payload specific face (e.g. camera) or relative required connections between two modules.

The mechanical reconfiguration is possible by the presence and operations of the Walking Manipulator. Equipped also with standard interfaces, it is able to manipulate and operate the modules (through the data and power interface). A high dexterity manipulator is recommended to be compliant with a lot of different manipulation or motion use cases along the structure (re-orientation of modules, non-planar displacement on the bus, obstacle avoidance...).

3.1.2 Data Reconfiguration

SpaceWire network reconfiguration is done from the SVC OBC using RMAP to update the routing tables when nodes are added/removed. There are several distinct scenarios to cover:

1. *Initial network discovery:* The SVC needs to find out the structure of the SpaceWire network and what devices are attached. To achieve this, a SpaceWire based network discovery service will be supported. The SpaceWire-PlugNPlay extension adds device identification and connect/disconnect event notification to the router design. For MOSAR, during the construction phase, the intention is for the OBC to have full control of the network traffic using RMAP commands. The OBC will query and configure each router to build a network map of device capabilities across the SpaceWire network and manage new devices being attached or detached to the network.
2. *Update of routing tables on addition/removal of modules:* Each unique node on the SpaceWire network will be allocated a unique SpaceWire Logical address as part of the network discovery process. Any node will be able to communicate to any other node using RMAP commands and the Logical address of the destination node. When a Spacecraft Module is added or removed, the routing tables in each R-ICU's routing table will be updated to reflect this change. By updating all routing tables on device connection/disconnection, the SpaceWire network will know the path to any other node. The SVC-OBC will update the router tables via RMAP and the updated network tree will be maintained by the SVC-OBC. SpaceWire network segmentation makes this approach scalable to networks significantly larger than the one required by MOSAR with 1000's of nodes supported.
3. *Support for walking manipulator:* As the walking manipulator moves across the SVC and CLT the SVC-OBC control for it will need to ensure that the correct SpaceWire port is used on the walking manipulator (i.e. the docked port). As this movement is planned it will be known at all times which port on the WM is currently docked. This allows use of both SpaceWire logical or path addressing in conjunction with the network map and WM planning will allow the SVC-OBC to address the correct WM port. Logical addressing will require updates to the WM entries of the R-ICU routing tables as a new WM joint is docked. Path addressing would also remove the



Detailed Design Document

need to continuously update the routing tables as the WM moves across the spacecraft and as only the SVC-OBC needs to access the WM there is no advantage to having a routing table entry that all nodes can use to address the WM. The path addressing header is not protected by CRC however and so this will be used as a secondary option.

The routing tables of the SMs can be updated by the OBC using the *"SpW Routing Table Set Entry"* telecommand. This sets the routing direction for a single logical address and is used during construction and WM travel operations to build and update the SpaceWire network topology.

3.1.2.1 Automatic Network Discovery and reconfiguration

The role of the automatic network discovery and reconfiguration in MOSAR is to detect faults in the SpaceWire network and reconfigure the network if required to route around faulty interfaces. This functionality is tested in the Scenario 4 demonstrator, where a link is disabled. There is also a further goal of being able to discover the structure of the MOSAR spacecraft, capabilities of each SM and the network topology at power-up.

The network reconfiguration process is executed by the CLT-OBC with the R-ICUs responding to network discovery TMTc to provide information about the capabilities of the SM, status of the SpaceWire router and allow settings such as the SpaceWire routing tables to be configured by the OBCs. An adaptation of the draft SpaceWire Plug and Play Protocol standard (22/03/2013) is used for SM capability and network status detection.

3.1.2.2 SpaceWire Plug and Play

SpaceWire PnP is a standardised set of registers designed so that routers and devices can be identified when they are added to the spacecraft's network. The standard also describes how the communication protocols and the applications provided on a device can be discovered, allowing a single device to support several of the protocols that can be run on top of SpaceWire.

The SpW PnP register set used in MOSAR is split into four groups:

1. Device Identification – registers that identify the device, vendor and also network status information for routers.
2. Vendor/Product Strings – human-readable descriptions of the device and its vendor (up to 8K per string)
3. Protocol Support – List of SpW protocols (and their vendors) supported by the device
4. Application Support – List of applications (and their vendors) supported by the device and the SpW protocol that is used to access the application e.g. camera service, RMAP.

The full set of SpW PnP registers have been translated to a ASN.1 file to support the transfer of this information via the RMAP TMTc interface between the OBC and the SMs.

3.1.2.3 SpaceWire PnP TMTc

The SpaceWire PnP standard defines an RMAP style interface that is used to access the PnP registers remotely. Write accesses from the PnP masters have some protection in the form of a Compare-And-Swap write, the master must provide the current fields value when performing a write, only if this value matches is the register allowed to be overwritten by the new data.

In MOSAR the PnP information is contained within the SpaceWire router which has its own TMTc interface. For ease of integration the SpaceWire PnP does not use the RMAP-style commands as provided by the standard but instead uses the existing TMTc infrastructure (which is RMAP based). SpaceWire PnP adds the following TMTc to the R-ICU:



Detailed Design Document

TM:

1. SpW PnP network status
2. SpW PnP Product strings, Protocols supported, Applications supported

The two TMs reflect the two use cases of the network reconfiguration:

- TM1 contains all the information needed for network topology discovery and is a small data packet. This TM may be read several times during network discovery due to the presence of loops.
- TM2 contains the information needed to determine the capabilities of the SM and is a large data packet. This is likely only to be read once per rediscovery once the network topology is found.

TC:

1. Update R-ICU routing tables
2. Set PnP Owner fields

These two TCs allow the results of the network discovery routing to be applied to SMs and allow the OBC to update the owner fields for debug purposes. As there is only one PnP master (the CLT-OBC) it is unlikely that this TC will be required for the autonomous reconfiguration process.

3.1.2.4 Network Reconfiguration Process

The network reconfiguration process is run by the OBC to allow it only to be enabled for the autonomous reconfiguration demonstrator. Once it is enabled it will trigger the reconfiguration process when a fault is detected (RMAP TMTC timeout or NACK received) or when the system is powered up.

An overview of the reconfiguration process is given in Figure 3-1. The routing algorithm to be used is TBC but given the small size of the MOSAR network and the centrality of the OBC-driven communication, it is likely to be a simple minimum spanning tree based approach such as Prim's or Kruskal's algorithms. The routing table TC is used to access the routing table on the R-ICU.

MOSAR Network Reconfiguration

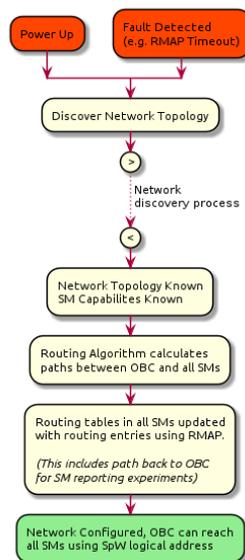


Figure 3-1 MOSAR SpW Network Reconfiguration Flowchart



Detailed Design Document

3.1.2.5 Network Discovery Process

Figure 3-2 details the network discovery process from the point of view of the OBC.

The process begins by clearing any previous knowledge of the network infrastructure. SpaceWire path routing is used for the network discovery process. Logical addressing is used for nominal RMAP TMTC operation as the logical address is protected by the RMAP header CRC. A stack (*Paths to Explore*) is used to keep track of unvisited network paths, this is first populated with the route from the OBC to the OBC R-ICU, typically a single hop over the Star-Dundee brick.

Once the algorithm arrives at a new SM, it reads the SpW PnP network information field TM. As part of this data is a R-ICU status code (so the discovery process can be halted in the case of R-ICU failure) and also the Node ID assigned by the MOASR consortium to each SM. This allows the SMs to be uniquely identified, keeping track of which SMs have been discovered is key for managing the loops in the network. The network information also includes the status of all SpaceWire interfaces connected to the R-ICU, any link that is active and running should be explored by the discovery process.

To explore links that may be inactive due to the remote end being powered down, the power output of active HOTDOCKs that have inactive links are enabled by the OBC. Some time is given for the remote SpaceWire and R-ICU to power up. Then, the networking information TM for the current SM is checked again to see if enabling these HOTDOCKs have brought any new SMs online. Active HOTDOCKs that do not power up new SMs can then be powered back down.

If the SM has already been visited, then its SM ID will be logged in the *Discovered SMs* array and the discovery can skip over any further exploration of this SM from this point. If it is not in this list, then it is added and its active links added to the *Paths to Explore* stack. As each link is explored, the link to SM ID relationship is logged.

By the end of this process a list of all SMs in the network will be produced. Each SM will have a list of its active links and the SM that is at the end of this link. This information can be compiled in a graph for analysis by the routing algorithm. Once discovery of the network topology is complete, a final step is to update the SM list with the capabilities of each node (payloads, types of HOTDOCK etc). This is done by reading the Component Information TM for each SM in the SM list.

The performance of the discovery process depends on if the SM is powered up or not. If the SM is already powered on, the dominating overhead is the time required to fetch the PnP TM from the interrogated SM to the OBC. This overhead is formed by:

1. The expected size of the PnP TM (max 32KB)
2. The expected throughput of the SpaceWire network (200Mbit nominal)
3. The expected time for the R-ICU to respond to the TM request (less than 1ms due to direct RMAP access via DMA).

Therefore, the discovery overhead for each node in the powered-on state is estimated to be less than 5ms. Given the small network in the MOSAR demonstrator, this allows the whole spacecraft to be discovered in under a second.

In the cold-start case, extra time is required at each SM to allow for each cPDU to stabilise power and for each R-ICU to boot up. This is expected to be bound to under 5-seconds per SM, but will be evaluated during the system testing phase in WP4.



Detailed Design Document

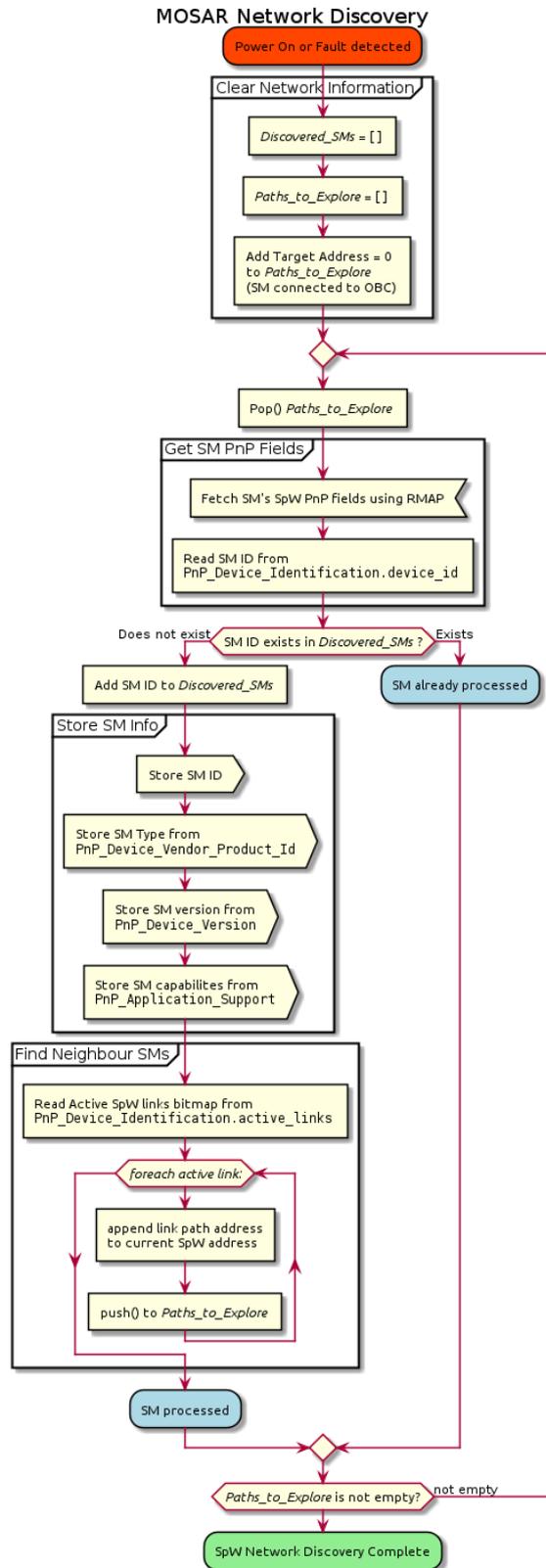


Figure 3-2 MOSAR Network Discovery (All SMs Powered On)



Detailed Design Document

3.1.3 Power Reconfiguration

In MOSAR, power distribution is achieved by the transfer of the electrical energy between the SM through the connected HOTDOCK interfaces. In order to configure the power distribution, each SM is equipped with a centralized power (and conditioning) distribution unit (cPDU), that provides the following functions:

- Local power distribution between the HOTDOCK interfaces of the SM
- DC/DC conversion from the main power bus to power the internal components of the SM/WM
- Intelligent powering sequence to reduce peak loads and provide advanced functionalities

The cPDU component and functions are described in section 5.8.

By controlling the cPDU and the distribution of power to the HOTDOCK interface points, the general spacecraft power layout can be configured. This provides the following features:

- Optimization of the system resources and power paths along the structure, for instance to create specific route for high demanding payloads
- Management of multiple power sources (generators, batteries), providing redundant configurations through the modules layout
- Automatic recovery from battery sources in case of failure of the main supply

3.2 Software Reconfiguration

3.2.1 General software reconfiguration strategy

The software reconfiguration approach is based on the definition of a finite number of possible configurations (*modes*) at design time. However, it does not appear necessary to restrict in advance the number of configuration changes (*mode transitions*) at that stage. All the possible configuration changes are thus allowed at design time.

This is obviously not the case at run time where the only software configuration changes that will be allowed, are those that are explicitly triggered either by a telecommand, a decision of the mission planner or, possibly in the future, a feedback coming from the network hardware reconfiguration plug and play feature. The triggering logics of software reconfiguration will be defined precisely when the various software functions will be integrated. Necessary software architecture trade-offs will be done at that stage if needed.

The configuration change feature is currently represented by a set of applicative software functions in the TASTE Interface View model. In the current prototype, it is represented by a `Configuration_Manager` component composed of an implementation of the mode automaton and an abstract representation of the configuration change logics to support mode transition guards in the future. With the current realization, each configuration sensitive component can enquire about the actual state of the system via the `configuration_status` interface to activate or deactivate itself.



Detailed Design Document

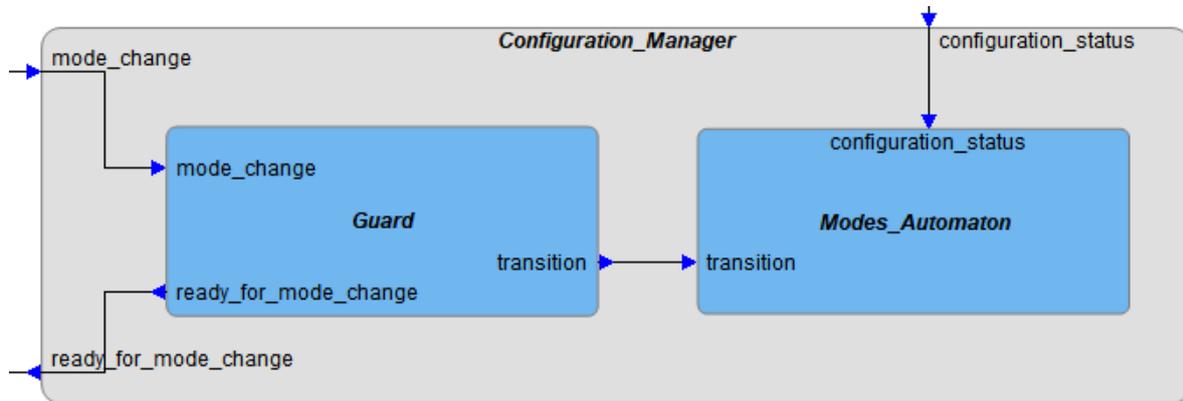


Figure 3-3: Software reconfiguration manager

The intent is that this Configuration_Manager becomes a generic reusable component that will have to be instantiated for each TASTE model that requires dynamic reconfiguration management. The actual source code for the Guard and Modes_Automaton should be automatically generated using the configuration related information that have been provided by the designer.

With the current prototype, only automatic code generation of the Modes_Automation component has been developed and integrated within the TASTE build process. Realization of the Guard component will become possible when the precise configuration change logics specification will become sufficiently precise.

In addition to this software architectural modelling approach, specific research work has been realised in the context of the project to introduce early verification capabilities on reconfigurable software architectural models.

The various options to manage software reconfiguration for the OBC were discussed in the Preliminary Design Document (D2.4). From these possible solutions, some prototyping work has been performed to enhance the TASTE toolchain, as part of the ESROCOS package. This work is described in the OG1-5 adaptations and extensions detailed design document [RD3]. This section contains the outcome of research work that has been realised in the context of the MOSAR project to study possible early verification capabilities on reconfigurable software architectural models. The integration of the proposed solutions into TASTE will then be also prototyped during the next phase of the project.



3.2.2 Introducing timing analysis of software reconfigurations

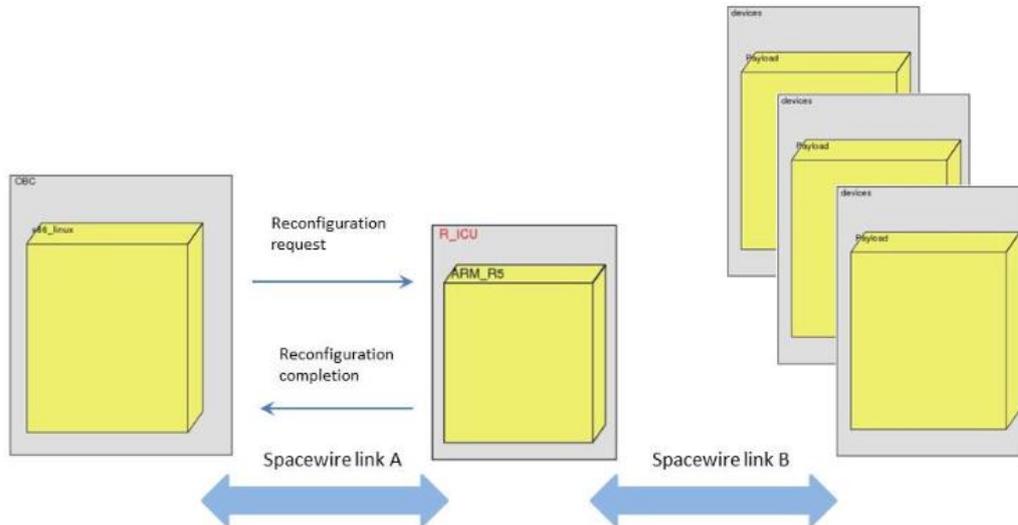


Figure 3-4: Considered MOSAR architecture for software reconfiguration

A MOSAR architecture contains critical components that must fulfil performance requirements such as meeting deadlines. If the software reconfiguration involves such critical components, having the ability to early evaluate the worst-case delay needed to perform the reconfiguration, provides a mean to predict when the components can be operational again.

Figure 3-1 shows a simplified view of a MOSAR architecture. As an example, reconfiguration requests may come from TM/TC on the OBC, generate messages on the SpaceWire link connecting the OBC to the R-ICU (link A in the figure), imply other messages between payloads and the R-ICU (link B), and finally, lead to messages from R-ICU to the OBC when the reconfiguration is completed. As we can see with this short scenario, computing a worst case on reconfiguration delay is complex due to the various components that are involved:

- We have various TASTE functions involved on the OBC that are competing for computing resources. We may have a similar issue on the R-ICU.
- Reconfiguration messages on the SpaceWire links are also competing with messages exchanged by the other MOSAR components.

To compute an estimation of the reconfiguration delay, we must then both analyse the timing of functions running on the nodes and also messages exchanged on the SpaceWire network. To make such schedulability analysis available to the MOSAR designer, we propose to extend the TASTE toolchain. This feature will provide a mean to TASTE designers to have an earlier estimation of the delay for reconfiguration when designing their MOSAR architecture with TASTE.

In the next sections, we present a method to simultaneously analyze SpaceWire messages scheduling with OBC functions scheduling and some means to integrate such analysis on the TASTE toolchain.

3.2.3 Overview of the schedulability analysis method

A MOSAR model built with TASTE is described by several views such as the Data View, the Interface View, the Deployment View, and the Concurrency View. The latter is an automatically generated compound AADL model that enables early verifications and source code generation. Figure 3-2 shows an overview of the approach we propose. To achieve schedulability analysis of a MOSAR TASTE model, the following steps must be applied:



Detailed Design Document

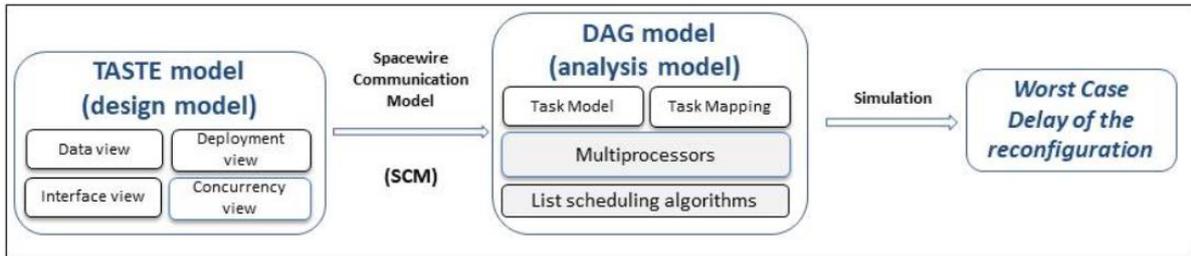


Figure 3-5: Worst-case reconfiguration delay computation process

- From the TASTE design model, we generate a DAG model (DAG stands for Directed Acyclic Graph) from which we can do the schedulability analysis. The produced DAG model specifies:
 - The communications on the SpaceWire links
 - The functions running on the TASTE nodes
- To transform both SpaceWire communications specification and TASTE functions specification in a DAG model, we propose a transformation algorithm called SCM (SCM for SpaceWire Communication Model). SCM is an extension of a transformation designed by the University of Brest for mesh communication devices such as Network-On-Chips [DRIDI 2019].
- With the DAG model produced by SCM, we can run classical schedulability analysis by scheduling simulations. In this context, scheduling simulations can be computed with list scheduling policies over the feasibility interval depending on frequency of TASTE cyclic functions. Such analysis lead to a proof of the timing property. Feasibility intervals are chosen according to [GOOSEENS 2016]. Such kind of schedulability analysis can be made with Cheddar [SINGHOFF 2009], an open-source scheduling analysis tool developed by the University of Brest and Ellidiss Technologies.
- Then, from this schedulability analysis, we can derive the worst-case reconfiguration delay.

3.2.4 Assessing schedulability with SCM

In the sequel, we explain how SCM will produce the DAG model from the TASTE model. First, we give the assumptions taken by SCM. Then, we specify rules specifying transformation of a TASTE model to a DAG model. A short example illustrates this algorithm.

3.2.4.1 Assumptions and notations

For the SpaceWire architecture, we assume the standard ECSS-E-ST-50-12C Rev.1 [ESA 2019]. Table 3-1 summarizes the assumptions related to the SpaceWire network considered by the SCM algorithm.

Properties	Values
Routing	Path or logical addressing
Switching mode	Wormhole
Arbitration policy	Fixed priority, Round Robin
Virtual channel	Virtual channel
Preemption level	Packet

Table 3-1: Assumptions for the SpaceWire network architecture

Communications between two TASTE functions on different TASTE nodes lead to several messages. In the sequel, we call a flow the set of messages exchanged between two TASTE functions. A flow i is noted R_i and is defined as follows: $R_i = \{O_i, P_i, D_i, NodeS_i, NodeD_i\}$, where:

- O_i is the release time of the messages composing the flow, i.e. the first time a message of the flow becomes ready to be transmitted.



Detailed Design Document

- P_i is the period of the message
- D_i is the deadline of the message i.e. the message must be available for the receiver before this time.
- $NodeSi$ the TASTE node running the transmitter function.
- $NodeDi$ is the node running the receiver function.

The DAG model is composed of a set of periodic tasks running on an abstract multiprocessor execution platform. Scheduling analysis of such models can be performed with list scheduling. List scheduling algorithms build list of tasks when assigning them their priorities. There are several means to determine the priorities of tasks such as Highest Level First, Longest Path and Longest Processing Time. Highest Level First with Estimated Time (HLFET), Modified Critical Path (MCP), Earliest Time First (ETF) and Dynamic Level Scheduling (DLS) which are examples of list scheduling algorithms [HAGRAS 2003]. Those algorithms can be applied on the analysis model we assume here.

A DAG is specified as follow. Let consider G , a DAG composed of n periodic tasks: $G = \{T_1, T_2, \dots, T_n\}$. Each task i , noted T_i , is defined as by $T_i = \{O_i, P_i, C_i, D_i, Node_i, E_i\}$, where:

- O_i is the first release time of the task i .
- P_i is the period of the task i .
- C_i specifies the capacity of the task i ; i.e. its Worst Case Execution Time.
- D_i is the task deadline to meet.
- $Node_i$ identifies the processing element running the task. A processing element models the computing resource on which tasks are run. The set of processing elements constitutes the abstract multiprocessor execution platform running the DAG to analyse.
- E_i introduces the precedence constraints of the task. For a given task i , the function E_i determines all the tasks j that receive messages from the task i

3.2.4.2 SpaceWire Communication Model (SCM)

Next, we detail the transformation rules composing the SCM algorithm. Assuming a TASTE model representing a MOSAR architecture, it will be transformed into a DAG by the following rules:

- Rule 1: hardware components
 - All SpaceWire router of the TASTE model will be removed in the analysis model.
 - All TASTE computing units are kept in the DAG analysis model and transformed as processing elements.
- Rule 2: links between hardware components
 - Each unidirectional link $e_{R_x R_y}$ in the SpaceWire network between two routers R_x and R_y of the TASTE model will be replaced in the DAG model by a new processing element named $PE_{R_x R_y}$.
 - Each unidirectional link $e_{PE_x R_y}$ in the SpaceWire network between a TASTE node and a router (respectively $e_{R_x PE_y}$ between a router and a TASTE node) of the TASTE model will be replaced in the analysis model by a new processing element $PE_{PE_x R_y}$ (respectively $PE_{R_x PE_y}$).
 - These new processing elements in the analysis model apply a scheduling algorithm compliant with the arbitration policy in the SpaceWire router.
- Rule 3: flows of control
 - Each thread of the TASTE concurrency view is replaced by a task in the analysis/DAG model.
- Rule 4: dependencies between flows of control
 - Each flow R_i of the TASTE model will be replaced by a set of nb tasks G_{R_i} , where $nb = nblink_i \times nbsize_i$ in the analysis model.
 - $nblink_i$ represents the number of links used by the flow R_i , and $nbsize_i$ represents the size of message of the flow R_i .



Detailed Design Document

$$G_{R_i} = \{ T_{R_i,1,1}, \dots, T_{R_i,a,b}, \dots, T_{R_i}, \text{nbsize}_i, \text{nblink}_i \}$$

- Rule 5: DAG task parameters
 - If the TASTE model contains couple of cyclic periodic functions T_{source} and $T_{\text{destination}}$ and if T_{source} sends messages of the flow R_i to $T_{\text{destination}}$ then applying SCM leads to the flow R_i transformed in the analysis model as a set of DAG tasks G_{R_i} .
 - Parameters of each DAG task $T_{R_i,a,b}$ of the task set G_{R_i} are computed as bellow.
 - For (a,b) in $[1, \text{size}_i] \times [1, \text{nblink}_i]$, $T_{R_i,a,b}$ is characterized by:
 - $O(T_{R_i,a,b}) = O(R_i)$
 - $D(T_{R_i,a,b}) = D(R_i)$
 - $P(T_{R_i,a,b}) = P(R_i)$
 - $\text{Node}(T_{R_i,a,b}) =$
 - $\text{PE}_{\text{PEsRs}},$ if $a=1$
 - $\text{PE}_{\text{RxRy}},$ if $1 < a < \text{nblink}_i$
 - $\text{PE}_{\text{RdPEd}},$ if $a = \text{nblink}_i$
 - $C(T_{R_i,a,b}) = \text{PD}_{\text{OnePacket/Onelink}}$, where $\text{PD}_{\text{OnePacket/Onelink}}$ is the time required to send one packet over only one link without considering the possible conflicts in the network.
 - $E(T_{R_i,a,b}) = \{ T_{R_i,a+1,b}, T_{R_i,a,b+1} \}$, if $a < \text{size}_i$ $b < \text{nblink}_i$
 - $T_{R_i,a+1,b}$ if $a < \text{size}_i$ and $b = \text{nblink}_i$
 - $T_{R_i,a,b+1}$ if $a = \text{size}_i$ and $b < \text{nblink}_i$
 - $T_{\text{destination}}$ if $a = \text{size}_i$ and $b = \text{nblink}_i$

We note here that:

- e_{RxRy} represents one of the physical links used by the flow R_i which is transformed to the processing element PE_{RxRy} in the analysis model.
- Some tasks of G_{R_i} have only one successor such as $T_{R_i,\text{size}_i,b}$, $T_{R_i,a,\text{nblink}_i}$ or $T_{R_i,\text{size}_i, \text{nblink}_i}$. The rest of the tasks have two successors.

3.2.4.3 Example

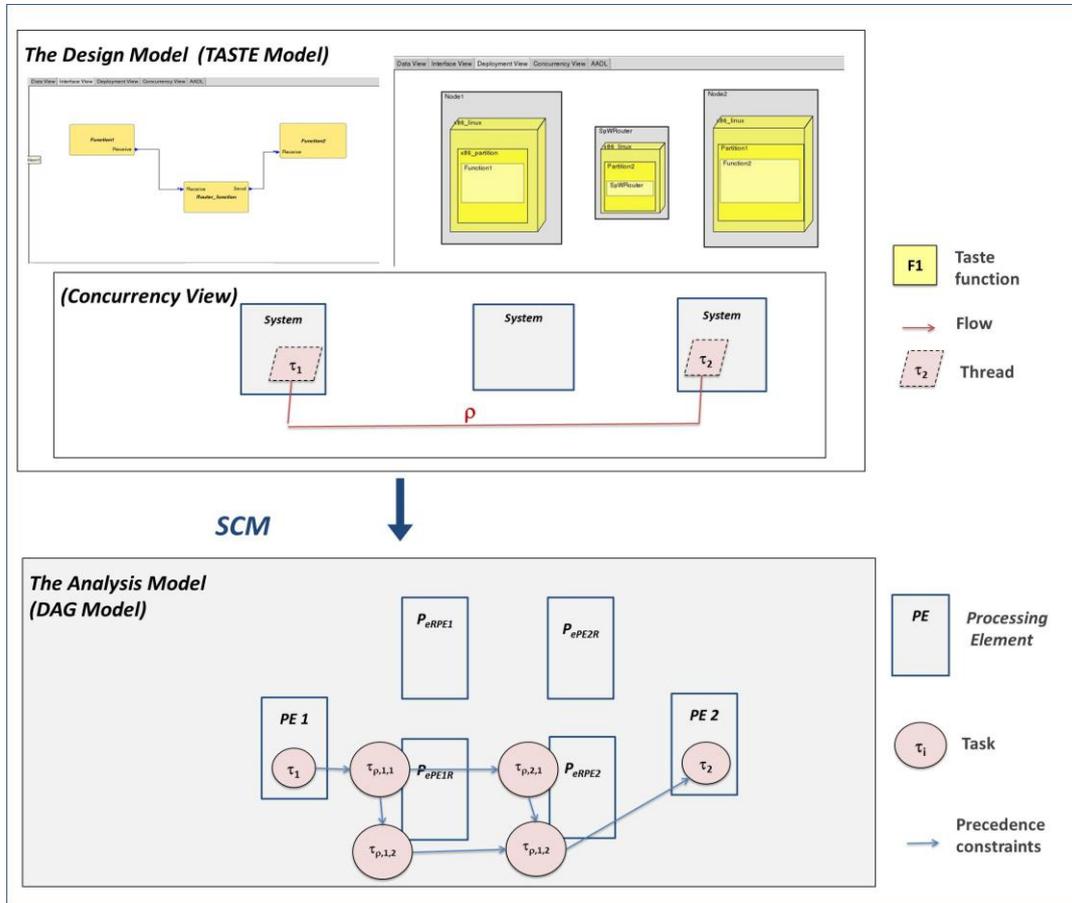


Figure 3-6: Example of applying SCM on a TASTE model

Figure 3-3 shows the TASTE design model (interface view, deployment view and concurrency view) and the corresponding analysis model (DAG model) generated with SCM. The purpose of this example is to explain the transformation rules of SCM. In this example, we consider two TASTE nodes (Node1 and Node2) connected by one router (SpWRouter) and several physical links. As shown in Figure 3-3, in each node, we have one TASTE function leading to one thread in the concurrency view. Figure 3-3 also shows the deployment view: Function1 is run on Node1 and Function2 on Node2. Function1 sends one flow to Function2 over the SpWRouter.

When applying SCM on this TASTE model, we get:

- Rule 1 + Rule 2: The router has been removed and the physical links are transformed in the analysis model to processing elements.
- Rule 3: Each cyclic/periodic function of the design model is replaced by a task in the analysis model.
- Rule 4 + Rule 5: R_1 uses 2 physical links. Thus, the flow R_1 has been transformed in the analysis model to a task set G_{R_1} of 4 tasks:

$$G_{R_1} = \{ T_{R_1,1,1}, T_{R_1,1,2}, T_{R_1,2,1}, T_{R_1,2,2} \}$$

$T_{R_1,1,1}$, $T_{R_1,1,2}$ run on the processing element PE_{PE1R} whereas $T_{R_1,2,1}$, $T_{R_1,2,2}$ run on the processing element PE_{RPE2} .



Detailed Design Document

The resulting DAG is composed of 6 tasks running on 4 processing elements. We consider a list scheduling algorithm for processing elements PE1 and PE2. We consider a fixed priority scheduling policy for the rest of the processing elements as the fixed priority policy algorithm is compliant with the arbitration in the SpaceWire network.

3.2.5 Integration into the TASTE toolchain

In this section, we explain our DAG model shown in the previous section will be handled by the TASTE toolchain. The timing analysis of a software reconfiguration by SCM will be implemented in the open source schedulability analysis tool Cheddar. Cheddar has already been integrated into the TASTE toolchain and can work on a generated TASTE Concurrency View. However, the Cheddar version included into TASTE is out of date and the initial prototyping will be based on AADLInspector, a commercial tool of Ellidiss that can fully operates with TASTE Concurrency Views through standard AADL files. Then, a proper update of Cheddar within the TASTE toolchain will be considered.

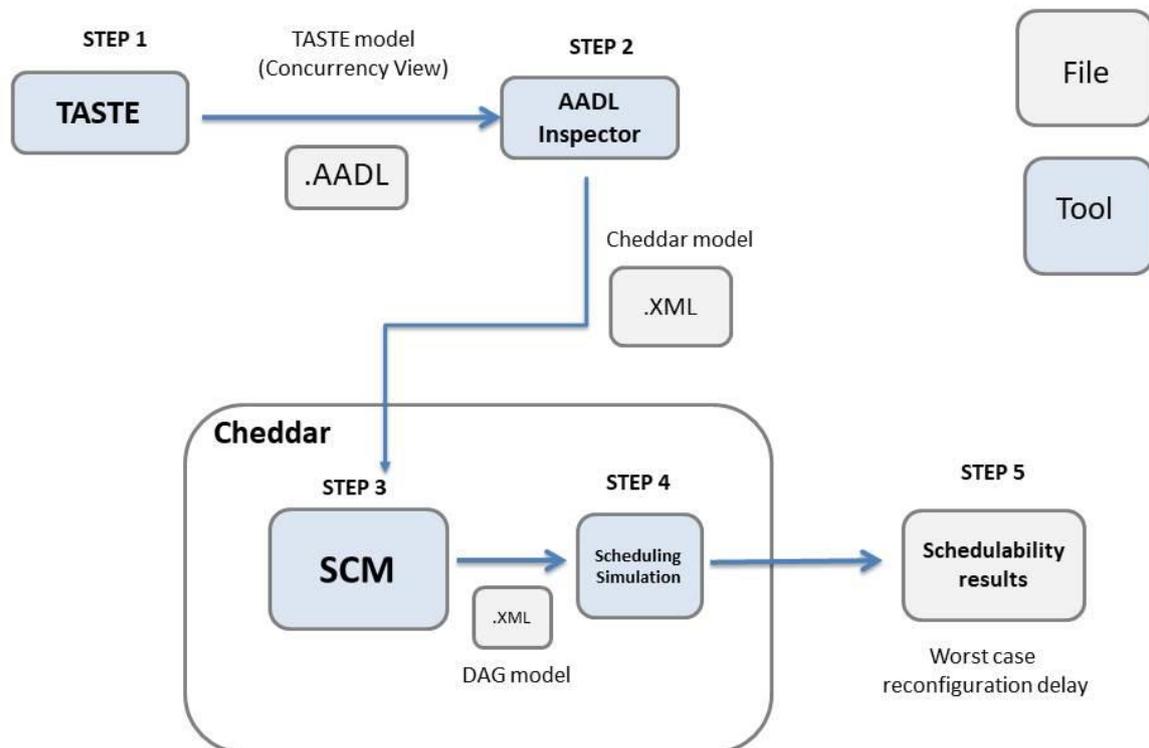


Figure 3-7: Integration of SCM into the TASTE toolchain

Figure 3-4 outlines how we expect to make early reconfiguration delay analysis available to TASTE users in a 5-step process:

- Step 1: When a MOSAR TASTE design is ready to be analysed, with TASTE, the AADL files modeling the concurrency view are generated.



Detailed Design Document

- Step 2: AADL files produced by TASTE can be directly read by AADLInspector, the AADL toolset of Ellidiss. Thanks to LMP, AADLInspector will produce Cheddar XML file for schedulability analysis.
- Step 3: In this step, SCM, which is implemented in Cheddar, produces the DAG model from the XML file printed by AADLInspector.
- Step 4: Finally, the DAG model, can be actually used to run schedulability, from which a worst-case reconfiguration delay can be computed.
- Step 5: The last step is to display the result to the TASTE user in a proper manner.

It is expected that this process will be run automatically, making it transparent for the TASTE user.

3.2.6 References

[GOOSEENS 2016] J. Goossens, E. Grolleau, L. Cucu-Grosjean. Periodicity of real-time schedules for dependent periodic tasks on identical multiprocessor platforms. *Journal of Real-Time Sys*, 52:808-832, 2016

[ESA 2019] ESA, ECSS-E-ST-50-12C Rev.1 Standard, SpaceWire – Links, nodes, routers and networks, 15 May 2019

[HAGRAS 2003] T. Hagra, J. Janecek, Static vs. dynamic list-scheduling performance comparison, *Acta Polytechnica* 43(6), 2003.

[SINGHOFF 2009] F. Singhoff, A. Plantec, P. Dissaux, J. Legrand. Investigating the usability of real-time scheduling theory with the Cheddar project. *Real-Time Systems*, 43(3), 259-295, 2009.

[DRIDI 2019] M Dridi, F Singhoff, S Rubini, JP Diguët. ECTM: A New Communication Model to Network-On-Chip Schedulability Analysis, Ada-Europe / Reliable Software conference conference, june 2019, Warsaw, Poland



4 System Operations

4.1 Ground Segment Operations

The robotic reconfiguration of the satellite envisaged in the MOSAR project is based on an operation plan previously computed on ground and verified by simulation. The scenario to be demonstrated is thus divided in two independent phases:

- Design and verification on ground: the desired configuration of the satellite is defined and validated, and a plan for the robotic reconfiguration is developed and verified in the simulator.
- On-orbit reconfiguration: the verified reconfiguration plan is uploaded to the on-board system and executed using the Walking Manipulator and the platform (or demonstrator) capabilities.

This section focuses on the first stage, while the second is addressed below in section 4.2.

The starting point for the planning of a reconfiguration mission is a set of available SMs. The definition of these SMs is previous to the start of the scenario demonstrated in MOSAR. In a future scenario where a reconfigurable spacecraft is in orbit and a servicing mission is being planned, it can be envisaged that the payload of the servicer spacecraft will be selected among a set of available APMs from satellite subsystem manufacturers, and APMs from specific clients. The characteristics and capabilities of the available APMs and ASMs must be known by the mission designer to perform the mission analysis and plan and validate the entire operation.

In the context of the MOSAR demonstrator, the definition of the SM characteristics and capabilities is a manual stage of configuring the different tools that support the demonstration. In order to ensure consistency among the different components of the system, the configuration is defined in a centralized database / configuration file, from which each software tool will extract its required information.

In addition to the knowledge about the SMs, some information on the characteristics of the SVC and CLT spacecraft, the WM and the environment must be also provided.

An initial list of the aspects that need to be considered is provided in Table 4-1 below. The table indicates the components that need each piece of information and the purpose served. It must be noted that the information is used not only by the components at the design and simulation stage, but also during the demonstrator operations.

Table 4-1: System characteristics relevant for configuration

Information	Element	Information users	Purpose
SM identifier	SM	All SW components	Unique identification of SMs
Capabilities and requirements: power, thermal, data routing and processing, etc.	SM	- Design Tool - Simulator	- Validation of the configuration. The Design Tool may check simple constraints, while the Simulator may be used to check the system behaviour in time - Simulation of the system behaviour in the steady state and during reconfiguration
Geometry of the SM (with different levels of fidelity)	SM	- Simulator - Robotic Arm SW (trajectory planning)	- Dynamic simulation - Visualization by the simulator



Detailed Design Document

Information	Element	Information users	Purpose
		- Vision Subsystem	- Computation of collision-free manipulator trajectories - Model matching
Physical parameters: mass, CoG, etc.	SM	- Simulator	- Dynamic simulation
Location and characteristics of the SIs: active, passive, mechanical, thermal, dummy	SM	- Simulator - Planner - Autonomy Agent	- Simulate the SI behaviour - Generate plans according to the constraints - Command the SIs according to their capabilities
SM placement restrictions	SM	- Design Tool - Planner	- Validate configuration - Plan according to constraints
CLT and SVC geometry (platform and SIs)	CLT/SVC	- Design Tool - Simulator - Robotic Arm SW (trajectory planning) - Planner - Autonomy Agent	- Available locations of the SMs - Simulation of the reconfiguration - Visualization - Computation of collision-free manipulator trajectories - Computation and execution of valid WM and SM relocation steps
CLT and SVC SI characteristics: active, passive, etc.	CLT/SVC	- Simulator - Planner - Autonomy Agent	- Simulate the SI behaviour - Generate plans according to the constraints - Command the SIs according to their capabilities
WM geometry	WM	- Simulator - Robotic Arm SW	- Simulation of the reconfiguration - Visualization - Computation of collision-free manipulator trajectories
WM dynamics	WM	- Simulator - WM Control SW	- Dynamic simulation of the robot - Control of the robot
Manipulation heuristics: reachability, constraints for operation in 1-g, etc.	WM	- Planner	- Generation of a reconfiguration plan compatible with the physical constraints of the system
Environmental parameters: orbit, illumination, visibility, etc.	Environment	- Simulator	- Simulation and visualization

Based on the description of the system and its environment, the design stage of the demonstration can start. The Figure 4-1 below presents an outline of the process, and the tools used by the mission analyst at each stage.



Detailed Design Document

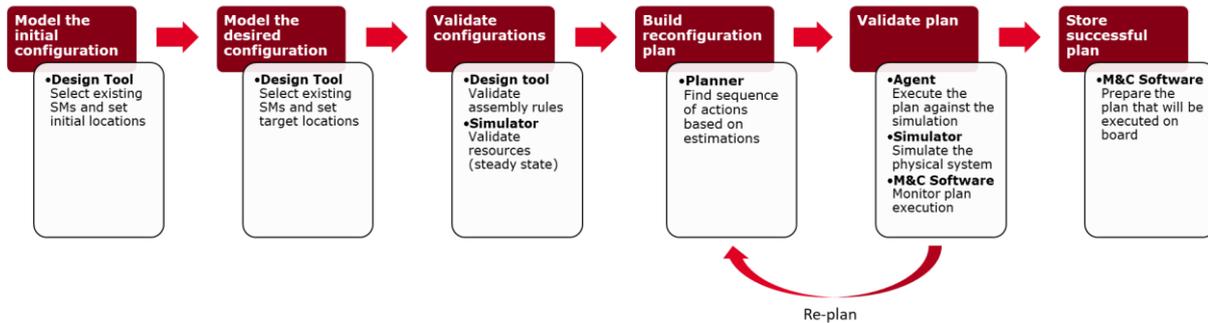


Figure 4-1: Design, simulation and planning stages

1. Model the initial configuration

The analyst uses the Design Tool (see section 5.1) to build a description of the initial configuration of the Client system (installed SMs, locations, connections, operational status, etc.), and the Servicer spacecraft (position of the new SMs and the WM at launch).

The chosen configuration is an input to the Simulator (see section 5.1) and the Planner (see section 5.2) tools.

2. Model the desired configuration

The analyst defines the desired configuration at the end of the scenario using the same process and tool. The result is an input to the Planner tool.

3. Validate configurations

The analyst checks that both initial and desired configurations are valid according to the system constraints.

The Design Tool can do an initial check of the configuration regarding the placement restrictions for each SM (if applicable) and the valid coupling of each pair of SIs (according to their type active, passive, etc.). Additional constraints may be implemented in the system, for instance regarding resource utilization (power, thermal, mass, data, etc.).

A second validation stage is performed using the Simulator. The operation of the system in the steady state (i.e., nominal operation before and after the reconfiguration of the client), using resource models of each SM, in order to assess if the target configuration performs as expected.

4. Build reconfiguration plan

Once the validity of the initial and desired configuration has been verified, the analyst uses the Planner to compute an initial robotic reconfiguration plan. The plan consists of a sequence of SM and WM relocation operations that lead from the initial to the desired configuration.

The Planner has the knowledge of the constraints for placing the SMs and for moving the WM (in particular, for demonstration in 1-g conditions). However, the Planner relies on estimation functions to calculate the cost in terms of duration or other aspects (e.g., energy usage). This means that the computed plan needs still to be validated using simulation.

5. Validate plan

The initial plan computed by the Planner is executed in closed loop with the Simulator. The Autonomy Agent (see section 5.2), identical to the one deployed on board, decomposes the plan in elementary steps that are executed by the Simulator.

The analyst uses the Monitoring and Control Software together with the visualization capability of the Simulator, to monitor the execution of the plan and investigate any issues that are detected.



Detailed Design Document

It is possible that the execution of the plan fails, as it has been obtained using a simplified model of the problem and does not reflect all the physical details modelled by the Simulator. In that case, the Planner executes in closed loop with the simulation in order to obtain an alternative plan. The process continues until a successful plan is found.

6. Store successful plan

The plan successfully tested is stored by the M&C Software for uploading to the on-board system during the actual demonstration. This plan is not necessarily the one initially computed by the Planner, but the result of possibly several replannings until the desired state is reached in the Simulator.

The scope of the plan is limited to the robotic reconfiguration of the system. Once the desired SM configuration is achieved, the on-board system needs to execute the actions to initialize each module and reach the operational state. This is out of the scope of the FES, and therefore of the computed plan.

4.2 Space Segment Operations

The Space Segment presents two main mode of operations, the CLT/SVC reconfiguration and the CLT nominal operations.

4.2.1 CLT/SVC Reconfiguration

The CLT/SVC reconfiguration operations consist of sequence of actions to go from an existing spacecraft configuration to a desired one, ready for operations. In the MOSAR demonstrator, the Servicer Spacecraft and its on-board computer manage the re-configuration operations. The Autonomy Agent running on the OBC-S executes the operation plan prepared by the MCC. As illustrated in Figure 4-2, it will trigger successive actions on the components at different level of the system architecture, with the Component manager in the OBC, the Component Control in the R-ICU/WM and the Component Controller at the lowest level. The specific role and scope of each level can be chosen according to the level of abstraction in the OBC and the autonomy of the components or SM (R-ICU). For instance, having generic command from the OBC component managers allows replacing the lowest level elements with different internal configuration or control.

The following section provides the list of the elementary actions that can be associated to form routines that will be often implemented during a plan. The full scenario executions consists then in a sequence of routines and elementary actions.

During re-configuration or spacecraft operations, the MCC can communicate with the OBC-S Telemetry and Telecommand service, to respectively monitor the plan operations and components parameters as well as control the execution of the Agent and the components.

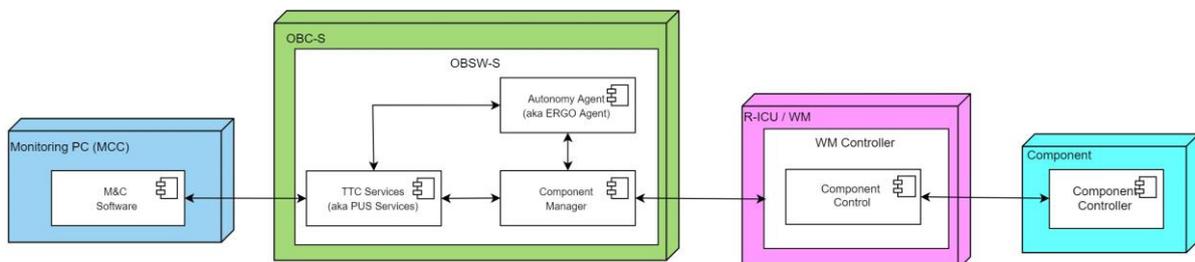


Figure 4-2: Re-configuration operation architecture



Detailed Design Document

4.2.2 CLT Nominal Operations

The CLT nominal operations consists mainly in the operations of the payloads of the spacecraft with TM/TC and data feedback (e.g. video images) to the OBC-C (on the CLT). Some (autonomous) re-configuration management can also be considered in the case of failure detection for the isolation of the faulty node and re-routing of power and data, without the need of intervention of the servicer spacecraft that could not be present anymore. As for above the TTC service allows to communicate with the MCC, for TM/TC of components and payloads, and data visualization / recording.

The transition between the two modes needs also to be managed (Client Management in Figure 2-10). Before starting the robotic re-configuration, the control/authority needs to be switched from the OBC-C to the OBC-S. After re-configuration, the OBSW-S should request the OBSW-C to reconfigure for the new architecture of the system and handle back control to the OBC-C.

Figure 4-2 provides a generic description of the software architecture, associated with a specific component of the system (subsystem, payload). The Component Control software starts automatically when the R-ICU is powered, initializing the corresponding Component Controller, initializing the SpW communication with the OBC (discovery, table routing) and going in *Idle* mode. In the *Idle* state, the payload is not operated and it allows to safely power off the SM. It periodically check if a TC is received from the Component Manager of the OBC, by polling the corresponding memory address (RMAP protocol). It periodically writes TM parameters and status to memory mapped to RMAP.

The PUS Console on the MCC provides an interface with user command to control the physical components, and to display the payload/component parameters. The TTC service running on the OBC-C will trigger the Management software to relay the received commands from the MCC (PolyORB-HI) to the Control software (R-ICU).

When the Control software receives TC for updating the subsystem, it will communicate with the local Controller (e.g. through CAN) that will manage the physical changes on the setup. The Management component polls the TM data of the Control software via RMAP. They are then published to the MMC through PolyORB-HI protocol, in order to be displayed on the Console.

4.2.3 Elementary Actions

This section defines and describes the list of elementary actions that will be recurrent in the re-configuration operations of the spacecraft. They mainly cover the operations of the WM, the connection of the HOTDOCK standard interface, the control of power transfer and the configuration of data routing.

The analysis of these actions, as well as the routines support the definition and implementation of the OBC software, as well ensuring that the hardware components full-fill the required functions to enable the operations.

4.2.3.1 Walking Manipulator Transfer Motion

The WM transfer motion is the action to move the WM free end-effector from its initial position to a target position in joint space. It is typically used for coarse collision-free motion of the WM during the different operations. As illustrated in Figure 4-3, the first step consists to configure the WM controller in Position mode, which is then replicated to the configuration of the joint drivers. Then the joint transfer trajectory is sent by RMAP command to the WM controller that will interpolate the sequence of points and define the successive joint position set-point for the joint drivers. Based on the feedback of the joint sensors TM, the WM controller confirms to the OBC the reaching of the goal position.



Detailed Design Document

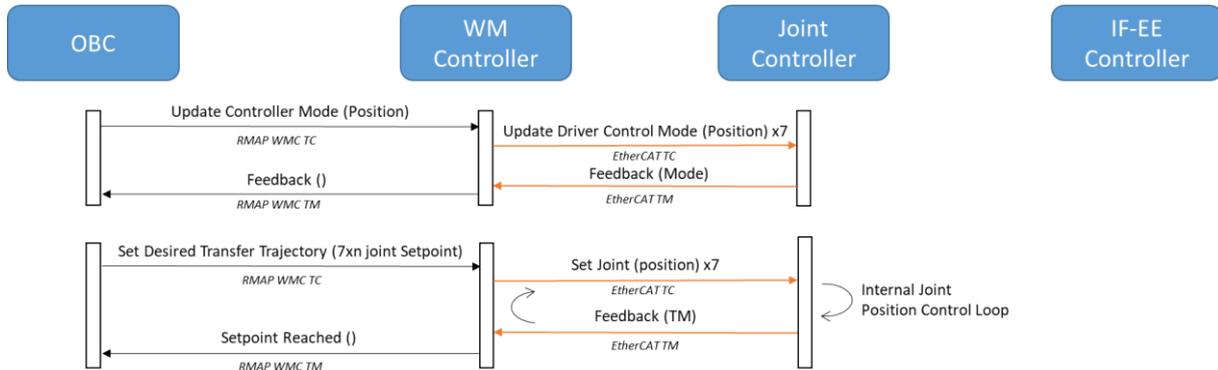


Figure 4-3: WM transfer Motion Sequence

4.2.3.2 Walking Manipulator Approach Motion

The WM approach motion is the action to perform the final approach of the WM end-effector to reach the position ready for the connection of the end-effector SI to the payload or satellite bus (also equipped with a standard interface). A similar motion is also applicable when a spacecraft module is manipulated by the WM to align with another module and/or the satellite bus.

The motion is based on Cartesian trajectory in order to ensure, as much as possible, the angle of approach inside the aperture of the HOTDOCK interface (+/- 60 degrees). In order to manage wrong initial alignment, an impedance control is applied to enable the guidance through the form fits of the SI, from the measurement of the contact forces (at joint level).

As illustrated in Figure 4-4, the first step consists to configure the WM controller in Impedance mode, which is then replicated to the configuration of the joint drivers. Then the Cartesian trajectory is sent by RMAP command to the WM controller that will performs kinematic conversions and dynamic computation to derive the joint current set-points for the joint drivers. Based on the feedback of the joint sensors TM (position, torque), the WM controller will iterate and update the set-point until the SI TM confirms the good alignment of them, ready for latching/connection. This information is confirmed to the OBC through the RMAP TM (either from the WM end-effector SI or the SM SI), which then sends the command to stop the process. This approach is more generic, However, we can also envisage to have a local validation of the alignment in the WM itself. In order to ensure contacts of surfaces, it is also possible to use the estimated contact force (from the joints torque sensors). There is however a margin of 2-3 mm that can be managed by HOTDOCK.



Detailed Design Document

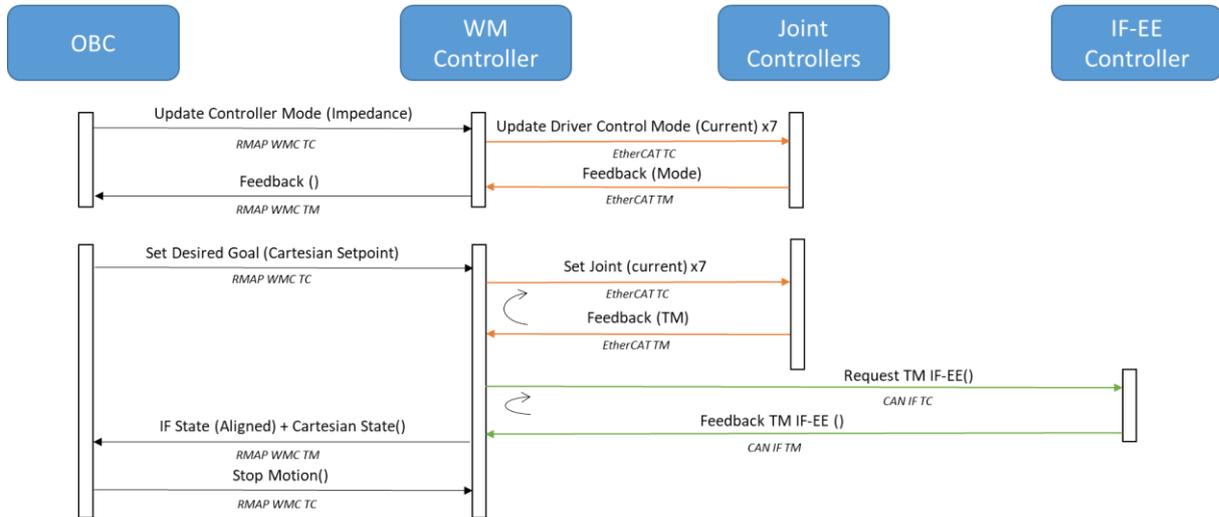


Figure 4-4: WM Approach Motion Sequence

4.2.3.3 Standard Interface Connection

The SI connection corresponds to the operation of latching and connection of the SI connector plate (data and power transfer interface). This operation, basically simple for the SI itself, requires however the configuration of the WM in impedance mode, such that it can adapt to the remaining adjustment motions during the connection process.

Figure 4-5 illustrates the typical sequence of actions. After the configuration of the WM in impedance mode, propagated to the joint drives configuration, a check is performed about the status of the alignment (which can be continuous during the process of connection). Once confirmed the OBC commands the respective SI to change its state to latch (only mechanical) or connection state (with also data and power interface). As above the commanded SI can be the WM EE or one of the SM or satellite bus.

The operation of connection also includes intermediated steps, locally managed by the HOTDOCK controller. This includes the detection of the orientation in order to configure the LVDS switch on the data interface connector (to ensure SpW communication with the 90deg symmetry), followed by the latching and connection steps.

As similar (inversed) sequence can be followed for the disconnection process.

4.2.3.4 Power Switch ON-OFF

The control of the power switches allows to enable or disable the transfer of power between the components. They can be either embedded in the HOTDCOK SI, for the WM EE, or in the cPDU component for the SM. They are typically controlled from the OBC, through RMAP TC that are translated in CAN TC. Voltage and current sensors on the active SI and cPDU allow to monitor and validate the operation of the switches.

The simple power switch command is also associated with local intelligence in the components, such that, when powered-on, the internal elements are activated in a controlled way, to reduce peak current and manage correctly the start-up sequence. This also allows to control the power-off sequence.



Detailed Design Document

4.2.3.5 R-ICU Data Routing Configuration

The routing tables of the R-ICUs can be updated by the OBC using the "SpW Routing Table Set Entry" telecommand. This sets the routing direction for a single logical address and is used during construction and WM travel operations to build and update the SpaceWire network topology.

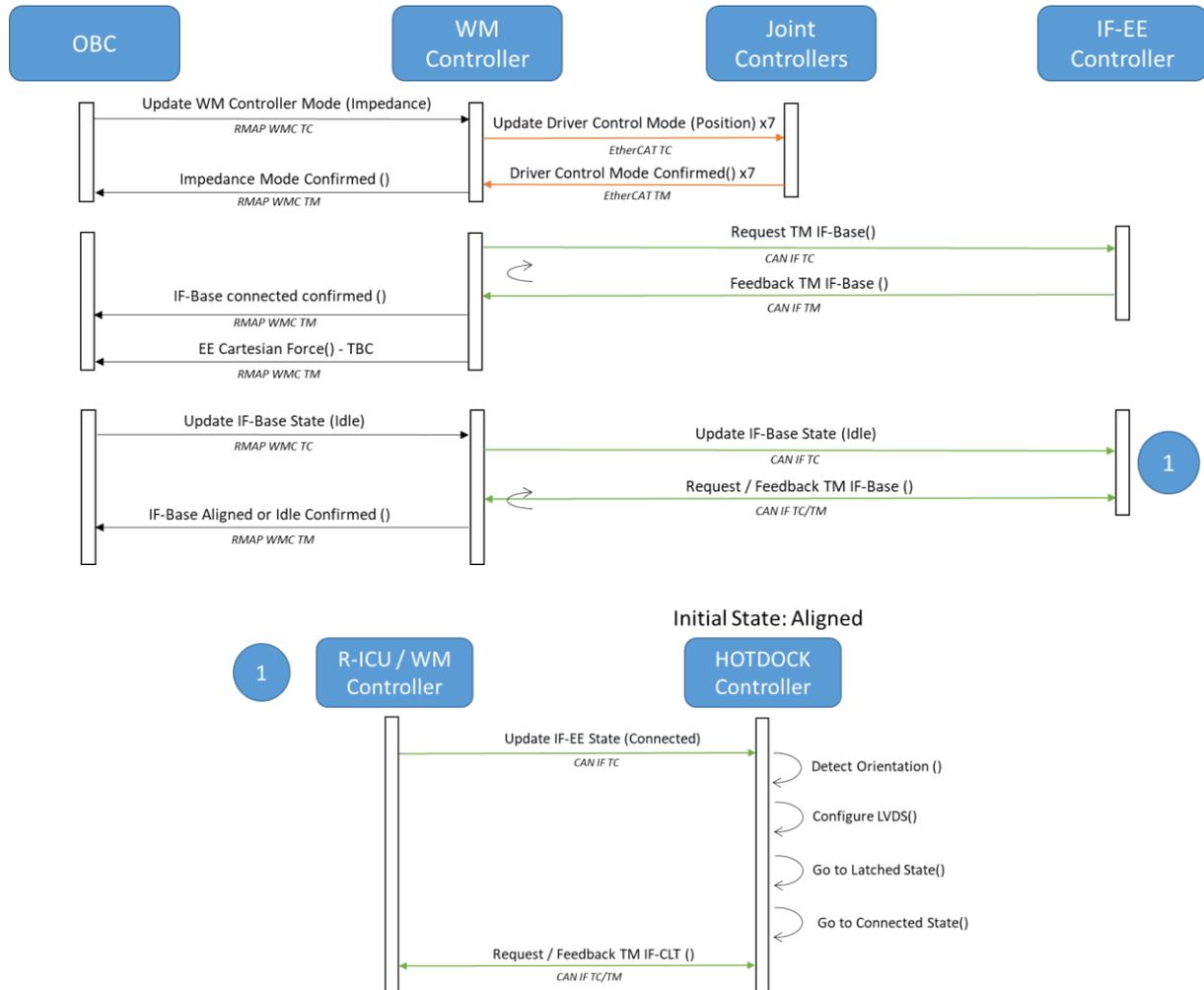


Figure 4-5: SI Connection Sequence

It can be triggered by a set of operation sequences, as part of the re-configuration process, or also through the automatic node discovery, when one module is activated or deactivated, in order to keep a valid network topology configuration.

4.2.4 Routines Sequences

This section illustrates the main routines of operations that will be frequently implemented during the re-configuration process. The purpose is to highlight the detailed sequence of actions and the required commands and feedback to support the development of the components and software involved in the process.



Detailed Design Document

4.2.5 System Start-Up – Initial Configuration

The following table provides an example of actions to perform a system start-up in order to reach a nominal configuration of the system, compatible with the further re-configuration operations, as described in the next sections.

The provided example corresponds mainly to the initial case, when the SVC is docked to the CLT, with the CLT bus equipped with SM1 and SM2, and the SVC bringing the WM, with its base connected to the SVC-SI-W port.

The following hypothesis are considered:

- The current configuration is known and the script of operations sequence is available (list of actions to reach the nominal spacecraft configuration after the initial power switch ON).
- The system is fully controlled and powered from the SVC, connected to the MCC. In this example the SM1 OBC-C (spacecraft OBC) is not involved

Step	Action (Autonomous, Operator or Manager)	Effects	Expected State / Required confirmation
1	The Operator switches ON the Main Power Supply.	1.1 OBC-S switches ON (through SVC cPDU 24V output) 1.1.1 SVC SpW Bridge switches ON 1.2 SVC cPDU controller switches ON 1.2 SVC R-ICU switches ON (through SVC cPDU 12V output) 1.4 CLT R-ICU switches ON (through CLT cPDU-1 12V output) <i>Note: there is a direct transmission of the 48V bus along SVC and CLT cPDU's</i> 1.5 CLT cPDU-1 and cPDU-2 controllers switch ON 1.6 CLT-SI-C controller switches ON (through CLT cPDU-1 24V output)	R1.1 Confirmation of communication from OBC-S with MCC through PUS service R1.2 Confirmation of communication between OBC-S and SVC R-ICU through RMAP TM R1.3 Confirmation of communication between SVC R-ICU and SVC cPDU through CAN TM. Status updated to OBC-S through RMAP TM <i>Note: During this action, there is not yet confirmation of communication to OBC-S, as the SVC R-ICU is not yet configured to address the CLT R-ICU (see after)</i> R1.5 Confirmation of communication between CLT R-ICU and CLT cPDU-1 and cPDU-2 through CAN TM. <i>Note: as above, not yet confirmation to OBC-S</i> R1.6 Confirmation of communication between CLT R-ICU and CLT-SI-C through CAN TM. <i>Note: as above, not yet confirmation to OBC-S</i>
2	The Operator initiates the start-up sequence, based on the knowledge of the spacecraft and servicer setup, with command from MCC to OBC-S	2.1 OBC-S executes the sequence (see next actions)	R2.1 MCC get confirmation of the execution of the sequence through PUS service.
3	OBC-S configures the SVC R-ICU and routing table to get access to the CLT R-ICU, through RMAP TC	3.1 The SVC R-ICU and routing table is configured. The OBC-S has now access to the CLT R-ICU, through RMAP	R3.1 Confirmation of status of CLT R-ICU to OBC-S, through RMAP TM.
4	OBC-S configures the CLT R-ICU and routing table, through RMAP TC	4.1 The CLT R-ICU is configured and can provides TM to OBC-S	R1.4-R4.1 Confirmation of communication between CLT R-ICU and OBC-S, through RMAP TM. R1.5-R1.6 Confirmation status CLT cPDUs and SI to OBC-S, through RMAP.
5	OBC-S switches ON CLT-cPDU power relay to power ON SM1, through RMAP TC	5.1 SM1 cPDU controller switches ON	



Detailed Design Document

		<p>5.2 OBC-C switches ON (through SM1 cPDU 24V output)</p> <p>5.2.1 SM1 SpW Bridge switches ON</p> <p>5.3 SM1 R-ICU switches ON (through SM1 cPDU 12V output)</p> <p>5.4 SM1 Active SIs controllers switches ON (through SM1 cPDU 24V output)</p>	<p>R5.2 Confirmation of communication from OBC-S on MCC through PUS service</p> <p>R5.3 Confirmation of communication between SM1 R-ICU and OBC-S, through RMAP TM.</p> <p><i>Note: The communication is directly possible as the CLT R-ICU routing has already been configured in the previous step.</i></p> <p>R5.1 Confirmation of communication between SM1 R-ICU and SM1 cPDU through CAN TM.</p> <p>Status updated to OBC-S through RMAP TM</p> <p>R5.4 Confirmation of communication between SM1 R-ICU and SM1-SI-A and SM1-SI-C through CAN TM.</p> <p>Status updated to OBC-S through RMAP TM</p>
6	OBC-S configures the SM1 R-ICU and routing table, through RMAP TC	6.1 The SM1 R-ICU is ready to communicate and route signal with other connected R-ICU.	R6.1 Confirmation status SM R-ICU to OBC-S, through RMAP TM
7	OBC-S switches on successively selected cPDU power relay to power ON other SM already in place.	Same effect than in Step 5 for the corresponding SM	Same confirmation to OBC-S than in Step 5 for the corresponding SM
8	OBC-S switches ON SVC-cPDU power relay to power ON the WM, which base is connected on the SVC-SI-W	8.1 WM OBC switches on	R8.1 Confirmation of communication between the WM OBC and OBC-s through RMAP TM
		8.1.1 WM SpW Bridge switches ON	
		8.2 WM SI Controllers switch ON	R8.2 Confirmation of communication between WM OBC and the two WM SIs through CAN TM.
			Status updated to OBC-S through RMAP TM
9			R2.1: The OBC-s confirms to MCC the success of the system start-up and configuration, through PUS Service

After these operations, the system is ready to perform the re-configuration operations, as described in the following sections.

As mentioned above, the set of operations were preliminary described through the sequence script, that has been applied by the OBC-S. It is however also envisageable to consider the use of self-reconfiguration capability in the case there would be a problem with the communication of one of the R-ICU (RMAP Time out) or if the configuration is not preliminary known. The advantage of the pre-defined plan is to allow to control the power and data routing along the system, at start-up.

This example considers that the configuration of the R-ICU and its routing table are done by specific command from the OBC-S through RMAP. It is also envisageable to predefined these information directly in the R-ICU, with still the possibility for the OBC-S to over-write it in case of need.

The same strategy can be applied for the nominal operations of the spacecraft, with the difference that the actions will be triggered by the SM1 OBC-C.

4.2.6 WM Re-Localization

The following table provide an example of successive operations for the re-localization of the WM, with a change of base. This enable the WM to move along the buses structures.

The following hypotheses are considered:



Detailed Design Document

- The WM is initially connected with its SI-A to the SVC-SI-W. The SI-B is free (this is not a strict requirement for the initial state. If SI-B is connected, initial operations can be performed to reach this point).
- The goal is to connect the WM through the SI-B to the SVC-SI-X
- The WM is initially powered ON (see previous sequence) but in disabled configuration (e.g. brakes activated)
- The R-ICUs tables are configured such that the OBC-S can communicate with the WMC

The sequence describes the main operations, effect and confirmation from and to the OBC-S, which manages the WM. Details about these operations and communications between the components are described in section 4.2.3.

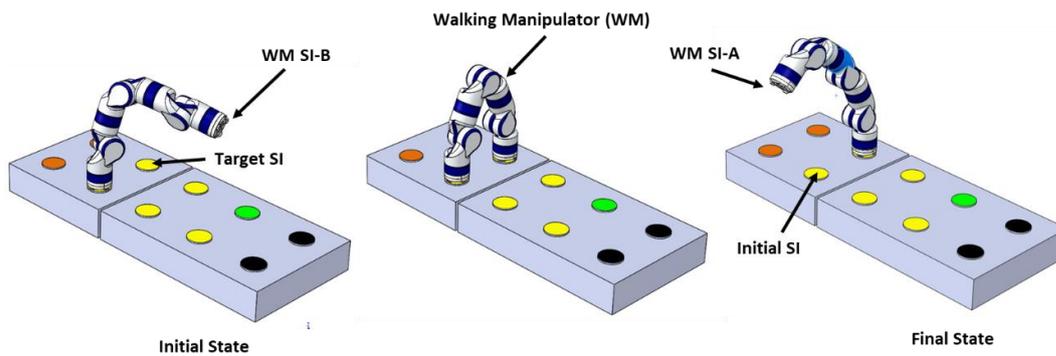


Figure 4-6: Routine 1 – WM re-localization between initial SI (SI-W) and Target SI (SI-X)

Step	Action (Autonomous, Operator or Manager)	Effects	Expected State / Required confirmation
1	OBC-S enables the WM for control and motion, through RMAP TC	1.1 WM joint drivers are enabled and configured to keep current position. Joint brakes are disabled	R1.1-1.2: Confirmation of the enabling of the WM to the OBC-S, through RMAP TM
2	OBC-S commands the WM for a transfer motion. See operation description in 4.2.3.1	2.1 The WM free extremity is moved from the current position to the target position for final approach	R2.1 Confirmation of set-point reached from the WMC to the OBC-S, through RMAP TM
3	OBC-S commands the WM for an approach motion. See operation description in 4.2.3.2	3.1 The WM SI-B is aligned and put in contact with SVC-SI-X	R3.1 Confirmation of the alignment of the SI (proximity sensors) and contact force, from the WMC to the OBC-S, through RMAP TM
4	OBC-S commands the connection of the SI-B to the SCV-SI-X See operation description in 4.2.3.3	4.1 The WM SI-B is connected to the SVC-SI-X, to allow power and data transfer	R4.1 Confirmation of the connection state of the SI-B, from the WBC to the OBC-S, through RMAP TM
5	OBC-S switches ON SVC-cPDU power relay connected to the SVC-SI-X, through RMAP TC	5.1 The WM can get power from the SVC-SI-X	R5.1 Confirmation of voltage and current transfer through the SI-B, from the OBC to the OBC-S, through RMAP
6	OBC-S configures the SVC R-ICU and routing table, through RMAP TC	6.1 The WM controller SpW communication is transferred to the SI-B interface	R6.1 Confirmation status WMC to OBC-S, through RMAP TM
7	OBC-S switches OFF SVC-cPDU power relay, connected to the SVC-SI-W, through RMAP TC (initial base)	7.1 The WM can not get anymore power from the SVC-SI-W. Only from SVC-SI-X	R7.1 Confirmation status WMC to OBC-S, through RMAP TM
8	OBC-S commands the disconnection of the SI-A from the SCV-SI-W See operation description in 4.2.3.3	8.1 The WM SI-A is disconnected	R8.1 Confirmation of the disconnection state of the SI-A, from the WBC to the OBC-S, through RMAP TM
9	OBC-S commands the WM for an approach motion. See operation description in 4.2.3.2	9.1 The WM SI-A is removed from the connection position with a controlled trajectory	R9.1 Confirmation of the removal of the SI (proximity sensors) and set-point reached,



Detailed Design Document

			from the WMC to the OBC-S, through RMAP TM
10	OBC-S commands the WM for a transfer motion. See operation description in 4.2.3.1	10.1 The WM free extremity is moved from the current position to the next target position	R10.1 Confirmation of set-point reached from the WMC to the OBC-S, through RMAP TM
11	OBC-S disables the WM for control and motion, through RMAP TC	11.1 WM joint drivers are disabled and joint brakes are enabled	R11.1: Confirmation of the disabling of the WM to the OBC-S, through RMAP TM
12	OBC-S switches OFF SVC-cPDU power relay, connected to the SVC-SI-X, through RMAP TC	12.1 WM is powered off	Confirmation of the absence of communication through RMAP

Steps 11 and 12 are optional as function of next set of operations.

4.2.7 SM Re-Localization

The following table provide an example of successive operations for the re-localization of an SM along the structure, commanded from the SVC OBC-S. With successive iteration of this routine, this allows the transfer of the SM between the SVC and CLT, or the re-localization of a module on the structure or other module.

The following initial conditions need to be full-filled to start the sequence:

- SM powered-off and is latched through the SM-SI -1 (SI initially connected to another SM or structure)
- The WM SI-A is connected to another SI on the spacecraft, with power and data transmission
- The WM-SI-B (end-effector) will be used to manipulate the SM through the SM-SI-2.
- The WM is powered on, is able to communicate with the OBC-S by SpW and is ready for operations
- The position of the WM and SM allow performing the desired trajectory between the initial and final position of the SM.

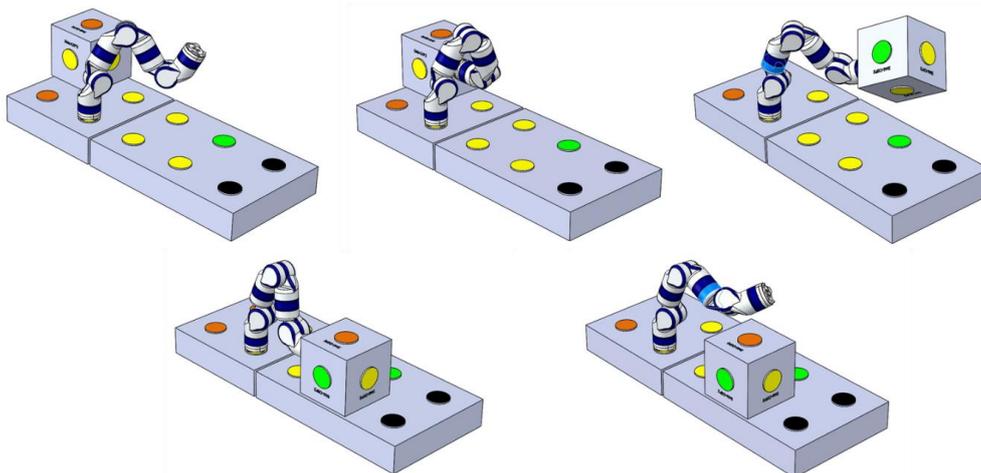


Figure 4-7: Routine 2 – SM re-localization



Detailed Design Document

Step	Action (Autonomous, Operator or Manager)	Effects	Expected State / Required confirmation
1	OBC-S enables the WM for control and motion, through RMAP TC	1.1 WM joint drivers are enabled and configured to keep current position. Joint brakes are disabled	R1.1-1.2: Confirmation of the enabling of the WM to the OBC-S, through RMAP TM
2	OBC-S commands the WM for a transfer motion. See operation description in 4.2.3.1	2.1 The WM free extremity is moved from the current position to the target position for final approach (in the vicinity of the SM)	R2.1 Confirmation of set-point reached from the WMC to the OBC-S, through RMAP TM
3	OBC-S commands the WM for an approach motion. See operation description in 4.2.3.2	3.1 The WM SI-B is aligned and put in contact with SM-SI-2	R3.1 Confirmation of the alignment of the SI (proximity sensors) and contact force, from the WMC to the OBC-S, through RMAP TM
4	OBC-S commands the connection of the SI-B to the SM-SI-2 See operation description in 4.2.3.3	4.1 The WM SI-B is connected to the SM-SI-2, to allow power and data transfer to the module	R4.1 Confirmation of the connection state of the SI-B, from the WBC to the OBC-S, through RMAP TM
5	OBC-S switches ON SI-B power relay, through RMAP TC	5.1 The WM transfers power to the SM, which powers-on. 5.2 SM cPDU controller switches ON 5.3 SM R-ICU switches ON (through SM cPDU 12V output) 5.3 SM-SI-1 controller switches ON (through SM cPDU 24V output)	R5.1 Confirmation of voltage and current transfer through the SI-B, from the OBC to the OBC-S, through RMAP <i>Note: During this action, there is not yet confirmation of communication to OBC-S, as the SVC R-ICU is not yet configured to address the SM R-ICU (see after)</i> 5.2 Confirmation of communication between SM R-ICU and SM cPDU through CAN TM. 5.3 Confirmation of communication between SM R-ICU and SM cPDU through CAN TM. <i>Note: as above, not yet confirmation to OBC-S</i>
6	OBC-S configures the intermediated R-ICU routing tables to get access to the SM R-ICU, through RMAP TC (the node discovery process can also be used)	6.1 The SVC R-ICU and routing table is configured (including intermediated R-ICU). The OBC-S has now access to the SM R-ICU, through RMAP	R6.1 Confirmation of communication between SM R-ICU and OBC-S, through RMAP TM. R5.2-R5.3 Confirmation status SM cPDUs and SI to OBC-S, through RMAP.
7	OBC-S commands the disconnection of the SM-SI-1 from the satellite bus See operation description in 4.2.3.3	7.1 The SM-SI-1 is disconnected	R7.1 Confirmation of the disconnection state of the SM-SI-1, from the WBC to the OBC-S, through RMAP TM
8	OBC-S commands the WM for an approach motion. See operation description in 4.2.3.2	8.1 The SM is removed from the satellite bus with a controlled trajectory	R8.1 Confirmation of the removal of the SI (proximity sensors) and set-point reached, from the WMC to the OBC-S, through RMAP TM
9	OBC-S commands the WM for a transfer motion. See operation description in 4.2.3.1	9.1 The SM is moved from the current position to the next target position (ready for final approach)	R9.1 Confirmation of set-point reached from the WMC to the OBC-S, through RMAP TM
10	OBC-S commands the WM for an approach motion. See operation description in 4.2.3.2	10.1 The SM-SI-1 is aligned and put in contact with the CLT-SI-C <i>Note: The SM SI to connect could be different from the previous one</i>	R10.1 Confirmation of the alignment of the SI (proximity sensors) and contact force, from the SM R-ICU to the OBC-S, through RMAP TM
11	OBC-S commands the connection of the SM-SI-1 to the CLT-SI-C See operation description in 4.2.3.3	11.1 The SM-SI-1 is connected to the CLT-SI-C	R11.1 Confirmation of the connection state of the SM-SI-1, from the SM R-ICU to the OBC-S, through RMAP TM
12	OBC-S switches OFF SI-B power relay, through RMAP TC	12.1 The SM is not anymore powered off. <i>Note: eventually, the SM could be powered first through the CLT interface if we want to keep it powered after the disconnection</i>	R12.1 Confirmation of current TM on WM-SI-B, from WBC through RMAP TM Confirmation of the absence of communication with the SM through RMAP



Detailed Design Document

13	OBC-S commands the disconnection of the WM-SI-B from the SM See operation description in 4.2.3.3	8.1 The WM-SI-B is disconnected	R8.1 Confirmation of the disconnection state of the WM-SI-B, from the WBC to the OBC-S, through RMAP TM
14	OBC-S commands the WM for an approach motion. See operation description in 4.2.3.2	9.1 The WM-SI-B is removed from the connection position with a controlled trajectory	R9.1 Confirmation of the removal of the SI (proximity sensors) and set-point reached, from the WMC to the OBC-S, through RMAP TM
15	OBC-S commands the WM for a transfer motion. See operation description in 4.2.3.1	10.1 The WM free extremity is moved from the current position to the next target position	R10.1 Confirmation of set-point reached from the WMC to the OBC-S, through RMAP TM
16	OBC-S disables the WM for control and motion, through RMAP TC	11.1 WM joint drivers are disabled and joint brakes are enabled	R11.1: Confirmation of the disabling of the WM to the OBC-S, through RMAP TM
17	OBC-S switches OFF SVC-cPDU power relay, connected to the SVC-SI-X, through RMAP TC	12.1 WM is powered off	Confirmation of the absence of communication through RMAP

This sequence is based on the assumption that the SM is initially connected to only one SI and that the SM SI are Active versions. The sequence can be adapted if multiple disconnections/connections are required and/or Active SI are not localized on the SM itself.

The sequence doesn't include the required configurations of the SM (e.g table routing), after the relocalization, and other components to make it ready for nominal operations with the CLT OBC-C.



5 Components Detailed Design

5.1 Design and Simulation Tool

The MOSAR Functional Engineering Simulator (FES, simulator) is built in Modelica to simulate the MOSAR scenarios. The focus of the simulator is the walking manipulator (WM, robot), which is used to build up a modular satellite from different modules on top of the satellites. The simulator assumes a docked state between the servicer and client satellite.

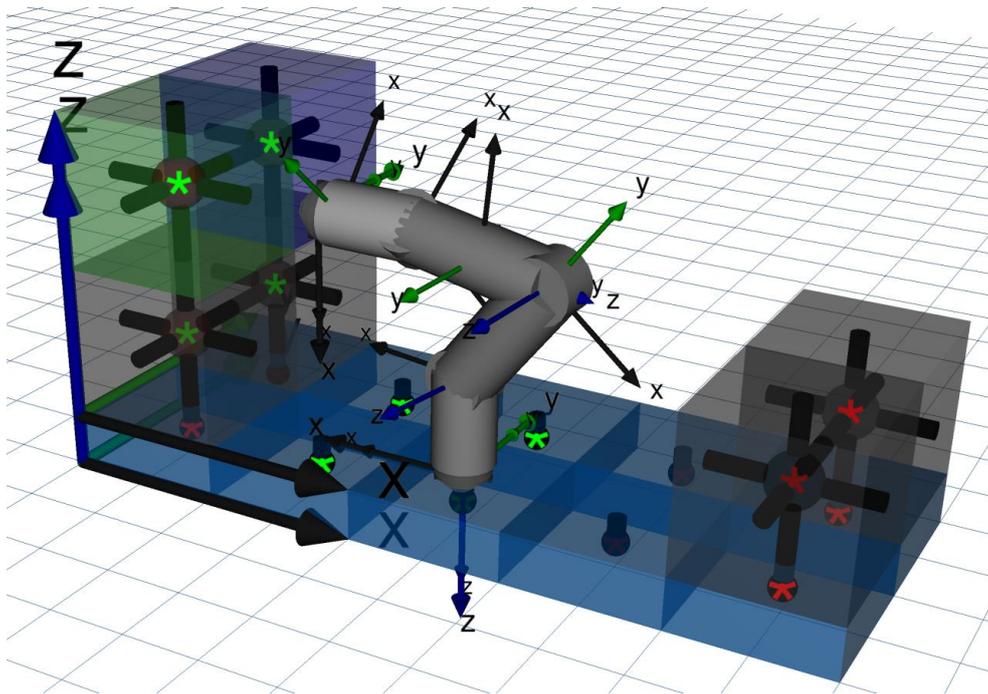


Figure 5-1: Visualization of the fixed FES in SimVIS

The simulator is based on the Modelica/Dymola technology. This technology provides a platform for object-oriented, acausal modelling of multi-physics systems which is required to solve the simulation objectives in MOSAR, in particular the system reconfiguration objectives. The simulator is based on DLR-SR Modelica libraries from various domains which are relevant for the MOSAR project:

- DLR SpaceSystems library
- DLR Robots library
- DLR RobotDynamics library
- DLR Visualization library

For the simulator two versions exist. One simulates the MOSAR demonstration setup on earth, the other the in-orbit simulation of the free floating satellite. In addition a design tool is available which can be used to validate a simulation setup before the actual simulation. Both versions include a detailed model of the WM (robot). The 7-axis robot arm implementation includes detailed drive train implementations including joint flexibilities, friction, motor and sensor models for forces, torques and position. The heat and power flow through the different satellite modules is considered and dynamically updated when the modular structure changes when the WM changes the location of a module. The WM can dynamically “walk” over the modules, using their connectors, without a restart of the simulation.



Detailed Design Document

The more advanced and complex version of the FES, the “Space version”, includes components to simulate the satellite in orbit in great detail. The satellite is free floating in space (LEO and GEO are possible) and also includes an implementation of an onboard GNC system for attitude control as well as a detailed orbit model.

A surface heat flow model is also included, in addition to a rework of DLR SR contact dynamics library to be compatible with Space version of the FES.

The Design Tool software is based on a variant of the FES simulator and uses similar models. The user can start the design tool directly from a Matlab script. The script will check parameter ranges for the user parameters and will make steady state simulations for the initial and final state of the scenario. It will warn the user if parameters or configurations are incorrect. The steady state simulations can be performed very fast compared to a full simulation run, such that the design tool can be used to check for validity before a full simulation using the FES.

The current implementation checks for the following aspects:

- Robot pose
- Static thermal balance
- Heat sink/source steady state analysis
- Produces warning if modules are not powered (no valid path to power source)

The script creates a report file and as well as direct command line output, which can be used by the user to find errors and to correct the scenario.



Detailed Design Document

5.2 Autonomy Agent and Planning

The Figure 5-2 below presents an overview of the tailoring of the ERGO Agent for the MOSAR demonstrator. The Agent is deployed on the Servicer OBC and manages the robotic reconfiguration.

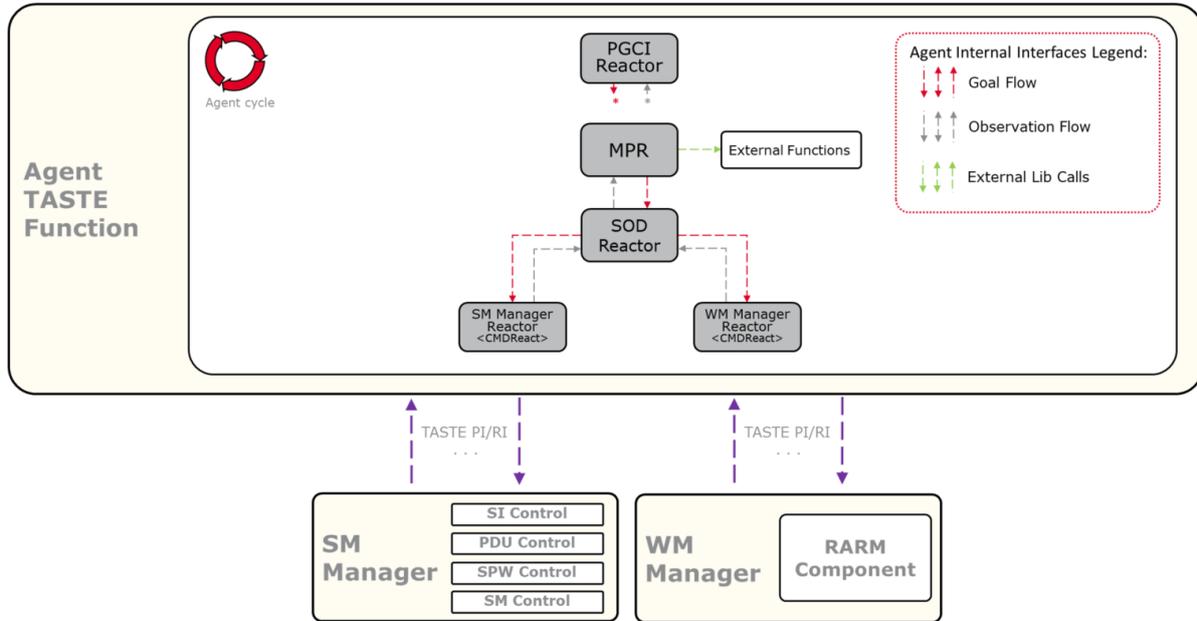


Figure 5-2: Overview of the ERGO Agent tailoring for MOSAR

The Agent and its interfaces are modelled as TASTE functions (which should be understood as components). The figure shows the Agent function and the functions in the Functional Layer that makes the capabilities of the system accessible to the Agent. In this case, two Functional Layer functions are defined: the SM Manager and the WM Manager, which respectively give access to the capabilities of the Spacecraft Modules (including HOTDOCK, power, data and SM configuration) and the Walking Manipulator.

The Agent is the component that plans and controls the actions of the system. This component will also handle all telecommands related to goals sent to robotic system from ground and telemetry generated on board and received from ground.

The architecture of the Agent consists of a controller, in charge of coordinating the different execution loops, and a set of different reactors, in charge of executing a specific control loop. The Agent is a generic component that has to be configured based on the needs of the application to be developed. This involves:

- Defining the reactors that will be part of the agent, and
- Defining the variables (or timelines) that each reactor owns and shares with other reactors.

For the MOSAR, the following set of reactor types are used:

- PGC I Reactor (PUS Ground Control Interface Reactor): reactive reactor that manages the communication with ground; it does not own any timelines, but it is subscribed to all the timelines in the system in order to report all the observations to ground and to enable direct command from ground of all the reactors



Detailed Design Document

- MPR Reactor (Mission Planner Reactor): deliberative reactor that computes the plan (on the plan calculation and verification stage on ground) and manages its execution (both for verification on the simulator and actual execution in the demonstrator)
- SOD Reactor (SM Operations Decomposer Reactor): reactive reactor that decomposes the plan goals into low-level ones
- CMD Reactor (Command Dispatcher Reactor): reactive reactor responsible for interfacing with the functional layer and performing elementary goals; two CMD reactors are defined:
 - SM Manager Reactor: controls the execution of operations on the SMs (mechanical, power and data connections of the SIs, and operation of each SM)
 - WM Manager Reactor: controls the operation of the WM (relocation and grasping)

The following sections detail the internal interfaces of the Agent and the design of the reactors.

5.2.1 Reactors Architecture

5.2.1.1 PGCI

The PGCI reactor is an adaptation of the original GCI (Ground Control Station) prepared to use PUS Services and data types instead of files for the communication with the Ground Control. The PGCI reactor is completed with a PGCI TASTE function, which is in charge of the data type translation. Together, they complete all the functionalities of the PGCI:

1. Serve as an entry point for the agent from the Ground Control, enabling the user to send different commands to the system and change the autonomy level.
2. Monitor all the internal timelines in the agent and send the corresponding telemetry to the Ground Control.
3. Serve as the integration point between ESROCOS PUS services and ERGO Goals and observations.

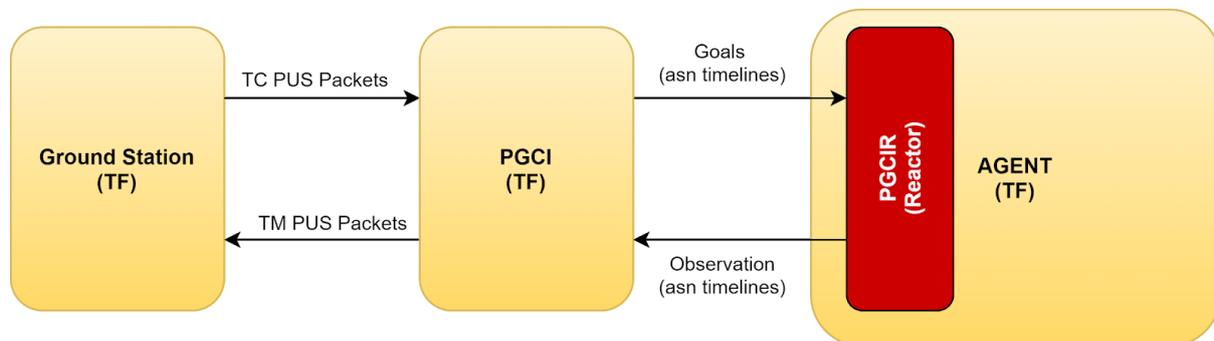


Figure 5-3: PGCI deployment

Following the same principle as in the agent, the telemetry messages are only sent when there is a change in the status. This approach limits the information exchange produced between the two computers (the Ground Station being deployed on ground).

5.2.1.2 MPR

This reactor is in charge of the high level autonomy goals and execution. The original Mission Planner Reactor from OG2 is in charge of two operations:

1. Generation of on-board plans given high level goals using Stellar planner.
2. Execution and monitoring of plans, either generated on-board or received from ground.



Detailed Design Document

In the MOSAR scenario, only the execution of predefined goals will be used on-board, meaning the on-board planning will be deactivated and the system will only work in E3 mode. The generation of plans will only be used on-ground, and the solution, once confirmed with the simulator, sent to be executed on-board.

The main components of the MPR are:

- **Deliberative reactor.** This is the main class that controls the rest of the internal components. It is derived from the Teleoreactor class, which is the base for all the reactors. It has all the interfaces and control mechanism needed to interact with the agent and maintain the control cycle and goals and observations handling.
- **PDDL parser and data types.** This component is composed by a set of classes and functions that are able to understand the PDDL format. Its main function is to parse the PDDL domain to understand all the conditions and effects of the actions so that they can be applied when executing a plan.
- **Executable plan, observer and dispatcher.** These functions and classes compose the module that is in charge of executing a plan (either received by Stellar or ground). The executable plan contains all the actions that compose the plan with its preconditions and post conditions. The observer is in charge of processing all the observations received and checking if they are compatible with the plan execution and when an action has finish so the next one can start. Finally, the dispatcher is in charge of sending the goals to the corresponding timelines using the observer's information.
- **Planner wrapper.** This class is responsible for the interaction with the Stellar planner. It calls for its execution and handles the result returned.
- **Stellar.** The planner that obtains a solution given a PDDL domain and problem.

5.2.1.3 SM Operations

The role of this reactor is to serve as an intermediate between the MPR and the CMD. This gives both the planning model and the functional independence from each other, allowing the first to handle high level actions while still commanding the latest at the low level needed with the command dispatchers.

To perform as a link, the reactor has two main functionalities:

1. Decompose the goals dispatched by the MPR into low level goals to be sent to the functional via the CMD.
2. Abstract the information received by the CMD as observation to high level information more easily usable by the MPR.

More specifically, this reactor will handle the HOTDOCKs commanding and walking manipulator to make sure all the operations needed to attach a SM to another or to the base are completed. It will also make all the necessary connections between SMs.

5.2.1.4 Command Dispatchers

The core of the CMD reactors is inherited without changes from ERGO. In MOSAR two CMD are loaded in the configuration files:

- Command dispatcher for the WM
- Command dispatcher for the HOTDOCK

For each timeline a set of interfaces is defined in TASTE with each of the TASTE functions in charge of handling WM planning and HOTDOCK control.



Detailed Design Document

For these interfaces the translation methods from Goal/Observation (specific to each timeline information, see section 5.2.2) to an analogue ASN.1 type has been defined.

Once the WM or a HOTDOCK are attached during a re-configuration process will be the functional layer the one in charge of the re-configuration sequences for both R-ICU and cPDU to manage the modifications in the layout that have been performed.

5.2.2 Robotic Arm

ERGO provides a Robotic Arm component (RARM) that implements the functionalities needed to plan and execute the movements of a manipulator. It plans trajectories and paths between points avoiding any collision.

The main particularity of the MOSAR scenario is that the WM is capable of walking over the structure, relocating and alternating the roles of its base and end effector. The ERGO software must be extended to support the MOSAR WM and Simulator, and to implement the new capabilities such as walking.

In addition, the scope and responsibilities of the manipulator control differ between ERGO and MOSAR. While in ERGO the RARM software interpolates the robot trajectory and commands the arm controller at fine trajectory level, in MOSAR the RARM software will generate a coarse trajectory and it will be the WM Controller that will interpolate it. The aim is to minimize latency.

In the MOSAR system, the WM is connected to the OBC-S through a chain of SpaceWire links. This can introduce an unpredictable latency that is not compatible with the impedance control of the arm. Impedance control is needed to push the standard interfaces together, compensating the possible misalignments to enable a successful locking of the mechanism. This is handled locally by the WM Controller, which reads the torque measurements and commands the joint motors at a very high rate.

Related to the latency topic, in the OBC-driven approach, total latency is determined by the number of router hops. The OBC will apply the routing topology and so will be able to provide optimal routing paths for the OBC - WM as the WM progresses across the spacecraft. The routers also have the possibility to prioritise WM traffic to reduce the impact of other traffic on the SpW network, this functionality will be included as an optional feature enabled by TC.

The ERGO Robotic Arm package is composed of a set of classes, mainly:

- **PathPlanner:** The PathPlanner class uses the library OMPL (Open Motion Planning Library) which implements various path planning algorithms in a generic and efficient way. It only needs to receive the description of the state space and an external function to check the validity of the sampled points.
- **CollisionChecker:** The collision checker provides functionality to load the robot model from URDF and keep track of all the objects in the scenario. Collisions are checked for a certain pose moving the arm virtually to the requested configuration and checking collisions with itself and the environment. The forward and inverse kinematics are computed purely based on the robot model. The forward kinematics moves the arm virtually to the requested joint configuration and retrieves the pose of the selected end effector. The inverse kinematics uses the default algorithm of the library, the returning position has to get checked for collisions afterwards.
- **Driver:** This package contains the classes associated to the movement of the robotic arm. The Driver class is an interface functionality that declares the desired methods based on the required functionality of the arm. To actually implement this functionality for each specific hardware, a class must inherit from Driver and implement the required functionality with the hardware API.
- **UseCaseDriver:** This class is a placeholder for the implementation of the specific use case; in MOSAR, two classes will be defined to drive either the simulation implementation or the actual WM.



Detailed Design Document

- **RARMManager**: Class that handles the execution of the rest of the components.

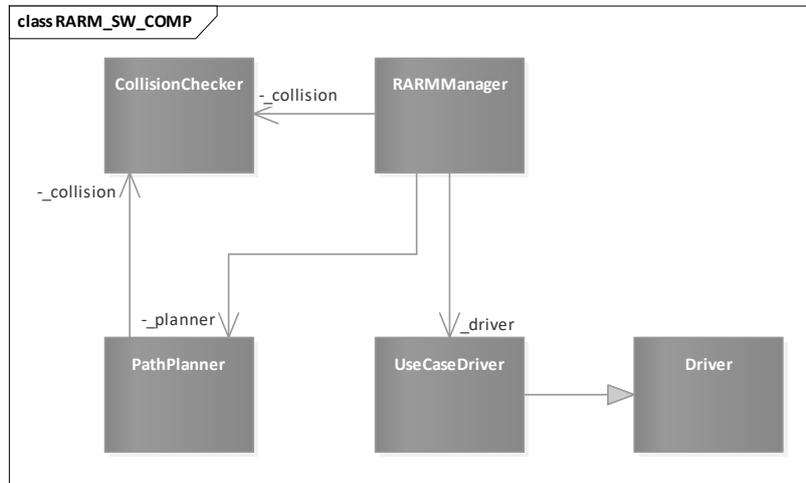


Figure 5-4: Robotic arm main components

This framework has as inputs from the rest of components the commands to be processed by the RARM manager, and which depend on the application being developed. In the case of MOSAR, in addition to the commands to move the end effector to a desired position and orientation, additional commands are needed to switch the end effector (allowing the arm to walk on the structure), and to indicate when the WM is carrying a SM.

The valid positions for the WM are constrained by the location of the SMs and the HOTDOCK interfaces that serve as attachment points. The trajectory to approach a given interface, however, varies according to the configuration of the system, in particular when a SM must be docked simultaneously with several standard interfaces.

The outputs of the RARM component are the requests sent to the WM or to the Simulator by the corresponding Driver class. An important change with respect to the original ERGO component is that the communication with the manipulator controller is interfaced at a different level. In ERGO, the RARM Driver commanded the arm controller at high rate, while in MOSAR the RARM component provides a coarse, collision-free trajectory to the WM Controller, and it is the latter the one that interpolates the trajectory and commands the actuators at a high rate. This avoids latency issues that might be introduced by the communication across several SpaceWire links.

The adaptations and extensions being implemented are:

- **RARMManager**: A new manager, based on the one developed in ERGO has to be implemented with the specific operations to be performed in MOSAR.
- **UseCaseDriver**: A new driver has to be implemented as interface between RARM SW and the WM OBC. This driver will have as an output a list of points that will be interpolated and executed on-board the WM.
- **CollisionChecker**: Adaptation to change configuration of the arm is needed. To walk from one slot to another the robot arm base and end effector will be exchanged.

The Driver components must be developed from scratch to interface with the MOSAR WM and its Simulator. As explained above, the interface with the WM Controller is done at the level coarse trajectory (instead of at the level of interpolated trajectory in ERGO). In addition, the MOSAR system does not rely on vision for the alignment of the end effector, so ERGO RARM Camera component will not be used.



Detailed Design Document

5.2.2.1 RARM Manager Adaptations

The Manager is in charge of loading and executing the rest of the components. It is split into two components:

- RARMManager – Handles the operations of the RARM components (Collision, Planner, Driver) making use of a more generic frame description. Works in generic Isometry transformation.
- MosarScenarioManager – Handles the adaptation of the RARMManager to the MOSAR scenario. Loads and handles the collisions for the scenario (SRV, CLT, SM, HOTDOCK) and add a layer so these new objects and the “slot” coordinates can be used for the specific operations. This component inherits from the more generic RARMManager and so overloads part of its methods.

Figure 5-5 shows the most relevant methods of both objects:

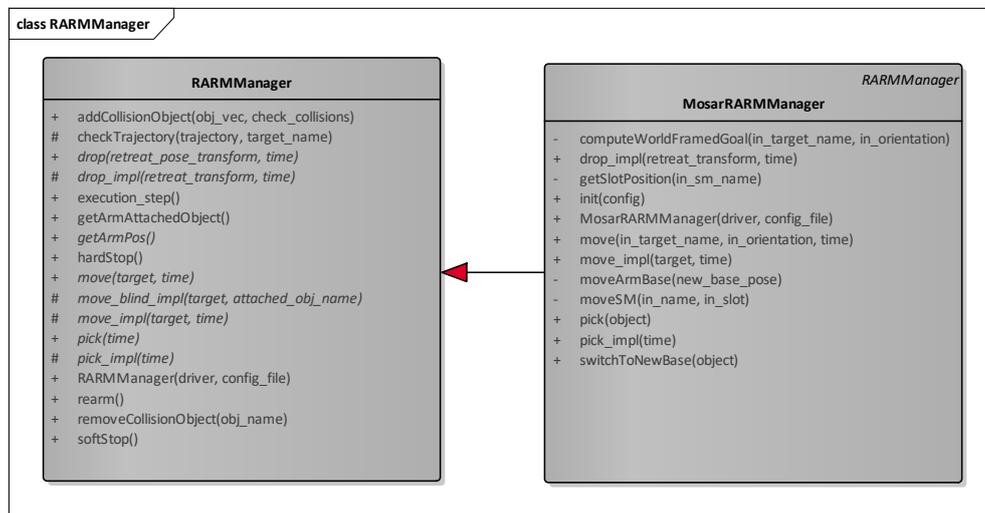


Figure 5-5: RARM Manager relevant methods

5.2.2.2 Driver Implementations

The Driver is in charge of interfacing with the robotic arm. Figure 5-6 shows the class hierarchy and the most relevant methods. Note that two different implementations are foreseen:

- WMSimDriver: Interfaces with the simulator
- WMDriver: Interfaces with the real hardware

From a high level perspective, a similar API is expected for both of them with different implementation according to the counterpart (actual WM or Simulator). The two implementations of the Driver interface are developed from scratch.



Detailed Design Document

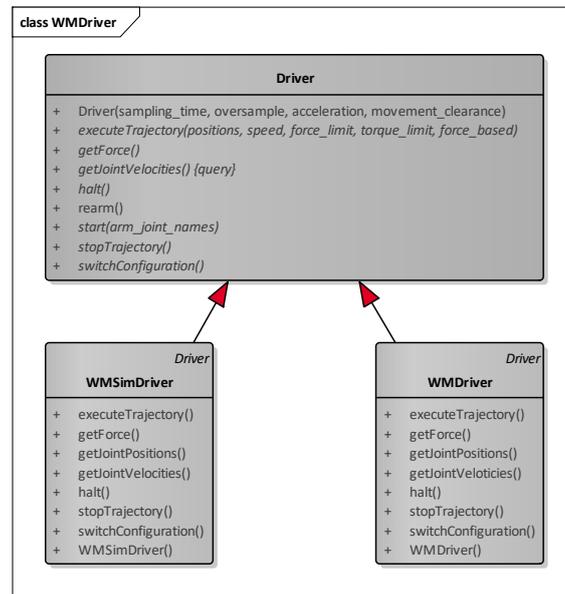


Figure 5-6: Driver specialization

5.2.2.3 Collision Checker

The Collision Checker is in charge of loading and handling collisions and arm model. Figure 5-7 shows the most relevant methods, which are further described below.

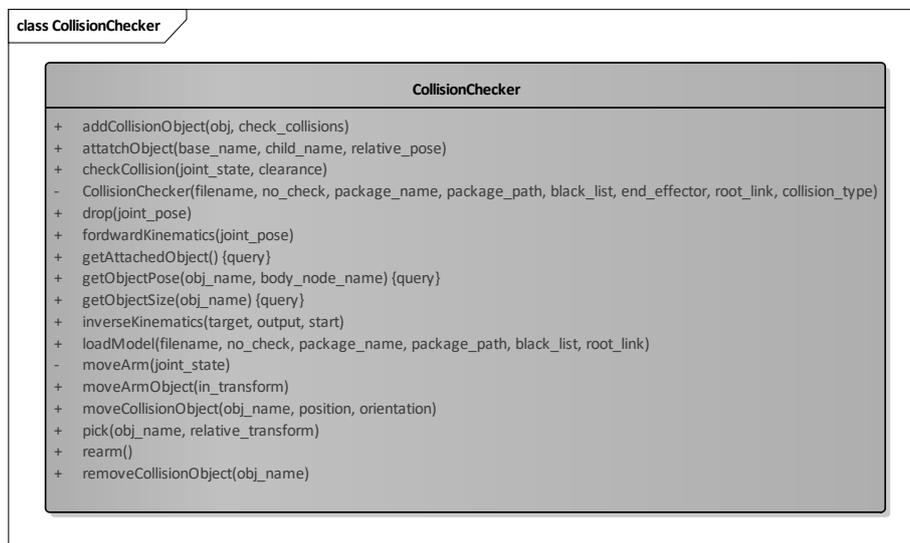


Figure 5-7: Collision Checker relevant methods



Detailed Design Document

5.3 Telemetry and Telecommand Service

The Telemetry and Telecommand (TTC) Service enables the operator to command and monitor the demonstrator from the MCC, both during the robotic reconfiguration of the system and in the nominal operation of the CLT platform before and after the reconfiguration scenario. The TTC Service is based on the ESROCOS PUS Services library, which implements a subset of the services defined in the PUS standard [RD6].

Two instances of the TTC Service are therefore defined:

- The CLT instance is used to monitor and command the SMs during nominal operations. In the demonstration scenario, it will enable the operator to determine when the SMs are operational, to receive telemetry and to command the functions provided by each module.
- The SVC instance is used to monitor and command the Autonomy Agent during the robotic reconfiguration.

In addition, the MCC runs the PUS Console application, which allows the operator to send TCs and visualize the TM received from the service instances.

The Figure 5-8 below shows the detail of the TASTE interface view with the Ground component (which includes the PUS Console tool linked as a library), and the two PUS services instances on the Client and Servicer spacecraft.

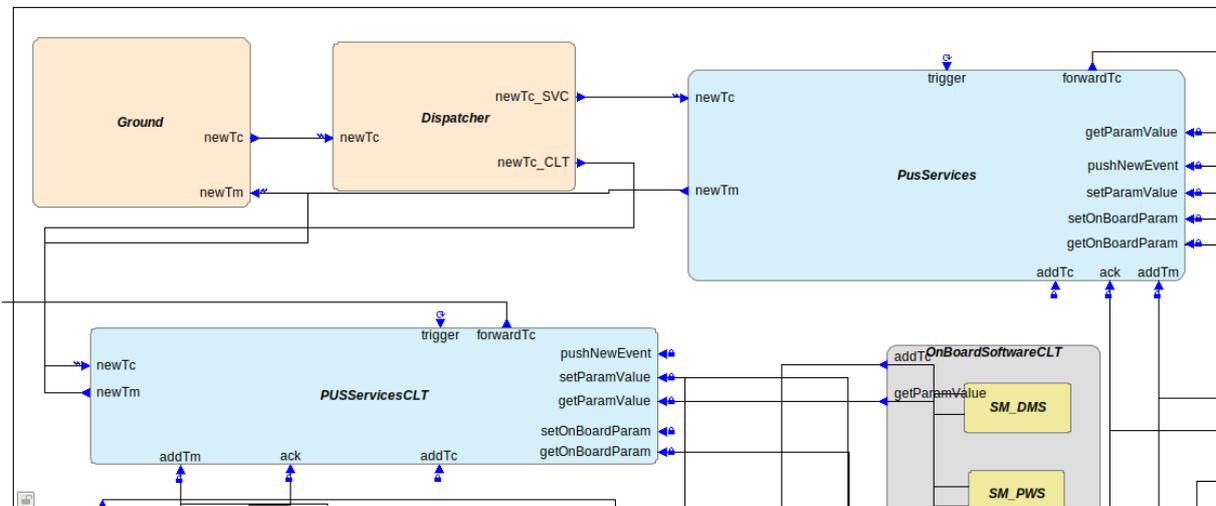


Figure 5-8: Ground and PUS Services instances

The PUS instances are composed of one TASTE function that implements the standard PUS services, and a set of functions that implement three mission-specific services: 200 (operation of the ERGO Agent, present only in the Servicer instance), 210 (RMAP telemetry and telecommand, present only in the on-board segment) and 220 (SW reconfiguration, present only in the on-board segment of the Client).

All RMAP interactions are centralized in a dedicated service. While having a dedicated service for SM to be controlled (and possibly for common components, such as HOTDOCK and R-ICU), it has been preferred to group all in one service for flexibility in case that the RMAP TM and TC interfaces change late in the project.



Detailed Design Document

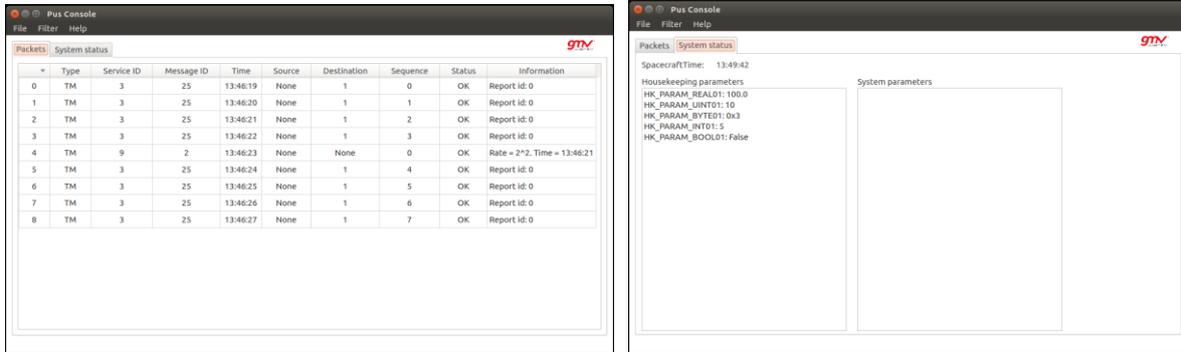


Figure 5-9: PUS Console (left: TM/TC log, right: housekeeping parameters)

A subset of the services included in the ESROCOS PUS library is selected for the MOSAR demonstrator, in addition to the mission-specific services developed from scratch.

Table 5-1: Selection of PUS services for MOSAR

Service Num.	Description and purpose	Type	Scope
1	Request verification Verification and acknowledge of TCs.	Standard	All instances
3	Housekeeping Retrieving of TM from system components (in particular, TM acquired through RMAP from SM subsystems and payloads, R-ICU, HOTDOCK and cPDU).	Standard	All instances
5	Event reporting Notification of component and system events.	Standard	All instances
8	Function management Execution of OBSW functions.	Standard	Optionally, to support testing
17	Test connection Testing of the PUS service connection to MCC.	Standard	All instances
18	On-Board Control Procedure Execution of complex procedures written in micropython.	Standard	Optionally, to support testing
23	File management Upload and download of files from the MCC (to be used in particular for plan and configuration files)	Standard	All instances
200	Autonomy agent management Communication with the autonomy agent (goals, and observations).	Mission-specific	Servicer-only, ground and space segments (for robotic reconfiguration)



Detailed Design Document

Service Num.	Description and purpose	Type	Scope
210	RMAP telemetry and telecommanding Retrieving of RMAP TM (to be reported by service 3) and sending of RMAP TCs.	Mission-specific	Only on the space segment, not relevant for the FES
220	SW reconfiguration Management of the reconfiguration of the system at software level.	Mission-specific	Only for the Client on the space segment, not relevant for the FES



5.4 Spacecraft Modules and Payloads

5.4.1 Spacecraft Modules Detailed Design

SITAEL defined the design of the 7 ASM/APM mechanical structures and harnesses starting from an approach leading towards the space application of the modules; afterwards, the design of the SMs for the ground demonstrator were derived and analyzed.

The design aimed at satisfying WP1 requirements, taking into account the targeted ground test scenarios. The exact number and positioning of the different HOTDOCKs variations, for each APM/ASM and test bench, were optimized and then reflected in the proposed preliminary design (see section 5.7.3). Specific ASM/APM avionics (SpaceWire Router, DC/DC converter, APM/ASM Control, HOTDOCK Controllers) and payload were also accounted for in this work.

Two different approach, in terms of mechanical loads, have been considered for the design of the SM:

- Space
- Ground (demonstrator)

The mechanical architecture is the same, Aluminium panels connected by means of cleats, the main differences are the materials and painting as described in detail in the following paragraph.

5.4.1.1 Space Design

The space design takes into account the space vibrational and thermal environment during the launch and orbit phase.

Different size of the SM has been considered in order to guarantee the compatibility to different class of payloads:

- 1U 400 x400x 400 mm
- 2U 800 x400x 400 mm
- 4U 800 x800x 400 mm
- 8U 800 x800x 800 mm

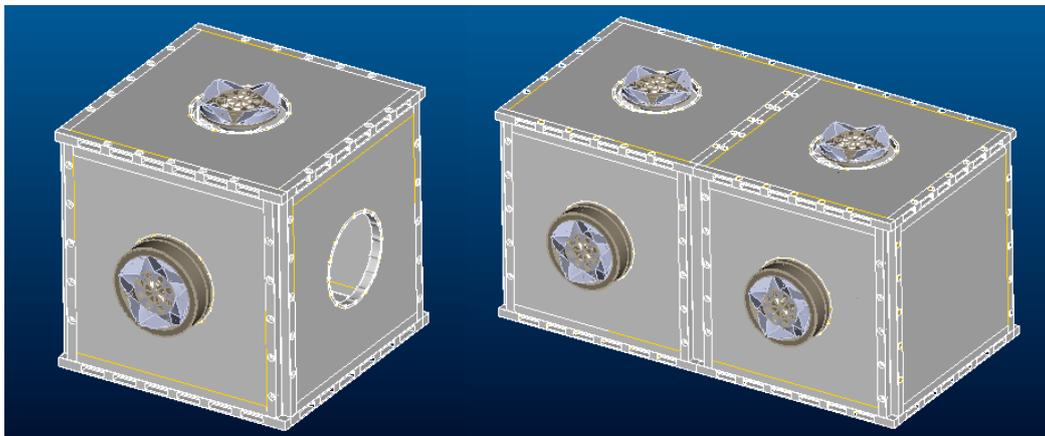


Figure 5-10: 1U SM - 2U SM



Detailed Design Document

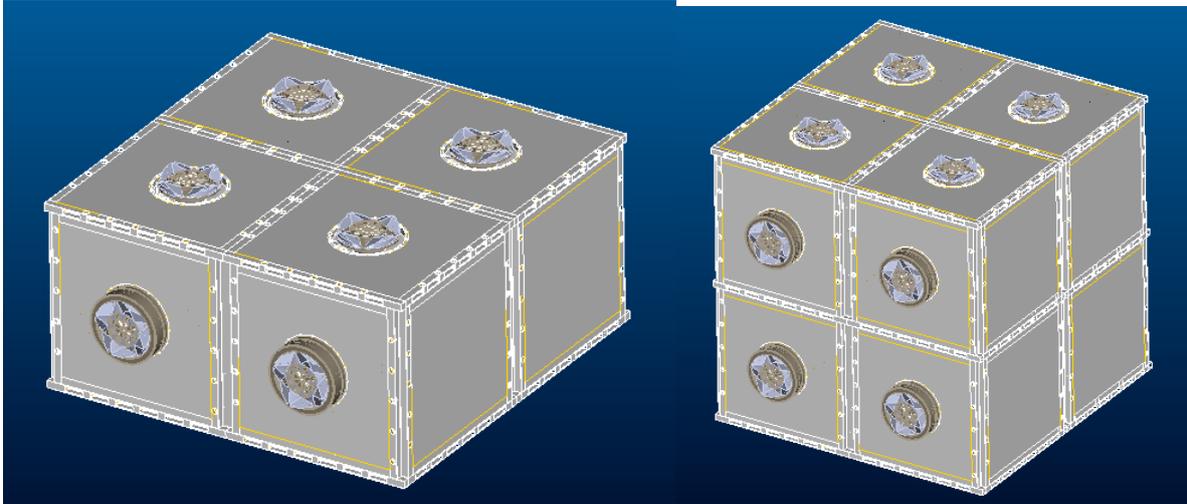


Figure 5-11: 4U SM - 8U SM

The main structure is composed by standard Al Sandwich panels (30mm of total thickness) connected by means of Aluminium beams as shown in Figure 5-12, giving more rigidity to the edge of the panels.

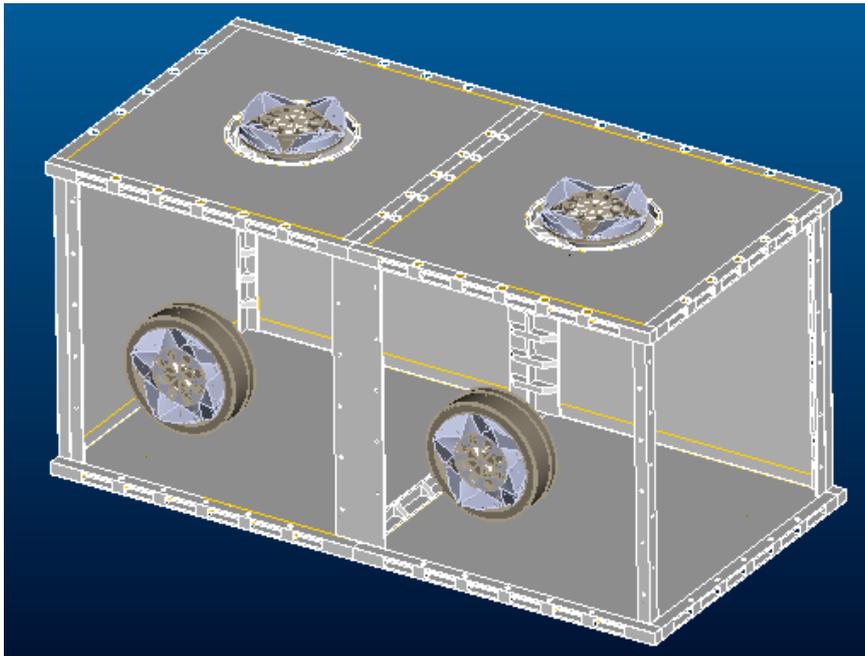


Figure 5-12: Connection beam

In order to guarantee the right stiffness to the mechanical interfaces with the hot dock an embedded aluminium Al 7075 T651 frame has been considered inside the sandwich panels as shown in Figure 5-13.



Detailed Design Document

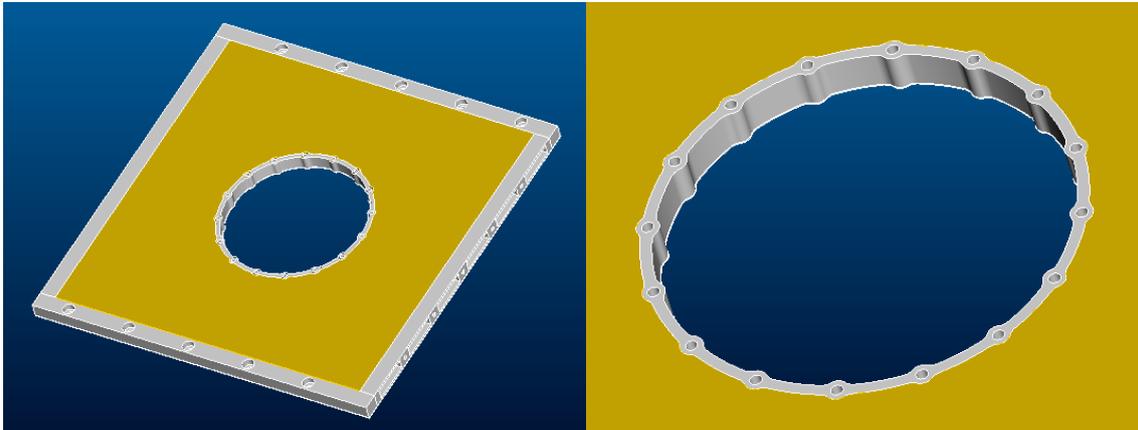


Figure 5-13: Sandwich panels internal Frame

5.4.1.2 Ground demonstrator Design

Modularity is expressed on the satellite structure as the capability to build the platform through standard thermo-mechanical modules assembled and interconnected together through HOTDOCKS. Six modules with different functionalities are designed in order to perform MOSAR demonstration, as listed below:

- SM1-DMS, containing the Data Management Subsystem
- SM2-PWS, containing the Power Subsystem
- SM3-BAT, containing the Battery Assembly
- SM4-THS, containing the Thermal Subsystem
- SM5-OSP1, containing Optical Payload
- SM6-OSP2, containing Optical Payload

A standard cubic shape for the SMs was chosen in MOSAR preliminary design phase with a baseline edge of 350 [mm]. The target implementation foresees the capability to integrate a HOTDOCK on each face of the SM. However, in order to simplify the system design according with ground demonstration, the integration of different models of HOTDOCK will be realized inside MOSAR (see section 5.7.3).

The baseline design of the SM features a customized panel with a central tray structure able to accommodate the module payload boards as well as the service boards. A second customized panel will be used as radiative panel in the Thermal Subsystem (THS) Module and even for the OSP1 and OSP2 Modules one of the lateral panels will be adapted in order to accommodate the payload cameras.



Detailed Design Document

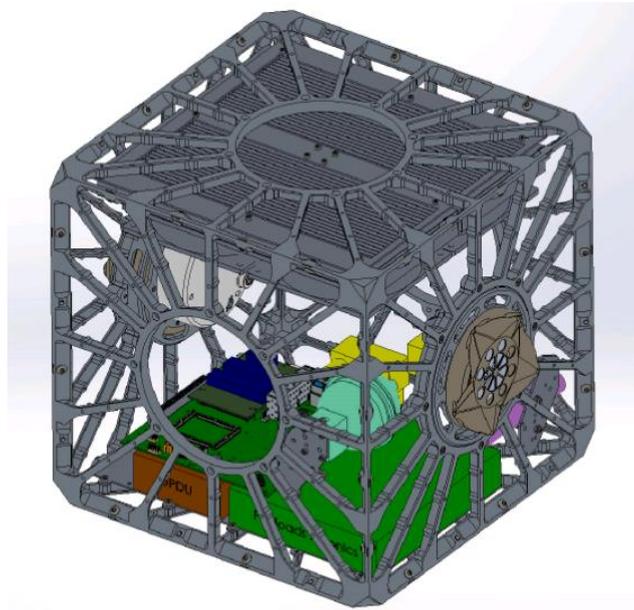


Figure 5-14: Ground demonstrator structure

5.4.2 SM1-DMS

5.4.2.1 Description

The SM1-DMS hosts the main on-board computer of the CLT that runs management software responsible for the management of the modular spacecraft mission and for the interfacing with the other subsystems and payload modules of the spacecraft.

The internal equipment of the SM1 is composed by:

- The R-ICU: is responsible for the local management of the module that includes
 - the local TM/TC with the active Standard Interfaces and the cPDU from RMAP
 - the interface with the OBC-C, through the SpW Brick (bridge)
 - the data communication and routing with the OBCs, through its embedded SpW router.
- The cPDU: is responsible to provide the required power voltage to the internal components, as well as control the power transmission between the Standard Interfaces.
- The module hosts the OBC-C for the management of the Client Satellite. Based on the NUC8i7INH platform with a Linux operating system, it is running an adaptation of ESROCOS software. It mainly manages the TMTC operations with the payload module, but also control the reconfiguration operations (e.g. the SM network discovery). It has a direct Ethernet connection to the MCC to represent the space data link for the PUS service with the ground segment.
- The SpW brick make the interface between the OBC-C and the spacecraft SpW network.
- The module is equipped with 2 active HOTDOCK interfaces.

The module is fixed on the structure of the CLT.



Detailed Design Document



Figure 5-15: OBC-C NUC board

Figure 5-16 illustrates the data architecture of SM1-DMS. The R-ICU is the central element, interfacing the SpW bus (with routing capability) and the internal SM CAN network (see section 5.9). All TM/TC and data transfer, between the OBC and R-ICU, are implemented with the RMAP protocol. From its TM/TC manager, the TM/TC are converted to either payload or CAN TM/TC, which are managed by the respective components controllers (Figure 5-42).

The module is equipped with an external CAN interface, allowing low-level control and telemetry of the components, for non-nominal operation purposes.

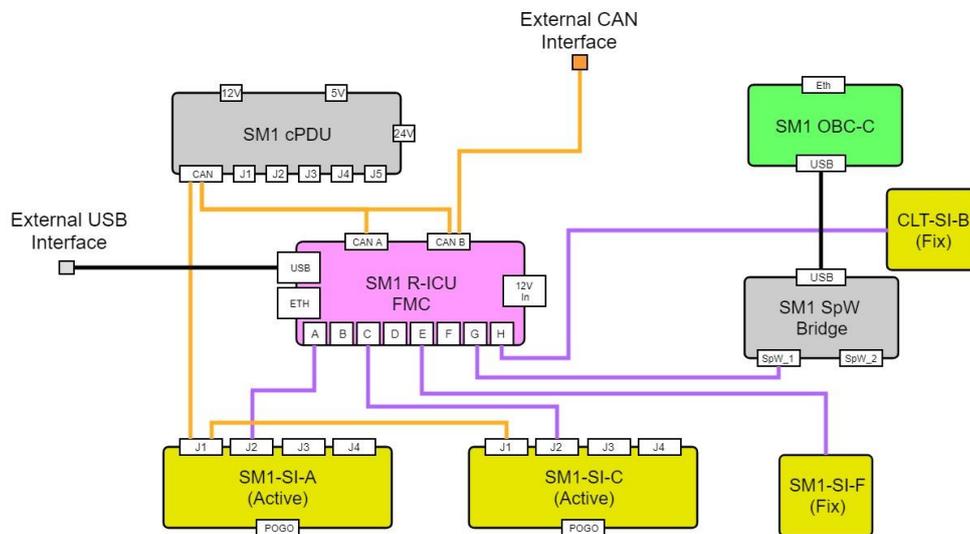


Figure 5-16: SM1-DMS data architecture

Figure 5-17 illustrates the power architecture of the SM1-DMS. The cPDU is the central element, providing the required power to the active components of the module. The SpW Birck is directly powered through the USB ports of the OBC-C. The cPDU controller allow to control the power sequence of the elements, when it is started for the first time. This allows to reduce power peaks and create sequence as function of the use case and application. The cPDU can distribute the 48V power bus along the different HOTDOCK interfaces.



Detailed Design Document

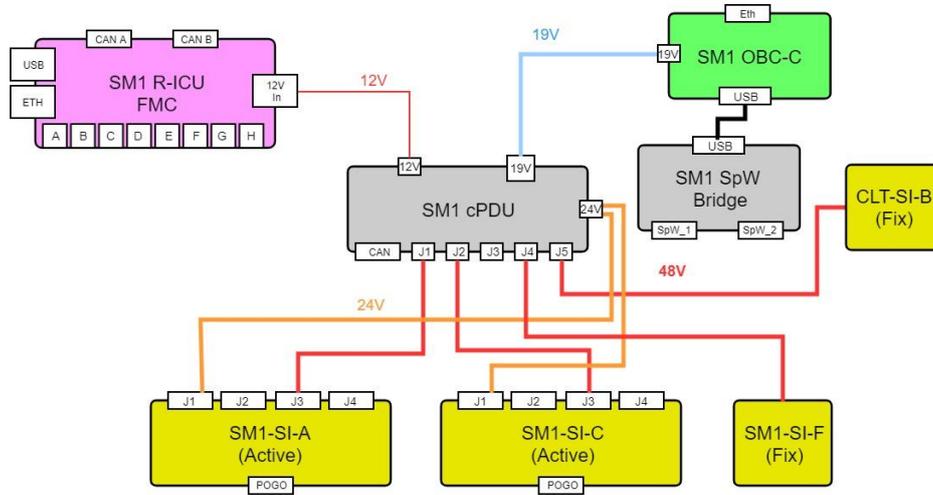


Figure 5-17: SM1-DMS power architecture

5.4.2.2 SM1-DMS Mass Budget

The following table estimates the mass budget for SM1 (including internal harness).

Table 5-2: SM1-DMS mass budget

Components	Mass [kg]	Safety	Mass w/ safety [kg]
Mechanical Structure	3.38	1.1	3.73
R-ICU	0.65	1.1	0.72
cPDU	0.38	1.1	0.42
Payload	0.68	1.1	0.72
HOTDOCK	3.46	1.1	3.81
Total	8.56		9.39

5.4.2.3 SM1-DMS Power Budget

The following table summarizes the power consumption of each active components of SM1-DMS, and identify the required current and consumption for each internal cPDU power line. It has to be noted that the table takes the assumption that the SM1-OBC is powered through the cPDU. There is also an option to power it directly from the main supply, through its own DC/DC convertor.

Table 5-3: SM1-DMS power budget

Components	Rail Voltage [V]	Average Power [W]	Peak Power [W]	Line Average Current [A]	Line Peak Current [A]	Bus Average Current [A]	Bus Peak Current [A]
R-ICU / FMC Board	12	10	20	0.83	1.67	0.21	0.42
cPDU	48	10	18.5	0.21	0.39	0.21	0.39
SM1-SI-A	24	5	10	0.21	0.42	0.1	0.21
SM1-SI-C	24	5	10	0.21	0.42	0.1	0.21
OBC	24	30	50	1.25	2.08	0.63	1.04
SpW Bridge	24	1.5	2.5	0.06	0.1	0.03	0.05
Total	N.A.	61.5	111	N.A.	N.A.	1.28	2.31



Detailed Design Document

5.4.3 SM2-PWS

5.4.3.1 Description

The SM2-PWS represents the power module of the Client Satellite. It regulates and balances the electrical power between the different sources of power (Solar Panel power conditioning and the internal battery of the SM3 Battery Module). The PWS is a heating element. It also provides a dedicated SI with a thermal interface to perform and demonstrate forced heat exchange with the Thermal Subsystem module SM4 (see below).

The internal equipment of the SM1 is composed by:

- The R-ICU: is responsible for the local management of the module that includes
 - the local TM/TC with the cPDU, from RMAP
 - the local management of the Thermal Subsystem making the conversion between the RMAP and CAN TMTC, and local control
 - the data communication and routing, through its embedded SpW router.
- The cPDU: is responsible to provide the required power voltage to the internal components, as well as control the power transmission between the Standard Interfaces.
- The Thermal Subsystem (see below)
- The module is equipped with 2 passive HOTDOCK interfaces.

The module is fixed on the structure of the CLT.

Figure 5-18 represents the SM2-PWS data architecture. It has the same design as SM1, except the presence of the Thermal subsystem, interfaced with the R-ICU by the local CAN bus. The presence of a dedicated R-ICU in SM2 was not initially foreseen, the module being then managed by the SM1 R-ICU. This is currently under evaluation, with the purpose to provide a more generic design to the full setup (each SM, having its own R-ICU controller).

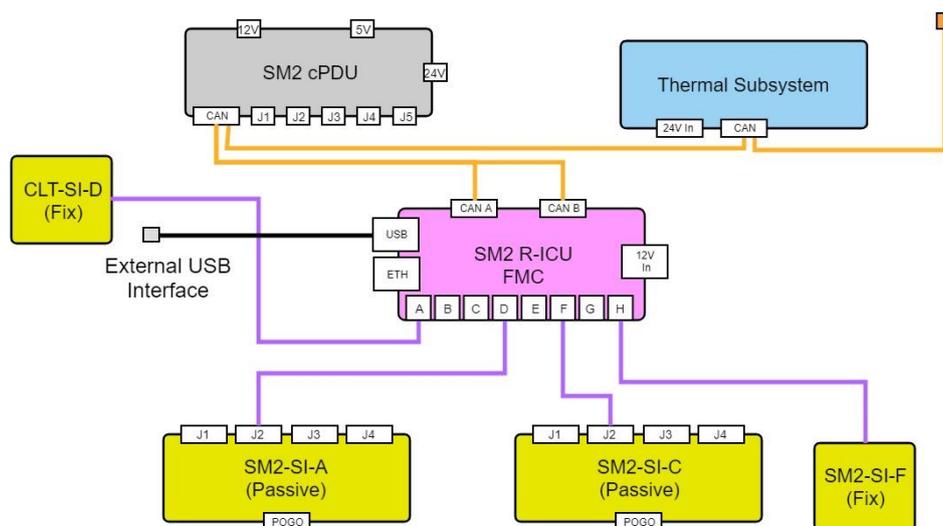


Figure 5-18: SM2-PWS data architecture



Detailed Design Document

Figure 5-19 represents the SM2-PWS power architecture. The main specificity is the presence of the 48V power supply that represents the power generator of the spacecraft. In the demonstration, the power supply will be external to the module and directly connected to the SM2-PWS power input of the cPUD. The heating element, that will generate the heat to dissipate during the demonstration, will be directly connected to the main supply (not provided by the internal SM power circuit).

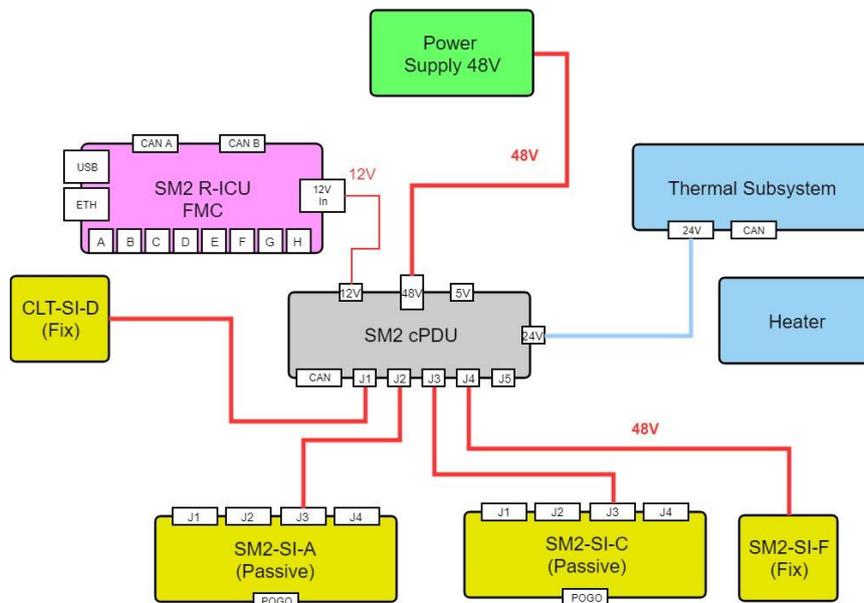


Figure 5-19: SM2-PWS power architecture

5.4.3.2 Thermal Payload

The purpose of the thermal payload is to control thermal exchange between SM with the goal to dissipate heat generated by the SM2-PWS in the SM4-THS (see below).

The thermal management implies the development of two components that are the fluidic thermal interface (embedded in the HOTDOCK interface to allow fluid transfer between modules) and the Thermal Subsystem, which is responsible for heat dissipation and control inside the SMs. The two main parts of this subsystem are described below

- 1) **Thermal IF:** MOSAR Thermal IF consists of 8 hydraulic connectors (four males and four females) integrated in a 3D printed part made of titanium (see figure below).

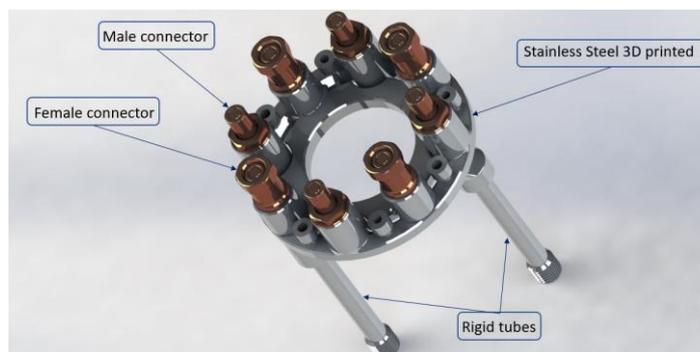


Figure 5-20: CAD model of Thermal IF



Detailed Design Document

The main characteristics of the thermal IF developed are summarized in the table below:

Table 5-4: Main characteristics of Thermal IF

Parameter	Value	Unit
90-degree symmetry	-	-
Maximum heat exchange	2500	W
Operational Temperature range	-40 to 100	°C
Maximum fluid flow	2.12	l/min
Maximum fluid pressure	15	bar
Minimum connection cycles	1000	-
Reference pressure drop	2	bar
Required connection force	400 ±10	N
Mass budget	0,181	kg
Heat exchanged between Cold and Hot line (+10°C)	0.2	W

- 2) **Thermal subsystem:** The thermal subsystem is located inside SM2-PWS and SM4-THS and form a close-loop circuit capable of transferring heat from the source of heat (7) located in SM2-PWS and dissipate this heat on the radiator (6) located in SM4-THS by means of a fans array (2). Note that by means of the thermal IF (9) heat is transferred from SM2-PWS to SM4-THS.

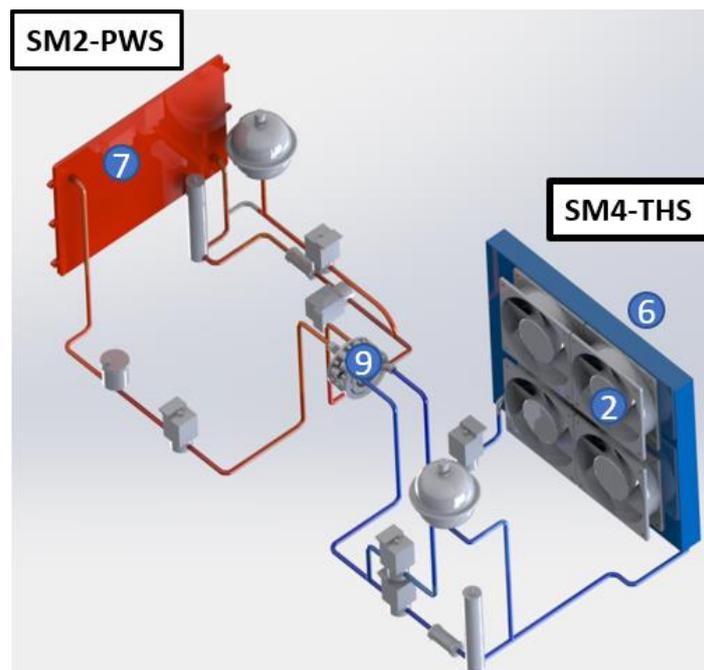


Figure 5-21: Thermal subsystem located inside SM2-PWS and SM4-THS.



Detailed Design Document

5.4.3.3 Thermal Subsystem Software Interface

The following tables provides the Battery Subsystem CAN TM/TC definition, for the communication with the R-ICU. The same data structure is shared between SM2 and SM4.

Table 5-5: Thermal subsystem TM HK1 – Status Data

Field Offset	Field Name	Length (bits)	Units	Description
0	Mode STATUS	8	Enum	Thermal Subsystem Status (Idle, configuration transition phases, operation)
1	VALVE_1_STATUS	1	bool	Valve 1 Status (Enabled/Disabled)
2	VALVE_2_STATUS	1	bool	Valve 2 Status (Enabled/Disabled)
3	VALVE_3_STATUS	1	bool	Valve 3 Status (Enabled/Disabled)
4	VALVE_4_STATUS	1	bool	Valve 4 Status (Enabled/Disabled)
5	VALVE_5_STATUS	1	bool	Valve 5 Status (Enabled/Disabled)
6	VALVE_6_STATUS	1	bool	Valve 6 Status (Enabled/Disabled)
7	VALVE_7_STATUS	1	bool	Valve 7 Status (Enabled/Disabled)
8	ERR_CODES	8	Enum	Mask coding error status
	TOTAL:	23		

Table 5-6: Thermal subsystem TM HK2 – Temperature Data

Field Offset	Field Name	Length (bits)	Units	Description
0	TMP_IN_COLD	16	°C	Input temperature of the radiator
1	TMP_OUT_COLD	16	°C	Output temperature of the radiator
2	TMP_IN_HOT	16	°C	Input temperature of heat exchanger
3	TMP_OUT_HOT	16	°C	Output temperature of heat exchanger
	TOTAL:	64		

Table 5-7: Thermal subsystem TM HK3 – Power and Pressure Data

Field Offset	Field Name	Length (bits)	Units	Description
0	Q_COLD	16	W	Thermal Power dissipated to ambient by radiator
1	Q_HOT	16	W	Thermal Power generated by heater
2	P_PWS	16	Pa	Pressure in SM2_PWS
3	P_THS	16	Pa	Pressure in SM4_THS
	TOTAL:	64		



Detailed Design Document

Table 5-8: Thermal subsystem TM HK4 – Flow and Motor Data

Field Offset	Field Name	Length (bits)	Units	Description
0	FLW_PWS	16	m ³ /s	Liquid flow of PWS pump
1	FLW_THS	16	m ³ /s	Liquid flow of THS pump
2	MOT_PWS	16	RPM	Pump speed PWS
3	MOT_THS	16	RPM	Pump speed THS
	TOTAL:	32		

Table 5-9: Battery TMC Codes – Packet Types List

			CAN TMC Codes - Packet Types List	
Type Code	Dir.	Type Label	Payload Fields Nbr.	Description
TELECOMMAND	TC			
0x00	TC	TC_UNK	0	Unknown value (Unused)
0x01	TC	TC_HK1	0	HK1 Request
0x02	TC	TC_HK2	0	HK2 Request
0x03	TC	TC_HK3	0	HK3 Request
0x04	TC	TC_HK4	0	HK4 Request
0x05	TC	TC_HKA	0	All HK Request
0x06	TC	TC_ATM	1	Auto TM [HZ] +1 payload field
0x07	TC	TC_START	0	Enable Subroutine to start sequence
0x08	TC	TC_STOP	0	Enable Subroutine to stop sequence and non-leakage strategy
0x09	TC	TC_EMERGENCY	0	Enable Subroutine to go in safe configuration
0x10	TC	TC_VALVE_1	1	Valve 1 control (enable/disable) (debug)
0x11	TC	TC_VALVE_2	1	Valve 2 control (enable/disable) (debug)
0x12	TC	TC_VALVE_3	1	Valve 3 control (enable/disable) (debug)
0x13	TC	TC_VALVE_4	1	Valve 4 control (enable/disable) (debug)
0x14	TC	TC_VALVE_5	1	Valve 5 control (enable/disable) (debug)
0x15	TC	TC_VALVE_6	1	Valve 6 control (enable/disable) (debug)
0x16	TC	TC_VALVE_7	1	Valve 7 control (enable/disable) (debug)
0x20	TC	TC_PUMP_PWS	1	PWS Pump speed set point (debug)
0x21	TC	TC_PUMP_THS	1	THS Pump speed set point (debug)
0x22	TC	TC_FAN	1	Fan control (enable/disable) (debug)
TELEMETRY	TM			
0x80	TM	TM_UNK		Unknown value (Unused)
0x81	TM	TM_ACK		Acknowledged Pckt
0x82	TM	TM_NAK		Not Acknowledged Pckt



Detailed Design Document

0x83	TM	TM_HK1		HK1 Response
0x84	TM	TM_HK2		HK2 Response
0x84	TM	TM_HK2		HK3 Response
0x84	TM	TM_HK2		HK4 Response

5.4.3.4 SM2-PWS Mass Budget

The following table estimates the mass budget for SM2-PWS (including internal harness).

Table 5-10: SM2-PWS mass budget

Components	Mass [kg]	Safety	Mass w/ safety [kg]
Mechanical Structure	3.38	1.1	3.73
R-ICU	0.65	1.1	0.72
cPDU	0.38	1.1	0.42
Payload	5.88	1.1	6.4
HOTDOCK	1.61	1.1	1.77
Total	11.91		13.1

5.4.3.5 SM2-PWS Power Budget

The following table summarizes the power consumption of each active components of SM2-PWS, and identify the required current and consumption for each internal cPDU power line.

Table 5-11: SM2-PWS power budget

Components	Rail Voltage [V]	Average Power [W]	Peak Power [W]	Line Average Current [A]	Line Peak Current [A]	Bus Average Current [A]	Bus Peak Current [A]
R-ICU / FMC Board	12	10	20	0.83	1.67	0.21	0.42
cPDU	48	15	28	0.31	0.58	0.31	0.58
Thermal Subsystem	24	68	122	2.83	5.08	1.42	2.54
Total	N.A.	93	170	N.A.	N.A.	1.94	3.54



Detailed Design Document

5.4.4 SM3-BAT

5.4.4.1 Description

The SM3-BAT is representative of a battery ORU module. It comprises a set of Lithium-ion batteries for the storage of electrical power in the spacecraft. It includes the electronics to manage the balance and regulation of their charge and discharge and to convert their voltage to the CLT buses voltages.

The internal equipment of the SM3 is composed by:

- The R-ICU: is responsible for the local management of the module that includes
 - the local TM/TC with the active Standard Interface and the cPDU, from RMAP
 - the local management of the Battery Subsystem making the conversion between the RMAP and CAN TMTC, and local control
 - the data communication and routing, through its embedded SpW router.
- The cPDU: is responsible to provide the required power voltage to the internal components, as well as control the power transmission between the Standard Interfaces.
- The Battery Subsystem (see below)
- The module is equipped with 1 active and 2 passive HOTDOCK interfaces.

The module is movable by the WM and is initially attached to the SVC, through the active HOTDOCK interface.

Figure 5-22 represents the SM3-BAT data architecture. It has the same design as SM1, except the presence of the Battery Subsystem, interfaced with the R-ICU by the local CAN bus

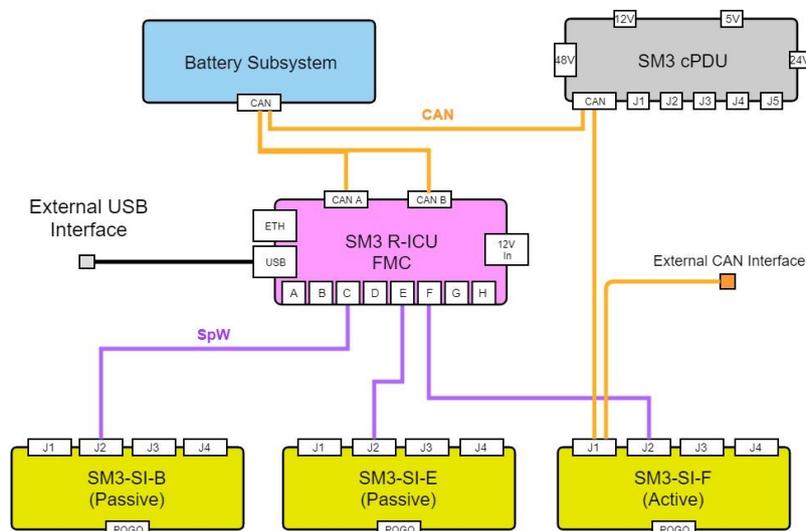


Figure 5-22: SM3-BAT data architecture

Figure 5-23 represents the SM3-BAT power architecture. The main specificity is the presence of the Battery Subsystem that interface the cPDU through the 48V bus. For the demonstration, the SM3 will illustrate the capabilities of charging and discharging of the battery pack, to power specific modules, through dedicated and controlled power path, through the power interfaces of HOTDOCK, and by the control of the cPDU switches. On top of the SM3 itself, we have designed the battery pack to enable the simultaneous operation of SM1-DMS and SM5-OSP1.



Detailed Design Document

The cPDU can distribute the 48V power bus along the different HOTDOCK interfaces. The module is equipped with an external power interface on the 48V bus, to replace the inputs from the HOTDOCK interfaces (for debugging purpose).

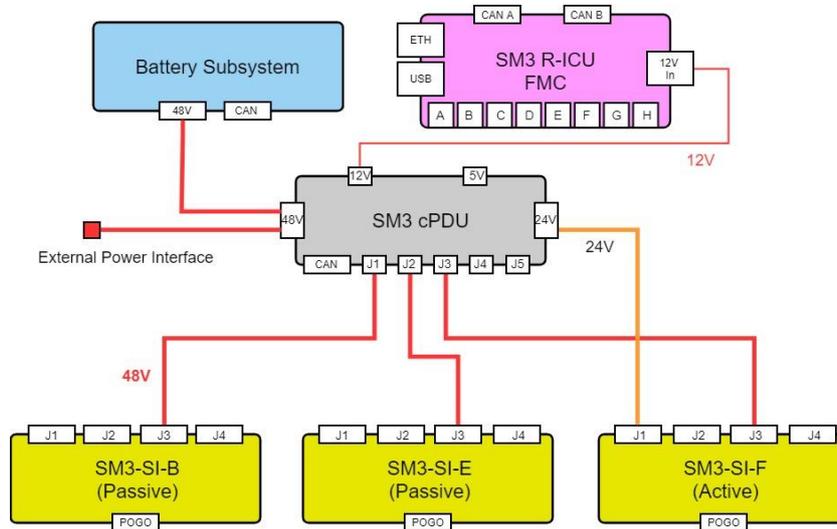


Figure 5-23: SM3-BAT power architecture

5.4.4.2 Battery Payload Subsystem

One of the main purposes of this system is to demonstrate power storage and charging in a modular spacecraft. During the demonstration the power shall be initially routed to charge the battery. At a certain point the power shall be cut and the battery shall take over as the main power source. Based on this scenario, the underlying requirements are the following:

- 48V interface to the main bus
- Battery capacity to support SM1 and SM5 for 30 minutes
- Switch between charging and discharging from the same bus (48V) autonomously or by command
- CAN interface

In order to supply 124.4W for 30 minutes with a 3S cell arrangement with a 2500mAh battery, it is required to have 2 strings in parallel. With a large safety margin, 4 strings in parallel are implemented with a total battery capacity of 10Ah.

Total weight of a 3S4P (3 Series 4 parallel) INR18650-25R cells is 540g.

Figure 5-24 shows the battery system architecture with its main power connections. The design is based on a battery controller system demonstrated in OG5.



Detailed Design Document

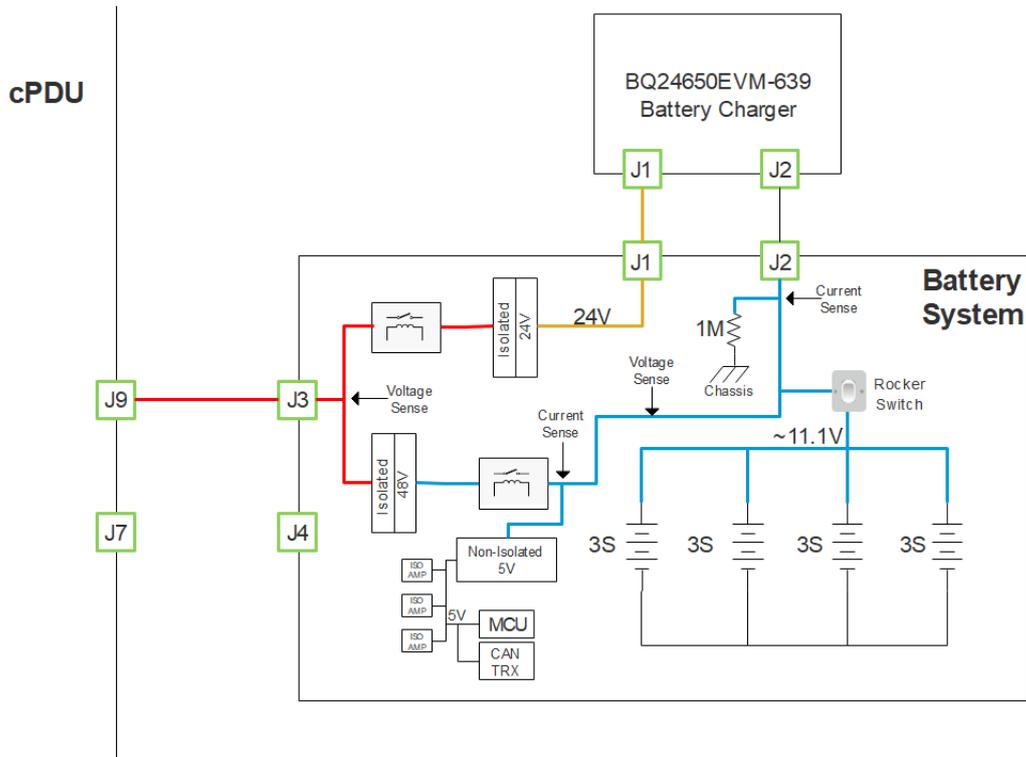


Figure 5-24: Battery power architecture

The battery system interfaces local CAN bus through the cPDU. The CAN commands from the R-ICU are received by a local microcontroller which interfaces current/voltage sensors and relays on the board. This allows passing telemetry back to the OBC and commanding for changing battery system's states.

5.4.4.3 Battery Payload Software Interface

The following tables provides the Battery Subsystem CAN TM/TC definition, for the communication with the R-ICU.

Table 5-12: Battery TM HK1 – Status Data

Field Offset	Field Name	Length (bits)	Units	Description
0	Mode STATUS	4	Enum	Battery Status (Standby, Charging, Discharging)
1	BAT_STATE	8	%	Battery state of charge
2	BAT_TEMP	16	Deg	Battery Temperature
3	ROCKER_RLY_STATUS	1	Bool	Rocker Relay Status (Enabled/Disabled)
4	CHARGER_RLY_STATUS	1	Bool	Charger Relay Status (Enabled/Disabled)
5	BUS_RLY_STATUS	1	Bool	Bus Relay Status (Enabled/Disabled)
6	ERR_CODES	8	Enum	Mask coding error status
	TOTAL:	39		



Detailed Design Document

Table 5-13: Battery TM HK2 – Power Data

Field Offset	Field Name	Length (bits)	Units	Description
0	BAT_V	16	mV	Battery Voltage
1	BAT_I	16	mA	Battery Current
2	BUS_V	16	mV	Bus Voltage
3	BUS_I	16	mA	Nus Current
	TOTAL:	64		

Table 5-14: Battery TMC Codes – Packet Types List

			CAN TMC Codes - Packet Types List	
Type Code	Dir.	Type Label	Payload Fields Nbr.	Description
TELECOMMAND		TC		
0x00	TC	TC_UNK	0	Unknown value (Unused)
0x01	TC	TC_HK1	0	HK1 Request
0x02	TC	TC_HK2	0	HK2 Request
0x06	TC	TC_HKA	0	All HK Request
0x08	TC	TC_ATM	1	Auto TM [HZ] +1 payload field
0x10	TC	TC_MODE	1	Update Battery Controller Mode
0x11	TC	TC_ROCKER	1	Enable/Disable Rocker switch
0x12	TC	TC_CHARGER	1	Enable/Disable Charger switch
0x13	TC	TC_BUS	1	Enable/Disable Bus switch
TELEMETRY		TM		
0x80	TM	TM_UNK		Unknown value (Unused)
0x81	TM	TM_ACK		Acknowledged Pckt
0x82	TM	TM_NAK		Not Acknowledged Pckt
0x83	TM	TM_HK1		HK1 Response
0x84	TM	TM_HK2		HK2 Response



Detailed Design Document

5.4.4.4 SM3-BAT Mass Budget

The following table estimates the mass budget for SM3-BAT (including internal harness).

Table 5-15: SM3-BAT mass budget

Components	Mass [kg]	Safety	Mass w/ safety [kg]
Mechanical Structure	3.38	1.1	3.73
R-ICU	0.65	1.1	0.72
cPDU	0.38	1.1	0.42
Payload	1.26	1.1	1.39
HOTDOCK	2.97	1.1	3.26
Total	8.65		9.51

5.4.4.5 SM3-BAT Power Budget

The following table summarizes the power consumption of each active components of SM3-BAT, and identify the required current and consumption for each internal cPDU power line.

Table 5-16: SM3-BAT power budget

Components	Rail Voltage [V]	Average Power [W]	Peak Power [W]	Line Average Current [A]	Line Peak Current [A]	Bus Average Current [A]	Bus Peak Current [A]
R-ICU / FMC Board	12	10	20	0.83	1.67	0.21	0.42
cPDU	48	3	6	0.06	0.13	0.06	0.13
SM3-SI-F	24	5	10	0.21	0.42	0.1	0.21
Battery Subsystem	48	18	36	0.38	0.75	0.38	0.75
Total	N.A.	36	72	N.A.	N.A.	0.75	1.5

5.4.5 SM4-THS

5.4.5.1 Description

The SM4-THS is responsible for the thermal management of the CLT SMs. It provides a SI with a thermal interface to allow heat exchange by fluid loop. The THS dissipates heat through ventilators. Interfacing the THS to the PWS will allow extracting heat from the PWS via a liquid cooled loop, and radiate this heat in ambient air through the radiator of the THS.

The internal equipment of the SM4 is composed by:

- The R-ICU: is responsible for the local management of the module that includes
 - the local TM/TC with the active Standard Interface and the cPDU, from RMAP
 - the local management of the Thermal Subsystem making the conversion between the RMAP and CAN TMTC, and local control
 - the data communication and routing, through its embedded SpW router.
- The cPDU: is responsible to provide the required power voltage to the internal components, as well as control the power transmission between the Standard Interfaces.
- The Thermal Subsystem (see SM2-PWS, section 5.4.3.2)
- The module is equipped with 1 active and 2 passive HOTDOCK interfaces.



Detailed Design Document

Figure 5-25 represents the SM2-PWS data architecture. It has the same design as SM1, except the presence of the Thermal subsystem, interfaced with the R-ICU by the local CAN bus.

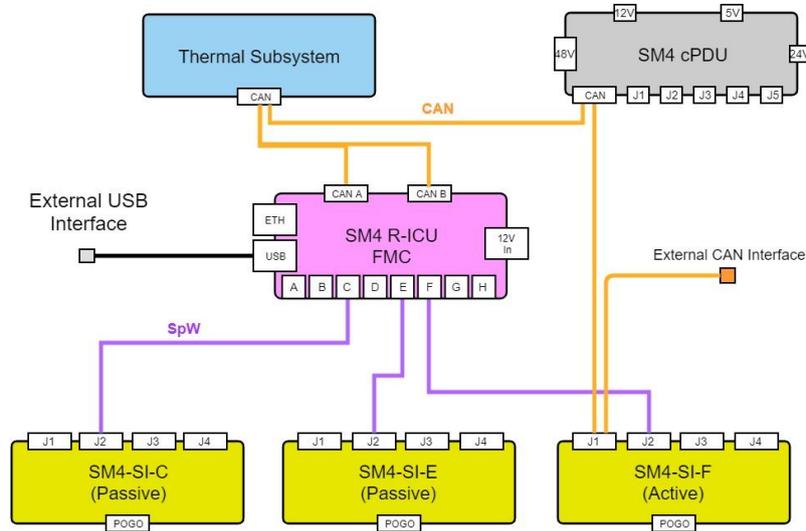


Figure 5-25: SM4-THS data architecture

Figure 5-26 represents the SM2-PWS power architecture. The Thermal Subsystem is powered from the cPDU with 24V and performs local conversion for its own components. The cPDU can distribute the 48V power bus along the different HOTDOCK interfaces. The module is equipped with an external power interface on the 48V bus, to replace the inputs from the HOTDOCK interfaces (for debugging purpose).

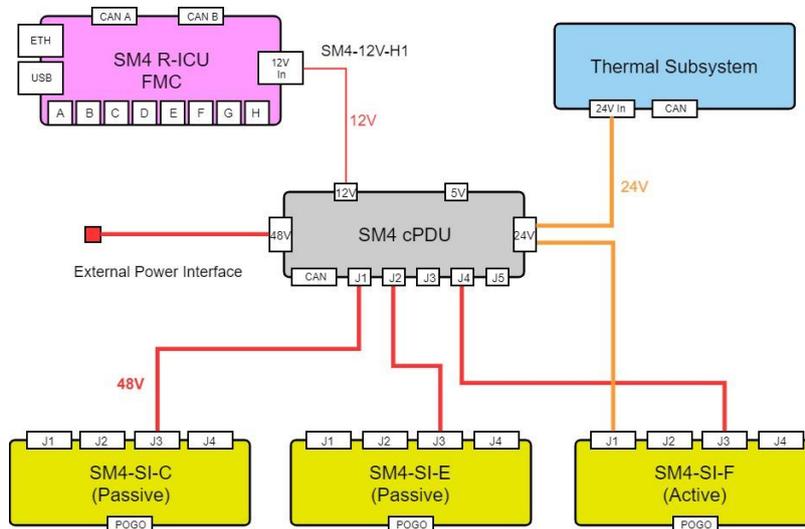


Figure 5-26: SM4-THS power architecture



Detailed Design Document

5.4.5.2 SM4-THS Mass Budget

The following table estimates the mass budget for SM4-THS (including internal harness). Table 5-17: SM3-BAT mass budget

Components	Mass [kg]	Safety	Mass w/ safety [kg]
Mechanical Structure	3.38	1.1	3.73
R-ICU	0.65	1.1	0.72
cPDU	0.38	1.1	0.42
Payload	5.29	1.1	5.82
HOTDOCK	2.97	1.1	3.27
Total	12.68		13.94

5.4.5.3 SM4-THS Power Budget

The following table summarizes the power consumption of each active components of SM3-BAT, and identify the required current and consumption for each internal cPDU power line.

Table 5-18: SM3-BAT power budget

Components	Rail Voltage [V]	Average Power [W]	Peak Power [W]	Line Average Current [A]	Line Peak Current [A]	Bus Average Current [A]	Bus Peak Current [A]
R-ICU / FMC Board	12	10	20	0.83	1.67	0.21	0.42
cPDU	48	21	36	0.44	0.75	0.44	0.75
SM3-SI-F	24	5	10	0.21	0.42	0.1	0.21
Thermal Subsystem	24	92	150	3.83	6.25	1.92	3.13
Total	N.A.	128	216	N.A.	N.A.	2.66	4.5



Detailed Design Document

5.4.6 SM5-OSP1

5.4.6.1 Description

The SM5-OSP1 is representative of an optical payload module. The purpose of the Camera Payload is to provide images of the surroundings of the spacecraft up to the MCC. In MOSAR, it is used for demonstration purpose.

The internal equipment of the module is composed of:

- The R-ICU: is responsible for the local management of the module that includes
 - the local TM/TC with the active Standard Interface and the cPDU,
 - the interface with the ZED camera and the local processing of the data
 - the data communication and routing with the OBCs, through its embedded SpW router.
- The cPDU: is responsible to provide the required power voltage to the internal components, as well as control the power transmission between the Standard Interfaces.
- The module is equipped with 1 active and 2 passive HOTDOCK interfaces.
- The ZED stereo Camera is the payload of the module, with the capability to render images from the surroundings of the spacecraft.

The module is movable by the WM and is initially attached to the SVC, through the active HOTDOCK interface.



Figure 5-27: ZED stereo camera

Figure 5-28 represents the SM5-OSP1 data architecture. It has the same design as SM1, except the presence of the ZED Camera, interfaced with the R-ICU by USB.

For the ZED camera payload, when the OSP Control software (running on the R-ICU) is commanded to *Operational* mode, it starts acquiring images from the camera using the driver interface of the Camera. The images are written to memory accessible through RMAP. The OSP Management component polls the *Frame Available* status of the OSP Control software via RMAP. When a frame is available, it commands the OSP Relay to read the image through RMAP and publishes it using the zeroMQ protocol to the OSP Visualization Component. This last receives each frame (ZeroMQ subscriber) and displays it in the GUI of the Visualization software on the MCC.



Detailed Design Document

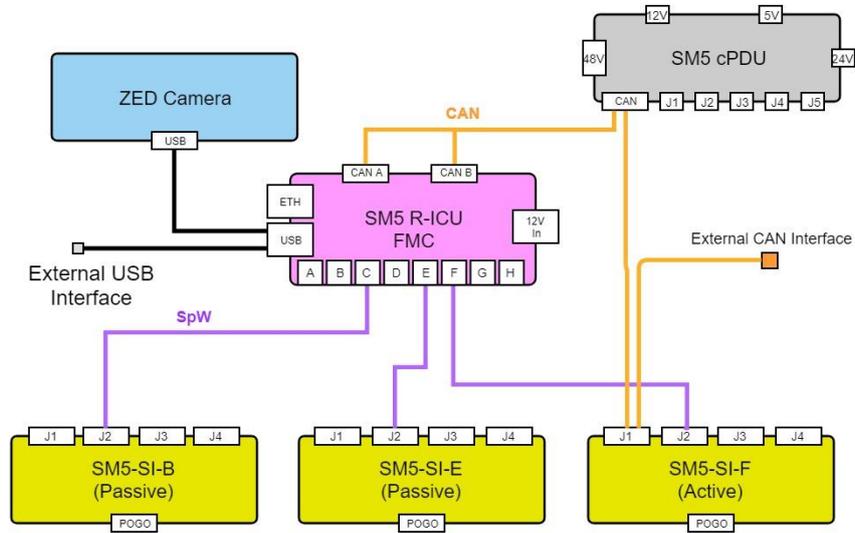


Figure 5-28: SM5-OSP1 data architecture

Figure 5-23 represents the SM3-BAT power architecture. The cPDU can distribute the 48V power bus along the different HOTDOCK interfaces. The module is equipped with an external power interface on the 48V bus, to replace the inputs from the HOTDOCK interfaces (for debugging purpose).

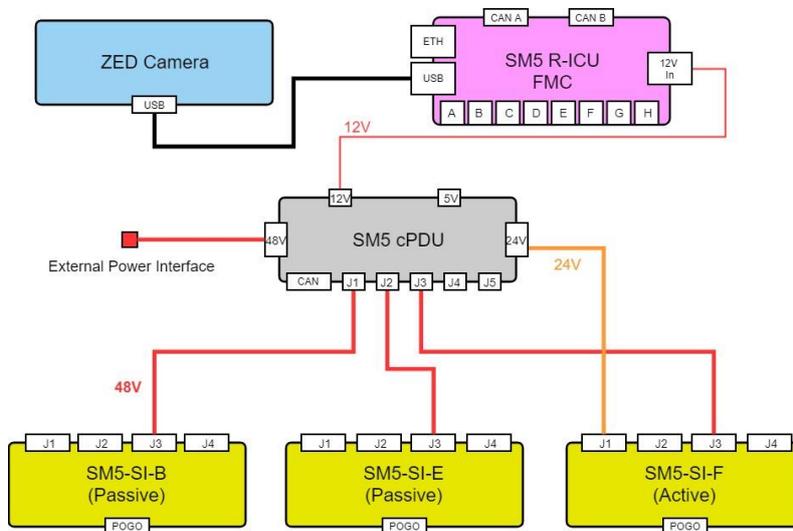


Figure 5-29: SM5-OSP1 power architecture



Detailed Design Document

5.4.6.2 SM5-OSP1 Mass Budget

The following table estimates the mass budget for SM5-OSP1 (including internal harness). Table 5-19: SM5-OSP1 mass budget

Components	Mass [kg]	Safety	Mass w/ safety [kg]
Mechanical Structure	3.38	1.1	3.73
R-ICU	0.65	1.1	0.72
cPDU	0.38	1.1	0.42
Payload	0.3	1.1	0.32
HOTDOCK	2.58	1.1	2.83
Total	7.29		8.00

5.4.6.3 SM5-OSP1 Power Budget

The following table estimates the power consumption of each active components of SM5-OSP1, and identify the required current and consumption for each internal cPDU power line.

Table 5-20: SM5-OSP1 power budget

Components	Rail Voltage [V]	Average Power [W]	Peak Power [W]	Line Average Current [A]	Line Peak Current [A]	Bus Average Current [A]	Bus Peak Current [A]
R-ICU / FMC Board	12	10	20	0.83	1.67	0.21	0.42
cPDU	48	4	7	0.08	0.15	0.08	0.15
SM5-SI-F	24	5	10	0.21	0.42	0.1	0.21
ZED Camera	12	3	3	0.25	0.25	0.06	0.06
Total	N.A.	22	40	N.A.	N.A.	0.45	0.84



Detailed Design Document

5.4.7 SM6-OSP2

5.4.7.1 Description

The SM6-OSP2 is representative of an optical payload module. It has a similar design than the SM5-OSP1 (section SM5-OSP1), except for the number of passive HOTDOCK interfaces that is here 3, in addition to the active one. As the SM5, this module is used as demonstration purpose, mainly to demonstrate the capability to replace a failed module.

We refer to section SM5-OSP1 for the description of the data and power architecture

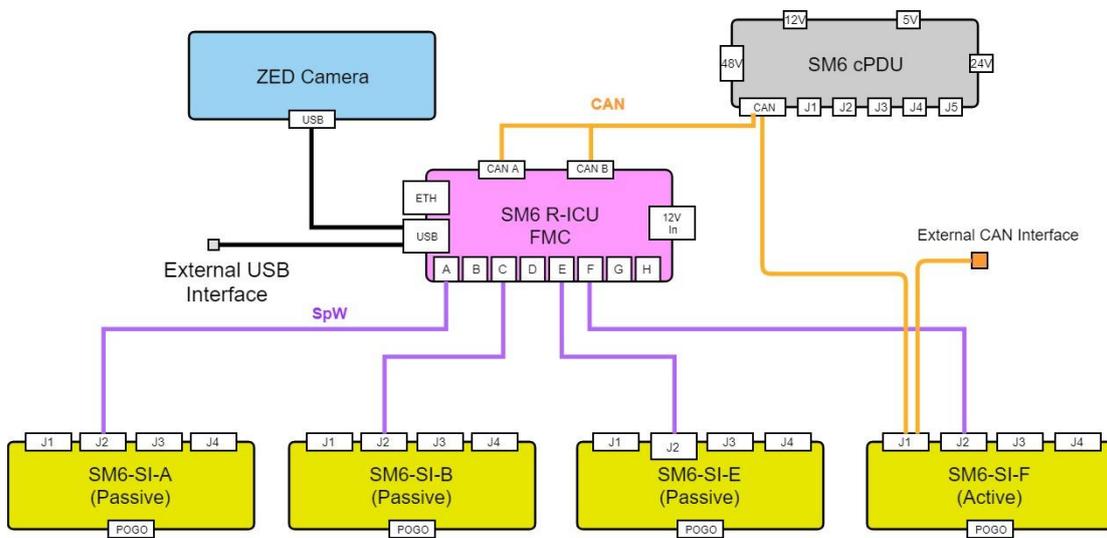


Figure 5-30: SM6-OSP2 data architecture

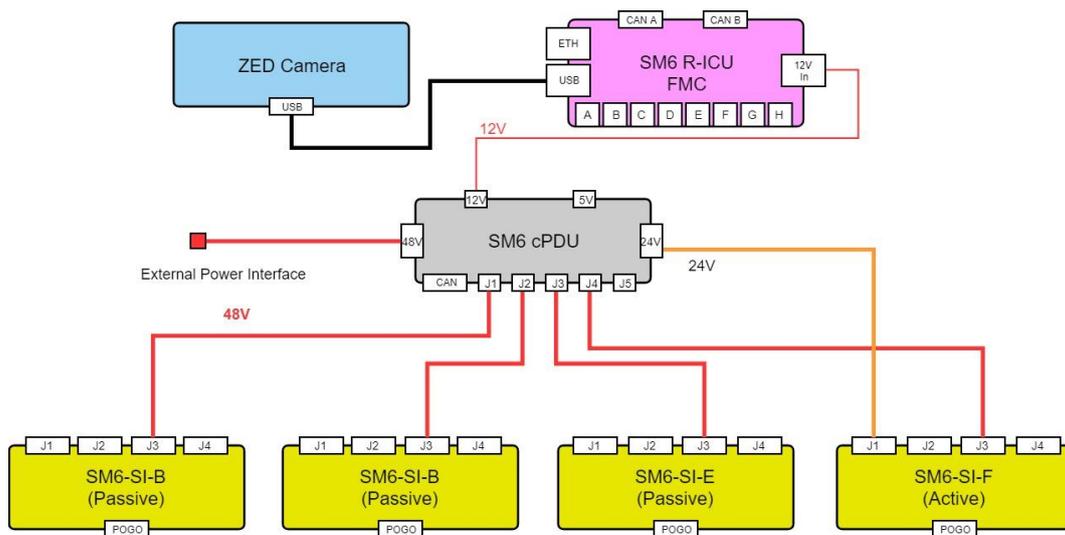


Figure 5-31: SM6-OSP2 power architecture



Detailed Design Document

5.4.7.2 SM6-OSP2 Mass Budget

The following table estimates the mass budget for SM6-OSP2 (including internal harness).

Table 5-21: SM6-OSP2 mass budget

Components	Mass [kg]	Safety	Mass w/ safety [kg]
Mechanical Structure	3.38	1.1	3.73
R-ICU	0.65	1.1	0.72
cPDU	0.38	1.1	0.42
Payload	0.3	1.1	0.32
HOTDOCK	3.49	1.1	3.84
Total	8.19		8.99

5.4.7.3 SM6-OSP2 Power Budget

The following table summarizes the power consumption of each active components of SM6-OSP2, and identify the required current and consumption for each internal cPDU power line.

Table 5-22: SM6-OSP2 power budget

Components	Rail Voltage [V]	Average Power [W]	Peak Power [W]	Line Average Current [A]	Line Peak Current [A]	Bus Average Current [A]	Bus Peak Current [A]
R-ICU / FMC Board	12	10	20	0.83	1.67	0.21	0.42
cPDU	48	4	7	0.08	0.15	0.08	0.15
SM6-SI-F	24	5	10	0.21	0.42	0.1	0.21
ZED Camera	12	3	3	0.25	0.25	0.06	0.06
Total	N.A.	22	40	N.A.	N.A.	0.45	0.84



5.5 SVC/CLT Buses

Two satellite buses are simulated in the frame of MOSAR. Respectively, 1 Servicer satellite bus (SVC) and 1 Client satellite bus (CLT). The Servicer Spacecraft (SVC) has the role to bring the new spacecraft modules and the WM up to the client satellite, and, after the docking operation, to manage the reconfiguration operations, based on an operation plan prepared and sent by the MCC. The Client satellite bus is a supporting structure for the spacecraft module, allowing the data and power routing distribution. For the MOSAR demonstration, the hypothesis is that the docking between the two spacecraft is already performed, as well as the data (SpW) and power connection.

5.5.1 Mechanical Design

The structures of the two buses will be simulated by the means of two test benches provided with the required HOTDOCK interfaces.

Two benches have been designed in MOSAR in order to simulate the SVC and CLT buses (Figure 5-32 SVC structure view). The benches will provide the required interfaces with the SM as well as with the WM and they will be featured with the required functionalities.

The parameters used as driver during the preliminary design phase of the bench are simplicity of the part manufacturing, flexibility of the integration of the benches and system versatility in order to avoid huge impact due to any accidental issue of a bench item during the simulation.

The mechanical parts composing the bench assembly are frame structure, interface panels, avionic shelf and structural cleats, as shown in Figure 5-32. As shown in the figure, each interface plate is independently integrated on the bus frame in so that they can be dismounted in case of problems with a minor impact on the entire system. Likewise, avionic shelves of the buses are designed in order to be fixed and unfix to the frame structure in any moment of the integration phase as well as if a problem will occur during the simulation.



Detailed Design Document



Figure 5-32 SVC structure view

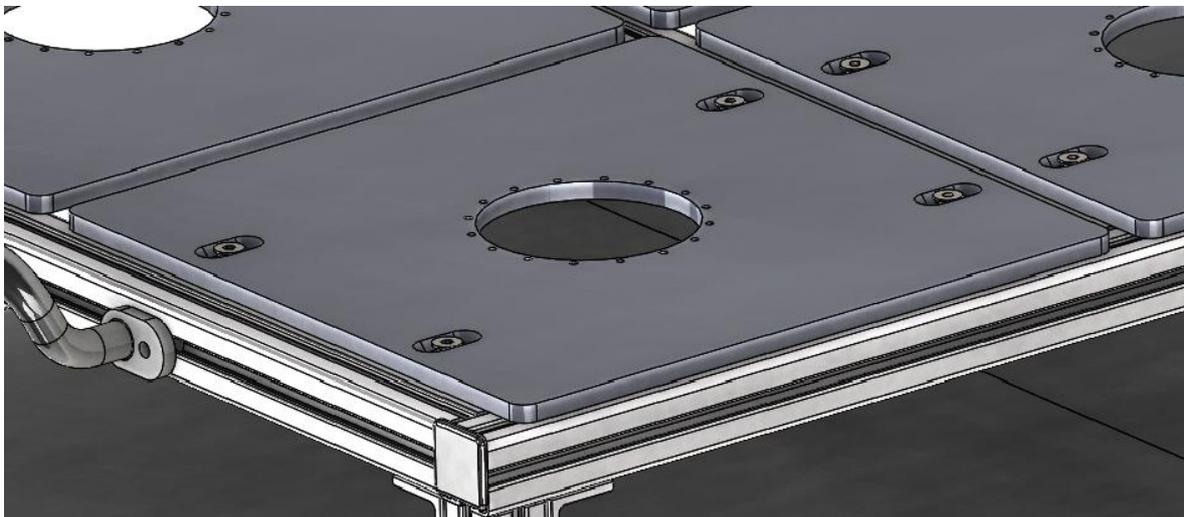


Figure 5-33: HOTDOCK interface plates

5.5.2 SVC Data and Power Architecture

Figure 5-34 represents the SVC/CLT data architecture. SM1 and SM2 are also included, to represent the static physical arrangement of the test setup.

The SVC bus includes the following elements:



Detailed Design Document

- The OBC-S that manages the reconfiguration operations both on the SVC and CLT, by controlling the WM motion, HOTDOCK interfaces and the power and data routing components. It interfaces the SpW bus through a USB SpW brick. It is also directly connected to the MCC, through Ethernet interface, to represent the space link between the ground and space segment. The SVC OBC-S is based on the NUC8i7INH platform (as in SM1), with a Linux operating system, running the Autonomy Agent, Components Managers and PUS services.
- The R-ICU controls the local cPDU, through the CAN interface, as well route the SpW signals to the CLT bus. With this configuration, the SVC has the possibility to take the full control of the client satellite, to perform the reconfiguration operations or other services.
- The SVC bus includes two passive HOTDOCK interfaces. As a storage solution during the transport of the modules, we could envisage to have only mechanical interfaces. However, the SVC is also bringing the WM. The passive interfaces are required to initiate the power and control with the robot. The two other mechanical interfaces are not represented, as they don't have any role in the data architecture.
- The cPDU ensures the power distribution to the R-ICU and OBC-S from the main power bus, and also the routing of the power to the passive HOTDOCK interfaces.

One of the port of the SVC R-ICU is used to communicate with the CLT bus through SpW, representing the data link of the docking interface. The SVC is powered by its own power supply (48V), which is, in the frame of the demonstrator, provided by the external EGSE.

5.5.3 CLT Data and Power Architecture

The CLT bus integrates only the elements related to its capability to route the data connection and power through its HOTDOCK interfaces. The control functionalities and power production/distribution are provide by the SM1 and SM2 modules. This approach has been selected to as much as possible illustrate the concept of modularity of the spacecraft. In practice, for the demonstration setup, these two modules will be permanently fixed to the structure (through representative HOTDOCK inter-distance). However, the architecture will propose the same level of data and power routing functionalities. Through the CLT R-ICU and cPDUs.

The CLT has 4 spot of HOTODOCK interfaces, allowing the connection of the additional modules or the walking manipulator, to support the re-configuration operations. One of the HOTDOCK is active to support the demonstration of multiple simultaneous HOTDOCK connections.

During nominal operation of the spacecraft (without the connection of the SVC), the power of the CLT components will be provided by SM2, while the control of the spacecraft will be managed by SM1.



Detailed Design Document

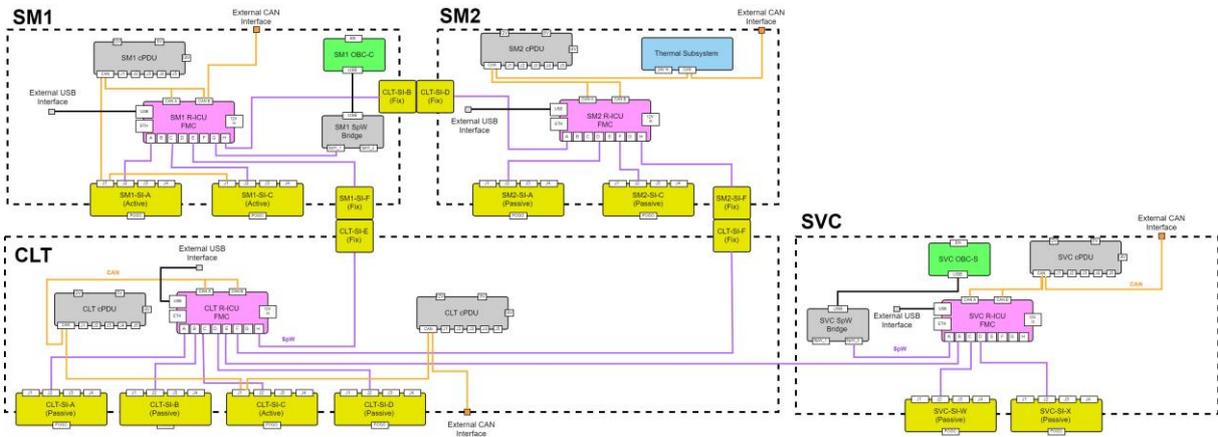


Figure 5-34: SVC/CLT data architecture (including fixed SM1 and SM2)

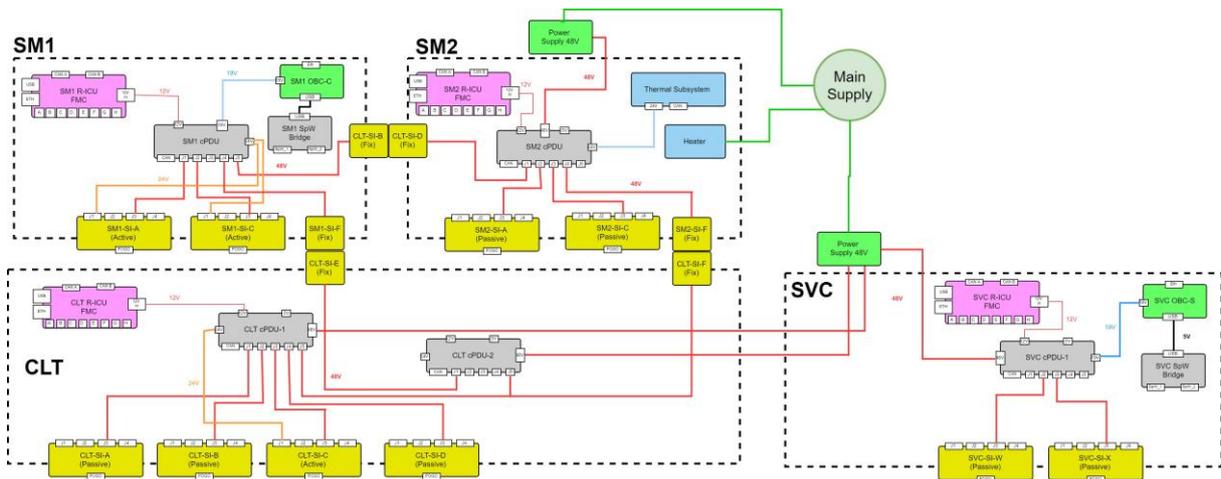


Figure 5-35: SVC/CLT power architecture (including fixed SM1 and SM2)



Detailed Design Document

5.5.4 SVC/CLT Power Budget

The following table summarizes the power consumption of each active components of the CLT/SVC, including the WM, and identify the required current and consumption for each internal cPDU power line. It has to be noted that power peaks are conservative, as not all components should be fully powered at the same time (see same comment in WM section 5.6).

Table 5-23: SVC/CLT power budget

Components	Rail Voltage [V]	Average Power [W]	Peak Power [W]	Line Average Current [A]	Line Peak Current [A]	Bus Average Current [A]	Bus Peak Current [A]
R-ICU / FMC Board SVC	12	10	20	0.83	1.67	0.21	0.42
R-ICU / FMC Board CLT	12	10	20	0.83	1.67	0.21	0.42
CLT cPDU 1	48	3	6	0.06	0.13	0.06	0.13
CLT cPDU 2	48	2	3	0.04	0.06	0.04	0.06
SVC cPDU	48	6	12	0.13	0.25	0.13	0.25
CLT-SI-F	24	5	10	0.21	0.42	0.1	0.21
SVC OBC	19	30	50	1.25	2.08	0.63	1.04
WM	48	77	380	1.6	7.92	1.6	7.92
Total	N.A.	143	501	N.A	N.A	2.98	10.44



Detailed Design Document

5.6 Walking Manipulator

5.6.1 Description

The MOSAR's walking manipulator aims at installing/releasing satellite modules and relocating itself over the satellite structures. The walking manipulator makes use of its end-effectors, equipped with HOTDOCK standard interfaces, to achieve its actions. Figure 5-36 illustrates an overview of the walking manipulator when moving a satellite module, as well as a representation of the preliminary design and components integration.

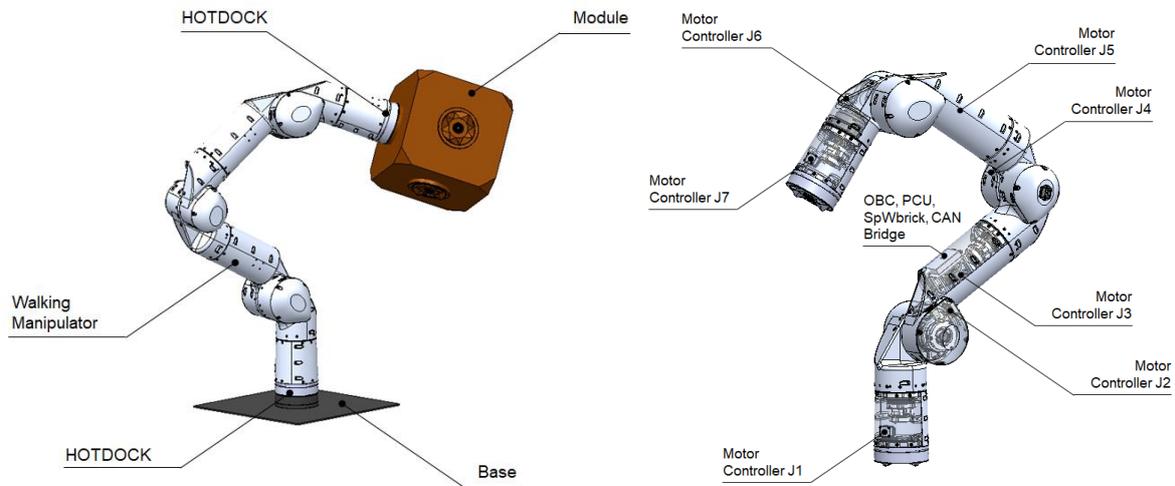


Figure 5-36: MOSAR walking manipulator

The walking manipulator kinematic is based on a human like structure with asymmetric joints, in order to optimize dexterity, range of motion and compactness. It presents a symmetric design between the base (side A) and the end-effector (side B) to allow self-relocation and walking, independently of the connected side.

The arm features seven active revolute joints based on brushless DC actuator and strain wave gear reducer. Each joint is equipped with position (incremental and absolute) and torque sensors to support the different control mode of operations, as well as an active brake.

Figure 5-37 illustrates the WM avionics architecture. The WM OBC Controller is the central component, playing the equivalent role as the R-ICU in the spacecraft modules (see Figure 2-10). It manages the control of the arm and the two end-effectors HOTDOC. As for the SM, the WM OBC interfaces the SpW bus (through USB/SpW brick) for TM/TC with the spacecraft OBC-S, through the RMAP protocol.

Each joint of the WM is equipped with a local controller for the closed loop position/current control of the actuator and the measurements of the joint sensors. All joint controllers are interfaced through an EtherCAT bus, managed by the WM OBC, which ensures the real-time exchange of information required for the control algorithm of the arm at 1 kHz.



Detailed Design Document

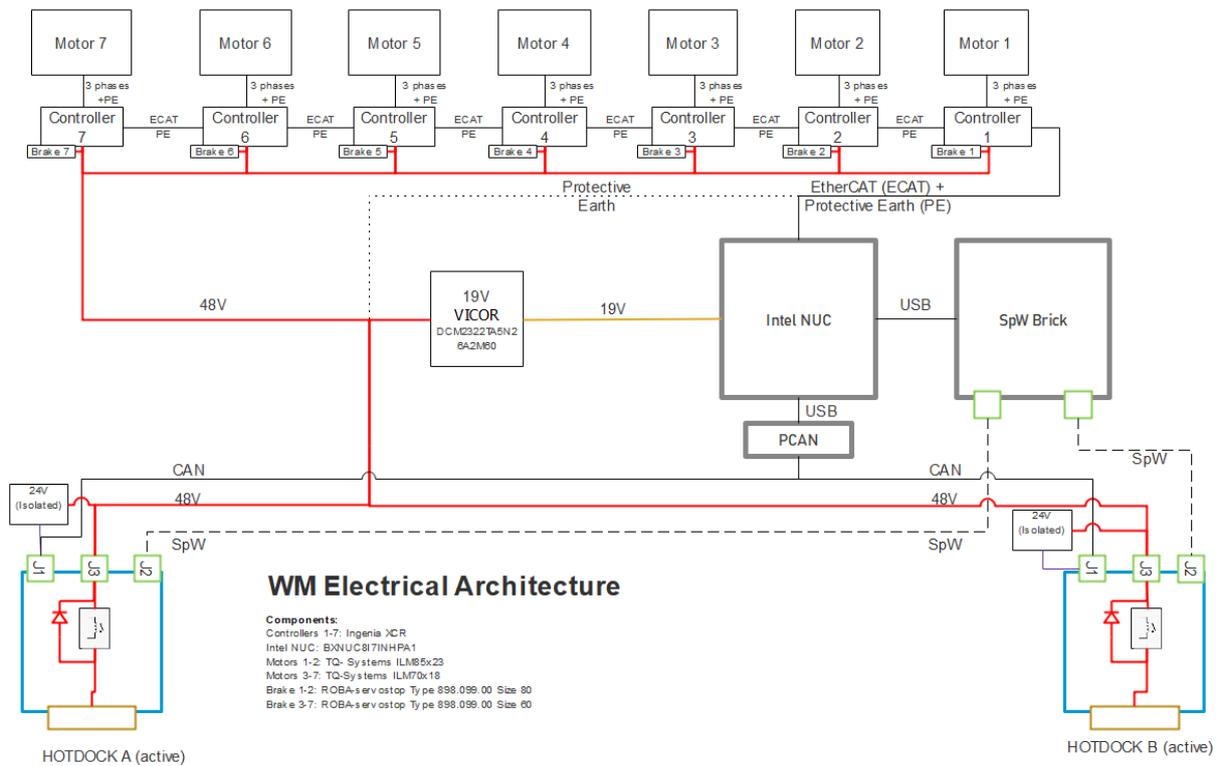


Figure 5-37: Walking Manipulator avionics / data architecture

5.6.2 EtherCAT route to space

The ISECG Technology Working Group Telerobotic Control of Systems with Time Delay published its newest Gap Assessment Report (RD9). Herein one aspect is related to communication requirements for any autonomous robot. The controlled system bandwidth is a very important performance indicator. It determines how precise, how efficient, and how fast a robot is able to execute its commands. Controlled systems are designed to achieve the highest possible bandwidth. In addition to sampling rates and data communication rates, the ability of feedback control software to handle robot dynamics plays a vital role in advancing controlled system bandwidth. This is particularly important for long and light robot manipulators.

In order to improve the controlled system bandwidth for advanced systems there is an inherent need for high-speed, low-latency data buses. This enables higher performance control loops and an overall increase in the self-reliance of robotic systems. Some operations require real-time performance, meaning that the travel times of signals need to be highly predictable. **The ISECG TWA values EtherCAT as optimized for high update cycles with a low jitter (highly relevant for robotics applications), real-time capable.** It criticizes the limit to 100 Mbit/s data rate. Today, EtherCAT G offers EtherCAT technology up to gigabit level and is fully compatible with EtherCAT core technology (RD10).

Future space robotics systems will benefit greatly from the introduction of bus technologies used in terrestrial industries and applications that meet the described requirements.

EtherCAT was already used by several projects on the International Space Station (ISS). Both in „Kontur 2“, a joint project of the German Aerospace Center DLR and the Russian Federal Space Agency ROSKOSMOS, as well as in the „Haptics“ projects of the European Space Agency (ESA) (RD11). The



Detailed Design Document

US MANTIS ISS payload will utilize Tethers Unlimited Inc. KRAKEN Robotic Arm. The internal arm communication also relies on the EtherCAT protocol (RD13).

The DLR Institute of Robotics and Mechatronics integrates the EtherCAT communication system in its 7dof space qualifiable robot arm CAESAR design for On-Orbit Services in LEO and GEO (RD12). The institute was testing the FB11xx containing the ET1100. For confidentiality reasons the final results are not fully published, but in case of a strong interest, Beckhoff will present the data.

Some test remarks:

- The tests rely on the ESCC22900 and ESCC25100 Standard-Rate"-Window: 4.84 Gy/h and. 254 Gy/h.
- Proton test: Fluence = $1.7E10$ Protons/cm²; Flux density = $2E7$ Protons/(cm².s); practice energies 68MeV, 50MeV, 30MeV
- Heavy Ion test ((Upset- und Latchup-Search) Fluence = $1E7$ Ionen/cm²; flux density = $35E3$ Ions/(cm².s); used Ions: Nitrogen (139 MeV), Iron (523 MeV), Krypton (768 MeV)

During radiation tests the system was operational and measured continuously status data of the ET1100 e.g. power (biased in-situ test).

For terrestrial applications Beckhoff provides an IPcore of the EtherCAT software as a commercial product. Currently, the DLR Institute of Robotics and Mechatronics is integrating this IPcore on a Polarfire-FPGA of Microchip, formerly Microsemi. The FPGAs are available as radiation hard and radiation tolerant.



5.7 HOTDOCK Standard Interface

5.7.1 General Overview

HOTDOCK is a standard robotic mating interface supporting mechanical, data, power and thermal transfer, in active and passive configurations (Figure 5-38). Its main application is to allow assembly and reconfiguration of spacecraft and payloads on-orbit and on planetary surfaces. It makes it straightforward to replace failed modules, or to swap payloads and provide chainable data interfaces for multiple module configurations.

A full feature HOTDOCK device provides the following interfaces, as illustrated in Figure 5-38:

- The mechanical interface that provides the alignment, connection and mechanical load transfer capabilities. It is composed of fixed elements (main body structure, form fit geometry) and a movable locking ring to allow connection with another device.
- The power interface, for the transfer of electrical power, through the central interface plate and POGO pin connectors.
- The data interface, for the transfer of CAN or SpW data, through the central interface plate and POGO pin connectors.
- The thermal interface, allowing fluid transfer between two devices.

HOTDOCK includes also its own internal PCB controller for local management (actuators, sensors, TM/TC communication) and connectors on the back, to access the power/data interface pins and the internal controller/powering of the device.

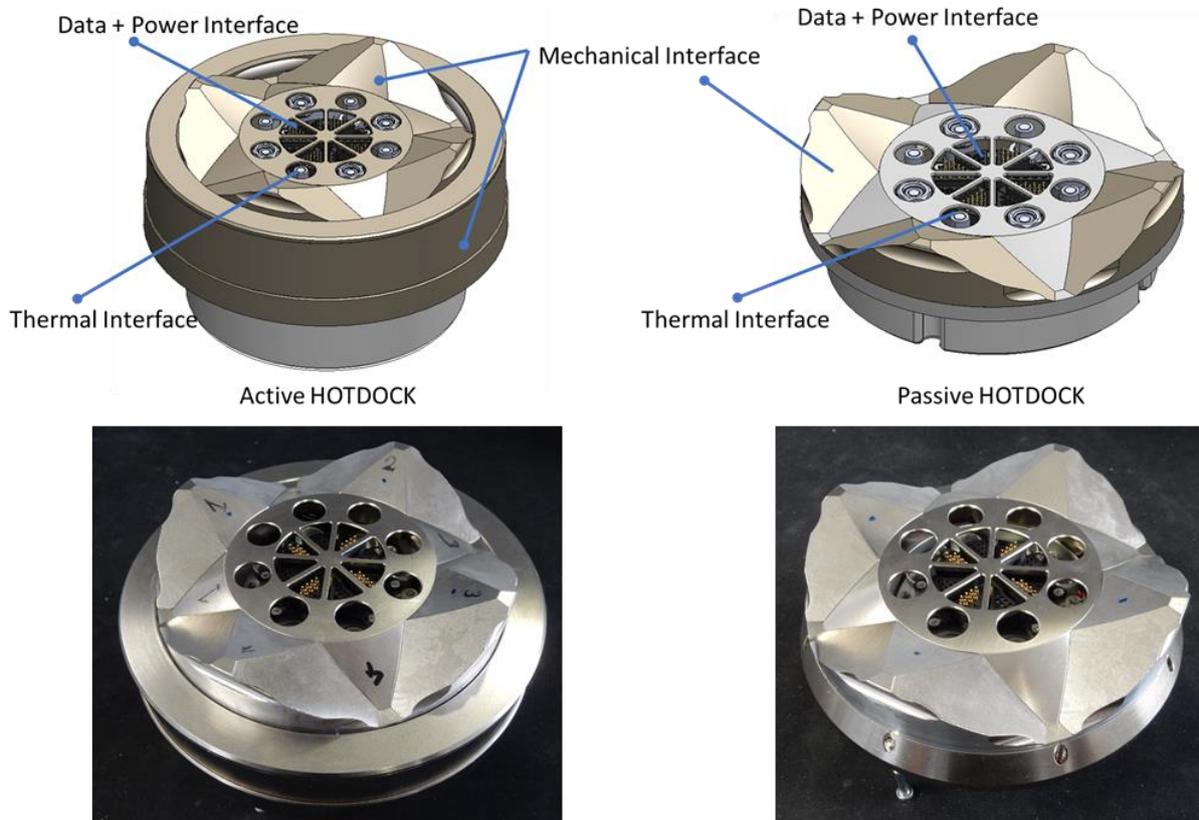


Figure 5-38: HOTDOCK Standard Interface (in Active and Passive configurations)



Detailed Design Document

5.7.2 HOTDOCK Declinations

HOTDOCK exists in different declinations, as function of the type of transfer capabilities.

Table 5-24 summarizes the different declinations:

- **HOTDOCK A (Active)**: provides the form-fit geometry, the active locking mechanism and the deployable power/data interface plate, with motor and PCB controller.
- **HOTDOCK AT (Active Thermal)**: active version including also thermal interface for fluid transfer
- **HOTDOCK P (Passive)**: simplified version of the active, which provides the form-fit geometry and a fixed power/data interface plate. There is no moving plate
- **HOTDOCK PT (Passive Thermal)**: passive version including also thermal interface
- **HOTDOCK M (Mechanical)**: simplified version, which provides only the for-fit geometry (no power/data interface plate or moving components)
- **HOTDOCK D (Dummy)**: 3D printed version for visual rendering (no mechanical load transfer)

Table 5-24: Features of the different declination of HOTDOCK

	Name	Visual	Mating	Mechanical Transmission	Data Transmission	Power Transmission	Thermal Transmission
	Active	✓	✓	✓	✓	✓	
	Active Thermal	✓	✓	✓	✓	✓	✓
	Passive	✓		✓	✓	✓	
	Mechanical	✓		✓			
	Dummy/Visual	✓					

The active declinations (HOTDOCK A and HOTDOCK AT) enable the mating process and are required to connect to another HOTDOCK device. The following connection options are possible, between two HOTDOCK declinations:

- **Active (Thermal) to Active (Thermal)**: for mechanical load, power, data (and Thermal) transfer
- **Active (Thermal) to Passive (Thermal)**: for mechanical load, power, data (and Thermal) transfer (the Active HOTDOCK plate is able to go in contact with the Passive connectors)
- **Active to Mechanical**: for mechanical load transfer

In terms of integration in the current projects, as well for future exploitation, the passive and mechanical versions have high interest, especially when talking about payload operations:

- They are smaller in height and lighter
- There is no active components (actuator, electronics)
- They are less expensive
- They be installed on a spacecraft/satellite with less integration constraints, waiting for an active one to connect to it (e.g. as part of a next mission). We think that this can be a great advantage for the early exploitation and diffusion of the technology (very low impact on spacecraft design)
- The reduced complexity and price of the passive/mechanical versions will facilitate the integration of more interfaces and connection points.



Detailed Design Document

- The mechanical version (without the central connector plate for data and power), can be envisaged when we need only to manipulate items (e.g. beam structures)

5.7.3 HOTDOCK Implementation in MOSAR

The HOTDOCK standard interface is used in MOSAR to inter-connect all the components of the system and enable their manipulation, fixation and data/power/thermal transfer:

- Equipped with HOTDOCK interfaces, the servicer spacecraft and the client satellite bus provides the structure for respectively the transfer of the spacecraft modules and assembly of the client satellite.
- The walking robotic arm manipulator allows the transfer and arrangement of the modules on the structures. Equipped with HOTDOCK interfaces at both end, it is able to connect to the modules, as well as, to walk along the spacecraft structure.
- The spacecraft modules, simulating payloads and equipment, are equipped with HOTDOCK interfaces to enable their connection to the spacecraft, the manipulator and the other modules.

The demonstration, foresees to demonstrate mechanical, data (SpaceWire), power and active fluid thermal transfer through HOTDOCK interfaces between the different components.

The following table provides the list and type of HOTDOCK interfaces embedded in the demonstrator for each MOSAR component (see Annex 6.1 for references):

Table 5-25: Standard interface selection for MOSAR components (O=optional)

SVC	W	X	Y	Z		
	Passive	Passive	Mech	Mech		
CLT	A	B	C	D	E	F
	Passive	Passive	Active	Passive	Fix	Fix
SM1-DMS	A	B	C	D	E	F
	Active	Fix	Active	Dummy (O)	Dummy (O)	Fix
SM2-PWS	A	B	C	D	E	F
	Thermal Passive	Dummy (O)	Passive	Fix	Dummy (O)	Fix
SM3-BAT	A	B	C	D	E	F
	Dummy (O)	Passive	Dummy (O)	Mech	Passive	Active
SM4-THS	A	B	C	D	E	F
		Dummy (O)	Passive	Mech	Passive	Thermal Active
SM5-OSP1	A	B	C	D	E	F
	Dummy(O)	Passive	Dummy(O)	Dummy (O)	Passive	Active
SM6-OSP2	A	B	C	D	E	F
	Passive	Passive	Mech	Dummy (O)	Passive	Active
WM	A	B				
	Active	Active				

Red letters highlight SM faces that are always accessible during the demonstration (e.g. for external interface access). The final amount of SI of each version is:



Detailed Design Document

Active	8
Active Thermal	1
Passive	15
Passive Thermal	1
Mechanical	5
Dummy	11 - Optional

5.7.4 HOTDOCK Detailed Design

The description of the HOTDOCK detailed design is provided in RD7 and RD8.

5.7.5 HOTDOCK Software Interface

The following tables provides the HOTDOCK CAN TM/TC definition, for the communication with the R-ICU or the WM Controller.

Table 5-26: HOTDOCK TM HK1 – Fast Polling Data

Field Offset	Field Name	Length (bits)	Units	Description
0	CUR_STATU	4	Enum	HOTDOCK Current State
1	REQ_STATU	4	Enum	HOTDOCK Requested State
2	RLY_STATU	1	Bool	Relay Status (Enabled/Disabled)
3	P25_STATU	1	Bool	P2V5 Status (Enabled/Disabled)
4	ORI_STATUS	2	Enum	Orientation Status (deduced from Proxy Hall Effects Sensors).
5	ABS_ENCOD	12	°	Absolute Encoder position
6	MON_TLM_I	16	mA	Power Transfer Current
7	MON_TLM_V	16	mV	Power Transfer Voltage
8	ERR_CODES	8	Enum	Mask coding error status
TOTAL:		64		

Table 5-27: HOTDOCK TM HK2 – Status Data

Field Offset	Field Name	Length (bits)	Units	Description
0	FSW_VER_1	4	N.A.	Flight SW Version - Major nbr.
1	FSW_VER_2	4	N.A.	Flight SW Version - Medium nbr.
2	FSW_VER_3	8	N.A.	Flight SW Version - Minor nbr.
3	AUTO_TM_S	4	Hz	Auto TM [Hz], default: 0 (disabled)
4	MUX_PIN_1	4	Enum	Mux Value for ADC sensors
5	DIV_PRX_1	8	%	Proximity Hall Effect 1
6	DIV_PRX_2	8	%	Proximity Hall Effect 2
7	DIV_PRX_3	8	%	Proximity Hall Effect 3



Detailed Design Document

8	DIV_PRX_4	8	%	Proximity Hall Effect 4
9	LVDS_PIN_1	4	Enum	LVDS-1 Mux Pins Values
10	LVDS_PIN_2	4	Enum	LVDS-1 Mux Pins Values
	TOTAL:	64		

Table 5-28: HOTDOCK TM HK3 – Temperatures Data 1

Field Offset	Field Name	Length (bits)	Units	Description
0	TMP_CPU_1	16	°C	SAM V71 CPU Temperature
1	TMP_NTC_1	16	°C	Temperature NTC 1 (located @....)
2	TMP_NTC_2	16	°C	Temperature NTC 2 (located @....)
3	TMP_NTC_3	16	°C	Temperature NTC 3 (located @....)
	TOTAL:	64		

Table 5-29: HOTDOCK TM HK4 – Temperatures Data 2

Field Offset	Field Name	Length (bits)	Units	Description
0	TMP_NTC_4	16	°C	Temperature NTC 4 (located @....)
1	TMP_NTC_5	16	°C	Temperature NTC 5 (located @....)
2	TMP_NTC_6	16	°C	Temperature NTC 6 (located @....)
3	TMP_NTC_7	16	°C	Temperature NTC 7 (located @....)
	TOTAL:	64		

Table 5-30: HOTDOCK TM HK5 – Motor Data (Debug Only)

Field Offset	Field Name	Length (bits)	Units	Description
0	MON_VBUS1	16	mV	Monitoring of the VBUS 1
1	FAULT_MOT	1	Bool	Fault MC Motor
2	HALL_EF_A	1	Bool	Hall Effect A
3	HALL_EF_B	1	Bool	Hall Effect B
4	HALL_EF_C	1	Bool	Hall Effect C
5	I_MOTOR_1	16	mA	Motor Current 1
6	MON_REC_V	16	mV	REC Neutre point Voltage
	TOTAL:	52		



Detailed Design Document

Table 5-31: HOTDOCK TMC Codes – Packet Types List

			CAN TMC Codes - Packet Types List	
Type Code	Dir.	Type Label	Payload Fields Nbr.	Description
TELECOMMAND	TC			
0x00	TC	TC_UNK	0	Unknown value (Unused)
0x01	TC	TC_HK1	0	HK1 Request
0x02	TC	TC_HK2	0	HK2 Request
0x03	TC	TC_HK3	0	HK3 Request
0x04	TC	TC_HK4	0	HK4 Request
0x05	TC	TC_HK5	0	HK5 Request
0x06	TC	TC_HKA	0	All HK Request
0x07	TC	TC_GOT	1	Go to State +1 payload field
0x08	TC	TC_ATM	1	Auto TM [HZ] +1 payload field
0x09	TC	TC_P2V	1	Enable/Disable P2V5
0x0A	TC	TC_RLY	1	Enable/Disable Relay
0x0B	TC	TC_LVD	2	Set LVDS Pins
0x0C	TC	TC_STO	0	Emergency Stop (motor)
TELEMETRY	TM			
0x80	TM	TM_UNK		Unknown value (Unused)
0x81	TM	TM_ACK		Acknowledged Pckt
0x82	TM	TM_NAK		Not Acknowledged Pckt
0x83	TM	TM_HK1		HK1 Response
0x84	TM	TM_HK2		HK2 Response
0x85	TM	TM_HK3		HK3 Response
0x86	TM	TM_HK4		HK4 Response
0x87	TM	TM_HK5		HK5 Response



Detailed Design Document

5.8 Centralized Power Distribution Unit

5.8.1 Description

In MOSAR, power distribution is achieved by the transfer of the electrical energy between the SM through the connected HOTDOCK interfaces. In order to configure the power distribution, each SM is equipped with a centralized power (and conditioning) distribution unit (cPDU), that provides the following functions:

- Power distribution between the HOTDOCK interfaces of the SM / WM
- DC/DC conversion from the main power bus to power the internal components of the SM/WM

The cPDU is the main component implicated in the power re-configuration operations that are used during the spacecraft re-configuration (managed by the OBC-S) or during nominal operations (managed by the OBC-C, e.g. for SM isolation in case of failure). The cPDU power architecture is illustrated below.

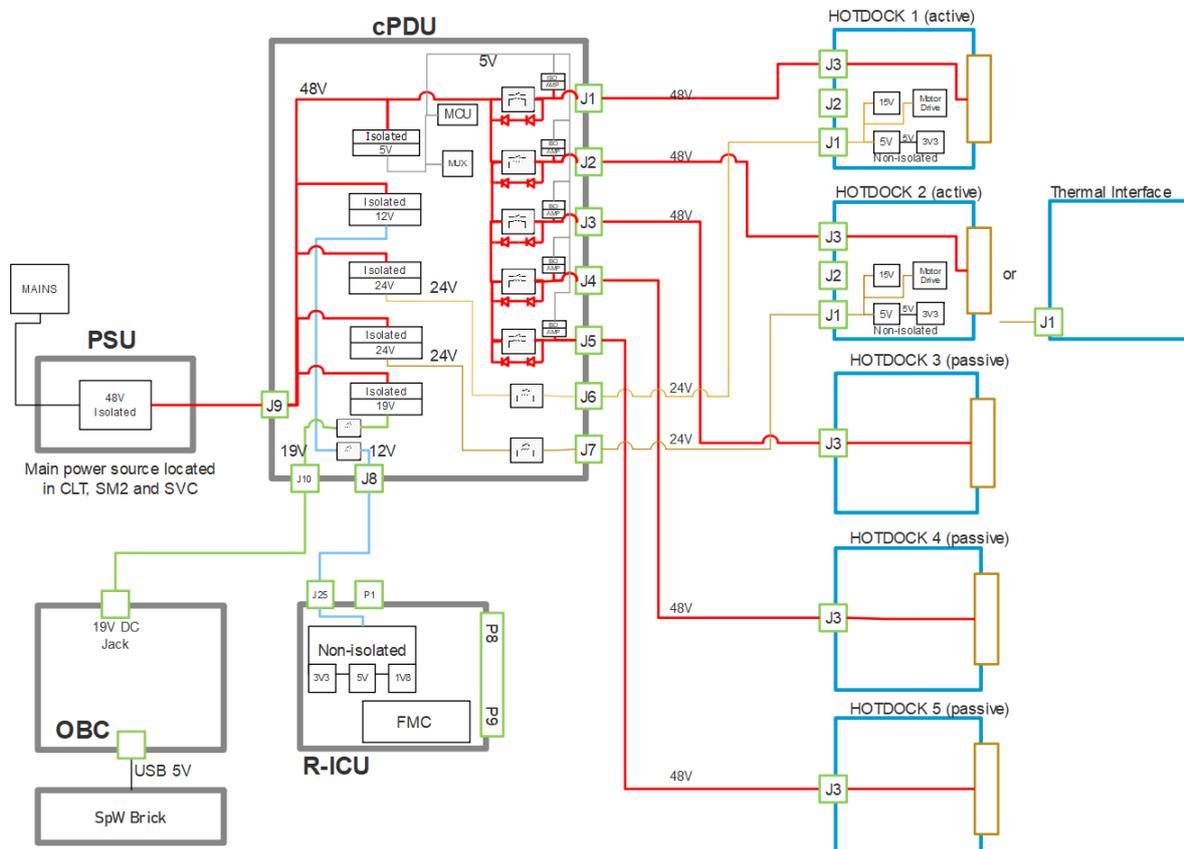


Figure 5-39: cPDU power architecture

Power switching capability is implemented for each HOTDOCK connection, to allow full control of power flow during assembly and at final state. Each HOTDOCK power line (J1 to J5) is equipped with a power relay which is bypassed with a diode. The relay provides the control of current flowing out of the SM, while the diode allows enough power to be transferred into the SM during first steps of power-up.

The cPDU can be fully controlled from the R-ICU through CAN TMTTC. In order to minimise inrush current when powering a new part of the modular spacecraft, the cPDU controller implements a set of switches



Detailed Design Document

that enable subsystem in a sequence. The sequential power-up also allows the cPDU to perform internal checks at each step.

cPDU will have different configurations, as function of the required DC/DC conversion for each of the spacecraft modules or spacecraft buses. Each DC/DC line has a relay, enabling the control of the powering of the SM internal component (e.g. R-ICU, payloads....).

5.8.2 cPDU Software Interface

The following tables provides the cPDU CAN TM/TC definition, for the communication with the R-ICU.

Table 5-32: cPDU TM HK1 – Fast Polling Data

Field Offset	Field Name	Length (bits)	Units	Description
0	STATUS	4	Enum	cPDU Status
1	J1_RLY_STATUS	1	Bool	J1 Relay Status (Enabled/Disabled)
2	J2_RLY_STATUS	1	Bool	J2 Relay Status (Enabled/Disabled)
3	J3_RLY_STATUS	1	Bool	J3 Relay Status (Enabled/Disabled)
4	J4_RLY_STATUS	1	Bool	J4 Relay Status (Enabled/Disabled)
5	J5_RLY_STATUS	1	Bool	J5 Relay Status (Enabled/Disabled)
6	12V_RLY_STATUS	1	Bool	DC 12V Relay Status (Enabled/Disabled)
7	24V_RLY_A_STATUS	1	Bool	DC 24V A Relay Status (Enabled/Disabled)
8	24V_RLY_B_STATUS	1	Bool	DC 24V B Relay Status (Enabled/Disabled)
9	LOAD_RLY_STATUS	1	Bool	Load Relay Status (Enabled/Disabled)
10	J1_TLM_V	8	V	J1 Voltage
11	J2_TLM_V	8	V	J2 Voltage
12	J3_TLM_V	8	V	J3 Voltage
13	J4_TLM_V	8	V	J4 Voltage
14	J5_TLM_V	8	V	J5 Voltage
15	ERR_CODES	8	Enum	Mask coding error status
	TOTAL:	61		

Table 5-33: cPDU TM HK2 – Status Data

Field Offset	Field Name	Length (bits)	Units	Description
0	FSW_VER_1	4	N.A.	Flight SW Version - Major nbr.
1	FSW_VER_2	4	N.A.	Flight SW Version - Medium nbr.
2	FSW_VER_3	8	N.A.	Flight SW Version - Minor nbr.
3	AUTO_TM_S	4	Hz	Auto TM [Hz], default: 0 (disabled)
	TOTAL:	20		



Detailed Design Document

Table 5-34: cPDU TMC Codes – Packet Types List

			CAN TMC Codes - Packet Types List	
Type Code	Dir.	Type Label	Payload Fields Nbr.	Description
TELECOMMAND	TC			
0x00	TC	TC_UNK	0	Unknown value (Unused)
0x01	TC	TC_HK1	0	HK1 Request
0x02	TC	TC_HK2	0	HK2 Request
0x06	TC	TC_HKA	0	All HK Request
0x08	TC	TC_ATM	1	Auto TM [HZ] +1 payload field
0x10	TC	TC_J1	1	Enable/Disable J1 switch
0x11	TC	TC_J2	1	Enable/Disable J2 switch
0x12	TC	TC_J3	1	Enable/Disable J3 switch
0x13	TC	TC_J4	1	Enable/Disable J4 switch
0x14	TC	TC_J5	1	Enable/Disable J5 switch
0x15	TC	TC_12V	1	Enable/Disable 12V line
0x16	TC	TC_24V_A	1	Enable/Disable 24V line A
0x17	TC	TC_12V_B	1	Enable/Disable 24V line B
0x18	TC	TC_Load	1	Enable/Disable Load line
TELEMETRY	TM			
0x80	TM	TM_UNK		Unknown value (Unused)
0x81	TM	TM_ACK		Acknowledged Pckt
0x82	TM	TM_NAK		Not Acknowledged Pckt
0x83	TM	TM_HK1		HK1 Response
0x84	TM	TM_HK2		HK2 Response



5.9 R-ICU and FMC Board

5.9.1 SpaceWire and CAN Interface FMC Card

The FMC add-on card developed for MOSAR adds eight SpaceWire ports to the unit, two CAN interfaces, 16 LEDs, 8 GPI pins and 8 GPO pins. All IOs and SpW interfaces are buffered on the FMC card. The FMC card is shown installed on the Trenz carrier card in Figure 5-40 with the two long IDC connectors interfacing to the SpaceWire and the short IDC interfaces to CANA and CANB. The GPI and GPO ports are located on a connector to the far left of the card.

The FMC cards have been tested as according to the test procedures detailed in RD5. The test setup tests all IO interfaces supported by the FMC card. As part of these tests and the RMAP IP is also tested as RMAP is used to interface with the LEDs in the test procedure. Test cables have also been produced which break the SpaceWire ports out into micro-D connectors for use with other SpaceWire equipment and a CAN interface breakout cable allows CAN devices to be prototyped with the system.

The user guide for the R-ICU development platform, detailing the use of these interfaces and cables along with HW/SW programming image creation is detailed in RD1.

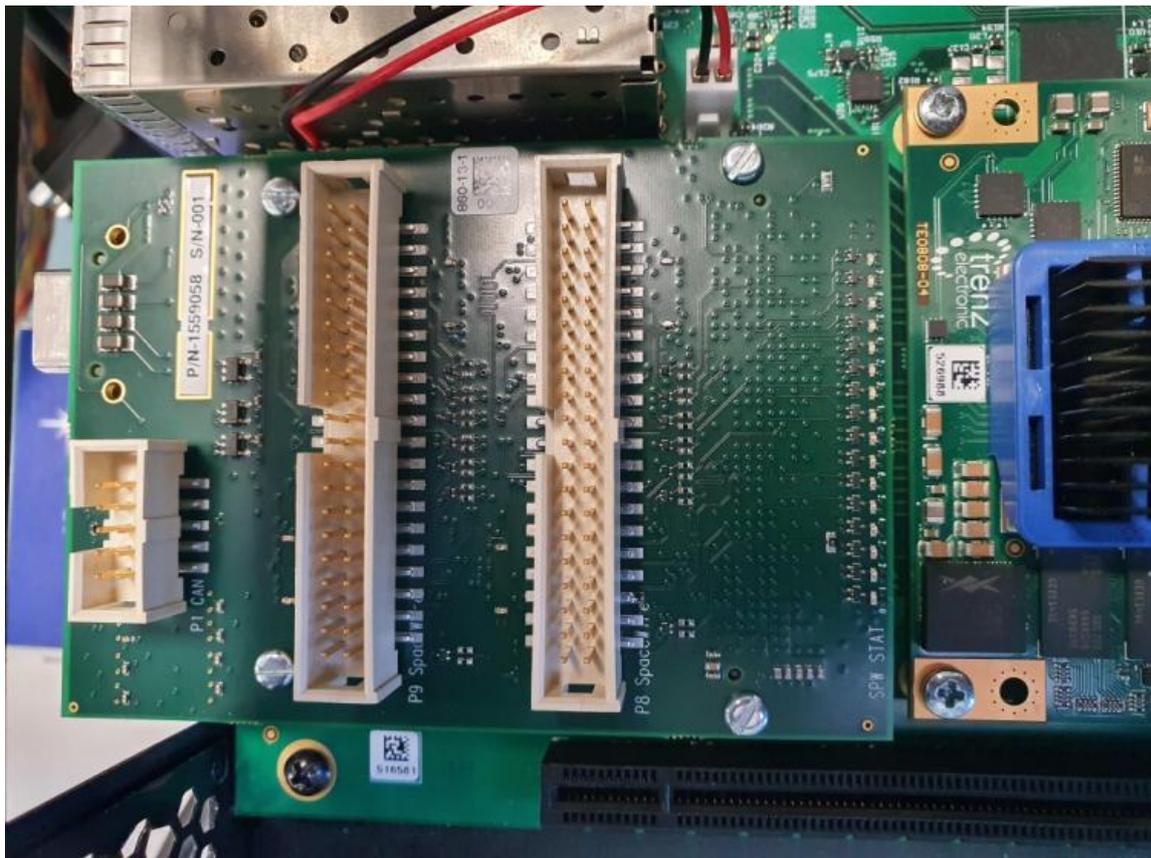


Figure 5-40 SpW and CAN FMC interface card installed on R-ICU carrier board

5.9.2 R-ICU Hardware Design

The R-ICU hardware is split between the built-in interfaces of the Zynq MPSoC and the SpaceWire interfaces implemented in the FPGA fabric of the Zynq. The general hardware architecture is shown in Figure 5-41. All SpaceWire IP cores are connected to the cache-coherent AXI interface of the MPSoC,



Detailed Design Document

allowing direct DDR4 access between the processing cores of the MPSoC and the DMA interfaces of the SpaceWire interfaces. This allows the RMAP interface direct access to the memory spaces for the TM/TC of the components running on the R5 processors and direct access to the optical payload outputs from the I3DS framework.

Time synchronisation is managed across the MOSAR system by SpaceWire timecodes. On reception of a timecode by a SpaceWire IP core, the time code is forwarded on to the other SpaceWire interfaces and internally to the timecode IP core. The timecode IP core accepts a time code if it is new to the system (i.e. has a larger value than a previously received timecode) and is used to synchronise with a local global tick counter which is running free.

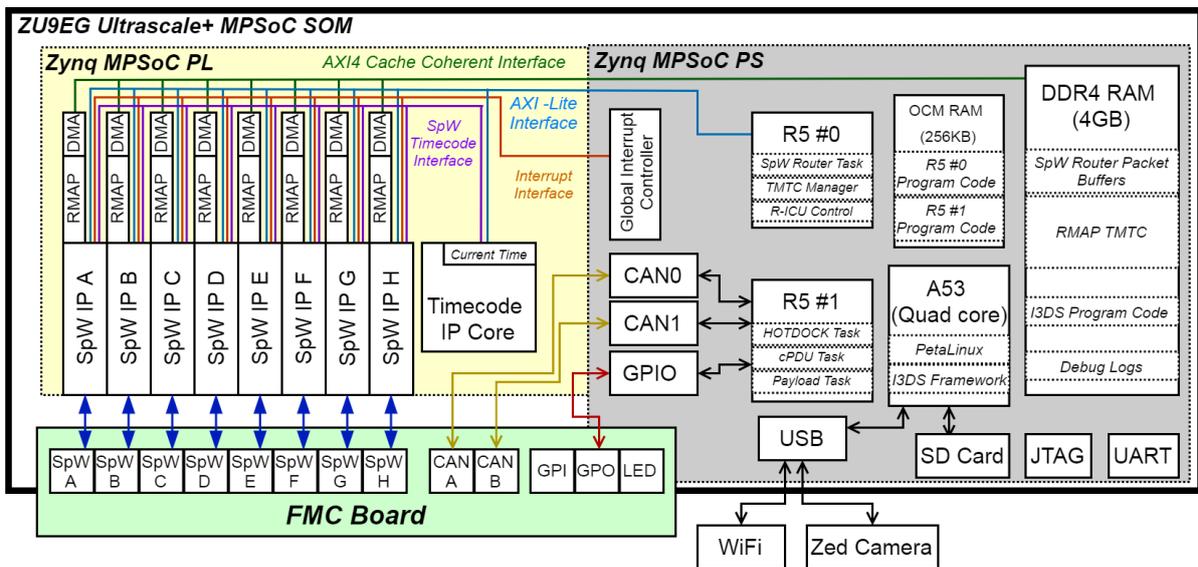


Figure 5-41 Hardware Design and Zynq MPSoC Components for the MOSAR R-ICU

The TAS-UK SpaceWire RMAP IP core has had several MOSAR specific extensions added to its functionality. Firstly is header delete functionality in the data payload of SpaceWire packets to allow the transmission and reception of SpaceWire packets to be done exclusively with DMA transfers, this significantly lowers to re-transmission latency despite the packet-switching nature of the MOSAR SpW router.

To support the RMAP-based TM/TC of MOSAR, the IP core has been further extended by adding support for RMAP transaction authorisation. An interrupt is issued when a RMAP header is received and the processing of the RMAP packet is paused. The processor can then check that the access address is valid and that the TM/TC residing at this location isn't currently being accessed by the R-ICU. If the TM/TC is being accessed then the RMAP transaction is stalled until the memory area is available. If the RMAP transaction address is not valid (does not line up with a valid TM/TC buffer address) or the RMAP pipeline stall times-out, then the RMAP IP core is instructed to terminate the RMAP transaction and send a "Command not valid or authorised" reply. This is in line with the RMAP command sequence for both read and write: stage 3 and 4 write/read data request/authorisation, as part of the RMAP ECSS standard ECSS-E-ST-50.

The CAN interfaces for SM platform and payload component control (HOTDOCK, cPDU, battery payload, Thermal payload) are provided by the CAN controller built in to the Zynq Ultrascale+ MPSoC. All CAN devices share the same CAN bus with the R-ICU's second CAN controller provided as a redundant option in a cross-strapped configuration for the flight version of MOSAR.



Detailed Design Document

The MPSoC USB controller is included for access to the Zed Camera and also to use a USB WiFi dongle for a debugging connection to the I3DS framework.

The hardware configuration and software images are loaded from the SD-card located on the Trenz carrier card by the Zynq MPSoC bootloader. This is configured to load the R5 software into the On-Chip-Memory (OCM) of the Processing System (PS), which can be isolated from the TMTC and SpaceWire router packet data contained in DDR4. Therefore, any buffer overflows in the TM/TC, incorrect RMAP writes or buffer overflows in the SpaceWire router DMAs can not affect the operation of the R5s. Due to the use of Petalinux in the I3DS framework, the DDR4 will be need be loaded with the I3DS program code. A Memory protection unit is present on the MPSoC and this could be configured to protect write access to the I3DS program code if needed.

The USB WiFi access would also allow remote update of hardware, R5 software and I3DS framework upgrades by writing new images to the SD card that can then be booted. This feature will be necessary if remote integration tasks are to be undertaken by partners that develop components that are connected to the R-ICU.

5.9.3 R-ICU Software Architecture

The R-ICU software is split between three processing systems, as shown in Figure 5-42. The allocation of software task to processor is determined by the hardware components that the task is responsible for managing and to partition the system to support easy addition of payload controllers or high-performance software applications in future systems:

1. R5 #0: The first R5 processor in the MPSoC is used for supporting the SpaceWire interface to the R-ICU and the TM/TC interface that is run on top of this using RMAP. As part of this role, the software SpaceWire router task is also run on this processor. As both the TMTC manager and the SpaceWire router need to access the registers of the SpaceWire IP cores, locating them on the same processor can ensure that they are protected using mutual exclusion to ensure registers are not modified whilst a routing or TMTC request is being handled. The R-ICU management task is also hosted by this processor as it will have modest processing demands and allows R5#1 to be dedicated purely to platform/payload software.
2. R5 #1: The second R5 processor hosts platform and payload control software for managing the SM specific devices attached to an R-ICU that provide a platform or payload role in the MOSAR spacecraft. This includes low-level control of: HOTDOCKs, cPDU, battery payload and thermal payload. Currently all of these devices are interfaced via CAN-bus and so this R5 is also responsible for managing the CAN controller of the MPSoC.
3. A53 #0 - #3: The quad-core A53 runs the I3DS framework which supports a number of sensing interfaces and sensor processing algorithms. The main role of this processor in the MOSAR demonstrators is to support the optical payloads. Functionality that the R5s undertake could be undertaken by this processor, especially as in the other SMs it is not used to support the payload. However, the A53 system is by far the most highest-performance aspect of the MPSoC and so the most power-consuming, therefore for SMs that do not need the high-performance aspect of the A53 it can be left powered down.

At this stage, the spare capacity of A53 also provides a role in developing the system by providing a debug interface that is accessible by a USB WiFi dongle and the SSH capabilities of Petalinux. This allows remote (including off-site) debugging of the SM component drivers running on the R5s through a live logging interface and also remote upload of new R-ICU software and programmable hardware (PL) images.



Detailed Design Document

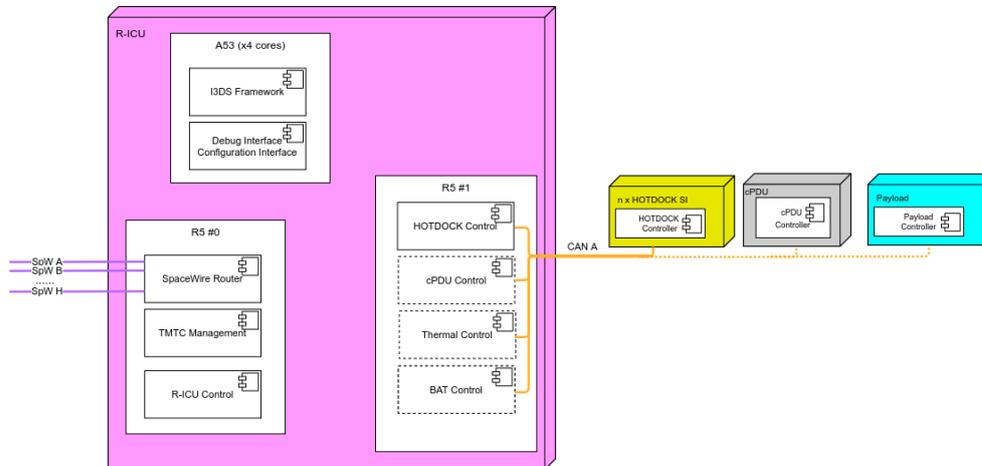


Figure 5-42 General overview of R-ICU software architecture

The interaction of these software tasks between the processors is to be achieved through the OpenAMP capabilities provided by the MPSoC. On each R5, FreeRTOS is used to schedule tasks with pre-emptive scheduling. This allows the support of both priorities to be given to tasks which require low response latency (e.g. SpW routing) and also support of periodic driven tasks (e.g. housekeeping data collection for TM or logging) to ease development where such aspects are required. Interrupts are deferred into message queues by the R5 interrupt handler to ensure that long ISR functionality does not bypass any scheduler constraints and to enable the scheduler to switch to a high priority task if the interrupt unblocks it.

Message queues are used to pass information between tasks and to distribute deferred interrupts. This allows a software design where tasks can block on a message queue until its service is required. This suits a TM/TC-driven approach well, the TM/TC manager will generate messages to the relevant component when TC requests are received and can use the same message queue to inform a component task when a TM is read by the OBC. A high level view of these queues and how they interact with the software tasks is shown in Figure 5-43 (inter-processor communication is currently not presented in this diagram but will take a similar message-based form).



Detailed Design Document

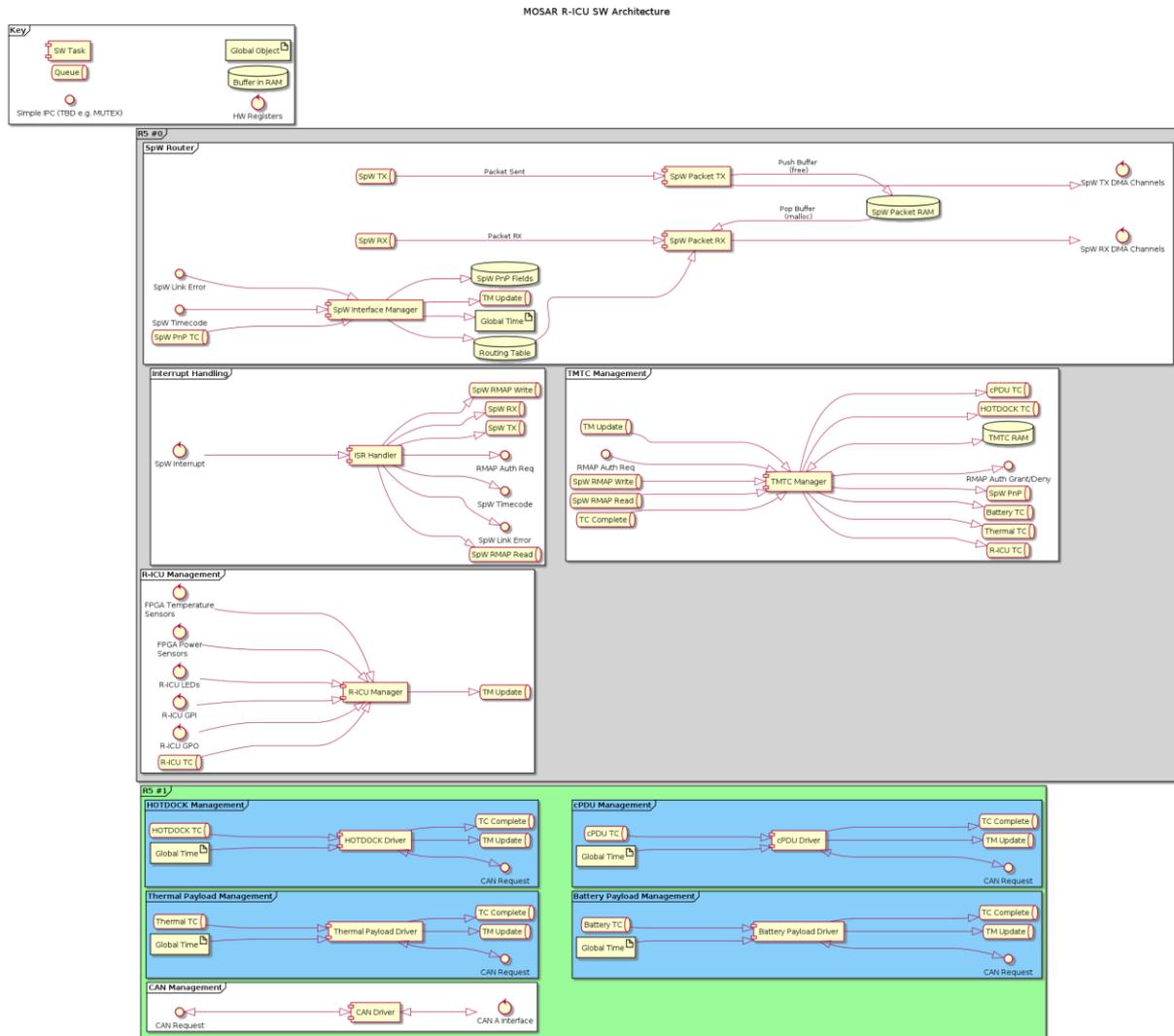


Figure 5-43 R-ICU detailed software architecture

5.9.4 SpaceWire PnP Support

The role of the automatic network discovery and reconfiguration in MOSAR is to detect faults in the SpaceWire network and reconfigure the network if required to route around faulty interfaces. The network reconfiguration process is executed by the CLT-OBC with the R-ICUs responding to network discovery TMTC to provide information about the capabilities of the SM, status of the SpaceWire router and allow settings such as the SpaceWire routing tables to be configured by the OBCs. An adaptation of the draft SpaceWire Plug and Play Protocol standard (22/03/2013) is implemented on each SM for SM capability and network status detection, this ensures that there is a potential future route to flight for the discovery process.

The R-ICUs in MOSAR are a hybrid device in the SpW PnP context as they act as both a router and a device endpoint. Therefore integration with other SpW PnP systems may present a challenge. However, as there are no other SpW PnP devices on the MOSAR network other than the R-ICUs, this is not seen as an issue. The use of SpW PnP register set is as follows:



Detailed Design Document

- a. Device Identification – used to show status of router, status HOTDOCK SpW connections. This register set is read by the network discovery process to discover what SpW links are active.
- b. Vendor/Product Strings – a textual representation of the SM configuration and component types attached (e.g. "SM2: R-ICU, HOTDOCK-A-PASSIVE, HOTDOCK-C-PASSIVE, HOTDOCK-D-FIXED, HOTDOCK-F-FIXED, THS PWS")
- c. Protocol Support – List of RMAP TMTC protocols that the R-ICU supports. These represent the ASN.1 file that should be used to access an SM component's TMTC interface.
- d. Application Support – This provides the list of R-ICU components that the SM implements and which RMAP TMTC protocol is used to interact with each one

These fields suffice to

- a) support discovery of the network topology
- b) support discovery of the capabilities of each SM.

In conjunction with the protocol support field it would be possible for an OBC to probe and communicate with an unknown SM if it supported all of the RMAP TMTC protocols that an unknown SM hosted.



5.10 Visual Processing System

The aim of the visual processing system is the structural inspection of the spacecraft modules and the validation of the final spacecraft reconfiguration.

As reported in [RD3] we highlighted three specific software functionalities:

- Detection of anomalies on the surface and interface of the spacecraft modules;
- Verification of correct reconfiguration of the satellite modules;
- Detection of anomalies on the surface and interface of the walking manipulator.

These functions will be limited by the minimum detectable anomaly, which will depend on the camera resolution and the numerical errors introduced by the algorithms. While it is possible to accept surface anomalies of the order of centimetres on the exterior of the spacecraft module and the walking manipulator, greater accuracy is required at the interface level, as deviations of the order of millimetres no longer allow modules to be assembled together.

Hence, our target accuracy depends on the specifications as follows:

- We aim to detect anomalies of 1cm on the surface of the spacecraft modules and walking manipulator;
- We aim to detect anomalies of 5mm on the spacecraft module and walking manipulator interfaces;
- We aim to detect anomalies of 1cm on the position of the modules after reconfiguration.

As occurrence of damage during operation is not expected, the detection of anomalies could be ideally done only once during operation. However, only a partial view of the spacecraft is available at one time, and the system is able to capture multiple views of an object as they move in the field of view. Hence, multiple detection attempts can be performed. The frequency of significant changes in the observed scene is not high, and it is possible to halt the walking manipulator during operation to allow proper analysis of the surfaces. Hence, the functions are not time-critical. Nonetheless, processing time should be reasonable and allow progress of the reconfiguration.

- We aim to process images at a frequency of 1 frame per minute.

In order to ensure detection of small defects, it might be necessary to enhance the functions with the ability of selective zooming. This is the capacity of controlling the camera focal length and view direction, in order to increase the resolution in one particular area of the field of view at the expense of a reduction of the field of view. In particular, we aim to detect a spacecraft component, turn the camera line of view toward them and increase the focal length so that the component occupies the entire field of view. A desirable 4-th technical feature is:

- Selective Zooming

This feature does not have an inherent accuracy requirement or time expectations. It should help the other main features meet their accuracy expectations, while not reducing the processing frequency below 1 frame per minute.

5.10.1 Main Processing Pipeline

The main processing pipeline or Data Fusion Processing Compound (DFPC) is illustrated in Figure 5-44.



Detailed Design Document

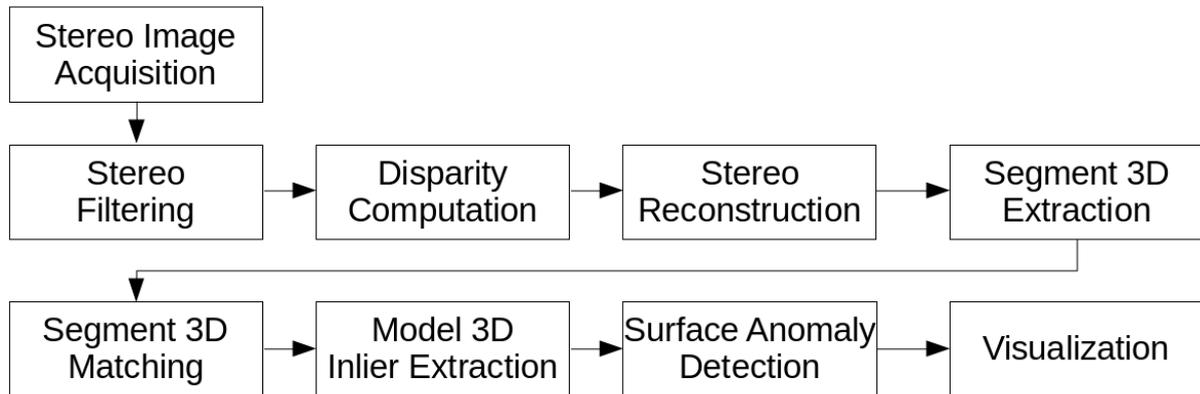


Figure 5-44: Surface Anomaly Detection Pipeline. This is the basic structure used for the implementation of all the other features.

The idea is to reconstruct a point cloud from a pair of stereo images, using the same process used in InFuse. Namely, we apply an undistorted and rectification filter, we compute a disparity image from the rectified images, and we convert the depth information contained in the disparity image in a point cloud representation.

After the point cloud is constructed, we extract 3D segments with a RANSAC process, and we attempt to match the extracted segments with the model segment with a second RANSAC process. Here, we make the assumption that model components can be described in term of large geometric polygons.

The result of a successful matching process is transform of the model in the scene. The transform will be used to align the model with the scene point cloud, compute inliers and detected anomalies.

5.10.2 Processing Line Variants

The main processing pipeline is used to detect anomalies on the surface of spacecraft modules. Table 5-35 lists the variants of the main DFPC required to implement the other specifications. Table 5-13 lists the main DFPC instances we aim to develop as standalone programs for the MOSAR demonstration.

Table 5-35: DFPCs Variants

DFPC Name	Input/Output	Variant
Module Anomaly Detection (Figure 5-45)	Input: A pair of stereo images, the models of the spacecraft modules, walking manipulator components or interfaces (for selective zooming) Output: A view of the scene point cloud, the detected modules, and the areas of anomaly.	The Segment 3D matching, model 3d inliers extraction and the surface anomaly detection are executed multiple times, once for each module we wish to detect, for as many times as necessary to discover all the modules.
Interface Anomaly Detection (Figure 5-46)	Input: A pair of stereo images, a hierarchical model of the spacecraft modules, the walking manipulator and interfaces. Output: A view of the scene point cloud, the detected modules and	The model are hierarchical, i.e. they have child submodules (representing the interfaces). Once a module is detected, the line extraction, line detection, inliers extraction and anomaly detection are repeated again for each



Detailed Design Document

	interfaces, and the areas of anomaly.	submodule, in the attempt to estimate a higher precision pose.
Reconfiguration Anomaly Detection (Figure 5-47)	<p>Input: A pair or stereo images, the models of the spacecraft modules, and the final reconfiguration of the spacecraft modules.</p> <p>Output: A view of the scene point cloud, a view of the ideal final reconfiguration with modules draw with different colors according to whether they were correctly detected in the correct position or not.</p>	The model 3d inliers extraction and surface anomaly detections steps are replaced with a reconfiguration anomaly detection stage. This stage takes all the detected modules, determines whether they respected a pre-planned configuration, and highlights any anomalies.

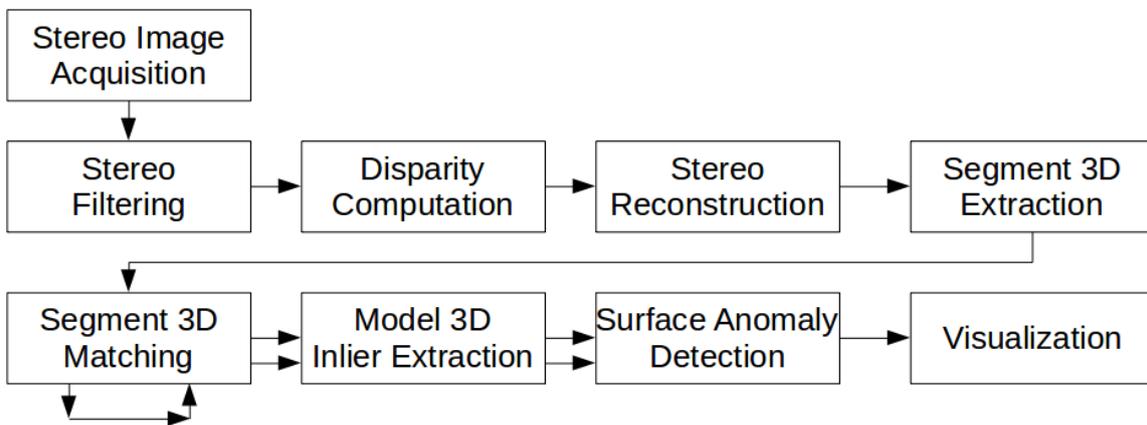


Figure 5-45: Module Anomaly Detection

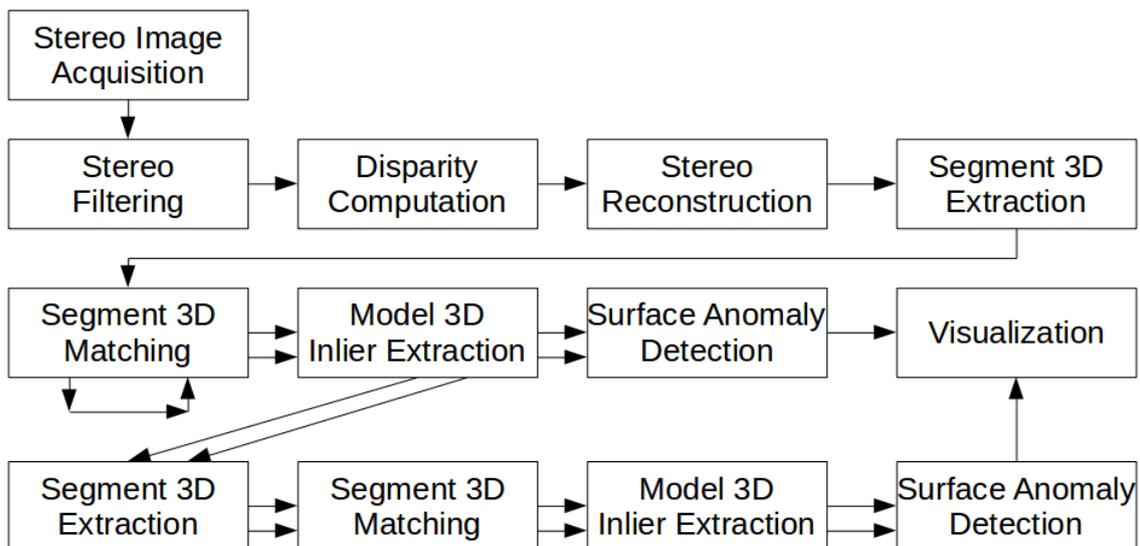


Figure 5-46: Interface Anomaly Detection.



Detailed Design Document

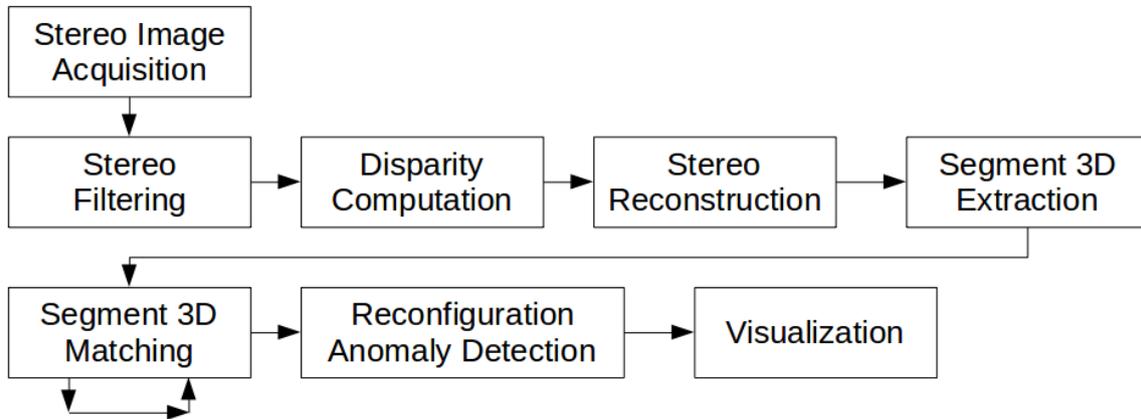


Figure 5-47: Reconfiguration Anomaly Detection.

Table 5-36: DFPCs Instances

Instance	DFPC	Input Models	Fulfilled Specification
1.1	Module Anomaly Detection	Only one spacecraft module mesh model (coarse model with no interfaces).	Surface Anomaly Detection of Spacecraft Modules
1.2	Module Anomaly Detection	All spacecraft module's mesh models (coarse models with no interfaces).	Surface Anomaly Detection of Spacecraft Modules
1.3	Module Anomaly Detection	All walking manipulator components (coarse models with no interfaces).	Surface Anomaly Detection of Walking Manipulator
1.4	Module Anomaly Detection for Selective Zooming	The interface mesh model.	Selective Zooming Surface Anomaly Detection of Interfaces
2.1	Interface Anomaly Detection	Only one hierarchical module mesh model, with one root mesh (the core model), and the interfaces as child models.	Joint Surface Anomaly Detection of Spacecraft Modules and Interfaces
2.2	Interface Anomaly Detection	All spacecraft module's mesh hierarchical models, with interfaces as child models.	Joint Surface Anomaly Detection of Spacecraft Modules and Interfaces
2.3	Interface Anomaly Detection	All walking manipulator module's mesh hierarchical models with interfaces as child models.	Joint Surface Anomaly Detection of Walking Manipulator and Interfaces
3.1	Reconfiguration Anomaly Detection	All spacecraft module's mesh models, and the reconfiguration plan as a map between module instances and poses.	Reconfiguration Anomaly Detection



Detailed Design Document

5.10.3 Software Architecture

We reuse InFuse software architecture and we distinguish between Data Fusion Nodes (DFNs) and Data Fusion Processing Compounds (DFPCs). Figure 5-48 shows the main software architecture, as it was designed in InFuse. DFN and DFPC Common interfaces define the main configuration and execution operations. DFN (respectively DFPC) interfaces define the input and outputs of each micro (respectively macro) data processing element.

In the pipeline illustrated in Figure Figure 5-44, each block represents one DFN interface and might have multiple implementations. In MOSAR, we plan to develop one implementation for each block, but additional implementations might be considered in case of low performance of the planned ones. We refer to D3.1 for the detailed description of each DFN.

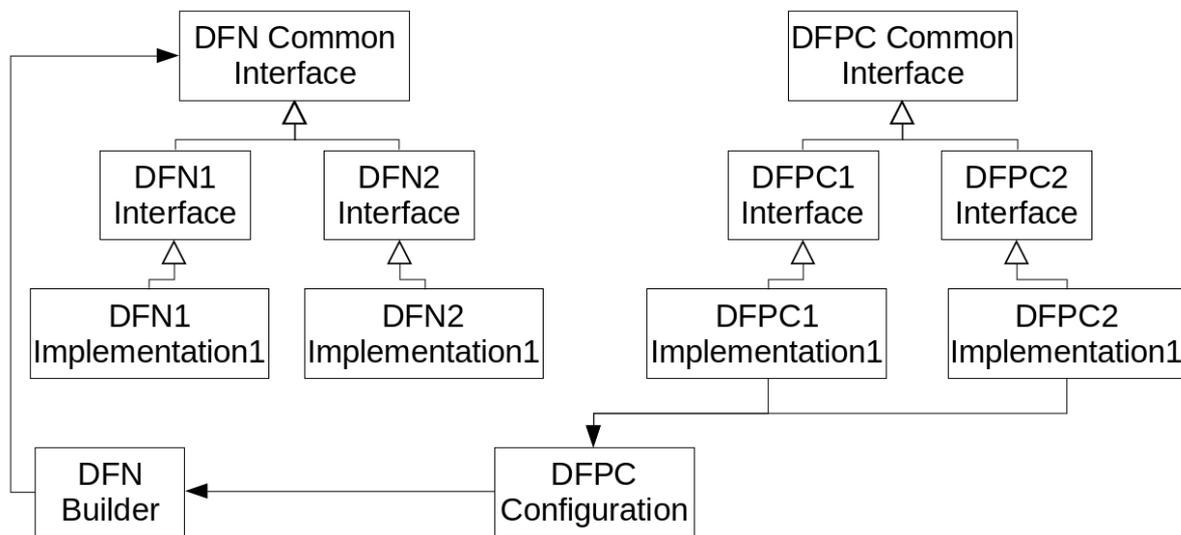


Figure 5-48: DFN and DFPC architecture

5.10.4 Visual Processing System Setup

As discussed in [RD1], the vision processing subsystem is composed of a vision processing platform, and a fixed stereo camera. Figure 5-49 displays the set-up of the vision processing system.

The processing platform is a standard computer with dedicated memory and processors, and input/output peripherals such a monitor, a keyboard, and a mouse. The processing platform will host a Ubuntu Operating System, and will execute the vision processing algorithms.

The stereo camera will be put on a stand at fixed pre-determined location at the side of the satellite mock. The camera field of view will cover the upper surface of the satellite, and non-occluded walking manipulator and spacecraft modules will be visible. The camera will have its own power source and will be directly connected to the PC with an USB or Ethernet cable for data transfer.

Space illumination conditions will be simulated by setting the demonstrator in a dark room with a black background. Room lighting conditions will approximate those that could be experienced in space, in a similar fashion to DLR's OOS-SIM simulator. A dark backdrop will be placed behind the demonstrator to absorb light and minimize features detected by the vision system. Two adjustable servicer LED lamps will be located at the side of the camera and will illuminate the visible side of the demonstrator. In a first scenario, we will assume that there is no sunlight, and that the earth albedo is simulated by room light. In a second scenario a third lamp will be positioned on a movable stand and will simulate the sunlight. We will evaluate the vision processing results under different light intensities of the two servicer lamps



Detailed Design Document

and different directions of the sunlight lamp. This will allow the performance of the vision system to be evaluated in realistic re-configuration scenarios.

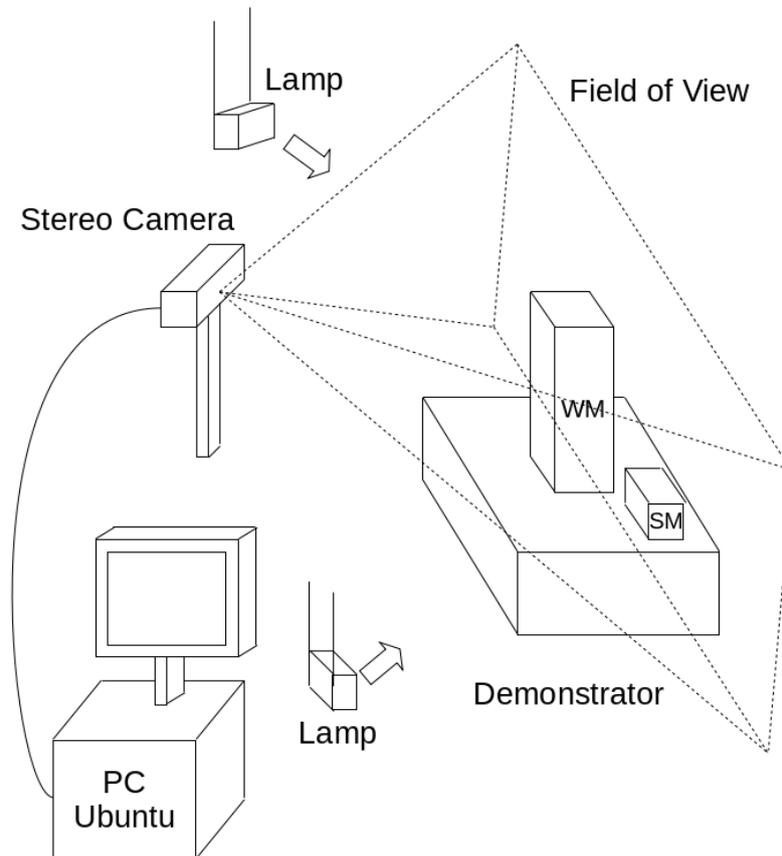


Figure 5-49: Hardware configuration of vision subsystem

5.10.4.1 Software Configuration

The vision processing algorithms will run on the Ubuntu Operating system hosted on the processing platform.

The Operating System will contain all the third party software libraries necessary to the correct execution of the vision processing algorithms (such as OpenCV, and Point Cloud Library). Alongside these libraries, InFuse will be available and will contain the core vision processing algorithms.

The final processing software will be composed of three components: (i) an image acquisition system that will receive the images from the camera and convert them into a format useful to processing; (ii) the functional algorithms for the actual image processing and computation of poses and 3D structures; (iii) an output display system that will show results on the monitor. Figure 5-50 shows the software architecture.

In addition to the software, a configuration file will be available for setting computational properties.

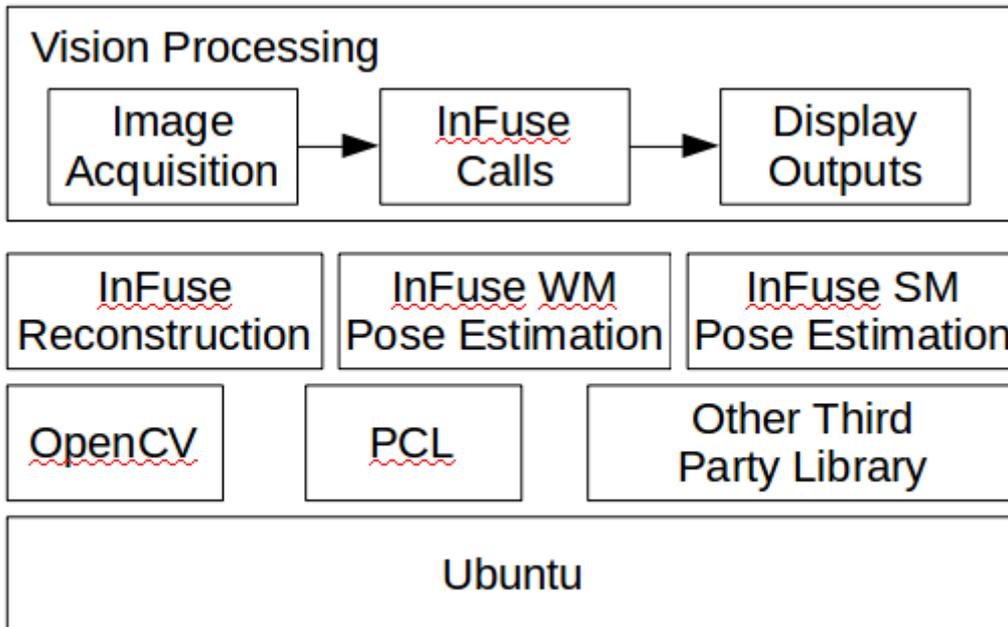


Figure 5-50: Software architecture of the vision subsystem

5.10.4.2 Hardware Components Selection

The following table lists the hardware components selected for the integration of the Visual Processing System with the demonstrator setup.

Table 5-37: Hardware Selection for Visual Processing System

Component	Description	Picture
Control Unit	The control unit is a Z2 mini workstation, with two 3.1 USB ports for fast data transfer, and NVIDIA graphic card to enable fast GPU processing.	



Detailed Design Document

Stereo Camera 1	<p>The first stereo camera is composed by two PTZ cameras attached to the same solid support. The cameras have a maximum resolution of 2.07 megapixels, and digitally variable focal length from 5.6mm to 98mm. The camera can be controlled though VISCA protocol on an RS422 interface. Ethernet cables and usb adapters are required to transmit data to the control unit. Each camera has its independent power supply.</p>	
Stereo Camera 2	<p>The second set of stereo camera is composed by two Basler cameras attached to the same solid support. The cameras are mono-chrome with a max resolution of 4096x3000 pixels. We will use 16 mm lenses. Ethernet cables and usb adapters are required to transmit data to the control unit. Each camera has its independent power supply.</p>	
Lamps	<p>3 Dimmable bi-color lamps. The lamps can cast light in two different colors and several level of intensity.</p>	



6 Annexes

6.1 MOSAR Setup Position References

This following figures provides the position reference definition of the HOTDOCK standard interfaces for the different components of the MOSAR setup that are the satellites mockups, SVC and CLT, and the spacecraft modules. Spacecraft modules faces are defined relatively to the initial MOSAR setup configuration.

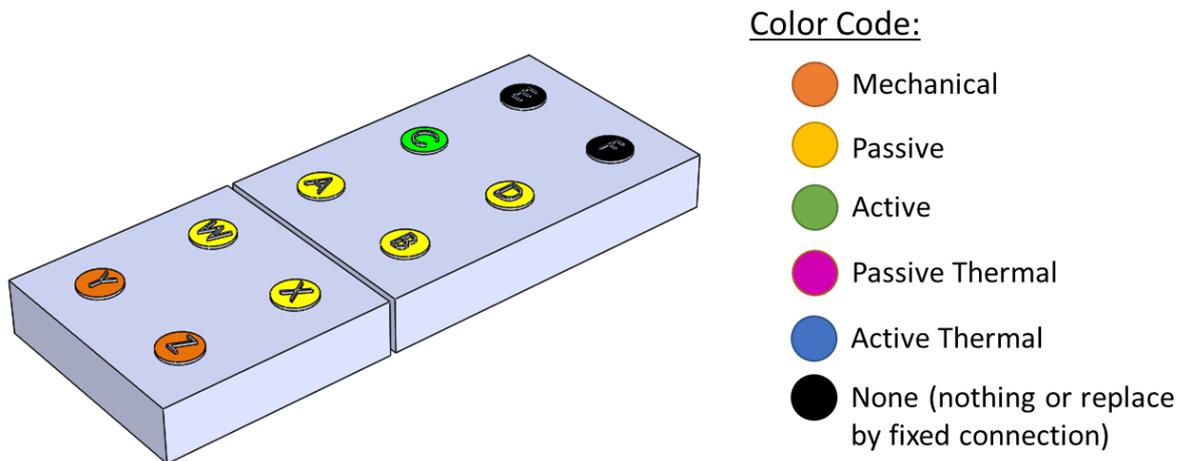


Figure 6-1: References of the HOTDOCK standard interfaces spot on SVC and CLT

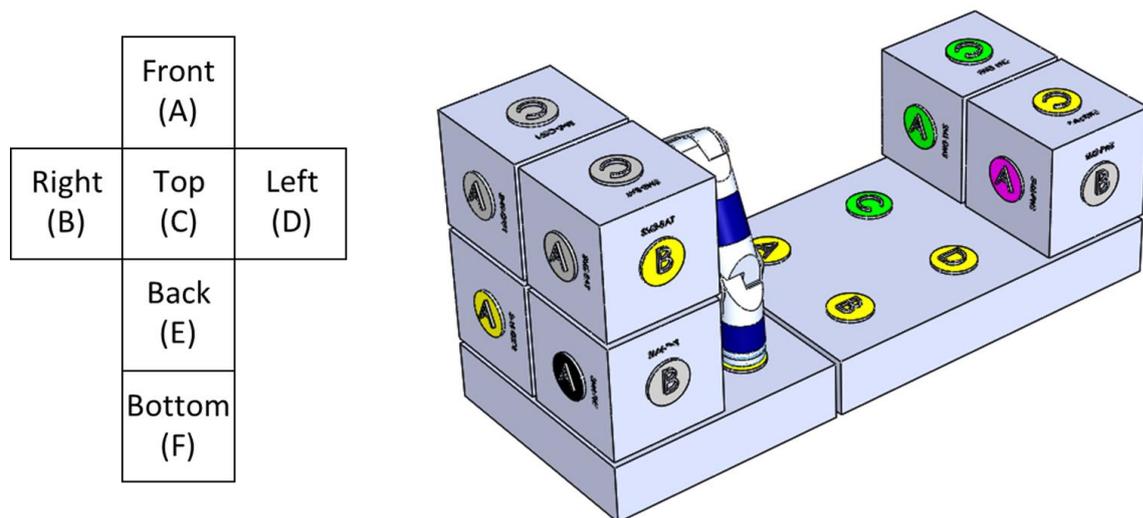


Figure 6-2: References of the spacecraft module faces (in the initial MOSAR setup configuration)



Detailed Design Document

End of Document
