estec

European Space Research and Technology Centre,
Keplerlaan 12201 AZ Noordwijk, The Netherlands
T +31 (0)71 565 6565F +31 (0)71 565 6040www.esa.int

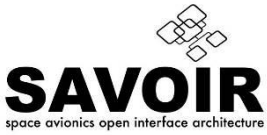
DOCUMENT

SAVOIR Flight Computer Initialisation Sequence Generic Specification

Prepared by	**SAVOIR**
Reference	**SAVOIR-GS-002**
Issue	**2**
Revision	**2**
Date of Issue	**18/11/2021**
Status	**Released**
Document Type	**Generic Specification**

European Space Agency
Agence spatiale européenne

Distribution **ESA Member States**



Disclaimer:

ESA does not provide any warranty whatsoever, whether expressed, implied, or statutory, including, but not limited to, any warranty of merchantability or fitness for a particular purpose or any warranty that the contents of the item are error-free. In no respect should ESA incur any liability for any damages, including, but not limited to, direct, indirect, special, or consequential damages arising out of, resulting from, or in any way connected to the use of this document, whether or not based upon warranty, business agreement, tort, or otherwise; whether or not injury was sustained by persons or property or otherwise; and whether or not loss was sustained from, or arose out of, the results of, the item, or any services that may be provided by ESA.

SAVOIR documents license:

The European Space Agency, on behalf of the participating members, holds copyright for all SAVOIR documents. No SAVOIR document may be disseminated outside the ESA member states in any form without the explicit written consent of ESA via a dedicated license.

Non-ESA members states or persons from outside ESA member states willing to access or use SAVOIR documents in whole or in part in their own documentation, should file a request to ESA. Only after agreement of the SAVOIR Advisory Group, the license is signed by ESA.

Note that:
• direct use of SAVOIR documents themselves should be made using quotation of SAVOIR documents, rather than rewriting.
• when SAVOIR documents text is used, SAVOIR copyright is acknowledged, quotations clearly identified in the document together with exact reference/version and potential modifications of SAVOIR documents used as source.

SAVOIR
space avionics open interface architecture

esa

APPROVAL

Title	
Issue 2	**Revision** 2
Author SAVOIR	**Date** 18/11/2021
Approved by	**Date**
SAVOIR Advisory Group	03/12/2021

CHANGE LOG

Reason for change	Issue	Revision	Date
Prepared for Public Review	1	1	18/08/2015
Public Review first batch of changes	1	2	30/08/2015
Public review second batch (harmonisation with GS-001)	1	3	30/09/2015
Public review third version (following discussion with industry of some Rids)	1	3b	12/10/2015
SAG endorsement review comments and editorials for publication	2	0	05/04/2016
Covered SoC with multiple-cores, FPGA and autonomous hosted payloads. Additional IF requirements to cover patch and dump operations with PUS services	2	1	05/05/2021
Public Review first batch of changes	2	2	18/11/2021

CHANGE RECORD

Issue 2	Revision 2		
Reason for change	**Date**	**Pages**	**Paragraph(s)**
See Review RID Tool			

SAVOIR
space avionics open interface architecture

esa

Table of contents:

1 INTRODUCTION

This document provides ESA requirements for the initialisation sequence of a typical Spacecraft On-Board Computer. The requirements are also applicable to computers belonging to other Spacecraft equipment such as mass memory, smart sensor/actuators, instruments or payloads.

The Boot SW is largely agnostic to the nominal function of the on-board unit, it deals mainly to the local processor resources, which are the same in any unit (star tracker, SSMM controller, ICU/DPU of an instrument), and any attempt to differentiate those units regarding the boot process is artificial and not necessary.

According to the hardware configuration of the targeted computer, the requirements of this document will be tailored to adjust to the hardware assumptions of section 4.3 that may be different, and the various parameters associated to some requirements (see 9 How to use this document).

Note*:* In the scope of the document, the term "payload computer" (as defined in 3.2) is representing any computer different from the On-Board Computer, it also covers any intelligent equipment embedding a microprocessor and running Flight SW. The main reason for this differentiation is that the OBC is the only one that can access to itself (therefore the redundancy assumption matters), whereas any other processor-based equipment can generally be accessed by the OBC.

The initialisation sequence is defined as all the operations executed by SW starting from the reset of the CPU up to the start of execution of the Application SW. This sequence can also be known as the Boot Software. For simplicity, the document shall refer to the Boot SW.
The Application SW, which is in charge of performing the mission operations, is considered an external SW item and it is not described in the present document.
The reference processor architecture used is the SPARC v7/v8, however, with exception of a few highlighted requirements, the concepts reported in the document can be applied also to other processors.
Requirements have been intentionally presented in a generic form in order not to be linked to a specific SW implementation. However, they come in the scope of the Avionics Functional Reference Architecture presented in SAVOIR-TN-001, which should be read together with this document, with the hardware assumptions specified in section 4.3.
Details not provided here, are assumed to be further specified in the project-specific boot software requirements baseline or technical specification to reflect needs and constrains given by mission, system or specific context (e.g. EDAC and interrupts enable, integrity test algorithm definition, cold/warm restart) (see 9).

2 APPLICABLE AND REFERENCE DOCUMENTS

2.1 Applicable documents

ECSS-E-ST-40C	Space Engineering - Software	6 March 2009
ECSS-Q-ST-80C	Space product assurance - Software product assurance	6 March 2009
ECSS-E-ST-70-41C	Telemetry and telecommand packet utilization	15 April 2016

2.2 Reference documents

SAVOIR-TN-001	SAVOIR Functional Reference Architecture	
SAVOIR-GS-001	SAVOIR generic OBC specification	
SPARC-V8	The SPARC Architecture Manual, Version 8	-
EA-2005-EEE-09-A	“Weak cells" in Hitachi HN58C1001 1Mbit EEPROM die – ESA Alert	16 September 2005

3 TERMS, DEFINITIONS AND ABBREVIATED TERMS

3.1 Acronyms

AIT	Assembly, Integration and Test
AOCS	Attitude and Orbit Control System
ASW	Application Software
BSP	Board Support Package
BIT	Built-In Test
CAN	Control Area Network
CPU	Central Processing Unit
CRC	Cyclic Redundancy Check
CSW	Central Software
EDAC	Error Detection And Correction
EEPROM	Electrically Erasable Programmable Read-Only Memory
EGSE	Electrical Ground Support Equipment
FDIR	Fault Detection, Isolation and Recovery
FPU	Floating Point Unit
FSW	Flight Software
HPC	High Priority Command
HPC-1	High Priority Command executed by HW without SW involvement
HPTM	High Priority Telemetry
HW	Hardware
I/O	Input/Output
MMU	Memory Management Unit

OBC	On-Board Computer
OS	Operating System
PLM	Payload Computer
PM	Processor Module
PROM	Programmable Read-Only Memory
RAM	Random Access Memory
RM	Reconfiguration Module
RMAP	Remote Memory Access Protocol
SDE	Software Development Environment
SGM	Safeguard Memory
SPARC	Scalable Processor Architecture
SpW	Spacewire
SW	Software
TC	Telecommand
TM	Telemetry
TTC	Telemetry, Tracking and Commanding
UART	Universal Asynchronous Receiver-Transmitter
WCET	Worst Case Execution Time

3.2 Terms and Definitions

Application Software	Generic term to identify the SW charge of performing the mission operations and running either on the OBC or on the Payload Computer.
Application storage memory	Read/write, non-volatile, data retention compatible with mission availability requirements (e.g. EEPROM, FLASH).
Board Support Package	SW package implementing specific support code for a given processor board that conforms to a given operating system.
Boot memory	Read-only, non-volatile, data retention greater than mission duration (e.g. PROM, write protected EEPROM).
Boot Report	Report containing the results of the Self-Tests executed by the Boot SW. Generated by the Boot SW.
Built-In Test	Test capability embedded in a device/equipment to autonomously verify its health status.
Cold restart	Start of SW execution following a computer power cycle. Computer resources need to be initialised.
Central Software	Application SW running on the OBC and in charge to manage the spacecraft
Death Report	Report containing the ASW and processor status at the time when an unrecoverable error is detected. Generated by ASW.
Essential Telemetry	See High Priority Telemetry
High Priority Telemetry	Hardware generated telemetry that is encoded directly into the OBC telemetry module.

Integrity test	Test of a memory region to verify it contains the expected data or code. It is typically implemented by means of a CRC or Checksum algorithm.
Interrupts	Signals generated asynchronously with respect to the SW execution by devices internal or external to the processor. Also referred as asynchronous traps.
Monitor	SW maintenance mode on Ground.
Nominal Sequence	Sequence of autonomous operations leading from processor reset/power on to ASW execution.
On-Board Computer	Flight computer managing the spacecraft. It is in charge of the AOCS, the TMTC space/ground communication and the management the other subsystems.
Payload Computer	Flight computer managing a specific payload subsystem. It is typically in charge of the payload science data processing, mass memory file system handling or equipment specific elaborations. In the scope of the document this term also covers star trackers and other intelligent equipment embedding a microprocessor and running Flight SW.
Post-mortem investigation	Analysis of processor working memory and Death Report following an unrecoverable error.
Processor Module	Module of the flight computer hosting the processor, the local memories and the interfaces required to communicate with the relevant equipment of the spacecraft.
Protected resources	Resource (e.g. SGM) capable to retain data in case of any recoverable or unrecoverable PM fault and in case of power loss.
Reconfiguration function	Autonomous function implemented in HW, capable to detect malfunctions (mainly) in the Processor Module and to perform recovery actions in order to maintain the spacecraft in a safe state.
Self-Test	Test performed autonomously during the nominal SW execution with the purpose of checking the health status of a device or processing resource. It could be a implemented by means of a device Built-In Test.
Standby	SW maintenance mode in flight.
Traps	Exceptions generated by the execution of a processor instruction (synchronous trap) or by the detection of a transition of dedicated HW signals (asynchronous trap) which causes the execution of specific privileged SW (trap handler).
Warm restart	Start of SW execution following a computer reset. Some computer resources (e.g. working memory) might retain information from previous execution.
Watchdog	Autonomous timer allowing the detection of failures that prevent the nominal execution of the SW.
Working memory	Read/write, volatile (e.g. SRAM, SDRAM).

3.3 Requirement Convention

Requirements defined in the document are presented following the structure presented hereafter:

SAVOIR.BOOTSW.<Function>.<Number>

<Requirement Title>

<Requirement text>

Note:	*<Note text>*
OptionInfo:	*<EQP>; <EQP>*
Assumption:	*< Assumption>; < Assumption>*
Parameter:	*<Parameter>; <Parameter>*
Requirement Rationale:	*<Rationale text>*
Verification Method:	*<Requirement Verification Method>; <Requirement Verification Method>*

where:
<Function> is an acronyms for the corresponding function amongst:

- BEF: Boot Execution Flow
- BIN: Boot INitialisation
- BTE: Boot Tests
- BAA: Boot Actions After tests
- BPF: Boot PerFormance
- BMM: Boot Memory Management
- IF: Interface
- IMP: IMPlementation
- STD: STanDard

<Number> is a progressive number.
<Requirement Title> is a short summary of the requirement
<Requirement Text> is the text of the requirement
<Note text> is an optional text meant to clarify or help in the interpretation of the requirement.
<EQP> is the equipment on which the requirement is applicable, amongst:

- OBC: On-Board Computer
- PLM: Payload Computer

< Assumption> is a precondition on the System and Operation for the applicability of the requirement
<Parameter> is the configuration parameter that needs to be defined in the actual project for the requirement to be applied.
<Rationale text> is the justification of the requirement
<Requirement Verification Method> is the proposed requirement verification method amongst:

- T: Validation by test;
- ROD: Validation by Review of the Design.

- A: Analysis
- I: Inspection

4 SOFTWARE OVERVIEW

4.1 Function and purpose

4.1.1 Nominal Sequence

Boot SW is the first part of SW executed when a processor module is powered up or reset. It is a standalone executable typically stored in a dedicated read-only memory (Boot Memory). It is in charge to initialise and test the processor module and to execute the Application SW after copying it from a non-volatile memory area into the processing working memory.

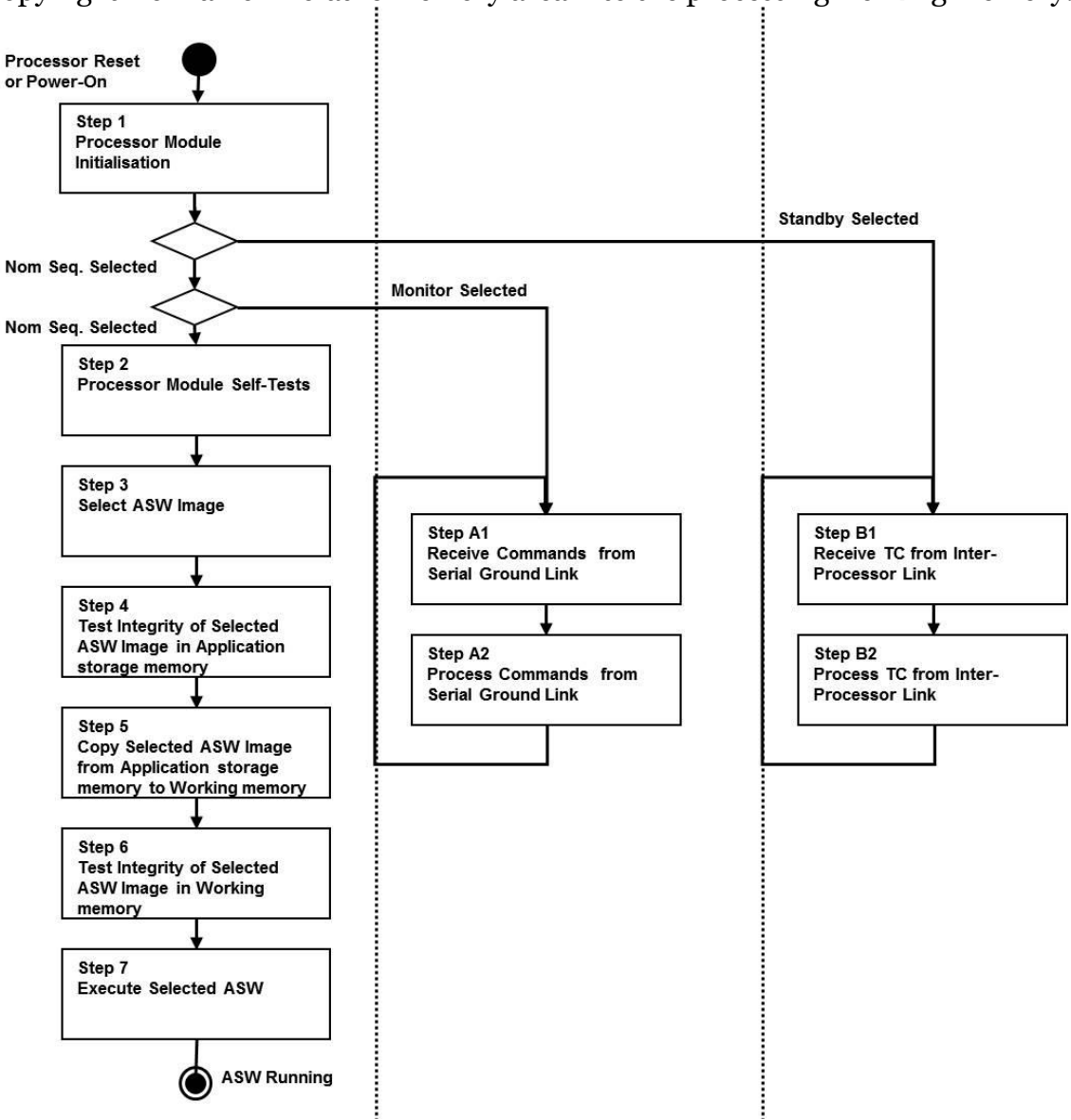


Figure 1 – Example of Initialisation Sequence

Boot SW and ASW are independent executables with their own separate code, data, stack and trap table areas in memory.
Depending on the computer platform, execution time requirements (i.e. the time required to boot the computer and start the ASW – this is mission dependent) and SW criticality, the Boot SW can run directly from Boot Memory (slow access time) or can be copied into working memory (fast access time) and executed from there.

In some occasions, to increase flexibility in the SW development process, part of the Boot SW following the processor initialisation, might be stored and executed from Application storage memory instead of Boot Memory. This allows the possibility of later refinements/bug fixing, although if not handled with care, it leaves open the risk for an unrecoverable corruption of the Boot SW during flight.

The initialisations and Self-Tests should put the processor in condition to execute the ASW and give information of the health status of the processor module (contained in the *Boot Report*) allowing Ground to investigate the source of possible malfunctions.
The Self-Tests should be limited to the processor module functions. Any external equipment must be under the responsibility of the ASW in order to keep the Boot SW as efficient as possible. For example SGM (with the exception of the *Boot Report* area) contains the context of the ASW saved before the last reconfiguration and its content and format have meaning only to the ASW, therefore it should not be accessed or modified by the Boot SW.

It is noted that *Boot Report* should not be confused with *Death Report*. The former is produced by the Boot SW and concerns the result of the processor module Self-Tests, the latter is produced by the ASW when an unrecoverable error occurs and reports the status of the processor at the time of the failure.
The OBC Boot SW stores the *Boot Report* in the safe guard memory. It is assumed that the OBC supports this function by providing a safe memory area powered by a power line independent from the processor power supply (see also Figure 2).
In case of computer supporting multiple autonomous reboots, the *Boot Report* should be managed in such a way that important information from one reboot is not overwritten by subsequent reports.

It is assumed that more than one ASW image is available in Application storage memory. This allows to update in-flight one image in order to correct malfunctions or to improve the SW performance and still to have the possibility to roll back to a second and safe ASW image not affected by the latest patches.
A common approach is that the Boot SW selects the ASW image to load in working memory after reading status bits set by Ground via HPC-1 commands. Some more elaborated mechanisms to autonomously select the image (e.g. based on the result of the integrity tests on ASW image or segments of it) are also possible, but should be adopted only when specific mission requirements justify it (e.g. survival of the spacecraft during long periods of time without Ground contact).

In general Boot SW is not supposed to perform any recovery action following the detection of a failure in a Self-Test. This is motivated by the fact that Boot SW should have a simple design, mainly independent from the FDIR strategy defined for a given mission. Moreover, Boot SW cannot be modified in flight, therefore any autonomous recovery action implemented in it might add unnecessary constrains to spacecraft operations afterwards.
The general strategy to leave FDIR out of the boot SW is still valid and generally accepted. Potential FDIR requirements would be addressed in the technical requirement specification, if needed by a specific mission and enabled by specific computer architecture. For example, the support to enable redundant memory banks depends on the computer architecture and the mission context.

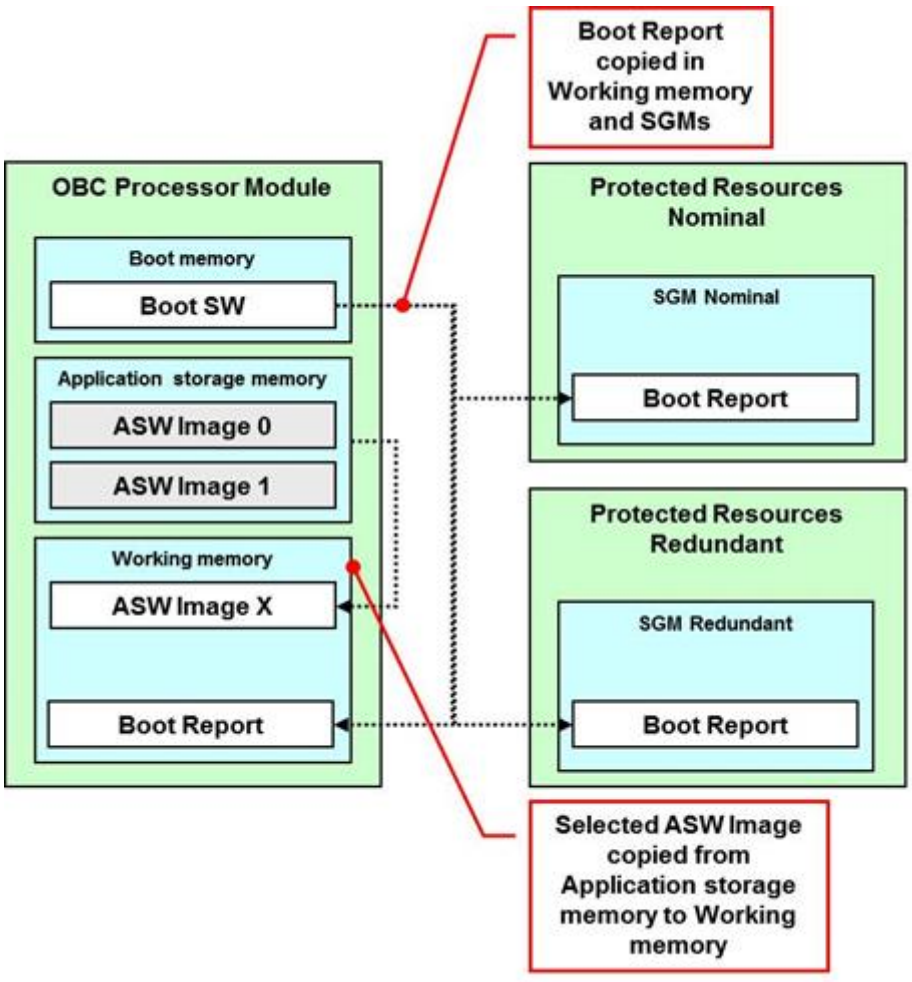


Figure 2 - Nominal Sequence memory context (OBC)

On OBC, during the initialisation sequence, external watchdog is used. Under the assumption that the hardware (e.g. Reconfiguration function) will mask the watchdog for a time larger than the Boot SW typical execution, the boot software does not need to refresh

the watchdog. HW insures the masking of the watchdog for the time needed to complete the sequence. This external watchdog is reset only by ASW, therefore, lack of external watchdog kick serves as a mean for detection of ASW start-up failure.
On PLM, the CSW running on the OBC can verify the completion of the initialisation of the payload computer by checking the state transition from initialisation to nominal mode or by monitoring the telemetry event reports produced during the payload initialisation.

4.1.2 Software Maintenance in Flight

The Boot SW is often required to provide SW support for the maintenance in-flight of the other SW items of the flight computer (e.g. ASW). This is justified by the fact that the Boot SW is a standalone executable and it is stored in a reliable and read-only memory (opposite to patchable and less reliable non-volatile memory where the ASW is normally stored).
The SW is maintained in flight by means of memory patch/dump/check operations commanded from Ground and the Boot SW supporting this functionality while operating in a *Standby* mode.
There are two main scenarios where the *Standby* mode is involved. One covers the maintenance of the SW located in one of the processor modules of the OBC (Figure 3), while the other covers the maintenance of the SW located on the processor module of a Payload Computer (Figure 4).

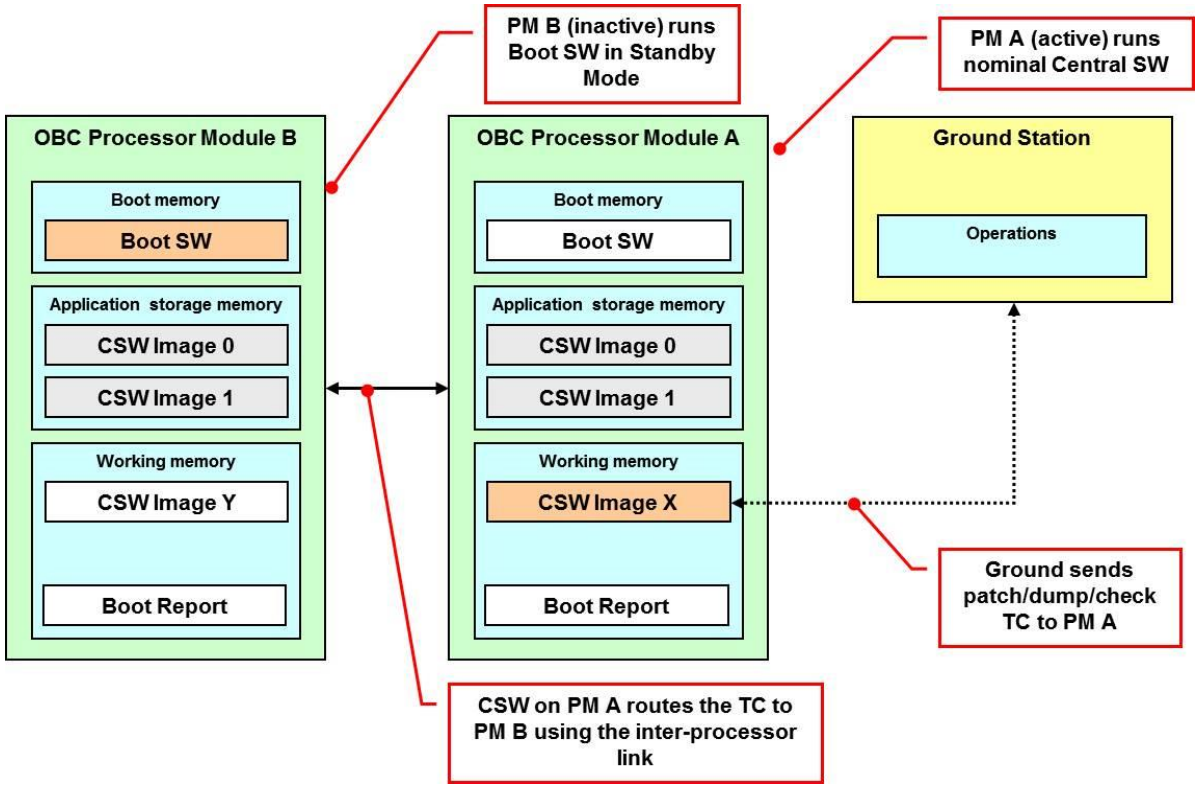


Figure 3 - Standby mode in OBC context

In the OBC scenario, the active processor module runs the nominal Central Software and manages the spacecraft including AOCS and TM/TC communication with Ground. The Standby mode runs on the inactive processor module that is in charge to process memory load/dump/check commands originated from Ground and routed to it by the active processor module via a dedicated inter-processor link (e.g. serial line, CAN bus, other equivalent means).

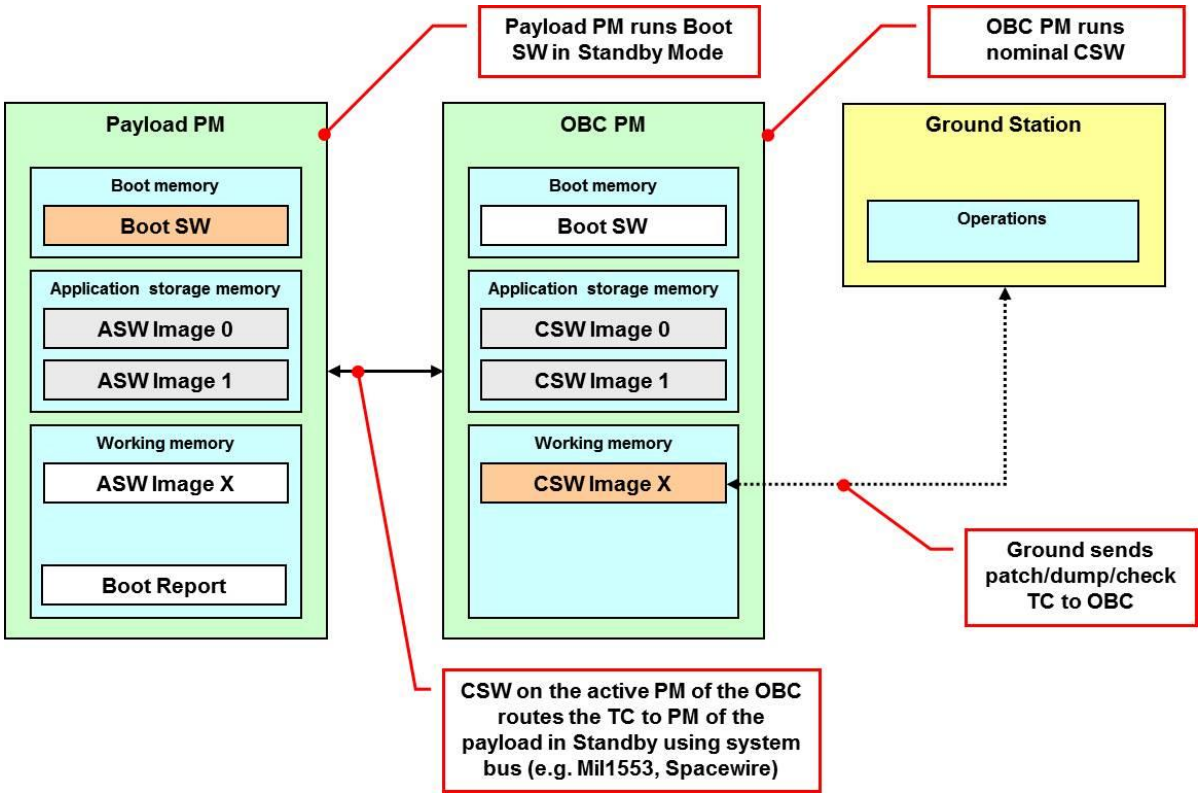


Figure 4 - Standby mode in payload computer context

In the Payload Computer scenario, the communication with Ground is usually managed by the OBC, which routes the memory load/dump/check commands to the Payload Computer using the system bus (Mil1553, Can Bus) and its related protocol. Some advanced payloads may have a direct communication link with Ground. In that case, the Standby scenario of the Payload Computer is similar to the one of the OBC.
When allowed by the mission/avionics system requirements, the communication protocol for the memory load/dump/check between OBC and Payload Computer should be kept as simple as possible, to avoid unnecessary complexity implemented in the *Standby* mode.

In both scenarios *Standby* mode could also be used to enable post-mortem dump of the working memory after processor module switch over or processor reboot, provided that the working memory banks in the faulty processor module are not switched off or erased prior to entering the *Standby* mode.

Standby enables the modification in flights of all other SW items and it is therefore classified as category B. Its specific characteristics depend on the computer platform and on the way the inter-processor link is implemented; however the *Standby* function should always be supported to ensure an adequate level of SW maintenance.

4.1.3 Software Maintenance on Ground

The Boot SW also provides SW support for the maintenance on ground of the other SW items of the flight computer.

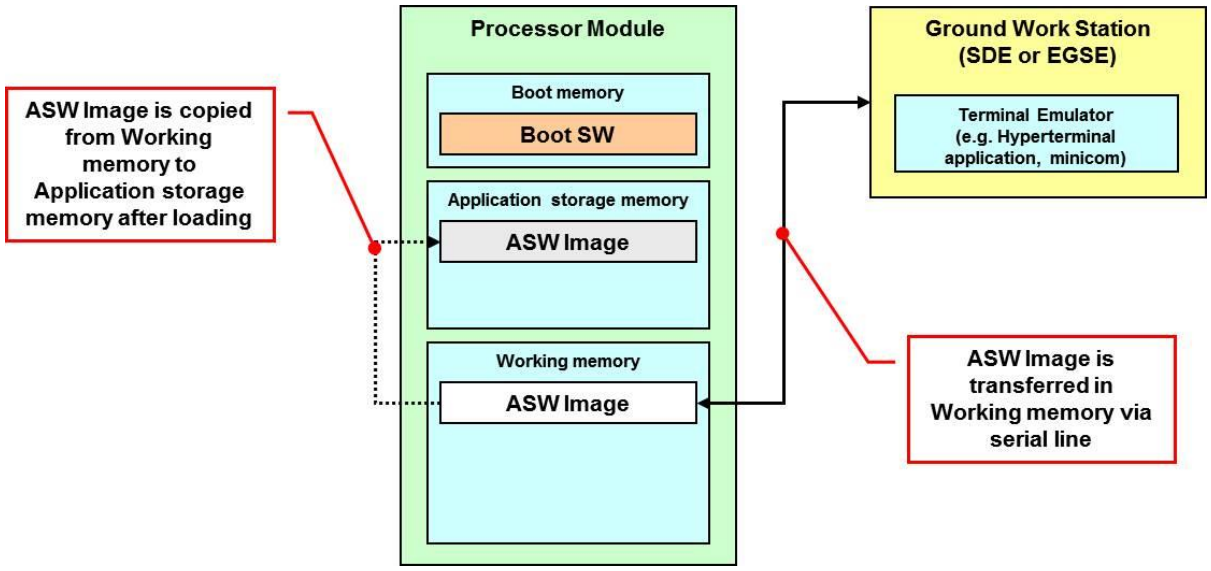


Figure 5 - Monitor context

The SW maintenance during ground operation mainly addresses SW debug or loading of the ASW image in the processor memory and it is provided by a Boot SW mode called *Monitor*. As shown in Figure 5, the *Monitor* mode supports communication between the processor located in the flight equipment and a ground workstation acting as SW Development Environment (SDE) or EGSE. *Monitor* usually has a shell-like interface requiring a simple back end on the host ground computer to interact with the operator and might have hooks to support high level debugging capabilities.

The SW criticality of the *Monitor* is lower than the rest of the Boot SW because it is used only on Ground. However, since it supports interfaces only foreseen in the ground configuration (e.g. serial line connected to the SDE), this mode should be deactivated in flight configuration and the mechanisms to start it (e.g. presence of test connector and time out on serial line) should be such that its activation will not be induced by a single failure on the computer.

The specific capabilities of the *Monitor* depend on the computer platform, but the functionality should always be present to ensure an adequate level of SW maintenance on ground. Monitor mode can be leveraged to introduce specific debugging capabilities, other than those offered by the hardware platform, when a specific debugging interface (e.g. JTAG) is not available.

In computers supporting debug capability in HW (e.g. DSU on LEON2/3 processors, SpW link with RMAP) the *Monitor* function might be embedded in the processor instead of being a SW item.

4.2 Environmental considerations

None.

4.3 Relation to other systems

A number of features presented above and further specified in the requirements section rely on HW functionalities provided by the computer platform. If some of this functionality is missing, some requirements cannot be fulfilled or will require unnecessary complexity to be added to the Boot SW.
In the following sections, a summary of the expected computer features applicable to OBC or Payload Computers is presented. These HW assumptions are preconditions for some requirements. These preconditions are highlighted when necessary in the requirements.

It is out of scope for this document to specify the HW requirements for the computer platform.

4.3.1 Common HW Assumptions

The following basic features are expected in both OBC and Payload Computers, in order to support the specified Boot SW functionality:

a. Boot memory: read-only, non-volatile, data retention greater than mission duration (e.g. PROM, write protected EEPROM) to store and execute the Boot SW (or at least the initial part of it).
b. Application storage memory: read/write, non-volatile, data retention compatible with mission availability requirements (e.g. EEPROM, FLASH) to store at least two ASW images.
c. Working memory: read/write, volatile (e.g. SRAM, SDRAM) to execute the ASW.
d. Capability to select the ASW image from Ground.
e. Capability to activate the *Standby* function from Ground.
f. *Monitor* interface for communication on Ground.
g. In case of multicore processors, capability to run the Boot SW from one core, while the others are kept in reset state.

Note: EEPROM data retention limitations should also be considered when deciding to use EEPROM instead of PROM for Boot SW (for further details see [EA-2005-EEE-09-A]).

Note: In a multi-core processor, the boot process is done by one core (i.e. single core processor configuration). Secondary cores(s) are disabled during the entire execution of the Boot SW. Initialisation of the secondary core(s) is performed by the ASW after it is started on the primary core.

Note: System-on-Chips are embedding one or multiple processing cores tightly coupled to reprogrammable logic (FPGA). If the configuration of the FPGA is required to initialise the computer (e.g. it includes the management of interfaces), the FPGA is expected to be configured prior the initialisation of the main processing core (e.g. by loading predefined configuration). Otherwise, the initialisation of the FPGA is expected to be performed by the ASW.

4.3.2 OBC HW Assumptions

As already mentioned the OBC is in charge to execute the CSW including the AOCS and TM/TC space/ground communication management.
The following basic features are expected in support to the Boot SW requirements:

a. A SW independent reconfiguration function capable to perform a recovery action (e.g. PM reset, switch over to redundant PM) in case of processor malfunctions.
b. Interface allowing the processor module to acquire, via a local access (e.g. parallel port or register), which recovery action has been executed by the reconfiguration function.
c. Processor module redundancy.
d. Capability to have both nominal and redundant processor modules powered on at the same time.
e. Protected resources (e.g. SGM) capable to retain data in case of any recoverable or unrecoverable PM fault and in case of power loss.
f. Arbitration (and cross-strapping) allowing nominal and redundant processors to access protected resources concurrently.
g. Capability for the nominal processor module to communicate with the redundant processor module and vice versa (e.g. inter-processor link such as UART or CAN Bus).
h. *High Priority Telemetry* (also known as *Essential Telemetry*) accessible by SW, but generated without SW intervention.
i. Interface allowing the processor module to check, via a local access (e.g. parallel port or register), if the processor module is set as active (i.e. executing Boot SW nominal sequence, then CSW) or inactive (i.e. executing StandBy SW).

4.3.3 Payload Computer HW Assumptions

The Payload Computer is a general term used in the document to identify the flight computers (different from the OBC) embedded in some of the spacecraft equipment (e.g. instruments, mass memories, star trackers) and in charge of specific functions.
Due to the difference with respected to the OBC in criticality, interfaces and redundancy, the following HW assumption(s) apply:

a. Communication link with OBC (e.g. Mil1553 Bus, UART, CAN Bus, SpaceWire).

4.4 Constraints

None

5 REQUIREMENTS

This section defines the minimum set of requirements applicable to every Boot SW product.

5.1 General

SAVOIR.BOOTSW.BEF.05

Boot software modes

The Boot software shall support the following modes:
- Nominal Sequence Mode, where the PM function loads an Application Software image.
- Standby Mode.
- Monitor Mode, where the PM function is controlled from a dedicated EGSE I/F.

Note:	*In Standby mode of operation, the PM function is controlled by the active PM.*
OptionInfo:	*OBC; PLM*
Assumption:	*Boot memory; Application storage memory; Working memory; Select ASW image from ground; Activate Standby from ground; Monitor interface*
Rationale:	*Different boot modes are required for active, inactive, or payload computer processor module and a Monitor mode is used on ground during development and AIT activities.*
Verification Method:	*T*

5.2 Functional requirements

5.2.1 Nominal Sequence

Requirements in this section address the execution flow steps performed by the Boot SW during the Nominal Sequence.
The Nominal Sequence does not contain the activation points for *Monitor* and *Standby* modes because the precise points in the sequence where to place these checks depend on the project and equipment requirements.

SAVOIR.BOOTSW.BEF.10

Boot software execution

The Boot SW shall be executed on processor reset (power-on reset, SW reset, error reset, watchdog reset).

Note:	*the handling of cold or warm restart conditions depend on project-specific requirements and are not described in this document.*
OptionInfo:	*OBC; PLM*
Assumption:	*Boot memory*
Parameter:	*ColdWarm*
Rationale:	*the requirement constrains Boot SW entry point and its allocation in memory (i.e. start from address oxOOOOOOOO, no other SW executed before).*
Verification Method:	*ROD; T*

SAVOIR.BOOTSW.BEF.15

Fast Boot Path selection

The Fast Boot Path selection shall be based on configuration data that are set prior to the PM reset/PM power-on.

Note:	*This can be data residing in the protected resource (SGM), data set by direct telecommand or data set by the Reconfiguration function.*
OptionInfo:	*OBC; PLM*
Parameter:	*FastBootPath*
Rationale:	*Fast Boot Path is used to bypass specified Nominal Sequence steps, e.g. PM Self Tests or Application Software integrity check.*
Verification Method:	*T*

SAVOIR.BOOTSW.BEF.20

Boot software nominal sequence

When Fast Boot Path is not selected, the Boot SW Nominal Sequence shall execute as minimum the following steps:

1. Perform processor module initialisations;
2. Perform processor module Self-Tests;
3. Select the active ASW image;
4. Test integrity of the selected ASW image in Application storage memory;
5. Copy the selected ASW Image from Application storage memory to processor working memory;
6. Test integrity of the selected ASW image in processor working memory;
7. Execute the selected ASW in processor working memory.

Note:	*It is expected to have at least two ASW images for each processor module. This is part of the variability expressed in section 9. ASW image can include patches.* *This mitigates limitations in Application storage memory data retention and allows having one image available for patching and another untouched to be used in case of contingency.*
OptionInfo:	*OBC; PLM*
Note:	*In case of multi-core processors, secondary core(s) are disabled during the entire execution of the Boot SW [SAVOIR.BOOTSW.BIN.245].*
Assumption:	*Application storage memory; Working memory; Select ASW image from ground*
Rationale:	*The requirement defines the minimal functionality:* • *Processor and processor module devices need to be initialised and tested in a controlled way before being enabled.* • *Most of the devices cannot be tested in the ASW context, since the I/Fs are enabled and in use. Self-Tests at initialisation time give also a higher level of confidence in the HW design/manufacturing, since the processor module devices are tested every time the equipment is powered-on.* • *ASW image integrity needs to be tested in order to give information about the cause of possible malfunctions during its early execution. The checks for Standby and Monitor activation are not mentioned because they can be placed in different points of the sequence, depending on mission requirements or extra features supported by the Boot SW.*
Verification Method:	*T*

SAVOIR.BOOTSW.BEF.22

Fast Boot Path sequence

When Fast Boot Path is selected, the Boot SW Fast Sequence shall execute a subset of the step of the nominal sequence.

Note:	*The selected subset of nominal sequence's step is defined following mission constraints.*
OptionInfo:	*OBC; PLM*
Parameter:	*FastBootPath*
Rationale:	*The Fast Boot path allows to reach nominal mode in a minimum time.*
Verification Method:	*T*

SAVOIR.BOOTSW.BEF.25

Active ASW image selection

The Boot SW shall be able to select the active ASW Image based on configuration data that are set prior to the processor module reset/power-on.

Note:	*In case additional on-board autonomy is required, a more elaborated selection algorithm can be specified as part of the mission requirements, e.g. based on the previous history of spacecraft (re)configuration.*
OptionInfo:	*OBC; PLM*
Assumption:	*Protected resource retaining data when fault or power loss*
Parameter:	*NumberASWimages*
Verification Method:	*T*

SAVOIR.BOOTSW.BEF.27

Active ASW image selection configuration data

The Boot SW active ASW image selection configuration data shall be controllable by direct TC from ground or by the Reconfiguration function.

Note:	*Boot Mode image selection configuration data can also include data to be used by the booted software image to select different operating modes, like Safe Mode.*
OptionInfo:	*OBC; PLM*
Assumption:	*Select ASW image from ground; Reconfiguration function; Reading recovery action in reconfiguration function*
Parameter:	*NumberASWimages*
Requirement Rationale:	*Ground should always be in the position to choose (override) the default ASW image to be loaded in working memory, this is usually done by means of relay status set by HPC-1.*
Verification Method:	*T*

SAVOIR.BOOTSW.BEF.30

Nominal sequence when failure

The Boot SW shall execute the steps of the Nominal Sequence also in case of failure in the Self-Tests.

OptionInfo:	*OBC; PLM*
Note:	*Additionally, since the final version of the Boot SW is expected to be implemented much earlier than the finalisation of the FDIR strategy system requirements, it is recommended to leave to the ASW the implementation of all the SW related FDIR functions.*
Requirement Rationale:	*Self-Tests provide information about the possible cause of a failure in executing the ASW, however there's always the chance that the ASW can run after a Self-Test failure (e.g. working memory test fails, but the failing locations are not used by the ASW).*
Verification Method:	*T*

5.2.2 Standby

The requirements in the section address the Boot SW functionality required to support SW maintenance in flight.

SAVOIR.BOOTSW.BEF.70

Standby function capability

The Boot SW shall support the following capability in flight, by means of separate equipment supporting the space-ground communication (*Standby*):

- to load, dump and check processor memories;
- to load, dump and check registers;
- to read the local Boot Report.

Note:	*the memory check function is meant to be equivalent to PUS service 6 memory check. It is recommended to use the same algorithm used in the integrity tests.*
Note:	*in the case of OBC, the inactive PM runs the Standby mode and the active PM manages the communication with Ground. The two PMs are meant to be connected via the inter-processor link.*
Note:	*in case of Payload Computer, the Payload Computer runs the Standby mode and the OBC manages the communication with Ground.*
Note:	*In case of multi-core processors, secondary core(s) are disabled during the entire execution of the Boot SW [SAVOIR.BOOTSW.BIN.245].*
OptionInfo:	*OBC; PLM*
Assumption:	*Boot memory; PM redundancy; Powering both nominal and redundant; if PM redundancy, inter processor link; Reading the active/inactive status by PM; if PLM, link with OBC*
Parameter:	*HardwarePatch*
Requirement Rationale:	*under SW maintenance the processor module can't control the spacecraft orbit, attitude and pointing. For this reason, a second equipment must be active and support AOCS and communication with Ground. Since it is not granted that the ASW can be successfully executed in case of failure, the maintenance needs to rely on a simpler and more reliable SW (stored in Boot Memory).*
Verification Method:	*T*

SAVOIR.BOOTSW.BEF.75

Standby function access

The Boot SW *Standby* function shall access only resources local to the processor module without interfering with the nominal spacecraft operations managed by the active PM.

OptionInfo: OBC

Requirement Rationale: *in the OBC, the Standby is executed by the inactive PM while the active PM manages the spacecraft. The inactive PM should not compete for spacecraft resources or interfere with the nominal spacecraft operations managed by the active PM.*

Verification Method: *ROD; T*

SAVOIR.BOOTSW.BEF.80

Standby function triggering

The Boot SW Standby function shall be triggered by the logical AND of two independent conditions.

Note: *Having two conditions to activate the Standby makes the system single point failure tolerant.*

OptionInfo: *OBC*

Parameter: *StbyTrigger*

Requirement Rationale: *Standby is normally an infinite loop waiting for commands sent via the inter-processor link. On OBC, if activated in flight on the active processor module, it will prevent to execute the ASW (i.e. AOCS/safe mode).*

Verification Method: *T*

SAVOIR.BOOTSW.BEF.90

Standby function activation condition

The Boot SW *Standby* activation condition shall be checked within the Boot SW Nominal Sequence.

Note:	*The requirement is not applicable if an automatic HW activation condition is foreseen.*
Note:	*If the Standby is implemented in HW, its activation might not be under SW control, in this case the requirement is not applicable.*
OptionInfo:	*OBC; PLM*
Parameter:	*StbyTrigger*
Requirement Rationale:	*the requirement constrains the designer to define a point in the Nominal Sequence where the double activation condition is checked. This allows a deterministic behaviour of the Boot SW in the execution of the nominal and non-Nominal Sequence.*
Verification Method:	*T*

SAVOIR.BOOTSW.BEF.95

Standby function activation

The Boot SW shall activate the *Standby* function based on conditions controlled by Ground.

Note:	*Activation conditions are based on external conditions set by Ground (e.g. relay status set by HPC-1).*
Note:	*The requirement is not applicable if an automatic HW activation condition is foreseen.*
OptionInfo:	*OBC; PLM*
Assumption:	*Activate Standby from ground*
Parameter:	*StbyTrigger*
Requirement Rationale:	*Standby is used by Ground for SW maintenance, therefore the activation conditions have to be under Ground control.*
Verification Method:	*T*

SAVOIR.BOOTSW.BEF.97

Standby function when failure

If activated after the Self-Tests, the Boot SW StandBy function shall be executed also in case of failure in the Self-Tests.

Note:	*Since the final version of the Boot SW is expected to be implemented much earlier than the finalization of the FDIR strategy system requirements, it is recommended to leave to the ASW the implementation of all the SW related FDIR functions.*
OptionInfo:	*OBC; PLM*
Requirement Rationale:	*Self-Tests provide information about the possible cause of a failure in executing the ASW. However, there's always the possibility that the StandBy function is executed after a Self-Test failure (e.g. working memory test fails, but the failing locations are not used by the StandBy).*
Verification Method:	*T*

5.2.3 Monitor

The requirements in the section address the Boot SW functionality required to support SW maintenance on ground.

SAVOIR.BOOTSW.BEF.40

Monitor function capability

The Boot SW shall support the capability to load and dump memories and registers, start and halt software execution, and reset the PM on Ground command (*Monitor*).

Note: More functions may be specified in the actual project specification.

OptionInfo:	*OBC; PLM*
Assumption:	*Monitor interface*
Parameter:	*HardwareMon; MonFunctions*
Requirement Rationale:	*in an embedded computer, the most common way to upload the SW in Application storage memory is with the support of a resident SW (i.e. stored in Boot Memory). For mass and cost reasons the Monitor is stored in the same memory area (Boot Memory) of the Boot SW.*
Verification Method:	*T*

SAVOIR.BOOTSW.BEF.50

Monitor function triggering

The Boot SW *Monitor* function shall be triggered by the logical AND of two independent conditions.

Note:	*the two activation conditions should be independent from the failure propagation point of view (e.g. one HW condition and one SW condition).*
OptionInfo: OBC; PLM	
Parameter: MonTrigger	
Requirement Rationale:	*Monitor is normally an infinite loop waiting for commands sent via a debug line. If activated in flight, it will prevent to execute the ASW. Having two conditions to activate the Monitor makes the system single point failure tolerant.*
Verification Method:	*T*

SAVOIR.BOOTSW.BEF.60

Monitor function activation condition

The Boot SW *Monitor* activation condition shall be checked within the Boot SW Nominal Sequence.

Note:	*the requirement is not applicable if an automatic HW activation condition is foreseen.*
OptionInfo:	*OBC; PLM*
Parameter:	*MonTrigger*
Requirement Rationale:	*This requirement constrains the designer to define a point in the Nominal Sequence where the double activation condition is checked. This, in turns, allows a deterministic behaviour of the Boot SW in the execution of the nominal and non-Nominal Sequence. If the Monitor is implemented in HW, its activation might not be under SW control, in this case the requirement is not applicable.*
Verification Method:	*T*

SAVOIR.BOOTSW.BEF.65

Monitor function activation

The Boot SW shall activate the *Monitor* function based on conditions being true only in Ground configuration.

Note:	*activation conditions are based on external conditions set on Ground (e.g. presence of test connector).*
Note:	*the requirement is not applicable if an automatic HW activation condition is foreseen.*
OptionInfo:	*OBC; PLM*
Parameter:	*MonTrigger*
Requirement Rationale:	*if Monitor is executed in flight, it will prevent to load the ASW and to command the computer itself from Ground, therefore the activation conditions should never be true in flight configuration.*
Verification Method:	*T*

SAVOIR.BOOTSW.BEF.67

Monitor function when failure

If activated after the Self-Tests, the Boot SW Monitor function shall be executed also in case of failure in the Self-Tests.

OptionInfo:	*OBC; PLM*
Note:	*Since the final version of the Boot SW is expected to be implemented much earlier than the finalization of the FDIR strategy system requirements, it is recommended to leave to the ASW the implementation of all the SW related FDIR functions.*
Requirement Rationale:	*Self-Tests provides information about the possible cause of a failure occurred while executing the ASW. However, there is always the possibility that the Monitor is executed after a Self-Test failure (e.g. working memory test fails, but the failing locations are not used by the Monitor).*
Verification Method:	*T*

5.2.4 Initialisations

The requirements in this section address the initialisation operations of the Nominal Sequence (ref. Step 1 of SAVOIR.BOOTSW.BEF.20). The order in which the following initialisations are executed is not strictly imposed.

SAVOIR.BOOTSW.BIN.180

Initialisation of integer unit

The Boot SW shall initialise Integer Unit processor registers.

OptionInfo:	*OBC; PLM*
Requirement Rationale:	*IU needs to be initialised to ensure a well-known and deterministic behaviour of the processor during the Boot SW execution.*
Verification Method:	*T*

SAVOIR.BOOTSW.BIN.190

Initialisation of floating point unit

The Boot SW shall initialise Floating Point Unit processor registers.

Note:	*the requirement is applicable if the processor has a FPU.*
OptionInfo:	*OBC; PLM*
Parameter:	*FPU*
Requirement Rationale:	*FPU needs to be initialised to ensure a well-known and deterministic behaviour of the processor during the Boot SW execution.*
Verification Method:	*T*

SAVOIR.BOOTSW.BIN.200

Initialisation of on-chip devices

The Boot SW shall initialise the registers of the On-Chip devices that are used by the Boot SW.

OptionInfo:	*OBC; PLM*
Requirement Rationale:	*On-Chip devices (and so the related registers) need to be initialised to ensure a well-known and deterministic behaviour of the processor during the Boot SW execution.*
Verification Method:	*T*

SAVOIR.BOOTSW.BIN.202

Disabling on-chip devices

The Boot SW shall disable the On-Chip devices that are not used by the Boot SW.

Note: the ASW is expected to initialize itself its own registers. This is usually done through the hardware-software layer.

OptionInfo: *OBC; PLM*

Requirement Rationale: *for a given application, some of the On-Chip devices provided by a System-on-Chip could not be required. Disabling them ensures a well-known and deterministic behaviour of the processor during the Boot SW execution.*

Verification Method: *T*

SAVOIR.BOOTSW.BIN.205

Initialisation of memory management unit

The Boot SW shall initialise MMU registers.

Note: *the requirement is applicable if the processor has a MMU.*

OptionInfo: *OBC; PLM*

Parameter: *MMU*

Requirement Rationale: *MMU (and so the related registers) need to be initialised to ensure a well-known and deterministic behaviour of the processor during the Boot SW execution.*

Requirement Verification *Method: T*

SAVOIR.BOOTSW.BIN.210

Trap handlers registration

The Boot SW shall register trap specific handlers for each of the following trap sources:

- reset
- data_store_error
- instruction_access_MMU_miss
- instruction_access_error
- r_register_access_error
- instruction_access_exception
- privileged_instruction
- illegal_instruction
- fp_disabled
- cp_disabled
- unimplemented_FLUSH
- watchpoint_detected
- window_overflow
- window_underflow
- mem_address_not_aligned
- fp_exception
- cp_exception
- data_access_error
- data_access_MMU_miss
- data_access_exception
- tag_overflow
- division_by_zero

Note:	*the requirement is applicable only to SPARC architecture. Ref. [SPARC-V8] page 76.*
OptionInfo:	*OBC; PLM*
Parameter:	*SPARC*
Requirement Rationale:	*the traps above can occur due to an unforeseen error conditions (HW or SW) or as normal processor behaviour. In case one trap is taken, it is essential that the processor behaves in a known and deterministic way.*
Verification Method:	*ROD*

SAVOIR.BOOTSW.BIN.215

Trap handlers registration when failure

The Boot SW shall register trap specific handlers for each of the trap sources related to error conditions.

Note:	*the requirement is applicable only to non-SPARC architecture.*
OptionInfo:	*OBC; PLM*
Parameter:	*SPARC*
Requirement Rationale:	*the traps above can occur due to an unforeseen error conditions (HW or SW) or as normal processor behaviour. In case one trap is taken, it is essential that the processor behaves in a known and deterministic way.*
Verification Method:	*ROD*

SAVOIR.BOOTSW.BIN.220

Default handler registration

The Boot SW shall register default handlers for every trap source where a specific handler is not defined.

Note:	*a possible implementation of the default trap is to record the trap type in processor memory (RAM) then, depending on the context, either attempt to continue nominal execution, or put the processor in a safe state (i.e. halt). The way to retrieve the information after processor restart is mission-specific (e.g. to foresee a warm restart when memory is not reset).*
OptionInfo:	*OBC; PLM*
Requirement Rationale:	*this is to ensure that the full trap table is defined in memory, so in case one trap is taken, the processor behaves in a known and deterministic way. It is not acceptable to have Boot SW sequential code overlapping with the trap table memory area (i.e. 0x00000000..0x0001000), because in this situation if an unforeseen trap is taken, the execution will be transferred into a "random" position within the Boot SW code.*
Verification Method:	*ROD*

SAVOIR.BOOTSW.BIN.230

Disabling memory EDAC

The Boot SW shall disable memory EDAC at start-up.

Note:	*the requirement is applicable to the memories where EDAC is supported.*
Note:	*during the execution of the Boot SW, after testing the EDAC functionality it is OK to enable it if required by the mission.*
OptionInfo:	*OBC; PLM*
Parameter:	*EDAC*
Requirement Rationale:	*read operations to non-initialised memory can lead to EDAC multiple errors if this is enabled. In addition, the EDAC function could depend on the memory configuration to be setup by the Boot SW at initialisation. Finally it is very likely that access to good data will trigger multiple errors in memory if the EDAC controller or the EDAC check bits memory area are enabled when in failure.*
Verification Method:	*T*

SAVOIR.BOOTSW.BIN.240

Disabling interrupts

The Boot SW shall disable interrupts at start-up.

OptionInfo:	*OBC; PLM*
Requirement Rationale:	*before interrupts are to be handled, the processor needs to initialise a number of registers (for example on SPARC, Trap Base Register, Processor State Register, external interrupts shape and polarity), it is therefore important to start the SW execution with interrupts disabled and to enable (and unmask) them only when the system is capable to handle them (i.e. after Self-Test).*
Note:	*In practice in the Boot SW execution context, this is limited to interrupts provoked by errors detected during Self-Tests (sometimes even self-injected). Note: The requirement addresses in particular interrupts (asynchronous traps) since they are not under SW control, but a similar concept can also apply to synchronous traps.*
Verification Method:	*T*

SAVOIR.BOOTSW.BIN.242

Initialisation by a Single Core

In case of multi-core processors, only one core must be responsible for the initialisation sequence.

OptionInfo:	*OBC; PLM*
Requirement Rationale:	*Ensuring a deterministic and easy to test behaviour. In case other cores would be active and executing SW or interacting with HW in other way, it could be difficult to demonstrate this deterministic behaviour.*
Verification Method:	*T*

SAVOIR.BOOTSW.BIN.245

Secondary core(s) states

In case of multi-core processors, secondary core(s) shall be disabled (i.e. powered-down) all time during the boot software execution.

OptionInfo:	*OBC; PLM*
Requirement Rationale:	*Only one core must be responsible for the initialisation sequence.*
Note:	*initialisation of the secondary core(s) is performed by the ASW after it is started on the primary core. This requirement is applicable only if such feature is available in the Chip.*
Verification Method:	*T*

SAVOIR.BOOTSW.BIN.250

Final configuration

The Boot SW shall set the processor module in a known and consistent configuration before executing the ASW.

OptionInfo:	*OBC; PLM*
Requirement Rationale:	*handover between Boot SW and ASW is critical due to the intrinsic difficulties in the debug and because the two SW items are typically developed by different entities. A well-defined configuration of the processor (registers content) before executing the ASW helps to isolate potential problems in a clean way.*
Verification Method:	*T*

5.2.5 Self-Tests

Requirements in this section address the Self-Test operations of the Nominal Sequence (ref. Step 2 of SAVOIR.BOOTSW.BEF.20).
In support to failure investigation or for performance reasons, some of the Self-Tests mentioned in this section might be skipped during the execution of the Boot SW in a given mode (e.g. warm restart, post-mortem investigations).

SAVOIR.BOOTSW.BTE.260

Testing processor functions

The Boot SW shall test the processor functions supporting Self-Test capabilities.

OptionInfo:	*OBC; PLM*
Note:	*For maintenance and failure investigation it is essential to have visibility from Ground of the health status of as much processor resources as possible.*
Requirement Rationale:	*processor functions designed to support Self-Test (e.g. EDAC, interrupts, cache) need to be tested at boot time because:*
Functions are not used and can be tested in isolation. On the contrary, this is not always possible during the execution of the ASW without effecting its state.	
Cache and EDAC Self-Tests, for example, require read/write to the memory and can't be done completely when ASW is loaded in working memory.	
Error and interrupt detection, for example, requires to have dedicated trap handlers to support the Self-Test. This is acceptable during bootstrap because nominally no interrupts or traps are supposed to be handled by design, while during ASW execution this requires to stop the management of error/interrupt handling.	
Verification Method:	*T*

SAVOIR.BOOTSW.BTE.265

Testing processor functions

if HW BIT is available, the boot SW shall report the built in test results in the boot report.

OptionInfo: *OBC; PLM*

Requirement Rationale: *Self-test can also be handled by HW feature (BITE) in this case the Boot SW is not performing the self-test procedure, but it reports the BITE result by fetching the contents of Flags & registers and putting them in the boot report.*

Verification Method: *T*

SAVOIR.BOOTSW.BTE.270

Testing processor module devices

The Boot SW shall test at least the processor module devices supporting Self-Test capabilities and used during the execution of Boot SW or ASW.

OptionInfo: *OBC; PLM*

Requirement Rationale: *the same justification as per requirement [SAVOIR.BOOTSW.BTE.260] applies.*

Note: *The requirement also addresses communication devices. In this case, Self-Tests done in loop-back mode allows to verify and isolate malfunctions on one side of the communication link, whereas during nominal ASW execution it might not be straight forward to identify the source of the communication problem (local or remote). In addition, loop-back tests can be executed in isolation when the processor/equipment is not active for maintenance purposes (i.e. Standby mode), without conflicting with the rest of the system.*

Verification Method: *T*

SAVOIR.BOOTSW.BTE.280

Testing integrity of boot software image

The Boot SW shall perform an integrity test (checksum/crc) on the Boot SW image (code and data) stored in processor module read-only memory.

OptionInfo: *OBC; PLM*

Parameter: *IntegrityCheck*

Requirement Rationale: *This allows isolating the source of a possible malfunction in support to in-flight failure investigations.*

Verification Method: *T*

SAVOIR.BOOTSW.BTE.290

Testing memory

The Boot SW shall test the processor module volatile read/write memories to detect functional and manufacturing problems.

Note:	*in case of EDAC protected memories, the EDAC checkbits memory banks are also covered by this requirement.*
Note:	*The Boot SW tests the volatile read/write memory addresses that it is going to use for its own purpose, before using it. In case the test fails, a strategy is defined in the actual Boot SW requirement document, e.g. to consider another area for its needs.*
OptionInfo:	*OBC; PLM*
Requirement Rationale:	*this allows isolating the source of a possible malfunction in support to in-flight investigations. In addition, since the tests at boot time are executed every time the computer is turned on, this allows to quickly identifying possible problems due to manufacturing during ground testing (e.g. address, control and data bus connections).*
Verification Method:	*T*

SAVOIR.BOOTSW.BTE.295

Disabling memory self-test

It shall be possible to disable the self-test of the processor module volatile memory.

Note:	*this enables the possibility for post-mortem investigations.*
OptionInfo:	*OBC; PLM*
Verification Method:	*T*

5.2.6 Integrity tests

The requirements in this section address the integrity test operations of the Nominal Sequence (ref. Steps 4 and 6 of SAVOIR.BOOTSW.BEF.20).

SAVOIR.BOOTSW.BTE.300

Testing integrity of original ASW image

The Boot SW shall perform an integrity test (checksum/crc) on the ASW image (code and data) to be copied from Application storage memory to processor working memory.

Note:	*the integrity test can be based on a single crc/checksum applicable to the whole ASW image or multiple crc/checksum associated to multiple blocks compositing the ASW image.*
OptionInfo:	*OBC; PLM*
Assumption:	*Application storage memory; Working memory*
Parameter:	*IntegrityCheck*
Requirement Rationale:	*this allows isolating the source of a possible malfunction in support to in-flight failure investigations.*
Verification Method:	*T*

SAVOIR.BOOTSW.BTE.310

Testing integrity of copied ASW image

The Boot SW shall perform an integrity test (checksum/crc) on the ASW image (code and data) in processor working memory after the copy.

Note:	*the integrity test can be based on a single crc/checksum applicable to the whole ASW image or multiple crc/checksum associated to multiple blocks compositing the ASW image.*
OptionInfo:	*OBC; PLM*
Assumption:	*Working memory*
Parameter:	*IntegrityCheck*
Requirement Rationale:	*this allows isolating the source of a possible malfunction in support to in-flight failure investigations. It is particularly important in the cases when the Application storage memory (where the ASW image is stored) is not EDAC protected.*
Verification Method:	*T*

SAVOIR.BOOTSW.BTE.320

Enabling memory EDAC

The Boot SW shall enable memory EDAC during memory and SW image integrity tests.

Note:	*the requirement is applicable to the memories supporting EDAC capability.*
OptionInfo:	*OBC; PLM*
Parameter:	*EDAC*
Requirement Rationale:	*EDAC is supposed to be enabled when data is accessed/copied for actual use, the result of the integrity test would not be fully representative if run with EDAC disabled.*
Note:	*Memory Self-Tests with EDAC disabled don't exercise the checkbits memory bank, if this is in failure, it will generate multiple errors in memory as soon as EDAC is enabled, for example, by ASW. A correct memory/integrity test will record in the Boot Report the EDAC error correction and/or detections.*
Verification Method:	*T*

SAVOIR.BOOTSW.BTE.330

Integrity test algorithms

The Boot SW shall use the same algorithm for all SW image integrity tests.

OptionInfo:	*OBC; PLM*
Requirement Rationale:	*this reduces effort in AIT and operations.*
Verification Method:	*ROD*

5.2.7 Actions after Tests

Requirements in this section describe the actions to be performed right after the execution of the related Self-Test (ref. Step 2 of SAVOIR.BOOTSW.BEF.20) or integrity tests (ref. Steps 4 and 6 of SAVOIR.BOOTSW.BEF.20).

SAVOIR.BOOTSW.BAA.340

After test reset

The Boot SW shall reset to a fixed pre-defined value the content of the processor module volatile read/write memories after the related Self-Tests.

Note:	*in case it is required for the equipment to support post-mortem investigation, the requirement is not applicable when standby is activated.*
OptionInfo:	*OBC; PLM*
Requirement Rationale:	*this is to have memory initialised to a well-defined state/content. It makes it easier to spot unexpected write access. It also reduces problems during ASW initialisation (BSS, Stack initialisation).*
Verification Method:	*T*

SAVOIR.BOOTSW.BAA.350

After test critical processor module use

The Boot SW shall enable and use critical processor module functions independently from the result of the related Self-Tests.

Note:	*Critical processor module functions are those functions used by the Boot SW that if disabled, prevent the completion of the Nominal Sequence, (e.g. working memory).*
Note:	*The identification of critical processor module functions is done with respect to mission requirement (e.g. maximum boot time, SEU tolerance).*
Note:	*Boot Report is made available anyway to identify and isolate the malfunction.*
OptionInfo:	*OBC; PLM*
Parameter:	*CriticalPMfunctions*
Requirement Rationale:	*if the processor doesn't complete the Boot SW sequence, OBC can't reach a state where the survival functions are active (for example AOCS safe mode, TTC pointing or payload service mode). For this reason, it is worth to try to complete the Boot SW execution also in case of some failure in the Self-Tests. If the failure leads to a malfunction, the reconfiguration or watchdog function will recover the processor module with a HW recovery.*
Verification Method:	*T*

SAVOIR.BOOTSW.BAA.360

After test non critical processor module use

The Boot SW shall enable and use the non-critical processor module functions only if the result of the related Self-Test is successful.

Note:	*Non-critical processor module functions are those functions used by the Boot SW that if disabled, do not prevent the completion of the Nominal Sequence e.g. EDAC or FPU.*
Note:	*The identification of non-critical processor module functions is done with respect to mission requirements (e.g. maximum boot time, memory SEU tolerance)."*
OptionInfo:	*OBC; PLM*
Parameter:	*CriticalPMfunctions*
Requirement Rationale:	*This aims to maximise the probability to complete the Boot SW sequence. For example if EDAC or cache are in failure, it is likely that unrecoverable exceptions will be issued when used, on the other hands the SW can run also without EDAC or cache it is better to leave these functions disabled and pay the associated penalties than failing the Boot SW sequence.*
Verification Method:	*ROD; T*

SAVOIR.BOOTSW.BAA.370

Boot report

The Boot SW shall provide the result of each test in the *Boot Report*.

OptionInfo:	*OBC; PLM*
Requirement Rationale:	*The Self-Tests results are needed to allow Ground to analyse and isolate possible malfunctions.*
Verification Method:	*T*

SAVOIR.BOOTSW.BMM.375

Boot report in telemetry

The result of the tests (*Boot Report*) shall be made available in Telemetry.

OptionInfo:	*PLM*
Assumption:	*If PM redundancy, inter processor link; if PLM, link with OBC*
Requirement Rationale:	*This is to ensure that the Boot Report is available to Ground (via the OBC) also in case of processor reset or if the ASW fails to execute.*
Verification Method:	*T*

SAVOIR.BOOTSW.BAA.380

Boot report summary

The Boot SW shall provide a summary of the *Boot Report* into High Priority Telemetry (HPTM Boot Summary).

Note:	*the requirement is applicable only if HPTM is present.*
OptionInfo:	*OBC*
Assumption:	*Essential telemetry*
Requirement Rationale:	*the Boot Report summary in HPTM is useful because the HPTM is formatted and transmitted to Ground without SW intervention, thus it could be available also in case of SW failure. Due to its simple interface and implementation, it is more reliable than the TM generated/controlled by SW.*
Verification Method:	*T*

SAVOIR.BOOTSW.BAA.385

Boot report summary content

As a minimum, the summary shall include the fast boot path selection (enabled/not enabled), the ASW image selection, and any type of error detected during PM self-tests.

OptionInfo:	*OBC*
Assumption:	*Essential telemetry*
Verification Method:	*T*

SAVOIR.BOOTSW.BAA.390

Boot progress report

The Boot SW shall report its progress in the Boot Report right after each test.

Note: *during the early initialisation of the processor resources, it is acceptable to delay the reporting of the first self-tests until working memory becomes available.*

OptionInfo: *OBC; PLM*

Requirement Rationale: *in case the Boot SW fails due to a malfunction, the Boot Report will be filled up to the point where the failure occurred. This allows correlating the source of the malfunction to the step in the Boot SW sequence and the function under test.*

Verification Method: *T*

SAVOIR.BOOTSW.BAA.400

Boot progress report in essential telemetry

The Boot SW shall report its progress in the HPTM Boot Summary after each test.

Note: *the requirement is applicable only if HPTM is present.*

OptionInfo: *OBC*

Assumption: *Essential telemetry*

Requirement Rationale: *the same justification as per requirement [SAVOIR.BOOTSW.BAA.390] applies.*

Verification Method: *T*

SAVOIR.BOOTSW.BAA.410

Boot reports alignment

In case copies of the same *Boot Report* are stored in different memory areas, the Boot SW shall keep these aligned during the progress updates.

OptionInfo: *OBC*

Requirement Rationale: *the same justification as per requirement [SAVOIR.BOOTSW.BAA.390] applies.*

Verification Method: *T*

SAVOIR.BOOTSW.BAA.420

Boot report integrity

The Boot SW shall allow checking the integrity of the *Boot Report*.

Note:	*Boot SW is not supposed to Self-Test the integrity of the Boot Report, but the method and/or structure used to generate the Boot Report should be such that it will be possible from Ground to distinguish between correctly generated results and corrupted data.*
Note:	*This also includes the progress update done after each Self-Test.*
OptionInfo:	*OBC; PLM*
Requirement Rationale:	*who reads the Boot Report should be able to understand if the content has been filled by the Boot SW, if it has not been updated at all or if it contains corrupted data. If the reliability of the data received is questionable, then the analysis resulting from that can't be fully trusted.*
Verification Method:	*T*

SAVOIR.BOOTSW.BAA.430

Multiple boot report

The Boot SW shall allow to store in protected resource (e.g. SGM) multiple *Boot Report*s in compliance with the number of autonomous reconfiguration sequences foreseen at system level.

Note:	*the requirement is applicable only if a protected resource (SGM) is present.*
Note:	*the requirement is applicable only to the active PM.*
Note:	*It is essential for Ground to identify and isolate root cause of the malfunction and to correlate the Boot Report content to the moment when the failure happened.*
Note:	*In case of multiple autonomous reboot, at each reboot a Boot Report should be available.*
OptionInfo:	*OBC*
Assumption:	*Protected resource retaining data when fault or power loss*
Parameter:	*MultipleBootReport*
Requirement Rationale:	*in case multiple reboots took place, it is likely that the root cause of the malfunction triggered the first reboot. The subsequent reboots might be the result of a cascade effect.*
Verification Method:	*T*

SAVOIR.BOOTSW.BAA.440

Boot report memory slot selection

The Boot SW shall select the memory slot where to store the *Boot Report* in the protected resource (e.g. SGM) depending on the last relevant recovery sequence executed by the reconfiguration function.

Note:	*The number of the reconfiguration sequence that triggered the reboot should be readable from a resource (status bits) local to the processor module and set by the reconfiguration function.*
Note:	*As reading operations of on protected resource (SGM) content require error handling and protections, it is advised not to rely on such resource for identifying which bootstrap sequences already took place .*
OptionInfo:	*OBC*
Assumption:	*Reconfiguration function; Reading recovery action in reconfiguration function; Protected resource retaining data when fault or power loss*
Parameter:	*MultipleBootReport*
Requirement Rationale:	*if the most complex part of the Boot SW is the reporting itself, it is likely that in case of failure this will fail. For this reason the algorithm used to select where to store the Boot Report depending on the reconfiguration sequence should be simple.*
Verification Method:	*T*

5.3 Performance requirements

SAVOIR.BOOTSW.BPF.450

Worst case execution time

Boot SW worst case execution time shall be compatible with reconfiguration time system requirements.

OptionInfo:	*OBC*
Requirement Rationale:	*execution time of the Boot SW is one of the contributions to the OBC reconfiguration time. WCET for the Boot SW should be driven by the mission reconfiguration time requirements and take into account the ASW and external equipment initialisation time.*
Verification Method:	*T*

5.4 Boot SW related PUS System requirements

It is expected to implement the following PUS services in the Standby mode.

5.4.1 PUS service type ST[01] Request verification

5.4.1.1 Acceptance and reporting subservice

SAVOIR.BOOTSW.IF.610

PUS service type ST[01] - acceptance and reporting subservice

The Boot SW shall host an acceptance and reporting subservice provider.

OptionInfo: *OBC*
Requirement Rationale: *Refer to [ECSS-E-ST-70-41C 6.1.2.2.3.a.1].*
Verification Method: *T*

5.4.1.2 Execution reporting subservice

SAVOIR.BOOTSW.IF.620

PUS service type ST[01] - execution reporting subservice

The Boot SW shall host an execution reporting subservice provider.

OptionInfo: *OBC*
Requirement Rationale: *Refer to [ECSS-E-ST-70-41C 6.1.2.2.3.a.2].*
Verification Method: *T*

5.4.2 PUS service type ST[03] Housekeeping

5.4.2.1 Housekeeping reporting subservice

SAVOIR.BOOTSW.IF.630

PUS service type ST[03] – housekeeping reporting subservice

The Boot SW shall host an housekeeping reporting subservice provider.

OptionInfo: *OBC*
Requirement Rationale: *Refer to [ECSS-E-ST-70-41C 6.3.2.2.1.a].*
Verification Method: *T*

5.4.3 PUS service type ST[05] Event reporting

5.4.3.1 Event reporting subservice

SAVOIR.BOOTSW.IF.640

PUS service type ST[05] – event reporting subservice

The Boot SW shall host an event reporting subservice provider.

OptionInfo: *OBC*
Requirement Rationale: *Refer to [ECSS-E-ST-70-41C 6.5.2.2.a].*
Verification Method: *T*

5.4.4 PUS service type ST[06] Memory management

5.4.4.1 Memory management

SAVOIR.BOOTSW.IF.650

PUS service type ST[06] – memory management service provider

The Boot SW shall host a memory management service provider.

OptionInfo: *OBC*
Requirement Rationale: *Refer to [ECSS-E-ST-70-41C 6.6.2.2.a].*
Verification Method: *T*

SAVOIR.BOOTSW.IF.660

PUS service type ST[06] – raw data memory management

The Boot SW related memory management service shall contain a raw data memory management subservice.

OptionInfo: *OBC*
Requirement Rationale: *Refer to [ECSS-E-ST-70-41C 6.6.2.1.1.a.1].*
Verification Method: *T*

SAVOIR.BOOTSW.IF.670

PUS service type ST[06] – check raw memory data

The raw data memory management subservice hosted by the Boot SW shall provide the capability to check raw memory data.

OptionInfo: *OBC*
Requirement Rationale: *Refer to [ECSS-E-ST-70-41C 6.6.3.5.a].*
Verification Method: *T*

5.4.5 PUS service type ST[17] Test

5.4.5.1 Test subservice

SAVOIR.BOOTSW.IF.680

PUS service type ST[17] – test subservice

The Boot SW shall host a test subservice.

OptionInfo: *OBCRequirement Rationale: Refer to [ECSS-E-ST-70-41C 6.17.2.2.a].*

Verification Method: *T*

5.4.6 PUS service type ST[20] Parameter management

5.4.6.1 Parameter management subservice

SAVOIR.BOOTSW.IF.690

PUS service type ST[20] – parameter management subservice

The Boot SW shall host a parameter management subservice provider.

OptionInfo: *OBC*

Requirement Rationale: *Refer to [ECSS-E-ST-70-41C 6.20.2.2.a]*

Verification Method: *T*

SAVOIR.BOOTSW.IF.700

PUS service type ST[20] – set parameter values

The parameter management subservice of the Boot SW shall provide the capability to set parameter values.

OptionInfo: *OBC*

Requirement Rationale: *Refer to [ECSS-E-ST-70-41C 6.20.4.2.a]*

Verification Method: *T*

5.5 Operational requirements

Not applicable.

5.6 Resources requirements

The requirements of this section address the usage of the on-board memory by the Boot SW.

SAVOIR.BOOTSW.BMM.100

Boot software processor initialisation storage

The Boot SW code and read-only data related to processor initialisation shall be stored in the processor Boot Memory.

Note:	*with processor initialisation is meant the step 1 of the Nominal Sequence depicted in [SAVOIR.BOOTSW.BEF.20].*
OptionInfo:	*OBC; PLM*
Assumption:	*Boot memory*
Requirement Rationale:	*the processor initialisation is high critical and it must be prevented to be changed in flight (due to operator error or malfunction). It is also noticed that PROM is more reliable than EEPROM for what concerns data retention.*
Verification Method:	*ROD*

SAVOIR.BOOTSW.BMM.110

Boot software storage

The Boot SW code and read-only data related to processor module Self-Tests, integrity tests, ASW selection/copy/execution and *Standby* function shall be stored in the processor Boot Memory or Application storage memory.

Note:	*with processor module Self-Tests and ASW selection/copy/execution are meant the steps from 2 to 7 of the Nominal Sequence depicted in [SAVOIR.BOOTSW.BEF.20].*
Note:	*Application storage memory area where the second part of the Boot SW is possibly stored should be write-protected in the flight configuration.*
Note:	*As schedule risk mitigation, it might be acceptable to implement a part of the Boot SW in Application storage memory to allow a later bug-fixing/SW update. Nevertheless, it is not acceptable to change these functions in flight.*
Note:	*The choice of Application storage memory or Boot Memory to store the second part of the Boot SW should also be driven by the Application storage memory data retention characteristics with respect to the mission requirements, [EA-2005-EEE-09-A].*
OptionInfo:	*OBC; PLM*
Assumption:	*Boot memory; Application software memory*
Parameter:	*BootSWExec*
Requirement Rationale:	*the same justification as per requirement [SAVOIR.BOOTSW.BMM.100] applies.*
Verification Method:	*ROD*

SAVOIR.BOOTSW.BMM.120

Boot software independence

The Boot SW shall be independent from the actual values of the ASW image parameters.

Note:	*changes in the ASW image (e.g. size) should not require modifications in the Boot SW code or read-only data.*
OptionInfo:	*OBC; PLM*
Requirement Rationale:	*changes in the ASW must not lead to re-programming the Boot Memory.*
Verification Method:	*T*

SAVOIR.BOOTSW.BMM.130

Boot report storage for ASW

The result of the tests (*Boot Report*) shall be stored in a predefined area of processor working memory.

OptionInfo:	*OBC; PLM*
Assumption:	*Working memory*
Requirement Rationale:	*the requirement constrains the designer to have a simple and well-defined interface with the ASW. This also allows the ASW memory usage to be defined in an early development stage.*
Verification Method:	*T*

SAVOIR.BOOTSW.BMM.140

Boot report storage for ground

The result of the tests (*Boot Report*) shall be stored in a predefined area of nominal protected resource (e.g. Safeguard Memory).

Note:	*If the protected resource is outside the perimeter of the PM (e.g. SGM), then the requirement is applicable only to the active PM. The inactive PM should not be able to perform any actions outside its perimeter in case it is faulty.*
OptionInfo:	*OBC*
Assumption:	*Protected resource retaining data when fault or power loss*
Requirement Rationale:	*this is to ensure that the Boot Report is available to Ground also in case of processor reset or if the ASW fails to execute.*
Verification Method:	*T*

SAVOIR.BOOTSW.BMM.150

Boot report redundant storage for ground

The result of the tests (*Boot Report*) shall be stored in a predefined area of redundant protected resource (e.g. Safeguard Memory).

Note:	*the requirement is applicable only if a redundant protected resource (SGM) is present.*
Note:	*If the protected resource is outside the perimeter of the PM (e.g. SGM), then the requirement is applicable only to the active PM. The inactive PM should not be able to perform any actions outside its perimeter in case it is faulty.*
OptionInfo:	*OBC*
Assumption:	*PM redundancy; Protected resource retaining data when fault or power loss; Access to protected resources from nominal and redundant*
Requirement Rationale:	*this is to ensure that the Boot Report is available to Ground also in case of reconfiguration to the redundant processor module branch.*
Verification Method:	*T*

SAVOIR.BOOTSW.BMM.160

Boot reports nominal and redundant storage

Boot Reports generated by nominal and redundant processors shall not interfere with each other in the protected resource (e.g. SGM).

OptionInfo:	*OBC*
Assumption:	*PM redundancy; Protected resource retaining data when fault or power loss; Access to protected resources from nominal and redundant*
Requirement Rationale:	*in case autonomous reboots are triggered on nominal and redundant processors, the Boot Report must not overlap each other in protected resource (SGM).*
Verification Method:	*T*

SAVOIR.BOOTSW.BMM.170

Boot report storage predefined area

The Boot SW shall use a predefined area of working memory and Application storage memory.

OptionInfo:	*OBC; PLM*
Assumption:	*Application storage memory; Working memory*
Requirement Rationale:	*this is to ensure since the early development stage that ASW will not overwrite the Boot Report or other memory areas used by the Boot SW during its initialisation or execution.*
Verification Method:	*ROD*

5.7 Design requirements and implementation constraints

SAVOIR.BOOTSW.IMP.460

Boot software sequential execution

The Boot SW shall be independent of an Operating System or run-time system.

Note:	*With the term run-time system it is intended any SW with functionality equivalent to a RTOS, like for example the ADA run-time system.*
OptionInfo:	*OBC; PLM*
Requirement Rationale:	*OS is needed mainly to handle multitasking, interrupts and complex I/O. These features are not managed by the Boot SW. Moreover, an OS is typically difficult to validate. Since the Boot SW has a simple sequential execution without external asynchronous events or I/O, the usage of an OS adds unnecessary complexity.*
Verification Method:	*ROD*

SAVOIR.BOOTSW.IMP.470

Boot software resources

The Boot SW shall be designed to use the minimal processor module resources necessary to fulfil the requirements.

Note:	*it is good practice to provide analysis and trade-off concerning the selected design justifying the usage of the processor resources during boot.*
OptionInfo:	*OBC; PLM*
Requirement Rationale:	*unnecessary complexity leads to unnecessary effort in development and validation. This has also impact in the OBC or Payload equipment delivery schedule, because Boot SW is normally stored in a one-time programmable device (Boot Memory) to be mounted in the computer before its final qualification campaign.*
Verification Method:	*ROD*

5.8 Security and privacy requirements

Not applicable.

5.9 Portability requirements

None.

5.10 Software quality requirements

SAVOIR.BOOTSW.STD.480

Boot software engineering standard

The Boot SW (including any SW libraries) shall be compliant to the ECSS-E-ST-40C.

OptionInfo:	*OBC; PLM*
Requirement Rationale:	*external SW items/library not under control of the developer should not be accepted for the Boot SW due to its criticality.*
Verification Method:	*ROD*

SAVOIR.BOOTSW.STD.490

Boot software product assurance standard

The Boot SW (including any SW libraries) shall be compliant to the ECSS-Q-ST-80C.

OptionInfo:	*OBC; PLM*
Requirement Rationale:	*the same justification as per requirement [SAVOIR.BOOTSW.STD.480] applies.*
Verification Method:	*ROD*

SAVOIR.BOOTSW.STD.500

Boot software coding standard

The Boot SW shall be developed in compliance with the applicable coding standards.

Note:	*ref. ECSS-Q-ST-80C clause 6.3.4.1.*
OptionInfo:	*OBC; PLM*
Requirement Rationale:	*coding standard should be applicable to the Boot SW development due to its criticality and it should be part of the review data pack during the early stage of the project.*
Verification Method:	*ROD*

SAVOIR.BOOTSW.STD.510

Boot software criticality category

The Boot SW (excepted the *Monitor* function) criticality shall be at least category B.

Note:	*ref. ECSS-Q-ST-80C annex D.1.*
Note:	*Boot SW criticality is meant to be confirmed by dedicated SW criticality analysis.*
OptionInfo:	*OBC; PLM*
Requirement Rationale:	*Boot SW is critical for the related equipment to work or being maintainable in flight. Failure in the Boot SW compromises the function of the equipment during the rest of the mission.*
Verification Method:	*ROD*

5.11 Software reliability requirements

Reliability requirements are considered mission dependent and out of the scope of the current document.

5.12 Software maintainability requirements

None.

5.13 Software safety requirements

None.

5.14 Software configuration and delivery requirements

None.

5.15 Data definition and database requirements

Not applicable.

5.16 Human factors related requirements

Not applicable.

5.17 Adaptation and installation requirements

None.

6 VALIDATION REQUIREMENTS

The validation method is directly provided for each requirement in section 5.

7 TRACEABILITY

The document defines top-level requirements. No traceability to upper level requirements is needed.

8 LOGICAL MODEL DESCRIPTION

Not applicable.

9 HOW TO USE THIS DOCUMENT

9.1 Overview

This document is meant to cover the minimal common set of user's requirements for the Boot SW.
With the exception of the OBC/PLM applicability, there are no explicit options in the requirement, whereas some requirements depend implicitly on a context. Moreover, several requirements or functional areas are likely to need additional features for the SW requirement specification to be produced by the supplier in the context of the related computer HW.
The topic listed below are points to be addressed when flowing down the requirements from RB (this document) to TS (supplier's document), or when complementing a project RB from this document. Note that, despite the document follows the template of the ECSS SRD, it is rather at a level of an SSS (Requirement Baseline).

9.2 Avionics and hardware environment

- Actual computer and avionics, impact on the hardware assumptions [see table below].
- [HardwareMon]: Existence of a debug function by hardware (e.g. DSU) [FN-40]
- [HardwarePatch]: Existence of a memory patching by hardware (e.g. RMAP) [FN-70]
- Watchdog hardware design [no requirement, see section 4.1.1]
- [BootSWExec]: Potential execution of Boot SW part in Application Storage memory or Working memory [FN-110]
- [SPARC]: Non-Sparc computers [FN-210; FN-215];
- [FPU] ; FPU or not [FN-190],
- [MMU]: MMU or not [FN-205];
- [EDAC]: EDAC or not [FN-230; FN-320]

9.3 Boot SW behavior

9.3.1 Mode management

- [ColdWarm]: Cold/Warm restart [FN-10]
- [FastBootPath}: Fast Boot path or not. If yes, list of skipped tests. [FN-15]
- [StbyTrigger]: How and when to trigger StandBy, also depending on the type of hardware (OBC, PLM) [FN-80; FN-90; FN-95]
- [MonTrigger]: How and when to trigger Monitor [FN-50; FN-60; FN-65]

9.3.2 Functionality

- [MultiBootReport]: Several Boot Reports handling strategy, Boot Report erasing strategy [FN-430; FN-440]
- [NumberASWimages]: Number of SW images. Potential algorithm to select the ASW image. [FN-20; FN-25; FN-27]
- [MonFunctions]: Additional functions of the Monitor, for software maintenance on ground
- [IntegrityCheck]: Integrity check method (crc/checksum) [FN-280; FN-300; FN-310]
- Watchdog usage [no requirement, see section 4.1.1]
- [CriticalPMfunctions]: List of hardware function not to use when they fail during the Self Tests [FN-350; FN-360]

9.4 Traceability Requirements vs variability

The following informative table intends to support the reader in using the document. Each requirement is traced:

- To the potential hardware assumption needed for its fulfillment,
- To the parameter that needs to be defined in the actual project.

Assumption: if the actual project does not feature the assumption for any reason justified in the project documentation, then the related requirements of this specification does not apply.

Parameter: when writing the actual project boot software specification, the parameter needs to be defined and the actual requirement is selected, adapted or completed accordingly.

Both are mentioned in the relevant requirement under the title "*Assumption*" or "*Parameter*".

	ASSUMPTIONS OPTION		Boot Memory	Application storage memory	Working memory	Select ASW image from ground	Activate StandBy from ground	Monitor interface	Reconfiguration function	Reading recovery action in reconfiguration function	PM redundancy	Powering both nominal & redundant PM	Protected resource retaining data when fault or power loss	Access to protected resources from nominal and redundant	If PM redundancy, inter processor link	Essential telemetry	Reading the active/ inactive status by PM	If PLM, link with OBC
General		FN-05	x	x	x	x	x	x										
Nominal	ColdWarm	FN-10	x															
	FastBootPath	FN-15																
	NumberASWimages	FN-20		x	x	x												
		FN-22																
	NumberASWimages	FN-25											x					
	NumberASWimages	FN-27				x			x	x								
		FN-30																
StandBy	HardwarePatch	FN-70	x								x	x			x		x	x
		FN-75																
	StbyTrigger	FN-80																
	StbyTrigger	FN-90																
	StbyTrigger	FN-95					x											
		FN-97																

Monitor	HardwareMon; MonFunctions	FN-40						x											
	MonTrigger	FN-50																	
	MonTrigger	FN-60																	
	MonTrigger	FN-65																	
		FN-67																	
Inits		FN-180																	
	FPU	FN-190																	
		FN-200																	
		FN-202																	
	MMU	FN-205																	
	SPARC	FN-210																	
	SPARC	FN-215																	
		FN-220																	
	EDAC	FN-230																	
		FN-240																	
		FN-242																	
		FN-245																	
		FN-250																	
SelfTests		FN-260																	
		FN-265																	
		FN-270																	
	IntegrityCheck	FN-280																	
		FN-290																	
		FN-295																	
In	IntegrityCheck	FN-300		x	x														

	IntegrityCheck	FN-310			x													
	EDAC	FN-320																
		FN-330																
AfterTest		FN-340																
	CriticalPMfunctions	FN-350																
	CriticalPMfunctions	FN-360																
		FN-370																
		FN-375													x			x
		FN-380														x		
		FN-385														x		
		FN-390																
		FN-400														x		
		FN-410																
		FN-420																
	MultiBootReport	FN-430											x					
	MultiBootReport	FN-440							x	x			x					
Perfo		PF-450																
Interface		IF-610																
		IF-620																
		IF-630																
		IF-640																
		IF-650																
		IF-660																

		IF-670																
		IF-680																
		IF-690																
		IF-700																
Resource		FN-100	x															
	BootSWExec	FN-110	x	x														
		FN-120																
		FN-130			x													
		FN-140											x					
		FN-150									x		x	x				
		FN-160									x		x	x				
		FN-170		x	x													
Design		IM-460																
		IM-470																
Quality		QT-480																
		QT-490																
		QT-500																
		QT-510																

Figure 6 Requirement traceability to hardware assumptions and parameters

10 INDEX