# Design and validation of the boot software for the instrument control unit of the PLATO mission

Baldacci, S., Granata, F., Serafini, L., Pannocchia, A., Azzali, M., et al.

S. Baldacci, F. Granata, L. Serafini, A. Pannocchia, M. Azzali, C. Del Vecchio Blanco, V. Zolesi, M. Focardi, R. Cosentino, G. Giusi, E. Galli, S. Pezzuto, "Design and validation of the boot software for the instrument control unit of the PLATO mission," Proc. SPIE 11443, Space Telescopes and Instrumentation 2020: Optical, Infrared, and Millimeter Wave, 114434X (13 December 2020); doi: 10.1117/12.2562209

Event: SPIE Astronomical Telescopes + Instrumentation, 2020, Online Only

# Design and Validation of the Boot Software for the Instrument Control Unit of the PLATO Mission

S. Baldacci [b], F. Granata [b], L. Serafini [b], A. Pannocchia [b], M. Azzali [b], C. Del Vecchio Blanco [b], V. Zolesi [b], M. Focardi [c], R. Cosentino [e], G. Giusi [d], E. Galli d, S. Pezzuto [d]

[b] Kayser Italia S.r.l, Via di Popogna 501, 57128 Livorno, Italy
[c] INAF-OAA, Arcetri Astrophysical Observatory, Largo E. Fermi 5, 50125 Firenze, Italy
[d] INAF-IAPS, Institute of Space Astrophysics and Planetology, Via del Fosso del Cavaliere 100, 00133 Roma, Italy
[e] INAF-FGG, Galileo Galilei Foundation, Rambla J. A. F. Pérez 7, 38712 Breña Baja, TF, Spain

## ABSTRACT

PLAnetary Transits and Oscillations of stars (PLATO) is a mission belonging to the European Space Agency Cosmic Vision program which objective is to find and study extrasolar planetary systems. PLATO is composed of 26 telescopes which will observe uninterruptedly Sun like stars in order to identify a periodic decrease of the star brightness indicating the possible transit of an exoplanet. The PLATO on-board Data Processing System (DPS) consists of an Instrument Control Unit (ICU) and several distributed Digital Processing Units (DPUs) connected together by a SpaceWire network. The ICU collects and compresses scientific data from the DPUs and it implements the main data interface towards the satellite for telemetry and telecommands. The focus of this paper is on the Boot Software (BSW) of the ICU. The BSW is executed on a LEON3FT processor to perform system initialization, hardware checks, telecommand/telemetry management and the start of the ICU Application Software (ASW) responsible of the PLATO sub-system management necessary for the mission objectives. ICU BSW is the only boot software on-board PLATO and its high criticality level requires stringent verification/validation activities and a high-quality control of the software product which is achieved through extensive quality plans, multi-level testing and static analysis of software code. This paper describes the BSW dependable architecture along with the methods used to achieve the required performances, including FDIR techniques. Two engineering models of the ICU are going to be developed and the foreseen functional and performance tests will be presented in this paper.

**Keywords:** PLATO, ICU, SW, Boot, LEON3FT, FDIR

## 1. INTRODUCTION

This paper describes the design, development and verification of the Boot Software (BSW) for the Instrument Control Unit (ICU) of the PLATO payload. A generic overview of the PLATO Data-Processing System (DPS, main core of the PLATO instrument) and a description of the ICU have been given in the initial sections. The ICU CPU board, where software is executed, is then introduced. Last sections focus on describing the ICU BSW starting from the architecture with particular attention to the Failure Detection, Isolation and Recovery (FDIR) functions and to the verification activities related to the BSW design, development and testing.
As the BSW has been classified on criticality level B, this impose a particular attention to the verification activities. An Independent Software Validation and Verification (ISVV) has been asked by ESA in order to meet its criticality level B requirements.

Space Telescopes and Instrumentation 2020: Optical, Infrared, and Millimeter Wave, edited by Makenzie Lystrup, Marshall D. Perrin, Proc. of SPIE Vol. 11443, 114434X · © 2020 SPIE
CCC code: 0277-786X/20/$21 · doi: 10.1117/12.2562209

## 2. PLATO OVERVIEW

PLATO is the selected M-class mission candidate of the European Space Agency's Science Cosmic Vision program foreseen to be launched by end of 2026. “Planetary Transits and Oscillations of stars” aims to characterize exoplanetary systems by detecting planetary transits and conducting asteroseismology of their parent stars. The major breakthrough to be achieved by PLATO will come from its strong focus on bright targets, typically with mV≤11. The PLATO targets will also include a large number of very bright and nearby stars, with mV≤8.
The primary science goals of PLATO are:

- The detection and characterization of exoplanetary systems of all kinds, including both the planets and their host stars, reaching down to small, terrestrial planets in the habitable zone;
- The identification of suitable targets for future, more detailed characterization, including a spectroscopic search for biomarkers in nearby habitable exoplanets
- A full characterization of the planet host stars, via asteroseismic analysis: this will provide us with the masses, radii and ages of the host stars, from which masses, radii and ages of the detected planets will be determined.

These ambitious goals will be reached by ultrahigh precision, long (a few years) and uninterrupted photometric monitoring of the visible light of very large samples of bright stars, which can only be done from space. The resulting high-quality light curves will be used on one hand to detect planetary transits and to measure their characteristics, on the other hand to provide a seismic analysis of the host stars of the detected planets, from which precise measurements of their radii, masses, and ages will be derived.
In order to meet these objectives, the instrumental concept has been based on a multi-camera approach. The telescope, based on a fully dioptric design and working in an extended visible light range is composed of 26 cameras connected to several DPUs (Data-Processing Units). PLATO DPUs consist of:

- 12 normal DPUs (N-DPUs). Each N-DPU is responsible for processing the data of 2 normal cameras.
- 2 fast DPUs (F-DPUs) gathered in one electronic box named FEU (Fast Electronic Unit). Each F-DPU is responsible for processing the data of one fast camera.

The ICU is responsible for the management of the payload, the communication with the Service Vehicle Module (SVM), the compression of scientific data before transmitting them as telemetry to the SVM. All communication links inside PLATO are based on a SpaceWire (SpW) network.
Figure 1 describes the main architecture of the PLATO payload.

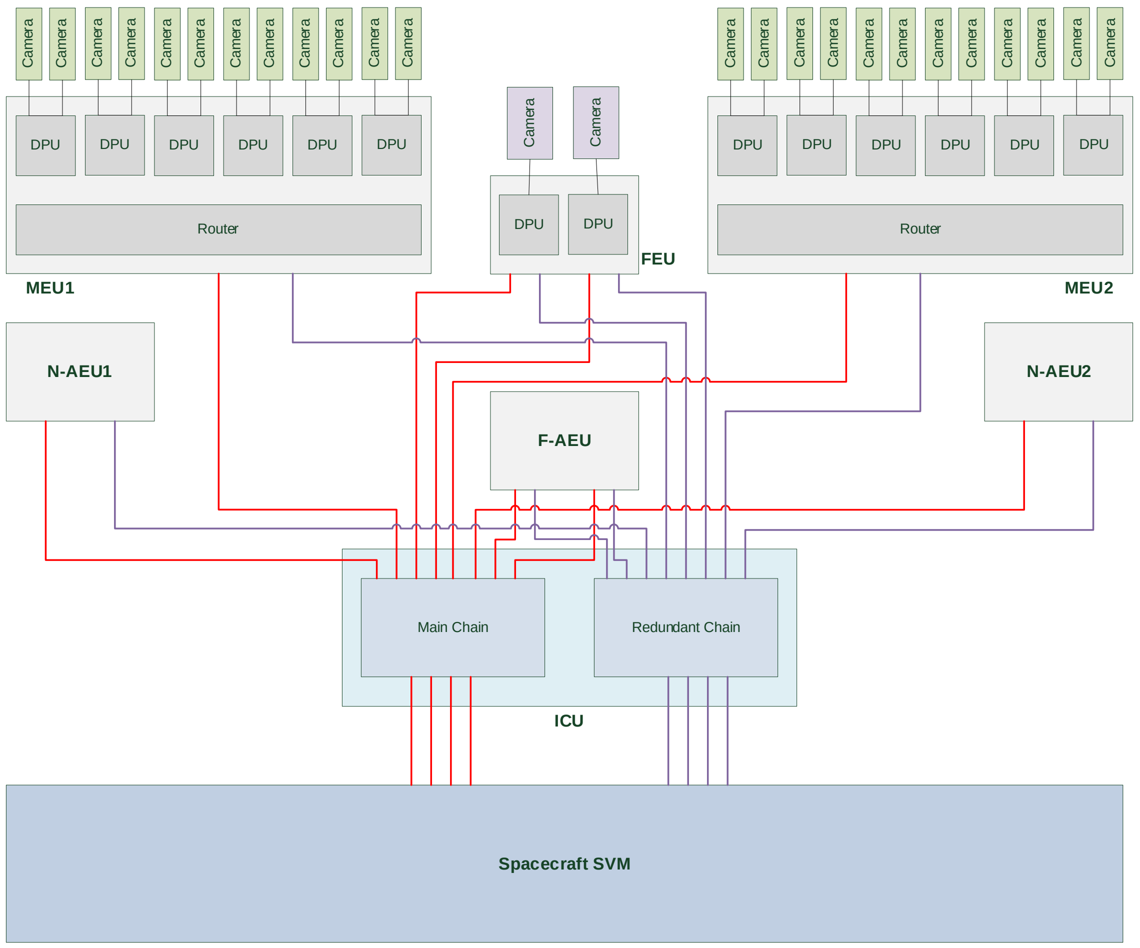

**Figure 1: PLATO Data Processing System**

## 3. ICU OVERVIEW AND ARCHITECTURE

The Instrument Control Unit (ICU) main functions are the following:

- Communicate with the Spacecraft (S/C) via SpW links for telecommands and telemetry.
- Execute Telecommands and forward them to the external PLATO Data Processing System (DPS).
- Collect scientific and housekeeping data from PLATO DPS.
- Compress scientific data and send them to the S/C.
- Monitor the status of the payload and provide it as telemetry.
- Perform FDIR procedures for PLATO payload.

The system architecture of ICU is described in Figure 2. ICU consists of an integrated mechanical box with all internal electronic modules connected by a back-plane for power and data sharing. It is based on 3 nominal + 3 redundant modules in cold redundancy configuration. Modules tagged as "A" are nominal and modules tagged as "B" are redundant:

- Nominal\Redundant power supply Unit (PSU-A\B): it receives the 28V from the S/C and it provides the secondary supply voltages to the A\B modules.
- Nominal\Redundant Processor and Mass Memory Unit (CPU/MMU-A\B): it is the main data-processing module of the ICU. It is responsible for payload management and control and for interfacing the S/C for Telecommands and Telemetry by means of the nominal 4 SpW links.
- Nominal\Redundant Router Data Compressor Unit (RDCU-A\B): it implements a SpW router to put in communication the CPU/MMU-A\B module with the rest of PLATO DPS by the nominal SpW links of Figure 2. It performs a data HW compression in order to be compliant to the compression requirements of the ICU.
- Single Back-Plane (BP): it allows the power distribution and the data sharing between the described modules.

RDCU-A and RDCU-B are the only ICU modules in cross-strapping configuration for power and data. The Boot software is executed in the processor board described in section 5 directly from the PROM memory.

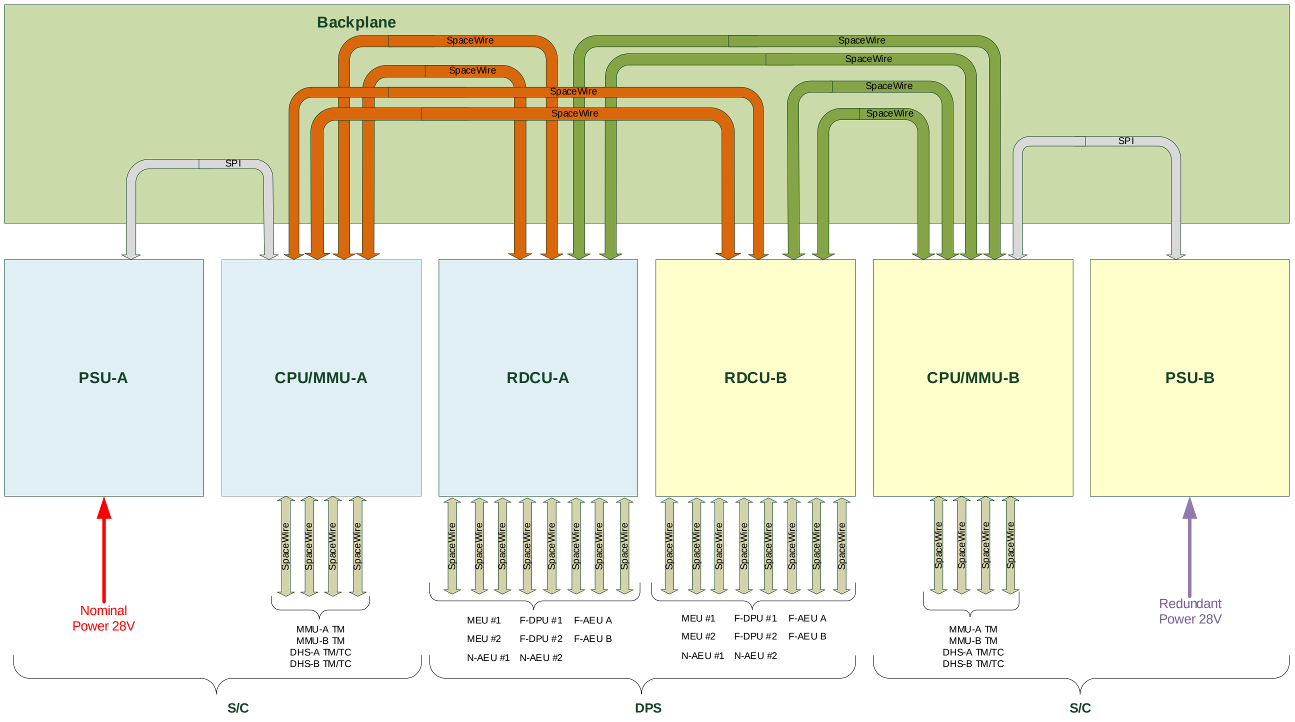

**Figure 2: ICU Architecture**

## 4. CPU/MMU BOARD DESCRIPTION

The CPU/MMU board, shown in Figure 3, is based on the UT700 (LEON3FT) SPARC V8 microprocessor running at 133 MHz, as required by the processing needs. The processor has 4 embedded SpW links allowing to directly communicate with the S/C SVM.
The UT700 microprocessor is connected, by a customized memory bus, to the following memories:

- 128kB of PROM: it is used to store the Boot Software (BSW);
- 16MB of Non-Volatile Memory (NVM), MRAM: it is used to store different images of the Application Software (ASW) of ICU and external DPUs. It will contain also configuration parameters and data;
- 512 MB of SDRAM: it is used to run the ASW and as buffer for data transfer;

SDRAM memories are EDAC (Error Detection And Correction) protected by the memory controller of UT700 with a bus of 48 bits wide (32+16 bits for EDAC function with Reed Solomon algorithm) at 66MHz. A FPGA is used to interface additional 512 Mbyte of SDRAM (32+8 bits for EDAC function with BCH algorithm) via PCI at 33MHz. FPGA is also used to extend the processing and interface capabilities of the processor implementing further four SpW interface links. In conclusion, the CPU/MMU board supports 8 SpW links compliant to [5] and 1 GByte of volatile mass memory.

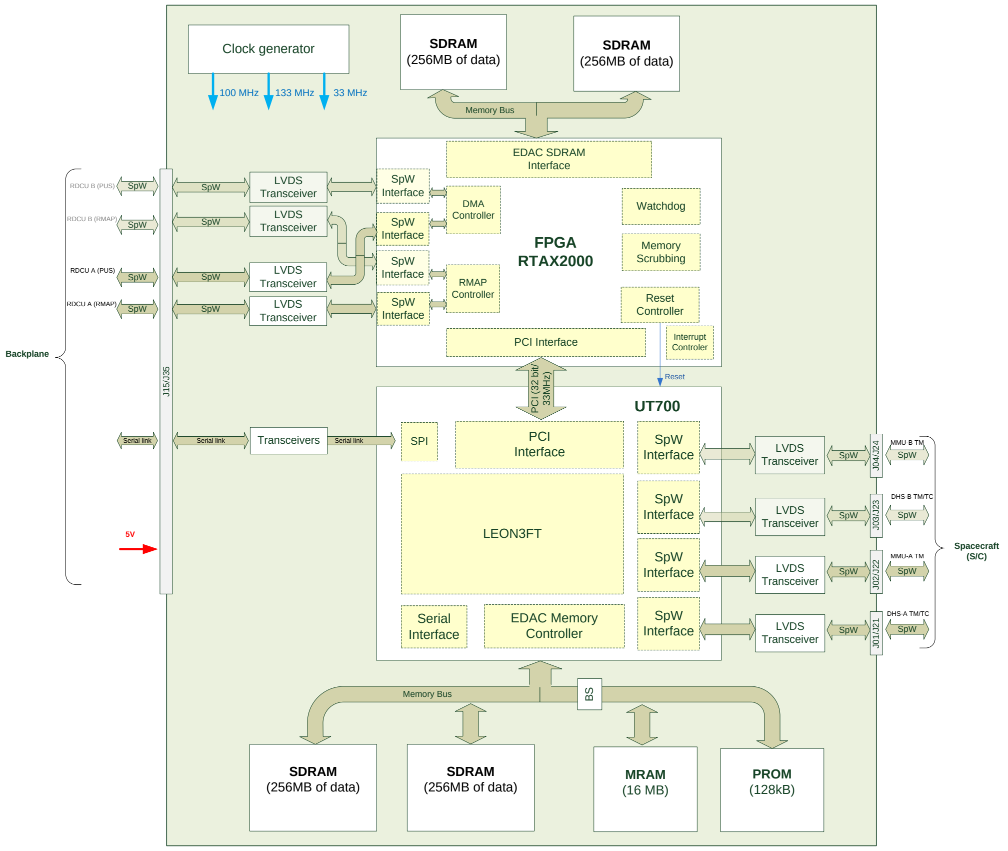

**Figure 3: CPU/MMU module**

The ICU flight software is composed of two distinct products:

- Boot software (BSW): the ICU BSW is the critical and low complexity software, which will be responsible for the maintenance and the boot of the ICU Application Software. It is classed in criticality category B [1].

- Application software (ASW): the ICU ASW is the software that implements the requirements needed to fulfil the PLATO scientific objectives. It is classed in criticality category C [1].

| **Category** | **Definition** |
|---|---|
| A | Software that if not executed, or if not correctly executed, or whose anomalous behavior can cause or contribute to a system failure resulting in: → Catastrophic consequences |
| B | Software that if not executed, or if not correctly executed, or whose anomalous behavior can cause or contribute to a system failure resulting in: → Critical consequences |
| C | Software that if not executed, or if not correctly executed, or whose anomalous behavior can cause C or contribute to a system failure resulting in: →Major consequences |
| D | Software that if not executed, or if not correctly executed, or whose anomalous behavior can cause D or contribute to a system failure resulting in: →Minor or Negligible consequences |

**Table 1: Software category according to [1]**

Next sections focus on the design, development and testing of the BSW.

## 5. BOOT SOFTWARE

The ICU BSW will be stored in the CPU/MMU PROM memory. BSW image will not be erasable or changeable during flight. Consequently, BSW shall be carefully developed and extensively tested before the final delivery of the ICU flight model. The objective of having the lightest software is to simplify the development process, to reduce the complexity of the tests and finally to minimize the technical risks. The BSW can receive the telecommands to update the application software and to fix it in case of problems. For this reason, the BSW is classified with the criticality level B whereas the ASW has the criticality level C.

The BSW tasks are limited to the internal CPU and PSU handling. There is no communication with the RDCU and, therefore, with the external PLATO units. The BSW is executed directly from PROM and it uses the SDRAM connected to the LEON3 as working RAM memory. The BSW reads the vital status (internal voltage, current and temperature) of the ICU by using the LEON3 SPI interface. Due to the critical nature of the BSW, no interaction with the FPGA and in general with the external DPS units are kept. This reduces the complexity and the memory occupation of the BSW increasing its reliability. In order to provide the S/C with the ICU status and to ensure a first level of intervention on the ASW, the BSW sets up a SpW communication with the S/C for Telecommand (TC) reception and Telemetry (TM) transmission.

### 5.1 BSW Main functions

The Boot Software runs after the power-on of the ICU (and so before the OS has booted) and the main tasks of BSW are:

- Check the integrity of the hardware (check of the memories).
- Initialize communication with the S/C.
- Manage internal Housekeeping (HK) data.
- Manage Telecommands (TC) and Telemetries (TM) from/to the S/C.
- Manage the files on the NVM.
- Launch the Application Software
    - Copy of the ASW from NVM to SDRAM and jump to the code

In addition to its main functions, the BSW implements a series of Packet Utilization Standard (PUS) services (see [6]) tailored to manage the ICU before the ASW has started. It is important to specify that only ASW is cross-complied with the Operating System.

### 5.2 BSW Architecture

After HW initialization and self-test, BSW generates a boot report to the Spacecraft with the results of the checks. BSW executes multiple Finite State Machines (FSMs) every single loop. The only exceptions to this context are the interrupt and trap handlers. A timer is used to generate a system clock for the internal timing of all FSMs. The time resolution is 1 ms. The use of the FSMs allows BSW to execute in parallel multiple operations that require execution time more than 20 ms. This means that each operation (acquisition, telecommand execution, …) will be completed after the execution of several steps according to the defined FSM and for each loop only a step of each FSM will be executed. The BSW is implemented as a single infinite loop (see in figure 4) and the main functions are:

- Initialization: it executes the initialization of processor interfaces
- Self-test: it executes the hardware checks of memories
- Acquisition: it acquires and monitor environmental data from the internal and external hardware resources (status flags, interface data, ADC data)
- Telemetry: it provides services to send telemetry packets to Spacecraft like housekeeping, heartbeat and telecommand responses.
- Telecommands: It allows the reception, validation, acceptance and execution of telecommand packets received by the Spacecraft.
- FDIR: this component contains the functions and routines for EDAC, CRC and error management

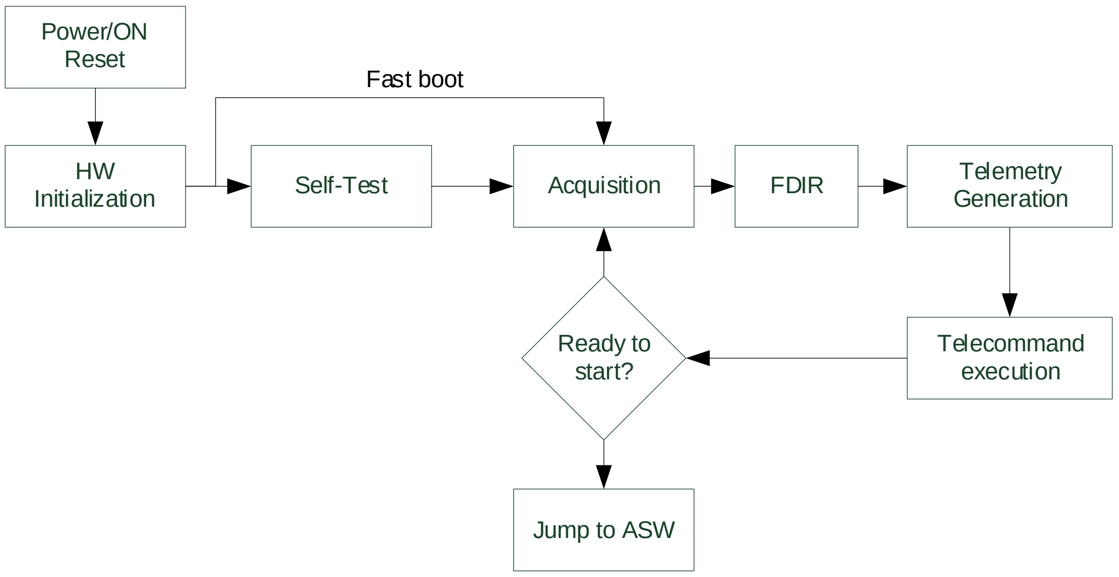

**Figure 4: BSW Main Loop**

The self-test procedure can be jumped in case of fast-boot option. Fast boot and other parameter settings (e.g.: ASW default image to be launched) can be performed writing in a configuration area included in the NVM by telecommands.

The communication protocol between ICU and S/C is based on CCSDS (Consultative Committee for Space Data Systems) [4] and PUS (Packet Utilization Standard) [6] over SpaceWire link for telecommands and telemetries.

Table 2 provides a measurement/estimation of the code and data size used by the BSW for the different types of memory present on the PLATO ICU CPU board. The computation presented are based on the Engineering Model (EM) implementation taking some margins to take into account possible variations.

| **Memory** | **Physical size** | **Code\data size** | **Margin** | **Usage** |
|---|---|---|---|---|
| PROM | 128KB | 90 KB | 30% | PROM is reserved for the BSW |
| NVM | 16MB | < 10KB | >99% | Storage of data to used during execution |
| SDRAM | 512MB | <8MB | >99% | Static data allocation |

**Table 2: BSW memory budget**

Each step of a finite state machine is designed to perform atomic operations in order to reduce the required time as much as possible. An estimation of the maximum execution time for each step is the following:

- Acquisition FSM: 5 ms
- Telecommand FSM: 3 ms
- Telemetry FSM: 5 ms
- Telecommands:
    - Memory Check FSM: 7 ms
    - Memory Dump FSM: 7 ms
- FDIR:
    - SDRAM Scrub FSM: 1 ms
- ASW Start FSM: 25 ms

When the last step "ASW Start FSM" has been activated, Telecommands, Memory Check and Memory Dump FSMs are not active.

## 6. BSW FDIR FUNCTIONS

BSW is designed taking into account the space environment where it shall operate. Specific FDIR functions are implemented at BSW level in order to increase the sub-system reliability and the failure-tolerance capability.

### 6.1 Memory Management

BSW is executed directly from PROM in order to reduce the probability of malfunctions due to SEU/SEFI in the SDRAM. PROM memory contains:

- BSW bit file
- Read-only default parameter values embedded in the BSW code
- Cyclic redundancy check (CRC) value of BSW

When BSW starts after power-on/reset, it reads the PROM area computing the CRC and it compares the CRC result with the expected value stored in PROM in order to verify the BSW image integrity. BSW uses the SDRAM memory for working data and for the Program Stack (less than 8Mbyte in total). This SDRAM memory area is fixed (it cannot be dynamically reallocated) and, for this reason, classified as critical.

NVM is managed at low-level implementing a sort of simplified file system. A predefined NVM structure (called NVM header) will be uploaded in this memory to allow the management of data files. NVM header and files are saved with CRC in order to check the memory integrity in any moment using a dedicated telecommand. BSW makes use of parameters and configuration data for its operation. These values are managed in NVM memory and they can be updated by telecommand. The default values of BSW configuration data are read-only and hard-coded in the BSW (as mentioned above). In case of corrupted NVM, BSW operates with the default hard-coded parameter values to be allocated in SDRAM (in the working area), it generates an event to inform the S/C and it waits a telecommand to upload a new NVM header (file structure) in MRAM.

CPU SDRAM memory is protected by EDAC logic using the reed-solomon algorithm. Each 32-bit word has a checksum of 16-bit and it is possible to detect and correct up to two 4-bit nibble errors in a 32-bit data word or 16-bit checksum. The classification of EDAC errors for LEON-based processor board consists of:

- Correctable errors: Corrupted bits in a DWORD or checksum can be detected and corrected by EDAC logic
- Uncorrectable errors: Corrupted bits in a DWORD or checksum can be detected but not corrected by EDAC logic

All Single Bit Errors (SBEs) can be detected and corrected by EDAC. All Multiple Bit Error (MBE) can be detected but only a part of them can be corrected by reed-Solomon algorithm.

CPU SDRAM memory is firstly initialized (data and checksum words) by the BSW. Then, BSW scans all SDRAM memory cells to detect possible EDAC errors after initialization during nominal execution. Only the SDRAM used by BSW is scrubbed with a dedicated software FSM:

- In case of a correctable error is detected, it will be corrected by the SW scrubber finite state machine.
- In case of an uncorrectable error is detected, BSW will generate a Death report in NVM with all useful information about the failure and a reset of the processor will be performed.

BSW also performs a self-test of the EDAC functionality during the boot phase.

SDRAM can be affected by stuck-bit errors. A bit in a SDRAM word can remain stuck to 1 or to 0 due to a charge particle hit (e.g. a cosmic ray). Stuck-bits are estimated as the less frequent single event effects (SEEs) during the PLATO mission timeframe and they cannot be corrected. BSW is designed to identify possible stuck-bits in the SDRAM after power-on during the boot phase. Stuck-bits are not directly recognized by the hardware and they can be seen as uncorrectable errors (unlikely) or correctable errors (likely). After memory wash, BSW scans the SDRAM memory to check possible HW corrupted words (by stuck bits) writing pre-defined values and reading back them to perform a comparison. In case of a detected error, BSW will mark the words as "HW corrupted" and it will communicate the errors to the S/C by a specific event. The memory scan is performed with two different data patterns for both data and checksum words.

## 6.2 Management of redundant SpW links

CPU/MMU board provides 4 SpW interface links to the Spacecraft. BSW uses only two links (one nominal and one redundant) for telecommands and HK telemetry and it maintains the other two unused links as disabled. The two active SpW links are configured in hot-redundancy. BSW starts sending HK telemetry in the nominal link, it waits a TC from one of the two links and then it sends telemetry where it receives telecommands. If the nominal link is in failure (not in running), BSW will send telemetry in the redundant one after reset.

## 6.3 Housekeeping Monitoring

BSW acquires external housekeeping (HK) parameters to monitor the HW activity and implement some basic FDIR procedures. Current, voltage and temperature of CPU/MMU and RDCU are monitored and in case of a parameter is out of the pre-defined limits for three consecutive times, a "event" packet is sent to the S/C with a specific severity level. At this point the recovery action is under responsibility of the Spacecraft that can decide to switch to the redundant chain of the ICU. The parameter thresholds are stored in a file in NVM and they can be modified.

## 6.4 Trap Handling

A trap is like an unexpected procedure/task call. A trap may be caused by an instruction-induced exception or an external interrupt request not directly related to a particular instruction. For each trap, a specific action was defined and a trap handler was developed. Basically, for the most of traps, in case of an occurrence of a "trap failure", BSW will write a death report in the NVM with the content of CPU registers and it will generate a reset.

# 7. BSW VERIFICATION AND VALIDATION

This section deals with the verification and validation process for the BSW design. Verification approach is composed of several activities performed through different means (e.g. reviews, inspections, analysis, prototyping, audits, specific tooling) and reported in a Software Verification Report at each project review.
Verification process started at the beginning of the project according to the presented plan and project milestones. Due to its criticality, BSW design is verified and validated by Kayser Italia and by an independent Organization. The Independent Software Verification and Validation (ISVV) is a procedure targeted at safety-critical software systems and aims to increase the quality of software products reducing the risks and costs through the operational life of the software. ISVV provides assurance that software performs to the specified level of confidence and within its designed parameters and defined requirements. ISVV activities shall be performed by independent engineering teams. While Kayser Italia aim to ensure that the software performs well against the nominal requirements, ISVV is focused on non-functional requirements such as robustness and reliability. ISVV results and findings will be fed back to the development teams for correction and improvement.

## 7.1 Verification of SW Specifications

A detailed BSW specification was written starting from the user baseline requirements provided by the ESA and DLR agencies, leading the PLATO project. For Each requirement specification, a single or multiple verification method was defined:

- R: Review of design
- I: Inspection of code
- T: test at board level or unit testing
- A: Analysis

Each requirement specification is traced from the input user requirement document and used to generate test and inspection procedures. The purpose of this activity is the check the Software Specification against the Requirements Baseline in terms of:

- Consistency of interna/external software interfaces
- Traceability between requirements baseline and software specification
- Verifiability of SW specifications analyzing that all the requirements have a valid feasible verification method associated
- Conformity to product metrics

## 7.2 Verification of SW Architecture

The purpose of this activity is the verification of the architecture design of the BSW to ensure a full consistency between the Software Components and to verify the complete traceability of the Software Specifications versus the Software Components (and vice-versa).

## 7.3 Verification of SW Source Code

BSW source code is verified to guarantee that the code is conformal to coding standards and product metrics using Static Analysis tool from LDRA® Toolsuite applying the rule set define in the MISRA-C:2012 Standards Model. The static analysis focuses on checking the application of the correct coding standard, measuring the complexity metrics of the developed code, checking the presence of unreachable code, checking the correct use of variables and data types, checking the presence of potential division by zero and checking the correct handling of error propagation (the value returned by the called subroutines).

## 7.4 BSW Validation

SW validation is based on standard testing approach and it consists of unit testing and system integration tests. Unit testing is a software testing method where individual units of source codes are tested to determine whether they are correctly developed. It is an important test activity where BSW functions can be verified more accurately providing unexpected inputs to check robustness and failure tolerance. The Unit testing is the program-based individual component verification process to ensure most of the development artifacts are independently tested for their expected behavior. LDRA® Toolsuite is used to implement the unit testing and this analysis is necessary also to verify the code coverage. The criticality level of the BSW requires at least the 85% of code coverage using the MC/DC criterion. Following a successful unit testing, a System Integration Tests is performed where all the interactions between SW components of the application are tested at board-level using Electrical Ground Support Equipment (EGSE) and Special Test Equipment (STE).

## 8. TEST RESULTS

The first release of BSW was successfully tested with the first Engineering Model (EM1) of ICU. EM1 implements only one electronic chain (the nominal) and a picture is shown in Figure 5.

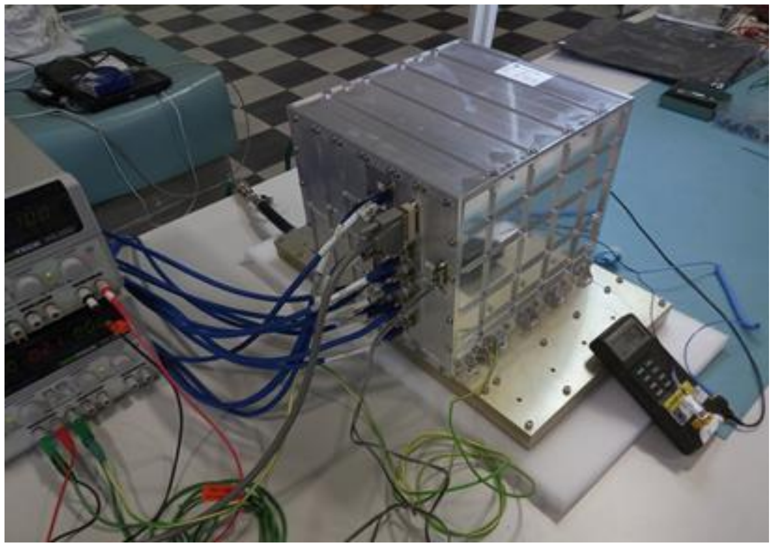

**Figure 5. ICU EM1 test setup**

BSW was tested with the test setup of Figure 6 at functional level. A CPU/Extension board was developed to provide the power to the CPU/MMU and the SPI interface for the ADC acquisition of housekeeping data emulating the PSU unit. A dedicated S/C simulator with 4 SpW link interfaces was developed to send CCSDS/PUS telecommands and receive CCSDS/PUS telemetry to/from BSW.

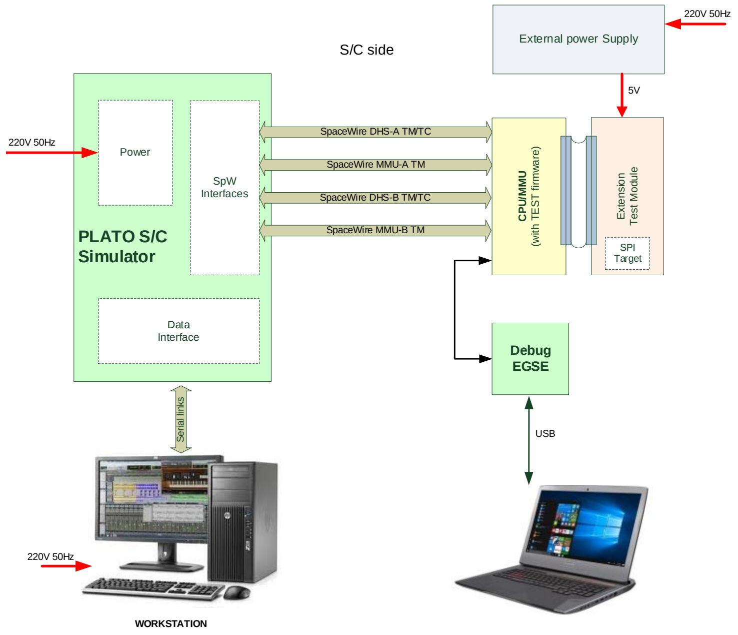

**Figure 6. BSW test Setup at board level**

A debug EGSE (Electrical Ground Support Equipment) is used to develop the BSW by JTAG or UART and to monitor the BSW activity in debug mode using a further UART port. The telemetry HK packet of BSW was defined in order to increase the observability of the internal BSW activities and verify the most of the functions by sending telecommands and checking the corresponding records of the telemetry.

All the required BSW functions were validated using the S/C simulator:

- Nominal boot with and without self-test and fast boot.
- Telecommand validation injecting errors
- PUS Service 3
- PUS Service 6
- PUS Service 5
- PUS Service 9
- PUS Service 190 (custom)
- PUS Service 192 (custom for NVM management)
- Off nominal conditions:
    - NVM header not available
    - Configuration data file not available
    - SpW link disconnection.

The BSW program is about 90 kbyte, not compressed and stored in PROM. It is executed directly from PROM and it uses only 8Mbyte of working DRAM area for data buffers, working variables and program stack. The main super-loop of BSW lasts about 15ms in the worst case: this value was measured with a dedicated setup using an oscilloscope to acquire a digital signal on a GPIO moved up/down by a dedicated version of BSW.

## 9. CONCLUSIONS

This paper introduced and described the PLATO payload focusing on the boot software of the Instrument Control Unit. BSW is the only boot SW within the overall data processing system of PLATO and, for this reason, it has been marked with the highest criticality level with respect to all application software included in the DPUs and in the ICU it-self. A BSW description was provided and the verification approach was presented. Tests on the first engineering model of ICU were performed validating the proposed architecture. Testing and verification will continue in the next phase of the project to increase the reliability of the software product, up to the ISVV closure.

## 10. ACKNOWLEDGEMENTS

A special acknowledgment to the Italian Space Agency (ASI) for the support to the Phases B/C of the PLATO Project, as defined within the ASI-Kayser Italia contract N° 2017-1-I.2-2020.

## REFERENCES

[1] ESA-ESTEC, ECSS-Q-ST-80C, "Software product assurance", 6 March 2009
[2] Steven M. Guertin, Member, IEEE, Gregory R. Allen, Member, IEEE, and Douglas J. Sheldon, "Programmatic Impact of SDRAM SEFI"
[3] SPARC International Inc., "The SPARC Architecture Manual", Version 8
[4] ECSS-E-ST-50-53C, "SpaceWire - CCSDS packet transfer protocol"
[5] ECSS-E-ST-5012C, "SpaceWire - Links, nodes, routers and networks"
[6] ECSS-E-ST-70-41C, "Telemetry and telecommand packet utilization"
[7] ECSS-E-ST-40C, "Software"
[8] M. Focardi, S. Pezzuto, R. Cosentino, G. Giusi, A. M. Di Giorgio, D. Biondi, C. Del Vecchio Blanco, L. Serafini, D. Vangelista, M. Steller, H. Jeszenszky, H. Ottacher, G. Laky, R. Ottensamer, F. Kerschbaum, M. Guedel, V. Noce, E. Pace, M. Pancrazzi, K. Westerdorff, G. Peter, B. Ulmer, R. Berlin, P. Plasson, I. Pagano, E. Tommasi, S. Natalucci "The design of the Instrument Control Unit and its role within the Data Processing System of the ESA PLATO Mission", SPIE 2018 - Astronomical Telescopes + Instrumentation, Austin Texas USA, 10–15 June 2018
[9] M. Focardi, S. Pezzuto, R. Cosentino, G. Giusi, D. Biondi, C. Del Vecchio Blanco, L. Serafini, D. Vangelista, "The PLATO Payload Data Processing System SpaceWire network", Compeng2018, Florence, 10-12 October 2018
[10] M. Rotundo, A. Leoni, L. Serafini, C. Del Vecchio Blanco, D. Davalle, D. Vangelista, M. Focardi, S. Pezzuto, R. Cosentino, G. Giusi, D. Biondi, Fanucci, "Simulation and Validation of a SpaceWire On-Board Data-Handling Network for the PLATO Mission", Compeng2018, Florence, 10-12 October 2018