Dedicated On-Board Computer for Active Debris Removal Mission

Michaël Juillard, Muriel Richard-Noca, Jean-Paul Kneib
EPFL Space Center
Ecole Polytechnique Fédérale de Lausanne (EPFL), 1015 Lausanne, Switzerland ; +41 21-693 40 55
michael.juillard@epfl.ch

ABSTRACT

This paper describes the first steps toward the implementation of a dedicated rendezvous payload on-board computer for the Active Debris Removal mission ClearSpace-1.

Challenges of ADR missions lay in their ability to first detect and track a target, then perform proximity operation and capture. It implies a variety of sensors which are needed for the Guidance, Navigation and Control of the spacecraft. Sensor outputs need to be processed to retrieve position and attitude estimation of the target, then results are transmitted to the GNC algorithms for precise navigation. To obtain accurate target information, the algorithms require a high input data rate and multiple sensor sources.

The EPFL Space Center and the start-up ClearSpace are working on a dedicated payload computer for their ADR mission. The mission will use a standard satellite bus developed for Earth observation and combine it with another on-board computer for all the tasks specific to the mission. The current testbench setup has the satellite bus physical processor board connected through SpaceWire and Ethernet to a simulator and a payload computer prototype. By implementing a Hardware-In-the-Loop setup, the team is able to assess various configurations for the satellite.

INTRODUCTION

In space, Active Debris Removal (ADR) missions can be seen as a mix between formation flying and docking but with additional constraints. These are usually found in the lack of information regarding the target and uncooperativeness of the object. In most cases, there is little knowledge regarding the state and rotation of the debris. Although optical ground tracking can provide estimation of the attitude (orientation of an object in a specific reference frame) and the shape of large satellites, it is impossible to resolve it for small satellites <2[m] from optical observations. For these reasons, the avionic of an ADR mission is crucial for gathering and processing information about the object. In addition, the spacecraft needs to be reactive and have a high level of autonomy especially in the final approach phase.

This publication focuses on the development of the payload computer as well as its interaction with the main On-Board Computer (OBC). The early design of the payload is shown, communication protocol is established, and a time management method is developed.

ClearSpace-1

ClearSpace-1 (CS-1) is a mission developed by the start-up company ClearSpace together with the EPFL Space Center. The goal is to rendezvous with VESPUP (VEga Secondary Payload Adapter Upper Part) and deorbit it. A similar mission was being designed by the start-up in an earlier phase and it was called *CleanSpace One (CSO)* [1]. This satellite will demonstrate technologies needed for future ADR missions. The development focuses on two main aspects. The first one is a capture mechanism able to retract and deploy multiple times, moreover it should perform a soft capture of the target. The second major development is the creation of a Payload On-Board Computer (POBC or simply payload computer) to host the different relative navigation and image processing algorithms. It should merge the data from various sensors needed for the approach and rendezvous phases.

To achieve a high degree of confidence during the final approach phase, the spacecraft needs to embark a variety of sensors with a wide spectrum of range and accuracy. These sensors will feed algorithms running in the satellite's POBC which will compute accurate information about the target. To guaranty constant tracking and prevent any biasing of the pose estimation, the algorithms require high input data rates from multiple sensor sources. Furthermore, due to orbital mechanism and potentially low ground coverage, data has to be analyzed on-board to satisfy a constant control of navigation. For the same reason, Collision Avoidance Maneuvers (CAM) and the majority of mission critical operations must be computed and performed autonomously.

For this mission, the idea is to use a standard avionic architecture coupled with a dedicated payload computer. In order to reduce the cost and development time needed for the satellite, it has been decided to use a pre-existing avionic platform and adapt it for the need of the mission. To preserve the integrity and consistency of the used platform, the team decided to focus on a dedicated payload that will have to handle all the remote sensors needed for the rendezvous. Critical parts of the software, such as the various image processing algorithms, will be implemented directly on this payload in order to benefit of dedicated hardware accelerators.

As said before, the mission will use a standard satellite bus developed for Earth observation and combine it with a Payload On-Board Computer for all the tasks specific to the mission. The setup includes the OBC with the physical processor board which is connected with SpaceWire and Ethernet to a simulator and the payload computer prototype. The simulator emulates instances of all the satellite's standard components such as power and telecommunication. In the first iteration, the payload core is a simple board running a minimal version of Linux that carries various tasks.

The primary function is to handle the multiple sensors dedicated to the detection, tracking, and analysis of the target. The dedicated payload computer acts as a controller and manages all the modes for the instruments as well as the Failure Detection Isolation and Recovery (FDIR). It also has the responsibility to control the data flow from the sensors to the different processing units. The secondary function is to manage the hardware accelerators linked to the payload and ensure the distribution of the data to the correct unit. The last function is to communicate and operate with the main satellite bus considering the requirements of the precise GNC. It provides processed data to the control algorithms of the spacecraft and reacts to the telecommand sent from the ground. The synchronization between the main bus and the payload core is capital in order to ensure correct navigation and prevent any collision with the target. Moreover, the payload should be responsive enough to manage all its tasks without creating issues when critical information is requested by the main bus. GNC algorithms for relative navigation are a typical example where minimizing latency is decisive. The interaction between the payload computer and the OBC is the main topic of this paper.

Flexible LEO Platform

The Flexible LEO Platform (FLP) [2,3] is an avionic platform design by Airbus DS Friedrichshafen. The first version has been sent to space with the "Flying Laptop" satellite from the University of Stuttgart. This satellite was launched in July 2017 in LEO and is still perfectly operational up to this day (June 2020).

In its first version, the platform contains two separated hardware boxes, one is the Power Control and Distribution Unit (PCDU) and the other is the On-Board Computer (OBC). Together, they are called the Combined Data and Power Management Infrastructure (CDPI). The PCDU has the power management, bus regulation and few controls over the OBC in case of failure. The OBC hosts the processor board, the communication board and an interface board. All parts together, the OBC controls the attitude, communications, the different sensors and includes the Failure Detection, Isolation and Recovery (FDIR). A new version of the platform (FLP2) is currently being developed in Friedrichshafen. This version will bring higher computation capability, a propulsion module and few other features.

In the process of creating the CS-1 mission, it has been decided to use the FLP2 platform as main avionic for the satellite. The reasons are its modular architecture and its high level of reliability. The exact implementation envisioned is presented later.

Literature Review

It is capital to understand the challenges of the various past and on-going space missions in order to develop a proper avionic that suits the need of the CS-1. The most interesting categories of mission is of course the Active Debris Removal (ADR) such as RemoveDEBRIS [4], e.Deorbit [5], NanoRacks-Remove Debris [6], and ELSA-d [7]. Similar types of problems can also be found with recent formation flying and docking mission considering an almost autonomous approach: ATV [8], [9], PRISMA [10], and AVANTI experiment [11].

The ATV [8], [9] (Automated Transport Vehicle) project lead by the European Space Agency (ESA) was created in the late nineties to resupply the International Space Station (ISS). In the end, five vehicles have flown and each of them was based on a similar design, but every iteration brought few improvements. Its main characteristic was its ability to almost automatically dock the ISS by using a set of different sensors. The safety of the vehicle and the capability to abort the final approach with up to two failures [12] was crucial. Moreover, it was using two sets of sensors for respectively the far range/formation flying and the final approach with a transition at around 300[m]. First relative GPS was used and then optic instruments (Videometers) based on Star-Tracker technologies with reflectors on the ISS [13]. Most of the control loop operated at 1[Hz] but few of the instruments provided data at 10[Hz]. Concerning the hardware architecture, the ATV avionics was fully redundant in all parts [14]. It had four main fault tolerant computers and each one was connected to four system buses.

Another important mission is the PRISMA [10] (Prototype Research Instruments and Space Mission Technology Advancement) project by the Swedish Space Corporation with the participation of many other actors. Launched in 2010, the goal was to demonstrate formation flying and rendezvous technologies for small satellites. The mission had two small satellites and successfully demonstrated different technologies such as relative GPS navigation, RF sensors for ranging, vision-based sensors with a Star Tracker. The main satellite was

a 150[kg] spacecraft with two redundant On-Board Computers and a variety of sensors. Its main computer board were based on a LEON processor and handles all the software including the GNC [15], [16]. The consortium wanted to achieve several objectives such as formation flying, autonomous rendezvous and proximity operation and all of them needed to use the processors aforementioned. Each of the phases had its own set of algorithms [17] and the data handling system is based on a LEON3 microprocessor with a 32-bit SPARC V8 architecture. It is a fault tolerant processor and is implemented on a FPGA which provides 20[MIPS] and it has a floating-point unit. It uses a CAN bus to communicate with the rest of the spacecraft.

The next major step in the domain is the AVANTI [11] (Autonomous Vision Approach Navigation and Target Identification) experiment by the DLR. The project was launched in 2016 in the BIROS satellite. The goal of this secondary scientific experiment was to demonstrate relative orbit determination with Angle-Only (AO) navigation, maneuvers planning and on-board safety monitoring. It used the CubeSat BEESAT-4 for its uncooperative rendezvous experiment. Its Star Tracker were used for vision-based navigation from far- to mid-range and to demonstrate autonomous rendezvous. This demonstration shows the feasibility of relative navigation with a unique sensor and low data-rate.

The satellite itself was based on an avionic platform called TET-X (direct upgrade of the TET-1) and built by OHB Sweden [18], [19]. The satellite's architecture was composed of four redundant computers that controlled every aspect of the spacecraft. Inside the computer, two separated nodes, respectively "worker" and "master" node, were created. The first one controlled the satellite and the second one supervised the correct operation. The AVANTI experiment could also use the Pre-Processing Unit (PPU) memory (80[MB]) designed for the other BIROS experiment. As previously mentioned, the AVANTI experiment used "standard" sensors for its relative navigation. Concerning the GNC, the algorithms were based on an analytic model and were updated every 30[s].

The last past mission in the domain is probably RemoveDEBRIS [4], mission that included Airbus, ArianeGroup, CSEM, SSTL and other actors. This mission was launched in April 2018 to the ISS and later released from it. The mission includes a net experiment, vision-based navigation and target inspection with a LiDAR, multiple optical cameras, a harpoon and also a dragsail. Most of these technologies would be beneficial for the CSO mission, especially the vision-based navigation and the target inspection.

The most known ADR mission is probably e.Deorbit [5] lead by ESA which is planned to be launched in 2025. Part of this mission is the HIPNOS (HIgh Performance avionics solutioN for advanced and complex GNC Systems) project [20] which aimed to developed avionics and algorithms for

ADR missions. The project's goal is to move out of the space grade processor (LEON3 or RAD750) and use COTS components to lower development prices and increase high performances. Moreover, due to the short time in orbit, they wanted to use non-Rad-hard component. Their vision was to use a System-On-Chip device with processor and FPGA integrated. The hardware used for their development was the Zynq MMp board with an FPGA and a dual-Core ARM Cortex CPU [20].

Regarding future missions, it is important to mention the ELSA-d satellite from Astroscale [7]. The mission is planned to launch in 2020 and wants to demonstrate rendezvous operation and capture mechanism with a small target. Their demonstration mission will be particularity interesting because of their maneuver for close rendezvous and the performances of their navigation system. The main difference is the use of a magnetic docking plate for the capture system limiting the type of objects they can capture and rules out most space debris.

CURRENT ARCHITECTURE

As stated in the introduction, the current avionic architecture is composed of a standard bus acting as the main On-Board Computer (OBC) connected to a dedicated payload computer. The standard bus is the FLP2 platform and its predecessor has been used as an Earth observation satellite that is still in orbit in 2020. It has been designed to be flexible and modular with all the nominal functions of satellite implemented, ranging from the AOCS (Attitude and Orbit Control System) to the communication protocol. This platform enables the ClearSpace-1 mission team to dedicate their times and resources toward the specific payload computer and the capture mechanism.

This main avionic platform is connected to the dedicated payload computer via SpaceWire and Ethernet. In FLP2, SpaceWire is the regular connecting method between the various boards of the avionic. However, in order to ease the implementation of the payload, it has the possibility to be connected with an Ethernet cable.

Regarding the software architecture, the FLP2 platform is mainly coded in C++ with the operating system based on a Real Time Operating System (RTOS), RTEMS (Real-Time Executive for Multiprocessor Systems).

More information on the platform architecture can be found in “A Combined Data and Power Management Infrastructure” and “The FLP Microsatellite Platform - Flight Operations Manual” by *Eickhoff, Jens and Al.* [2], [3].

Hardware

The current testbench setup hosted at the EPFL Space Center is a simplified replica of the one found at Airbus DS Friedrichshafen. It has the processor board connected to a SpaceWire router and then an Ethernet-SpaceWire Bridge. The Ethernet connection is then plug into a regular router connecting to the satellite’s simulator and the prototype payload. The Figure 1 shows the layout.

The setup also included a development workstation connected with UART and JTAG to the processor board for development purpose. Connected to the Ethernet router is also the ground station workstation (not shown in Figure 1) which is not being used at the moment but will enable future development.

The following paragraphs are dedicated to the main hardware parts regarding the current state of the project. The first one is the processor board that hosts the flight software, then there are the payload prototype and finally the SpaceWire router and bridge.

Processor Board: For this setup, the processor is the GR712RC development board build by *Cobham Gaisler AG* [21]. It is a Dual-Core LEON3FT SPARC V8 processor with several input/output ports and other functionalities. As the name states, the processor has a fault tolerant design and it has also a radiation tolerant technology for space application. According to the specification, it can resist up to 300[krad] of Total Ionizing Dose (TID), has Single-Event Latch-Up (SEL) immunity and Single-Event Upset (SEU) tolerance. The Figure 2 shows the architecture of the board.

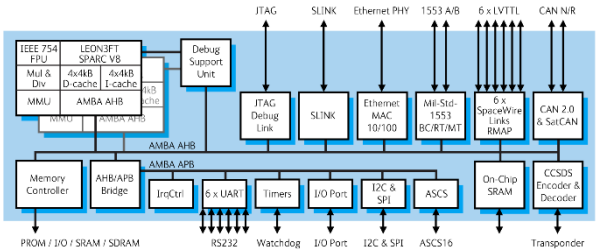

Figure 2: GR712RC Block Diagram [21]

©Cobham Gaisler AG

This two LEON3FT processor cores have 16[KiB] of data and 16[KiB] of instruction cache. They contain two IEEE-754 floating point units and have each a 192[KiB] EDAC protected on-chip memory. They have various interfaces including 6 SpaceWire links with RMAP target. The processor can have a maximum clock frequency of 100[MHz] and the SpaceWire links can go up to 200[Mbps].

The development board also contains an 80[Mbit] SRAM and a 64[Mbit] FLASH memory. It has of course few power and clock circuits as well as all the interface connectors for the SpaceWire, CAN bus, Ethernet, JTAG and UART.

This development board acts as the main OBC of the testbench setup. In the next part, it is referred to as processor board or OBC.

Payload Prototype: In the first iteration of the payload computer, a Raspberry Pi 4 [22] has been chosen. It offers a great flexibility and a good range of Input/Output (I/O) ports. This model has Quad core Cortex-A72

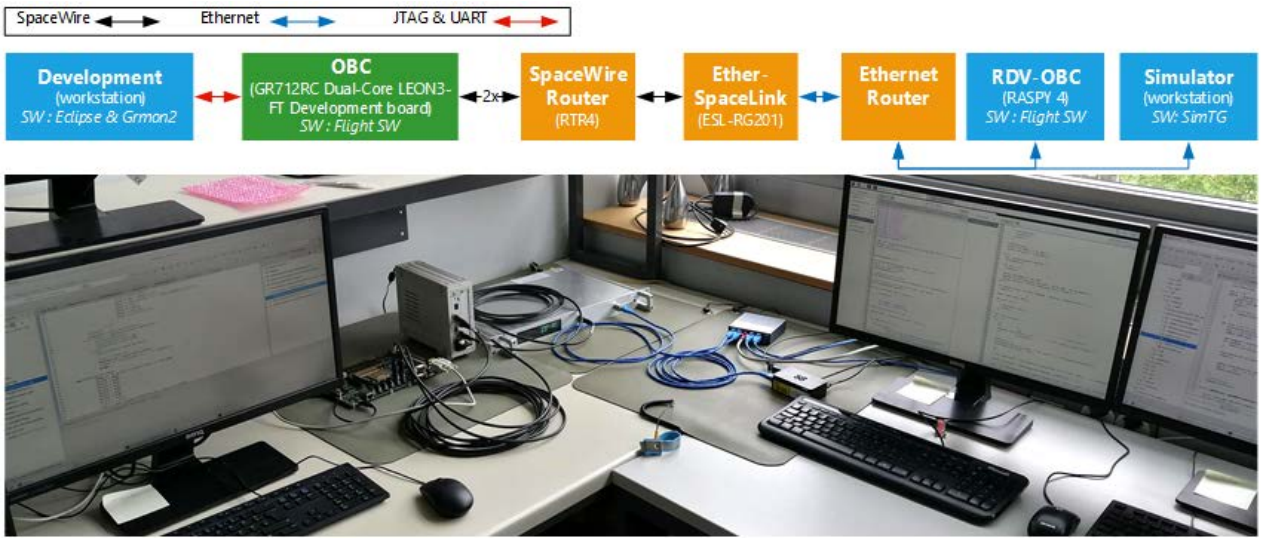

Figure 1: Current setup

(ARM v8) 64-bit System-on-Chip with a clock frequency of 1.5[GHz] and 8[GB] of SDRAM. It also includes an Ethernet port and 40 GPIO pins.

The Ethernet port is capital for this design since it is the way of communication with the processor board and the rest of the setup. The GPIO pins will be useful later in the development when prototypes of modules have to be connected to the payload.

Even though the capabilities of this small computer are quite limited in comparison with the main avionic board, its cost and flexibility allow for a rapid development. The main purpose at this point is to test the communication between this computer and the main processor board. Of course, limitation in the development will quickly rise and it will have to be replaced with a more powerful computer. Nevertheless, the communication protocol and the task management can stay consistent.

SpaceWire Router and Bridge: In order to connect the processor board with payload computer as well as the satellite's simulator, it is needed to use SpaceWire Router and mostly an Ethernet-SpaceWire Bridge. For this setup, the company 4Links [23] is providing both of them.

The router used is the RTR4 module which contains 4 SpaceWire interfaces. Its main goal is to route the signal coming from/to the processor board to the Ethernet-SpaceWire Bridge. In the current design of the setup, this module could be omitted since there is only the processor board that needs to be connected with SpaceWire. However, the testbench will evolve and the possibility to add other hardware elements is critical. One possibility could be to move the payload computer from an Ethernet interfaces to a SpaceWire one.

The second module is the Ethernet-SpaceWire Bridge called EtherSpaceLink-RG201. In this version of the setup, it contains a single SpaceWire port as well as a single Ethernet port. Its goal is to manage the transfer of packet to/from SpaceWire from/to Ethernet network. This allows to connect various modules on the Ethernet side without the need of dedicated SpaceWire interface for each of them. Regarding the SpaceWire side, it uses the Remote Memory Access Protocol (RMAP). For the Ethernet side, TCP/IP protocol is required to talk to the bridge, however it is something handled by most operating system. This connection is particularly useful for the satellite's simulator which is a simple workstation and only have Ethernet ports. The second advantage is for the payload computer that can be a generic small board without any dedicated hardware interfaces for the SpaceWire connector. This module will soon be updated with a combination of the Diagnostic SpaceWire Interface (DSI) and the Single Link Recorder (SLR) called DSR (Diagnostics SpaceWire Recorder).

Flight Software

In the following section, the software used is a direct heritage of FLP2 Flight Software [2], [3] (FSW) provided by Airbus to the EPFL Space Center and the start-up ClearSpace. The setup used this already flight proven software as a base to construct the code for the handling of the dedicated payload computer. It has to be noted that the payload computer in itself does not inherit from the FLP2 Flight Software.

As mentioned, the software consists of two main instances. On one side there are the modules which are directly integrated in the FLP2 FSW and they need to handle the communication coming from/to the payload. In particular, there are the data request, data verification, the modes changes of the payload and the potential High Priority Command (HPC). On the other side, there is the prototype payload that also needs to receive and send packets to the main OBC, handle the various sensors connected to it and allocate the internal or external resources to the algorithms.

Regarding the FLP2 FSW, it is based on the RTEMS (Real-Time Executive for Multiprocessor System) software which allows to use all the functions needed for a Real-Time Operating System (RTOS). The FSW is written in C++ with Object-Oriented Programming (OOP) design in mind. The modules developed for the handling of the dedicated payload follow this rule.

With OOP, there is a high level of encapsulation and heritage between modules. In the code architecture, almost all subsystems have a controller instance that manage the module in general. They also contain an assembly module that regroups and commands all the device handler. These device handlers are controlling directly a single hardware part such as a reaction wheel.

Another important concept of the FLP2 FSW is the datapool (OBSW DP). In order to save variables that need to be accessible from different place, the software stores them in the datapool. It is physically located in the RAM and it contains mainly the state of the spacecraft. It is the location where data produced by the payload computer will be stored.

Regarding the communication protocol, the FLP2 FSW has implemented the Packet Utilization Standard (PUS) developed by ESA. The PUS Services allow a standard way of communication between the ground and the spacecraft. The goal is to extend this implementation to include the need of the dedicated payload, however this is not planned in the first setup.

The final critical point of the software is the implementation of the Failure Detection, Isolation and Recovery (FDIR) in the code. This topic is not covered in this publication and further information about the existing implementation in the FLP can be found in "The FLP Microsatellite Platform - Flight Operations Manual" by *Eickhoff, Jens and Al.* [2].

The Figure 3 shows the general architecture of the FLP1 Flight Software. In this first version, the payload was still a module inside the main avionic core. In this diagram, the datapool is shown as OBSW DP, there are also the FDIR and various handlers.

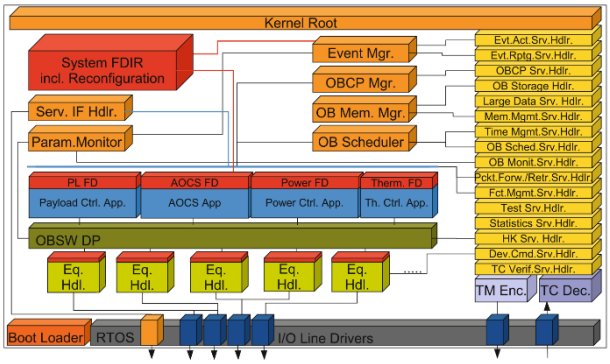

Figure 3: Flight Software of FLP1 Block Diagram [2] ©Jens Eickhoff

Concerning the software for the dedicated payload, the goal is to follow the same type of architecture in order to benefit from the advantages of FLP2 platform. Moreover, by having a similar structure of code, it is easier to transfer module from one architecture to the other and having common module between the two. As in the first phase of the development, the payload has only few programs developed in C++. It also uses standard libraries especially for the TCP/IP socket management.

OBC-PAYLOAD INTERACTION

This section is dedicated to the interaction between the dedicated payload computer and the processor board. In the following subsection multiple aspects of the interaction are presented with the constraints associated to them. Similar concepts were already developed at the Airbus DS Friedrichshafen [24] and some have been adapted for the need of the mission.

The principle behind the communication is a Master-Slave architecture with the main avionic core (or On-Board Computer) being the Master. Since the OBC handles all the primary function of the satellite and most importantly the FDIR, it is crucial that it remains constantly fully operational. Moreover, it has the task to control the telecommunication with the ground as well as handling the High Priority Command (HPC) in case of emergency or failure. In addition, this architecture is a good option to keep the integrity of the FSW developed by Airbus DS Friedrichshafen. Nonetheless, the dedicated payload computer has still the critical task of providing accurate relative navigation data when in close proximity operation with the target. All the algorithms and sensors dedicated to the rendezvous phase are handled by the payload and the information computed in it should be transmitted to the main avionic core. Since the OBC handles the AOCS and propulsion part, it needs the information of the payload to compute the maneuvers and to react according to the state and attitude of the targeted debris.

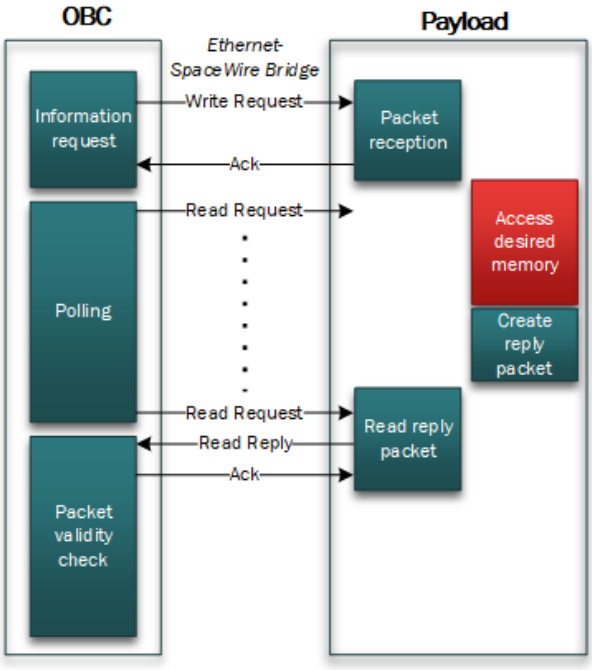

Figure 4: Communication Protocol

By defining the payload computer as Slave in the architecture, all the interactions are handled by the OBC. By consequence, it has to create a request RMAP packet every time information is needed. The protocol follows this sequence. If the OBC needs information from the payload, it creates a write request packet and it is sent to the payload. The payload receives the packet and transmits immediately a small acknowledge (ACK) packet. The payload accesses the information required in its own memory/datapool and creates a reply packet. In the meantime, the OBC starts sending read request packets to the payload in periodic manner after receiving the previous acknowledge packet (It is called the polling sequence). These packets are received by the payload and the read reply packet is sent to the OBC as soon as

available. The OBC receives the read reply packet and analyses the correctness of the information. If the information is valid, it sends an acknowledge packet otherwise a new write request is created. The Figure 4 shows the protocol in a simplified way.

The following subsection addresses the various requirements and details of the interaction.

Packet transformation SpaceWire to Ethernet

This part is specific to the usage of payload connected with Ethernet to the processor board. As shown in Figure 1, the dedicated payload computer is connected with first Ethernet and then SpaceWire to the main avionic core through an Ethernet-SpaceWire Bridge. In this case, the role of the bridge is to correctly redirect packet coming from one side or the other, however the encapsulation of these packets are done inside both modules.

From the SpaceWire side, all the packets are transmitted using RMAP (Remote Memory Access Protocol). The decision is directly inherited from the FLP2 FSW architecture that already uses this protocol internally. In order to keep the consistency with the rest of the avionic, it was decided to use it for the communication. Nonetheless, few additions needed to be done to communicate with the payload computer. Since it is connected with Ethernet to the Ethernet-SpaceWire Bridge, it has been decided to use TCP/IP protocol for this part. It implies the OBC also needs to handle and controls the TCP/IP part. To achieve this, the data has to be encapsulated into first a TCP/IP packet and then the RMAP. A schematic of this encapsulation can be seen in Figure 5. To be noted that this concept was already explored at Airbus DS [24]. As shown in the figure, the TCP/IP header are added in front of the Data packet and then encapsulated inside an RMAP packet. In order for this to work, the OBC needs to have prior knowledge of the IP address of the payload.

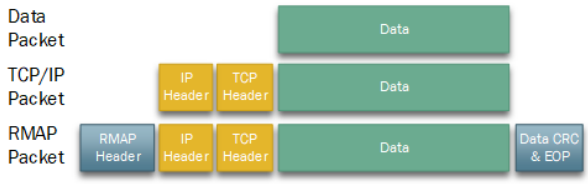

Figure 5: Packet encapsulation TCP/IP & RMAP

When the packet mentioned before is received by the Ethernet-SpaceWire Bridge, the RMAP Header and end of frame are removed and then it is routed accordingly to the TCP/IP information. Next, the TCP/IP packet is received by the payload and decoded. For the return path, the payload computer does not need to handle RMAP packets. Indeed, the transmitted information needs to be written inside a buffer in the Ethernet-SpaceWire Bridge and then requested by the processor board. It means the dedicated payload computer simply needs a TCP/IP connection to the bridge in order to received and send information. Overall, it simplifies the communication protocol between the two entities.

One of the disadvantages of the TCP/IP protocol is the timing issue. It generally guarantees the correct delivery of the packet, but the time of transmission can vary. Nonetheless, this protocol could allow multiple payloads to be connected on the same network and work in parallel. A possible solution to this problem is the restriction to only SpaceWire connection between the payload computer and the OBC. However, it would increase the complexity and the cost of the payload.

Reaction time of payload computer

A problem related to the time sharing is the availability of the computed data in the dedicated payload computer. It has to be remembered that the various instruments required for the rendezvous phase have different frequency rate as well as computation time. It implies that pre-process data from the sensors are computed at a different time and then the pose estimation algorithm needs some additional time to compute its result. Even though this timing is mostly deterministic, they need to be considered.

The classical example is the processor board requesting position information to the payload. In this case, the processor could need the relative distance to the target as well as the rotation rate of the target with both information coming from different sensors. The request is emitted at a time t_0 and received by the payload at $t_1 = t_0 + \delta t_{transmission}$. At this point, the payload will access its memory and retrieve the needed information. The relative distance has been computed at an earlier time t_2 with the use of the radar sensor information, however the information is valid at the time of measurement t_3.

$$t_3 = t_2 - \delta t_{distance\ algorithm\ computation\ time} - \delta t_{radar\ sensor\ preprocess}$$

Similarly, the rotation rate estimation has been done at time t_4 with measurements from the camera at t_5.

$$t_5 = t_4 - \delta t_{rotation\ rate\ algorithm\ computation\ time} - \delta t_{camera\ sensor\ preprocess}$$

These two measurements are encapsulated in a packet at t_6 and sent to the processor board with this timestamp. Finally, the processor board receives the information requested at $t_7 = t_6 + \delta t_{transmission}$. Nevertheless, the information in the packet have been computed at respectively t_2 and t_4 with time $t_7 - t_2 \neq t_7 - t_4$. At this point, it is obvious that the processor board needs

information relative to t_2, t_3 and t_5, t_4 in order to correctly compute its desired maneuver. For this reason, it is capital that data packets include not only the time at which they have been created but the time of measurement as well as the time of computation end. The Figure 6 is a summary of the above explanation. It shows the timeline with the various timings as presented in the example.

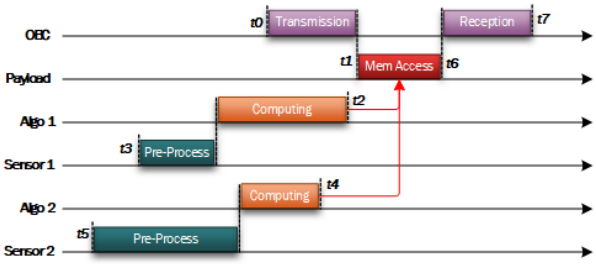

Figure 6: Timing of measurements and transmission

Time synchronization method OBC-Payload

One of the most constraining challenge of the OBC-payload communication is the timing or synchronization. Indeed, it is crucial for both parts to work autonomously, however the exchange of information required precise timing. The typical example for this is the handling of a critical phase such as the last meter of the rendezvous with the targeted debris. In the last phase, the spacecraft needs a high level of control regarding its attitude and will correct any disturbance with its AOCS. This subsystem is of course handled by the main avionic platform. Nonetheless, the pose estimation and relative attitude between the chaser and the target are provided by the dedicated payload. Indeed, all the rendezvous sensors are directly controlled by this computer and the data retrieved is used by algorithms that are also held in some parts of the payload computer. Overall, the main avionic platform needs information stored in the dedicated payload in order to compute its correct attitude in space relative to the target. Since the distances during the last phase are small, any disturbances can be fatal and thus the attitude computation should be as precise as possible. It means that the main avionic platform should know exactly when the last pose estimation of the target has been done in order to deduce its own location in space and execute the proper maneuver. From this, it is obvious that timing is decisive. The previous subsection has shown the importance of correct timestamps for the measurement, this subsection is about the synchronization of the payload and OBC.

In the configuration shown in Figure 1, there are no direct circuit linking the payload to the main avionic core. Everything is transmitted through SpaceWire and Ethernet. By consequence, there are no possibilities to have a dedicated clock circuit that can share time information through the whole spacecraft.

In this case, the most straight forward method is simply to add timing information inside the packets that are being transmitted. In this approach, it would imply that each packet that transits between the processor board and the payload have a timestamp incorporated in it. This timing information is used mainly by the processor board to correctly update its current attitude in space. As explained before, the AOCS needs this information in order to compute the exact position relative to the target at any moment in time. Once updated, the spacecraft can propagate its attitude with a Kalman filter or similar algorithm to deduce its current position. The Figure 7 shows the construction of a data packet with multiple information in it and is based on the example of Figure 6. In this example, there are a general timestamp for the packet as well as timestamps for every measurements and computed values.

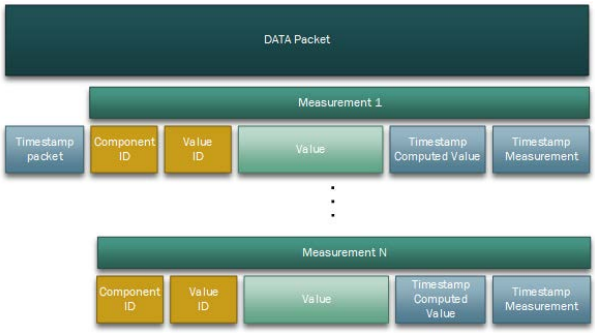

Figure 7: Structure of Data Packet with multiple measurements

For this configuration, it implies both entities (processor board and payload) need to have access to their respective clock information and this point is not about synchronization but only access. Another problem that needs to be tackled is the precision of the clock. As mentioned in the previous part, it is capital to have a correct timing for all the measurement to prevent errors in the pose estimation. Small mistakes in the attitude computation could lead to erroneous maneuver and jeopardize the mission. To solve this problem, the clock information needs to be shared at microseconds precision and thus increasing the size of the packet transmitted.

The last point of this section is the synchronization of clocks. In order to minimize the error across the system it is crucial that the whole spacecraft shares a common clock. However, it is obviously not possible to share a single clock signal through the entire system without creating important slew rates. For this reason, the implementation of at least two clocks is needed. There is

the central one for the core avionic module and another one for the dedicated payload computer. These two clocks should have the same rate and indicated the same time at any point of the mission. To guarantee their accuracy respectively to each other, a periodic packet is sent from the processor board to the payload computer with the exact current time in order to correct the drift. For the correctness of this update, it is important to have a determined transmission time for the packet. Since all the packets are sent over the same bus, the processor board must prevent the transmission of any other information at the sending time to prevent delay of this particular packet. Nonetheless, it is known that correct timing for TCP/IP packet can be difficult and thus more investigations need to be done on this topic.

Handling & Verification of data in OBC

As discussed earlier, the OBC requests information to the payload in a two steps process. First a write request is created in order to ask specific information to the computer payload and then a read request is sent to retrieve the requested data.

The next step is then to extract the data from the RMAP packet and analyze it. For the second packet (read reply), the goal is to check the integrity of the data and the correctness of the packet. First, the header and the end of packet are analyzed to ensure the validity of the packet. If the encapsulation is correct, the OBC controls the structure of the data sent and the timestamp of the packet. If the structure matches the requested information of the write request packet, then the information correctness can be checked. If one of these four tests returns an error, the OBC will start again its communication protocol with the payload.

Regarding the data correctness, the two mains objectives are to verify that each measurement is in a correct range and that the timing is plausible. The first check is controlling the ID of the components and values match hardware part of the payload. Then it is the verification related to the timing of the measurements and computed values. The timing of all information is checked against a defined table and if the difference is too large, an error is created. In this case, the communication protocol is reset. The values are then compared with a range table stored in the memory and corresponding to the actual phase of the mission. If the measurement received differs too much from the previous one or if it is not plausible to obtain such value, then the communication protocol is started again. At this point, it is worth mentioning that the two previous cases can occur because of an issue with the payload itself or one of the instruments. Nevertheless, this case is not discussed in this subsection and more information on the verification can be found later in this publication.

Once all the checks have been done, the obtained data can finally be stored in the datapool of the FLP2 FSW. A summary of the operation is shown in Figure 8. It shows the process flow of the decision and the different errors that can be generated.

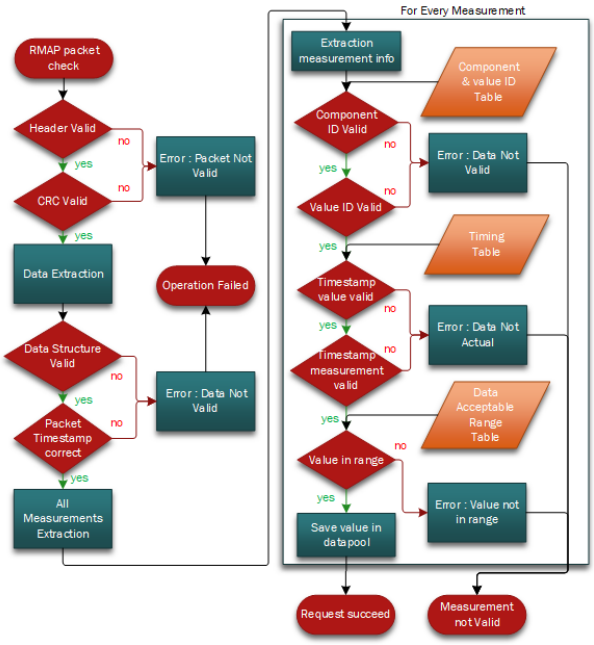

Figure 8: Validation of read reply packet in the OBC

Task simulation on payload

At this early phase of the setup development, the list of sensors that is needed by the spacecraft during the rendezvous phase is still not entirely defined. The requirement regarding the measurement and the type of information has been established but the selection of these instruments is in process. In the same category, the various algorithms needed for the pose estimation and the GNC are being developed in parallel by the start-up and some partners. Moreover, the implementation of the algorithm in the dedicated payload is being discussed and various options are considered in order to optimize power consumption, accuracy, timing constraint and development cost.

In an earlier study [25], analyses of the various architecture configurations for the payload has been done. This publication has shown advantages and disadvantages of different architecture and the effect of multiple parameters on the system. Based on the result, it is possible to create a simulation of the task in this dedicated payload.

For the first iteration of the setup, the important point is to simulate the sensors output and the computation time

of the algorithm. In order to simplify the process, a list of output is generated prior to the simulation with potential results from the various algorithms with a defined timestamp associated to them. In order to have consistency, the results are created in accordance to the scenario of the publication mentioned before [25]. The created list is then called by the payload computer and access as if the sensors and algorithm were creating the data. The information is then stored in the payload datapool and ready to be requested by the processor board. A block diagram of the prototype payload architecture is shown in Figure 9.

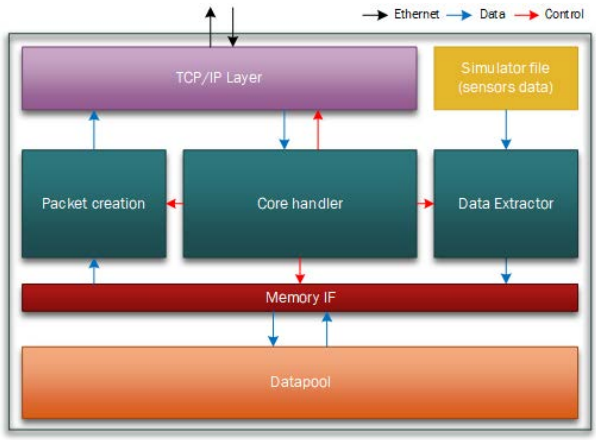

Figure 8: Block diagram of the payload computer architecture

This simplified version of the payload computer allows the development of interaction with the processor without being dependent on the implementation of the algorithm and the drivers for the sensors. By decoupling the algorithm part and the communication part, the team is able to develop a reliable communication protocol without suffering from changes in the algorithm development or hardware modification.

The downside of this method is the reduction of processing time and resources utilization by the payload computer. Since no real algorithms are being run by the payload computer, all the processing resources are allocated to the communication and datapool handling part. It implies that timing constraint and latency are not matching the one of the final implementations. Nevertheless, the hardware used for the payload computer is not representative of the final one either. It means an implementation of more real algorithm in the actual version of the payload computer will not correspond to the constraint on the final payload computer. This trade-off has been decided to speed up the development of the communication part of the payload and verify the concept of operation.

CHALLENGES

This section addresses the various challenges laying in the previous explained processor board and payload interaction. The goal is to present the current issues with the implementation and possible solutions to these problems.

High Priority Command from OBC to payload

As explained in the previous subsections, the timing between the OBC and the payload is decisive. In order to have the correct attitude of the spacecraft, the main avionic core must have timestamps associated to measurements as close as possible to the reality. Despite this required accuracy, there are few cases where the platform needs information as quickly as possible. One example could be that the previous computed attitude state of the spacecraft shows that a collision with the target is imminent. To escape this situation, the spacecraft reacts with a maneuver to leave its collision trajectory. Nonetheless, it still needs to verify the correctness of the new trajectory and ensure that the collision will not happen. For this verification, the processor board needs to request again information from the sensors of the payload as quickly as possible. In this case, a regular data request from the OBC could be too slow. For this reason, a High Priority Command (HPC) protocol has been envisioned. The goal of this extra protocol is to access in a shorter period of time and with priority to the required information. The following paragraphs describe possibilities of implementation.

One possibility is the use of the regular SpaceWire and Ethernet connection, however it imposes already a lot of restriction on the timing. The work could be done at the reception and processing of the request by the payload computer. If the request can be handled faster in this computer, a timing amelioration could be achieved. It would imply the modification of the task management by the payload and probably the stop of most processes inside it. For example, it could allocate all its resources to the received request while blocking the thread of the other parallel tasks. This could indeed improve the timing reaction of the payload computer, but a majority of the delay are still inside the network and with the encapsulation process.

A second possibility would be to add a direct data line connecting the OBC with the payload computer. This solution would indeed greatly reduce the latency by removing most of the communication layer needed when going through SpaceWire and Ethernet. Nonetheless, it will create additional hardware interfaces and the complexity of the information transfer will rise. Moreover, the verification and validation of this architecture will drastically increase in difficulty. In

addition, it will create issue in the FLP2 FSW integrity and impose additional constraint on the payload side. For all these reasons, the second solution is right now not planned to be implemented.

Another aspect of the HPC is the type of information request that could be done. Since the timing is the main driving factor, the information requested should be as small as possible. Nevertheless, it should contain the critical data needed for the spacecraft. At this point, a more intensive analysis should be done with the AOCS teams to determine the type of information that could be needed by the OBC with the HPC.

Finally, the verification of the data integrity should not be forgotten. Even though the process needs to be the fastest, it is critical as well that the information delivered is correct. By looking back at the example discussed before, the HPC is used when mission critical events are happening. In the case of the trajectory correction, it is at the same level of importance that the information provided to the AOCS is the most recent and perfectly accurate. If the HPC cannot guarantee the second point, it could create potentially fatal scenario for the mission. One could be the wrong estimation of closure rate to the target and thus the collision with it.

OBC polling and redundant information transmission

In itself, this subsection is only partially an issue. Regarding the state of the spacecraft and information available to the avionic core, it does not create a problem that the OBC requests updates at a high frequency. It will happen that the data requested has already its most recent value stored in the datapool. By having a regular polling sequence for most of the information generated by the dedicated payload computer, the FSW of the OBC guarantees to have regular updates on the data computed. This process is particularly useful to check the consistency of the instrument and algorithms on the payload and monitor any changes in the trend of the data. An example is the beginning of the relative navigation of the spacecraft. When the chaser switches to an absolute positioning in space with the GPS signal and Star Tracker to a relative positioning with the target, the payload computer will start computing values. By constantly polling the payload computer, the OBC can monitor the error due to the initialization of the pose estimation algorithm and the difficulty to measure precisely the target state at a long distance. The monitoring is important to prevent any modification of the chaser trajectory based on none accurate measurements of the rendezvous sensors. Furthermore, the OBC can monitor if the algorithms are converging to a steady solution or if they need to be reset. It can also request a restart of a sensor if the information seems to be erroneous.

The second aspect is the resources allocated to the polling during various phases. As shown in a previous analysis [25], the payload computer is generally busier when the chaser is close to the target. In addition, the main avionic core can have a similar trend in processing resources utilization. In the last phase of the rendezvous, the AOCS is constantly used and the communication with the ground is at its peak to maintain some human control on the satellite. For these reasons, it is important to keep the polling sequence at a reasonable rate to prevent overloading the main avionic core. Nevertheless, it should still guarantee a regular update of all parameters to prevent any collisions. It is again at this point that a HPC could be useful.

In conclusion, the request of redundant information by the OBC is not a capital point in the development of the mission. The validity of the data as well as the precise timing should remain the driving factors.

Integrity check of data packets

One of the challenges of the OBC regarding the interaction with the dedicated payload computer is checking the validity of the information received. As discussed in a previous section, the data should be verified accordingly to its associated timing and the plausibility of its value.

For the first point, the time of measurement could simply be compared with the actual time of the spacecraft. If the information is considered to be too "old", then the OBC starts again the data request protocol. This simple verification can prevent the system to be updated with data from sensors being idle or turned off. Indeed, the payload computer simply provides the last available measurement or computed value in its memory and it is up to the OBC to check it. If the OBC tries to retrieve an information that the payload is no longer updating, it could cause an issue. Since the default mode is to restart the data request protocol if the information is considered not valid, the OBC could be blocked in a loop of data request. One solution is to request the status of all payload components if too many data request fail. By checking the availability of the sensors/algorithms where the data is produced, the OBC can react accordingly. Of course, it could be that an error has occurred in the payload and few components need to be restarted. This is addressed in a following subsection.

The second point to check is if the data received is plausible. It implies more logic due to its complexity since the validity of most measurements depend on the range to the target. Moreover, few sensors might have a larger uncertainty when reaching the limit of their application. It is important that each mission phase uses the correct set of sensors in their respective range. In

addition, the avionic core should have a previous knowledge of what value to expect from which sensor depending on the mission phase. A table with the range of expected value will have to be computed prior to the mission on a simulator and then stored in the OBC memory. Nonetheless, the possibility to update the table after the launch should be kept. Assuming the avionic core has such a list, the check could be done by requesting the modes of the payload and its algorithms/sensors and matching them with the table. As in the previous case, the OBC will start again the data request protocol if the data does not fit in the expected range. The same behavior as for the timing validity process should be applied in case of multiple failures.

Critical situation detected by payload computer

One of the disadvantages of the Master-Slave architecture of the spacecraft is the inability of the payload computer to propagate an event to the main OBC. A typical example could happen in the final approach phase. At this point, the OBC is polling the payload computer at a certain rate but an event can occur in-between two polling. Another possibility is that the OBC is requesting a different information at the time and will not need the crucial data before another defined time. For this example, it is assumed that a piece of the debris detached itself from the target and is going straight to the chaser. The payload computer is constantly computing the range to the target at a high frequency and will probably detect this extra debris rapidly. Nevertheless, the polling frequency of the OBC has a lower rate and the information can be missed by the main avionic core. In this very specific case, it would be decisive for the payload computer to inform the processor board. If not, the satellite might not be able to perform an escape maneuver in time.

A partial solution to this problem could be to add a flag in each packet being transmitted to the main avionic core indicating if a crucial event is happening. However, it would work only if the OBC is indeed requesting data from the payload computer. In the case of the action happening between two requests, the problem would remain.

The second solution to this problem would be to increase the polling frequency of the OBC. As discussed in a previous subsection, this cannot be done past a certain limit at which the processor board would be overloaded. It is up to the design team to find a compromise between resources allocation and polling frequency.

It is arguable that this scenario has a low probability of happening and more analysis should be done to evaluate it. Nevertheless, the flag in the data packet is viable option that could solve partially the problem.

Failure Detection, Isolation and Recovery

This topic has already been addressed briefly earlier. It is capital for the main avionic core to control and monitor the payload computer. It is as well crucial for the payload computer to do the same with all the sensors and hardware accelerators if any. It means the full avionic system needs to have two separate FDIR entities with their respective action and procedure and they should be able to communicate. Moreover, it must be remembered that the power control is done by Power Control and Distribution Unit (PCDU) of the FLP2 platform and that the payload does not communicate with this module directly.

As envisioned, the FDIR of the payload computer will probably have limited action. For example, the integrity check of the data is handled by the OBC and the powering of the module should be requested as well by the OBC. Nonetheless, the FDIR can still have a range of action. Regarding the detection and isolation, the payload computer can already monitor the status of the various instruments by requesting their housekeeping data. For the recovery part, software reset is initiated by the payload computer on its own or from an OBC request. The "hard" reset should be initiated by the PCDU on request from the OBC by turning off the power of the targeted component.

FUTURE DEVELOPMENT

The first step is to verify the interaction protocol established in the setup. Various functions have been created in both the FLP2 FSW and the payload SW and interaction between them should be controlled. By accessing memory output of both the module during all scenarios testing, it is possible to ensure the correct transmission of information.

The second step is to verify the various timing constraints of the data exchange protocol. As explained in the previous part, it is crucial to monitor the time needed by a packet to be transmitted back and forth. Moreover, early assessments can be done regarding the memory access of the payload as well as data integrity checks in the OBC.

The third step is the implementation of the concept presented in the challenges section. These concepts first need to be refined and discussed with the corresponding team members. Once clearly established, the procedure developed can be incorporated into the FSW. Following their implementation, early verification should be done.

The next steps are not enumerated since they can happen any order. Among them, there is the amelioration of the FDIR concept for both the payload and the OBC. The one presented previously is an early solution and needs

to be polished. Moreover, a deeper analysis of the system needs to be performed in order to reveal the various cases of failure and the recovery methods associated to them.

One important part of the setup will be the implementation of the ground control loop. This interaction already exists for the standard FLP2 testbench setup and it needs to be modified to take in consideration the payload. The interfaces of the ground control will be modified and the communication protocol with the spacecraft updated. Once these actions are done, an operator will be capable of commanding the payload computer through the OBC.

In parallel of these implementations, the upgrade of the payload computer should be considered. As explained in the introduction, the raspberry pi is only a preliminary prototype of the payload to ease the first development on the setup. By upgrading to a more realistic prototype, the possibility to verify timing constraint will be possible. Moreover, the software could be adapted to the new hardware and the allocation of resources will be more realistic.

In the same line, a new payload computer will imply the incorporation of the various algorithms needed for the rendezvous phase. In a first phase, they will be fed by pre-computed data but the possibility of adding true sensors is not so far behind. Adding realistic hardware parts will permit the verification of the overall loop as well as the transmission protocol between payload and OBC.

CONCLUSION

The communication management between the OBC and the dedicated payload computer is a decisive point for the future avionic system of the mission. It is a bottleneck that can lead to many issues if not implemented in the correct manner. As shown multiple times, the main OBC is highly dependent on the results of the payload computer for mission critical part of the rendezvous. The information computed by the payload computer should be retrieved easily and in a periodic manner by the main avionic core in order for the AOCS and other subsystems to work properly.

In this publication, the concept of interaction between the OBC and the payload has been presented with emphasis on the timing of the measurements and the clock synchronization. A validation procedure has been established and a data packets format created. Moreover, the implementation of High Priority Command (HPC) and the Failure, Detection, Isolation and Recovery (FDIR) has been discussed.

The next step is the verification of the various concepts in the Hardware-In-the-Loop setup with timing as the main driving factor. Future hardware upgrades are also planned to increase the representativeness of the testbench relatively to the Active Debris Removal mission, ClearSpace-1.

References

1. M. Richard, L. Kronig, F. Belloni, ..., and H. Shea, “Uncooperative rendezvous and docking for MicroSats,” In 6th International Conference on Recent Advances in Space Technologies, RAST, 2013.
2. Eickhoff, Jens (Ed.), “The FLP Microsatellite Platform - Flight Operations Manual,” Springer, 2016.
3. Eickhoff, Jens (Ed.), “A Combined Data and Power Management Infrastructure,” Springer, 2013
4. J. L. Forshaw, G. S. Aglietti, N. Navarathinam, H. Kadhem, T. Salmon, A. Pisseloup, ... and S. Barraclough, “RemoveDEBRIS: An in-orbit active debris removal demonstration mission,” Acta Astronautica, vol. 127, pp. 448-463, 2016.
5. R. Biesbroek, L. Innocenti, A. Wolahan, and S.M. Serrano, “e. Deorbit-ESA’s Active Debris Removal Mission,” Proceedings of the 7th European Conference on Space Debris, ESA Space Debris Office, p.10, 2017.
6. T. Debus, and S. Dougherty, “Overview and performance of the frontend robotics enabling near-term demonstration (FREND) robotic arm,” AIAA Infotech@ Aerospace Conference and AIAA Unmanned... Unlimited Conference, p. 1870, 2009.
7. C. Blackerby, A. Okamoto, K. Fujimoto, N. Okada, J. L. Forshaw, and J. Auburn, “ELSA-d: An In-Orbit End-of-Life Demonstration Mission,” In Proceedings of the 69th International Astronautical Congress, Bremen, Germany: International Astronautical Federation, 2018.
8. G. Personne, “ATV GNC synthesis: overall design, operations and main performances,” Guidance, Navigation and Control Systems, Vol. 606, January 2006.
9. J. Fabrega, M. Frezet, and J. L. Gonnaud, “ATV GNC during rendezvous,” Spacecraft Guidance, Navigation and Control Systems, Vol. 381, p. 85, February 1997.
10. S. Persson, S. D’Amico, and J. Harr, “Flight Results from PRISMA Formation Flying and Rendezvous Demonstration Mission,” Proc. Of

61st International Astronautical Congress, Prague, CZ, Vol. 2, 2010.

11. G. Gaias, J. S. Ardaens, and C. Schultz, “The AVANTI experiment: flight results,” Proceedings of the 10th International ESA Conference on Guidance, Navigation & Control Systems, European Space Agency, ESTEC Noordwijk, The Netherlands, 2017.
12. J. Gonnaud, J. Sommer, and M. Tsang, “APRK GNC Design and Performances Evaluation for ATV Rendezvous,” Proceedings of the 3rd International Conference on Spacecraft Guidance, Navigation and Control Systems, pp. 26-29, November 1996.
13. L. Blarre, N. Perrimon, C. Moussu, P. Da Cunha, and S. Strandmoe, “ATV videometer qualification,” Proc. of the 55th Int. Astronautical Congress, Vancouver, Canada, October 2004.
14. O. Durou, V. Godet, L. Mangane, D. P´erarnaud, and R. Roques, “Hierarchical fault detection, isolation and recovery applied to COF and ATV avionics,” Acta Astronautica, Vol. 50(9), pp. 547-556, 2002.
15. P. Bodin, R. Larsson, F. Nilsson, C. Chasset, R. Noteborn, and M. Nylund, “PRISMA: an in-orbit test bed for guidance, navigation, and control experiments,” Journal of Spacecraft and Rockets, Vol. 46(3), pp. 615-623, 2009.
16. P. Bodin, R. Noteborn, R. Larsson, and C. Chasset, “System test results from the GNC experiments on the PRISMA in-orbit test bed,” Acta Astronautica, Vol. 68(7-8), pp. 862-872, 2011.
17. S. D’Amico, P. Bodin, M. Delpech, and R. Noteborn, “PRISMA,” Distributed Space Missions for Earth System Monitoring, pp. 599-637, Springer, New York, NY, 2013.
18. K. Brieß, W. Bärwald, E. Gill, H. Kayal, O. Montenbruck, S. Montenegro, ..., and H. Venus, “Technology demonstration by the BIRDmission,” Acta Astronautica, Vol. 56(1-2), pp. 57-63, 2005.
19. “ESA eoPortal BIROS,” https://directory.eoportal.org/web/eoportal/satellite-missions/b/biros , Last access 14.05.20202.
20. G. Lentaris, I. Stratakos, I. Stamoulias, K. Maragos, D. Soudris, ..., and D. Gonzalez-Arjona, “Project HIPNOS: Case study of high performance avionics for active debris removal in space,” VLSI (ISVLSI), 2017 IEEE Computer Society Annual Symposium, pp. 350-355, July 2017.
21. Cobham Gaisler AB, “GR712 Development Board User Manual,” Rev. 0.13, 2018-03-08
22. Raspberry Pi Foundation, “Raspberry Pi 4 Specification,” https://www.raspberrypi.org/products/raspberry-pi-4-model-b/specifications/, Last access 27.05.2020
23. 4Links, Products, https://www.4links.co.uk/index.php/portfolio, Last access 27.05.2020
24. S. Hegde, “Science Data Preprocessor and IP-based Access in the FLP2 Satellite Testbench,” 2017, Master Thesis, Technical University of Kaiserslautern
25. M. Juillard, M. Richard-Noca and J. Kneib, “Simulation Tool to Study High Performance Avionic for Active Debris Removal Missions,” 2019 IEEE/AIAA 38th Digital Avionics Systems Conference (DASC), San Diego, CA, USA, 2019, pp. 1-10