EUROPEAN WORKSHOP ON ON-BOARD DATA PROCESING (OBDP2021), 14-17 JUNE 2021

PLATO DPS: STATE OF THE ART ON-BOARD DATA PROCESSING FOR EUROPE'S NEXT PLANET-HUNTER

Claas Ziemke [(1)], Ulrike Witteck[(1)], Gisbert Peter[(1)], Philippe Plasson[(2)], Emanuele Galli[(3)], Bernd Ulmer[(4)], Roland Ottensamer [(5)], Harald Ottacher[(6)], James Windsor[(7)]

(1) German Aerospace Center (DLR) - Institute of Optical Sensor Systems, Berlin, Germany
(2) Laboratoire d'Etudes Spatiales et d'Instrumentation en Astrophysique (LESIA), Meudon, France
(3) Istituto di Astrofisica e Planetologia Spaziali (INAF IAPS), Rome, Italy
(4) Ingenieurbüro Ulmer, Frankfurt (Oder), Germany
(5) Universität Wien – Institut für Astrophysik, Vienna, Austria
(6) Österreichische Akademie der Wissenschaften - Institut für Weltraumforschung, Graz-Messendorf, Austria
(7) European Space Agency - ESTEC, Noordwijk, The Netherlands

ABSTRACT

State-of-the-art optical space instruments are capable of producing data rates that considerably exceed the available down-link capacities. The potential data rates are increasing as the sensor technologies progress. These trends require a significant data reduction on-board, most appropriately at the source of the data, the instrument. Furthermore, the high availability needed to fulfil the science goals require a high degree of robustness. In this paper we will discuss these trends using the example of the architecture of the data processing system of the PLATO payload. PLATO (PLAnetary Transits and Oscillations of stars) is the third medium (M3) mission in ESA's Cosmic Vision programme. The goal of the PLATO mission is to detect terrestrial exoplanets in the habitable zone of so-lar-type stars and characterize their bulk properties. The PLATO instrument is comprised of 24 identical refracting telescopes each equipped with 4 CCDs, plus 2 additional telescopes which aid the space-craft fine-pointing. Altogether the PLATO payload is comprised of over 27 CPU cores, over 30 FPGAs and 6 Space-Wire Routers. In order to master this complex system of interacting software and hardware we use standardized protocols, pre-qualified operating systems, Model-Based Systems-Engineering tools and techniques, and code-generation. We will give an overview of the aforementioned tools and techniques, discuss their benefits and pitfalls and share the lessons learnt so far in the development of the PLATO instrument.

1. DESIGN DRIVERS FOR ON-BOARD DATA PROCESSING ARCHITECTURES

Comparing past present and future exo-planet survey missions, it can be seen that the key characteristics are all following a clear trend towards better performance and higher data-rates. The most obvious trend in the described mission is the increase in the number of CCDs and consequently the number of pixels that could potentially be processed and/or downlinked. While CoRoT [1][2] and TESS [4] are in low-earth orbits that allow high bandwidth downlinks, Kepler [3] and PLATO are stationed at the L2 approximately 448900 km from earth. This limits the amount of available bandwidth. These factors lead to a high demand of on-board data-reduction. There are three possibilities in data reduction. First and most obvious is the fact that only interesting parts of the CCDs (targets) are downlinked while the rest of the CCD is discarded. The second is compression. All the mentioned missions are compressing the pixel data before downlink and on-board storage. This compression is lossless (or quasi-lossless in the case of Kepler) as lossy compression would lead to problems in the further processing of the data on-ground. The third and most radical solution is to not downlink the pixel data of targets at all but instead do the first steps of the scientific data analysis already on-board. In the case of exo-planet search this is the calculation of the total brightness (flux) and the center of brightness (centroid) of stars. As the demand for the number of observed targets is increasing and the compression of noise (after a first step of taking the difference between two consecutive exposures, the resulting pixel stream is basically noise) is mathematically limited, the only possibility to further increase the number of observable targets is the implementation of on-board data processing. While a typical PLATO target is a 6x6 pixel image (72 bytes) the flux of a star is (simplified) essentially a single value of 4 bytes (some details of this can be found in chapter 3).

Table 1 Key characteristics of past and future missions

Mission	Launch	CCDs	MPixel	Targets	Down-link
CoRoT [1][2]	2006	4	16.78	12k	18 kbps
Kepler [3]	2008	42+4	94.62	170k	N/A
TESS [4]	2018	16	67.11	>10k	N/A
PLATO	~2026	104+8	1952.6	3600k	5280 kbps

2. THE PLATO DATA PROCESSING SYSTEM

In the following section we will describe how the PLATO data processing system (DPS) is designed to deal with the challenges described in the previous section. We will describe the system architecture, the units comprising the DPS and give an overview of the major data processing and reduction steps from the CCD read-out to the mass-memory storage.

1.1 System architecture

The PLATO payload is comprised of telescopes, front-end electronics (FEEs), data processing units (DPUs), ancillary electronics units (AEUs) and the instrument control unit (ICU). There are 24 "normal" cameras and FEEs, nominally operated at a 25 second cadence, and 2 "fast" cameras and FEEs operated at a 2.5 second cadence. 12 normal DPUs are grouped into two main electronics units (MEUs) together with two SpaceWire routers each, while the two fast DPUs are bundled in the fast electronics unit (FEU). The Fast DPUs produce a fine-guidance quaternion for the space-craft attitude control system and also produce additional science data products. Furthermore, the instrument has its own power conditioning AEUs for very precise power conditioning of the FEEs, highly accurate clock signal distribution to the FEEs and synchronized CCD read-out triggering. All the interfaces between the units and between the ICU and the space-craft are high speed SpaceWire serial links. Except the synchronization lines there are no discrete, analogue or low speed lines in the instrument. The architecture of the PLATO payload is shown in Fig. 1 below.

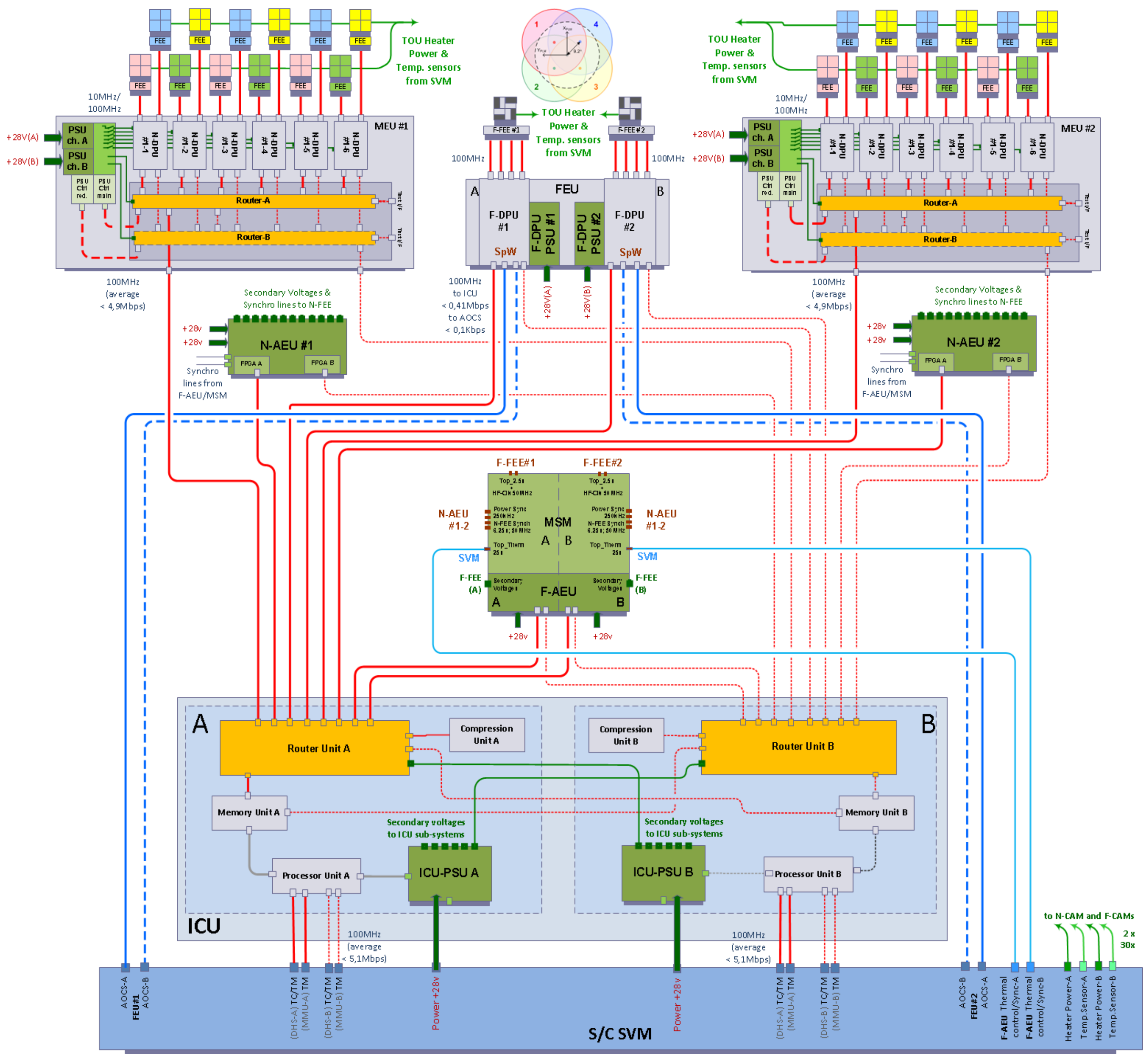

Figure 1 PLATO payload architecture overview [5]

Normal front-end electronics

Each normal camera is equipped with full-frame 4 CCDs and is served by the normal-front-end electronics (N-FEE). The N-FEE is connected to the DPU through a single SpaceWire line and has two main functions. The first is of course the control of the CCDs (power conditioning, clocking, read-out, analogue-digital conversion etc.). The N-FEEs are triggered every 6.5 seconds to read out a single CCD. This leads to the overall cadence of 25 seconds for the normal cameras. The second function of the N-FEE is to select the pixels of interest for the science products. This function is called "windowing". A window list of up to 150.000 entries can be uploaded to each N-FEE. The N-FEE then cuts out the selected pixels and only transfers the selected pixels to the DPU. The N-FEE can also be commanded to transfer full CCD frames for calibration and characterization purposes. In this case a full CCD is transferred over two 6.5 second cycles. The N-FEE also implements other calibration functionalities such as reverse-clocking and trap-pumping. The N-FEEs are powered and clocked by the N-AEUs [6].

Normal data processing units

Each normal data processing unit (N-DPU) controls two N-FEEs. It is equipped with a dual-core LEON3 processor and 256 Mbytes of SDRAM. Each core serves an N-FEE and runs an independent instance of the real-time operating system RTEMS. The main functions of the N-DPUs are, to command and configure the N-FEEs, to monitor the N-FEE and CCD health, and of course to execute the science data processing described in the sections below. 6 N-DPUs serving 12 N-FEEs are bundled into the MEU and are connected via two redundant SpaceWire links with two SpaceWire routers inside the MEU. Each router is connected with one side of the ICU via a SpaceWire link. The N-DPUs secondary voltages are conditioned by the MEU's internal power supply [7].

Fast front-end electronics

Each fast camera is equipped with 4 frame transfer CCDs and is served by the fast-front-end electronics (F-FEE). The functions of the F-FEE are very similar to the N-FEEs with the main difference of a 2.5 second trigger pulse and a simultaneous read-out of all 4 CCDs. To achieve a low latency in the fast processing chain and enable the simultaneous readout of all 4 CCDs, the F-FEEs are connected to the fast data processing units (F-DPUs) through 4 parallel SpaceWire lines. As the N-FEEs, the F-FEEs also can be commanded to transfer full CCD frames for calibration purposes. In this case the data of a single CCD is transmitted over two SpaceWire links simultaneously. The F-FEEs are powered and clocked by the F-AEU [8].

Fast data processing units

Each fast data processing unit (F-DPU) controls a single F-FEE. It is equipped with a LEON2 processor, 8 Mbytes of SRAM for the on-board software running the real-time operating system RTEMS and a companion FPGA which is used to reconstruct the windowed pixel data received from the F-FEE through the 4 SpaceWire links. The pixel data is stored in a dedicated 512 Mbyte of SDRAM. The main function of the F-DPUs is to command and configure the F-FEEs and to provide fine guidance quaternions to the space-craft with a maximum latency of 3.75 seconds relative to the middle of integration via a dedicated SpaceWire line, the timing is shown in Fig. 2 below. Furthermore, the F-DPUs are also providing windowed pixel data as science products. Both F-DPUs are bundled into the FEU and are connected via two redundant SpaceWire links to the ICU [9].

Instrument control unit

The instrument control unit (ICU) as the name suggests is controlling all the units of the instrument itself short of the primary power supplies that are controlled by the space-craft. It does control however, the secondary power supplies of the N-DPUs and the FEEs via the AEUs. The second main functionality of the ICU is the compression of all science data products. For this purpose, it is equipped with a hardware compression unit. The science data received from the DPUs is temporarily stored in a dedicated 512 Mbyte SDRAM and is subsequently compressed by the compression hardware. The compressed science data then is sent to the mass memory unit (MMU) of the space-craft. Only windowed pixel data is compressed in hardware. The other science products are compressed by software in the ICU [10].

Normal and fast ancillary electronics units

The ancillary electronics units (AEUs) main functions are the precise conditioning of the FEE secondary voltages and the provision of a synchronization pulse that is used in the FEEs to trigger the simultaneous read-out of all the FEEs. As the AEUs are not considered part of the DPS, the AEUs are not described in great detail in this paper. Nevertheless, due to the high demands on the precision of synchronization signals and secondary voltages, they are an integral part of the PLATO payload [11].

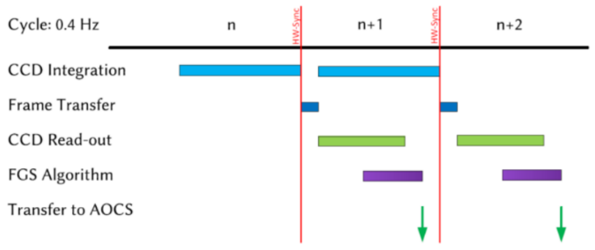

Figure 2 Fine-Guidance processing timeline [12]

3. PLATO SCIENCE DATA PRODUCTS

In the following paragraphs we will give a short overview of the science data products of the PLATO payload. The reason for the different data products is to reduce the data-rates even further compared to providing only imagette data. The selection of the kind of data product for each target can be commanded to the DPUs in order to saturate the available down-link. The on-board data-flow of science data products is shown in Fig. 3 below.

1.2 Imagettes and auxiliary data

Imagettes are unprocessed pixel data received by the DPUs from the FEEs. The only processing done on-board is the reordering of pixels into continuous pixel streams, grouping into science packets and compression. The details of the compression are described in the section "Compression" below. As the imagettes are the raw data from the CCDs, they are the primary science data product of PLATO. In addition to the target imagettes the following auxiliary data is transmitted: Electronic offset, background values (pixels with no stars), smearing patterns (resulting from the exposure during readout), charge-transfer inefficiency parameters and finally house-keeping parameters like temperatures bias-voltages etc. Imagettes can be between 4x4 and 12x12 pixel.

1.3 Fluxes

In order to observe even more targets then can be down-linked as imagettes, the DPUs can be commanded to process targets on board. One possibility is to calculate the flux of a target and only transmit the flux as a single number. This achieves a data reduction factor of approximately 7[1]. A configurable mask is applied to the received pixels prior to the flux calculation. The electronic offset correction, the background subtraction and the smearing correction is also applied on-board. The fluxes then can be averaged over different cadences of 1, 2 or 24 exposures. The fluxes then can be ordered into a time-series on ground to generate light curves.

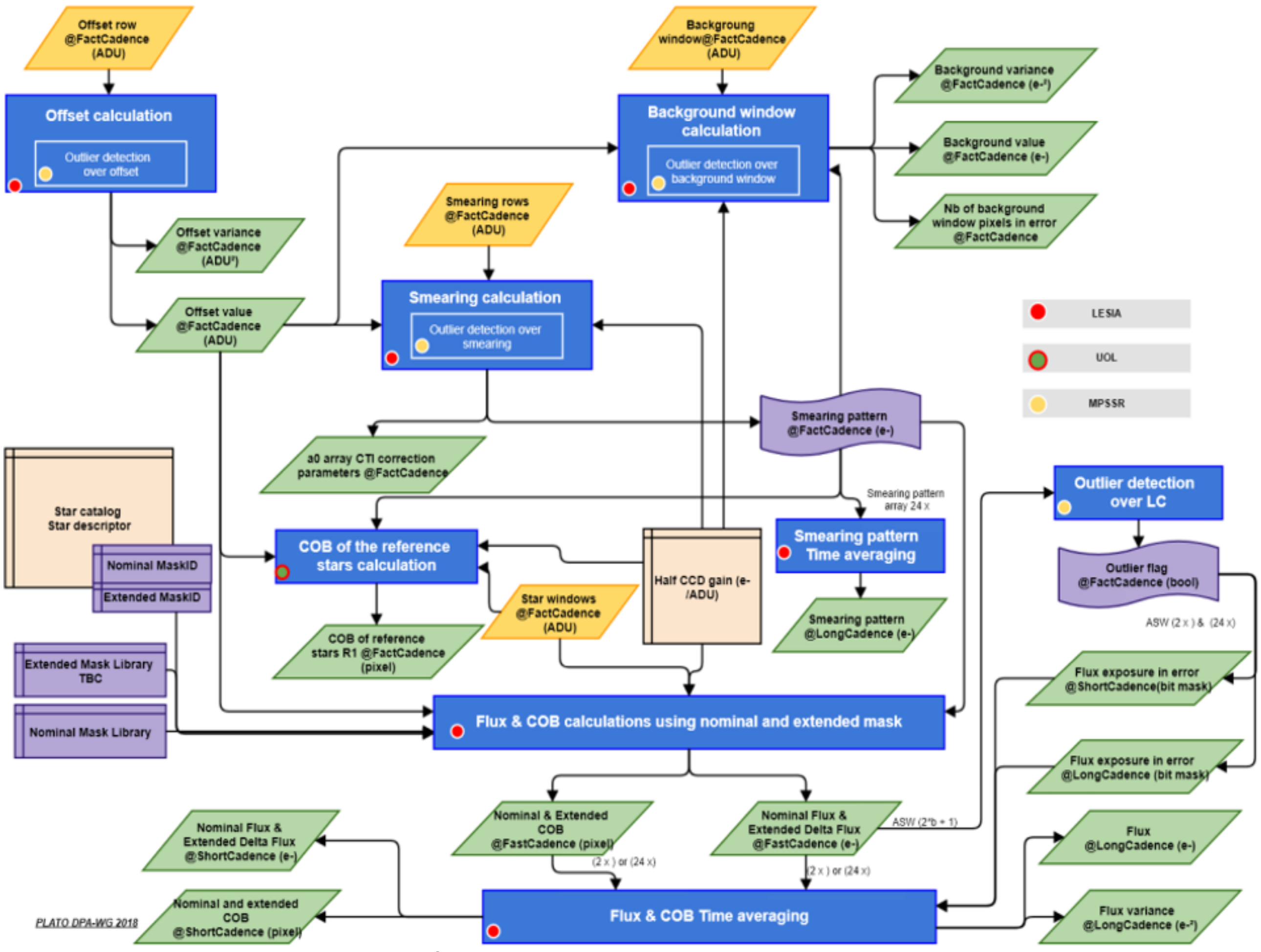
Offset row @FactCadence (ADU)
Offset calculation
Outlier detection over offset
Offset variance @FactCadence (ADU²)
Offset value @FactCadence (ADU)
Backgroung window@FactCadence (ADU)
Background window calculation
Outlier detection over background window
Background variance @FactCadence (e-²)
Background value @FactCadence (e-)
Nb of background window pixels in error @FactCadence
Smearing rows @FactCadence (ADU)
Smearing calculation
Outlier detection over smearing
LESIA
UOL
MPSSR
Smearing pattern @FactCadence (e-)
a0 array CTI correction parameters @FactCadence
Smearing pattern array 24 x
Star catalog Star descriptor
Nominal MaskID
Extended MaskID
COB of the reference stars calculation
Half CCD gain (e-/ADU)
Smearing pattern Time averaging
Outlier detection over LC
Outlier flag @FactCadence (bool)
ASW (2 x) & (24 x)
Star windows @FactCadence (ADU)
COB of reference stars R1 @FactCadence (pixel)
Smearing pattern @LongCadence (e-)
Extended Mask Library TBC
Nominal Mask Library
Flux & COB calculations using nominal and extended mask
Flux exposure in error @ShortCadence(bit mask)
Flux exposure in error @LongCadence (bit mask)
Nominal Flux & Extended Delta Flux @ShortCadence (e-)
Nominal & Extended COB @FastCadence (pixel)
Nominal Flux & Extended Delta Flux @FastCadence (e-)
ASW (2*b + 1)
(2 x) or (24 x)
(2 x) or (24 x)
Flux @LongCadence (e-)
Nominal and extended COB @ShortCadence (pixel)
PLATO DPA-WG 2018
Flux & COB Time averaging
Flux variance @LongCadence (e-²)

Figure 3 PLATO on-board science data-flow [13]

[1] The concrete data-reduction factor depends on the size of the input imagette and the cadence of the flux.

1.4 Centroids

A second kind of processed data product is the centroids. In order to generate a centroid, the N-DPU calculates the center of brightness of the target. Again, a mask is applied to the received pixel data and the corrections are applied as describe in the paragraph above, prior to the calculation. The centroid is transmitted as the x- and y-coordinate of the center of brightness relative to the middle of the window. This achieves a data reduction of approximately 5[2].

1.5 Meta-data removal

In early definitions of the PLATO science data products, it was foreseen to include in the science product, the static meta-data (target ID, number of pixels, etc.) of each product. As this data does not change over time and would be transmitted repeatedly, it was decided to remove this meta-data and only transmit it on request. This increases the available bandwidth for raw science data and also the efficiency of the compression. The amount of meta-data that was removed is shown in Fig. 4 below.

1.6 Full frame image

For calibration purposes the PLATO instrument will also produce full frame images. These full-frames will be used in order to characterize the CCDs and calculate the point spread functions of the target stars and to aid the scientists in the target selection process. Cosmetic defects of the CCDs (bright and dark pixels) can also be detected and targets selected accordingly. It is planned to down-link a full field of view image after each observation and after each quarterly roll (PLATO is rotated every 90 days by 90 degree in order to cope with the orbital dynamics in a similar way as Kepler). These full frame images will also be published as science products.

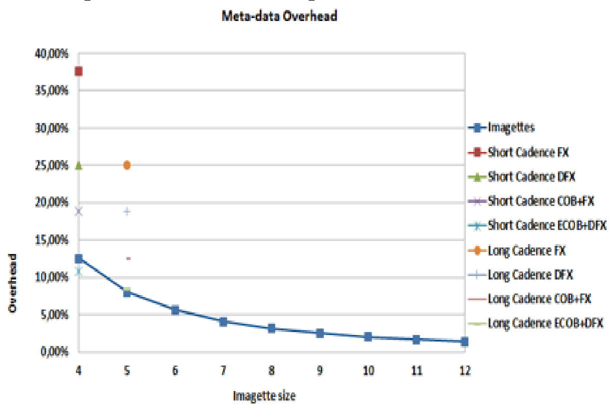

Figure 5 Meta-data Overhead

[2] See above, also the centroids are combined in packets with fluxes so the data reduction can benefit from shared meta-data.

1.7 Compression

In order to further increase the number of targets that can be observed and down-linked, the imagette data products are compressed in the ICU. The compression algorithm used for this is using Golomb coding. Golomb coding is a lossless compression method and is an optimal prefix code for alphabets with a geometric distribution [14]. Golomb coding is suitable for data in which the occurrence of small values is significantly more likely than large values. In order to ensure this property, the dynamic range of the data first is reduced by calculating the difference between the data and a data mode. Because this can lead to negative numbers and negative signed numbers in conventional two's complement are large unsigned numbers, the negative numbers are not represented as two's complement but instead interleaved with the positive numbers (0,1,-1,2,-2,3,-3,4,-4…)[3]. This difference is then compressed using the Golomb code in a dedicated compression hardware. Using these techniques, a lossless compression of a factor of 3.5 (or slightly better) can be achieved. The data-flow of this algorithm is shown in Fig. 5 below. In regular intervals the model is updated and down-linked in order to maintain a high compression factor and to minimize the impact of possible corruption in the transfer of compressed science data packets. A similar algorithm is also implemented in the ICU software to allow for compression of other science data products. Kepler has achieved compression ratios of more than four, but has implemented a re-quantization in the compression which is technically not lossless. Nevertheless, Kepler also has shown that this re-quantization is "quasi-lossless" as no information relevant for the science data processing is lost. The baseline for PLATO is not to use re-quantization and instead do more on-board science data processing in order to achieve the science goals of the mission as described in chapter 1 above.

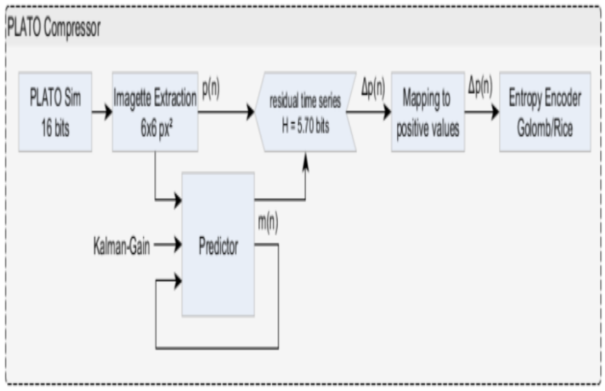
PLATO Compressor
PLATO Sim 16 bits
Imagette Extraction 6x6 px²
p(n)
residual time series H = 5.70 bits
Δp(n)
Mapping to positive values
Δp(n)
Entropy Encoder Golomb/Rice
Kalman-Gain
Predictor
m(n)

Figure 4 PLATO Compressor Data-flow [15]

[3] The formula for positive numbers is N'=2*N and for negative numbers N'=(abs(N)*2)+1.

4. DATA PROCESSING SYSTEMS ENGINEERING

In the following section we describe important design principals, tools and techniques that have been applied in order to design, specify, and implement the complex PLATO DPS system. The following examples are not exhaustive and do not include the mandatory application of standards such as ECSS-E-ST-40C or similar.

1.8 Principles and philosophy

One of the most important principles in system engineering is to clearly define responsibilities and interfaces. For the PLATO payload the functional allocation is straight forward, as described in section 2 above. The FEEs manage the CCDs and provide the windowing functionality, the DPUs manage the FEEs and process the pixel data in order to reduce data volume. The AEUs provide the high-precision sync pulses and condition the secondary power, while the ICU manages the whole payload, stores the software binaries and configuration data and provides the hardware accelerated compression. In terms of operations and FDIR these responsibilities are very similar. Each unit in the FEE => DPU => ICU => Spacecraft => Ground-station chain either detects (and reports) a failure or has the ability to isolate or even recover the failure. This is achieved by using standardized interfaces, in this case standardized PUS services as described below.

1.9 Hard- and software interfaces

As described above, clean interfaces are one of the key design goals for systems-engineering. Standards help in achieving this, because standards are usually well documented and understood. The following hard- and software interfaces are used in the PLATO payload[4].

SpaceWire

SpaceWire is a well-established serial interface, widely used in the space community [16]. As such, the usage of SpaceWire is neither especially innovative nor noteworthy. However, the fact that the PLATO payload does not feature any low-speed interfaces such as MIL1553, together with the fact that only the synchronization lines between the AEUs and the FEEs are discrete signal lines and the usage of the RMAP protocol for remote-terminals without software, greatly reduces the complexity of the system as a whole. Furthermore, this reduces the reliance on complex EGSE systems during AIT.

[4] In addition to the described standardized SpaceWire protocols, PLATO uses a proprietary protocol exclusively between the FEEs and DPUs.

RMAP

The Remote Memory Access Protocol (RMAP) is a standardized way to access memory region of remote units without requiring the remote unit to run a software [17]. RMAP is used for two distinct purposes. The first is to send data to and acquire data from “non-intelligent” units such as the AEUs, the MEU power-supply units and the SpaceWire routers. The second is, to boot the DPUs. The remote booting of the DPUs by loading the software images directly into RAM by the ICU, allows to significantly reduce the hardware complexity of the DPUs, as the DPUs do not need to be equipped with non-volatile memory. Furthermore, this reduces the operational complexity, as no boot software is needed in the DPUs and the central storage of the application software images greatly simplifies the application software image maintenance.

CPTP

The CCSDS Packet Transport Protocol (CPTP) is the second standardized protocol on top of the SpaceWire standard [18]. In the case of the PLATO payload, all CCSDS packets transported between the payload units and between the payload and the spacecraft have been specified to be Packet Utilization Standard (PUS) compliant. An advantage of using the CPTP instead of RMAP or other possible proprietary protocols is the fact that the CPTP is an asynchronous protocol. This means, that there is are very little dependencies (for example in timing, memory regions, etc.) between the different units, which simplifies the implementation. A disadvantage however is the lack of acknowledgement in the protocol layer which is handled by the PUS itself.

PUS

The Packet Utilization Standard (PUS) specifies a set of functionalities and packet formats that should be used in accordance to a mission specific tailoring [19]. The usage of these standardized services allows a very efficient re-use of not only on-board software but also EGSE and ground segment software. Furthermore, the PUS defines a set of services (the monitoring service, the event reporting service and the event action service) that in combination allows for a very efficient way of implementing complex FDIR scenarios. This decoupling of FDIR scenarios from the application software design and implementation details also makes it possible to flexibly evolve the FDIR scenarios in a late stage of the project without needing to change the specification of the software itself. Lastly, the usage of PUS packets for (nearly) everything also makes testing of specific units without the availability of others easy, as the only thing needed is an EGSE that allows to send and receive PUS packets.

1.10 Requirement engineering tools

The usage of requirement authoring and management tools has become standard practice in space-projects and because of this will not be described in detail in this paper. For PLATO, IBM DOORS was chosen as it is widely used in the industry. Between the different stakeholders in the project, the requirement baselines are exchanged directly through DOORS modules, but in order to integrate the requirements into the other systems-engineering tools described below, the requirement interchange format (ReqIF) [20] can be used.

1.11 Interface engineering tools

One of the interesting aspects of the PLATO DPS architecture is the great number of functionally identical hardware units. Each DPU for example implements the same PUS services and subservices, but naturally is allocated different APIDs. The PLATO Payload TMTC database engineering tool allows to define the packet structure of PUS packets and implements many features that are needed to efficiently work with the great number of units and APIDs in the system. For example, it allows to define a packet structure only once and then allocate this packet to as many APIDs as needed. Also, it is possible to define different calibration curves for the same parameter, one for each unit the parameter is allocated to. The TMTC database is implemented using the Eclipse Modeling Framework (EMF) [21] and uses a Connected Data Objects (CDO) repository as backend. The usage of CDO allows essential features such as revision control, change tracking and auditing in a transparent way [22]. Furthermore, as described in the following paragraphs, the leverage of the Eclipse ecosystem allows for a seamless integration of the TMTC database with other tools used in the system engineering context. This is shown in Fig. 6 below.

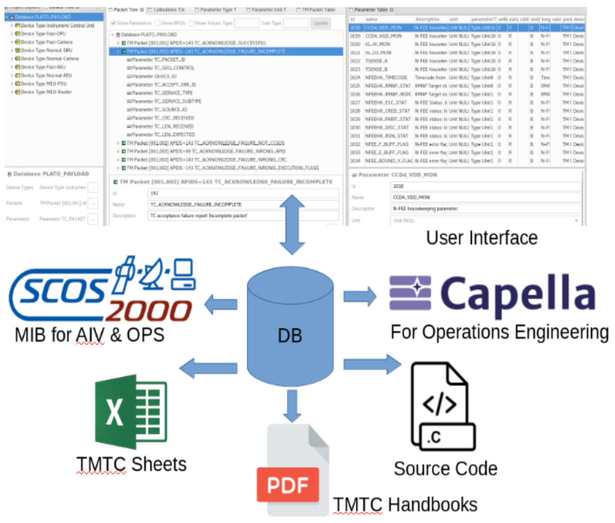

Figure 6 TMTC database export features

Model to Text Generation

The TMTC database uses the Acceleo Model to Text (M2T) transformation language in order to efficiently implement export functionalities [23]. Currently exporters to the SCOS2000 import ICD format (MIB) [24], a latex exporter for generation of TMTC handbooks and a generator for on-board source code are available. The usage of these exporters guarantees consistency between the MIB, the documentation and the implementation of the TMTC handling on-board. Furthermore, the generation of the on-board software code allows the on-board software engineers to concentrate on the important features as the amount of boilerplate to be written by hand is greatly reduced.

1.12 Operations engineering tools

Another important aspect of the systems-engineering process of space-missions is the planning of complex operational scenarios. Especially in the case of PLATO where due to the complexity of the system, there are a great amount of interactions. The open-source model-based systems-engineering (MBSE) tool was chosen for PLATO (not only for operations engineering, but also other tasks, such as interface definition, requirement-to-design tracing etc.). Capella also is part of the Eclipse ecosystem, which makes the interface between Capella and the TMTC database easy to implement [25]. Furthermore, Capella already has a ReqIF import interface, which allows to directly trace specific items to requirements. Operational and FDIR scenarios are modeled in Capella. A study is planned in order to investigate the integration between Capella and the DLR tool PROTOS, which will be used in order to produce the operation procedures in the MOIS XML format that has been requested by ESA [26]. In case this is not possible, the procedures will have to be translated manually.

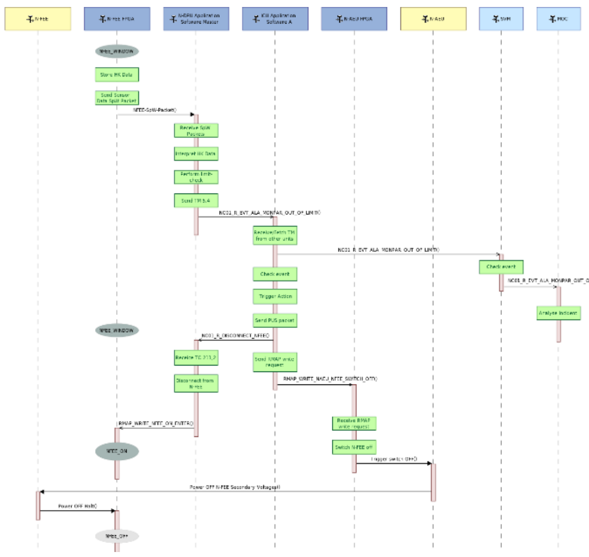

Figure 7 Example FDIR scenario modeled in Capella

REFERENCES

[1] Auvergne, M. Et al.: The CoRoT satellite in flight: description and performance. A&A 506, 411–424 (2009).

[2] Vandermarcq, O.: At the heart of the COROT mission operations. SpaceOps 2008 Conference (Hosted and organized by ESA and EUMETSAT in association with AIAA)

[3] Koch, D. G. Et al.: KEPLER MISSION DESIGN, REALIZED PHOTOMETRIC PERFORMANCE, AND EARLY SCIENCE. The Astrophysical Journal Letters, 713, 79–86, (2010)

[4] Schlieder, J. Et al.: TESS Guest Investigator Program - TESS Observatory Guide. Version 1.1, TESS Science Support Center, NASA Goddard Space Flight Center, Greenbelt, MD, (2017).

[5] Ziemke, C., Et al.: PLATO-DLR-PL-RS-0006, PLATO (On-Board) Software System Specification, Issue 2.3, German Aerospace Center, Berlin, (2019).

[6] Hailey, M., Et al.: PLATO-MSSL-PL-DD-0001, N-FEE Design Description, Issue 6.0, UCL Department of Space and Climate Physics Mullard Space Science Laboratory, Surrey, (2021)

[7] Plasson, P., Et al.: PLATO-LESIA-PL-DD-0004, N-DPU Application Software Design Document (SDD), Issue 1.1, Observatoire de PARIS Section de MEUDON – LESIA, Paris, (2019)

[8] Koncz, A., Et al.: PLATO-DLR-PL-DD-001, F-FEE Design Description, Issue 1.3, German Aerospace Center, Berlin, (2016).

[9] Witteck, U., Et al.: PLATO-DLR-PL-DD-0002, F-DPU Software Design Document, Issue 1.0, German Aerospace Center, Berlin, (2019).

[10] Galli, E., Et al.: PLATO-INAF-PL-DD-0007, ICU-ASW Design Document, Issue 1.3, INAF - IAPS, Rome, (2021).

[11] Koch, M., PTO-ASPE-MA-RS-2000, PLATO F-AEU Specification, Issue 4, Advanced Space Power Equipment GmbH, Salem, (2021)

[12] Grießbach, D., Et al.: PLATO-DLR-PL-RP-0003, Fine Guidance System Performance Report, Issue 3.3, German Aerospace Center, Berlin, (2020).

[13] Samadi, R. Et al.: PLATO N-DPU: Architecture and data flows of the on-board science data processing pipeline, Issue 2.1, Observatoire de Paris, Paris, (2019).

[14] Golomb, S. W: Run-length encodings. IEEE Transactions on Information Theory, IT--12(3):399—401, (1966).

[15] Ottensamer, R. Heiss, M. Loidolt, D.: PLATO Data Compression Concept, Issue 1, University of Vienna, Vienna, (2019).

[16] ECSS-E-ST-50-12 - Space engineering - Space-Wire - Links, nodes, routers and networks, Issue C, ECSS, (2008).

[17] ECSS-E-ST-50-52 - Space engineering - Space-Wire - Remote memory access protocol, Issue C, ECSS, (2008).

[18] ECSS-E-ST-50-53 - Space engineering - Space-Wire - CCSDS packet transfer protocol, Issue C, ECSS, (2008).

[19] ECSS-E-ST-70-41 - Space engineering - Telemetry and telecommand packet utilization, Issue C, ECSS, (2016)

[20] Requirements Interchange Format (ReqIF), Issue 1.2, OMG, (2016)

[21] Merks, E. Et al.: EMF: Eclipse Modeling Framework, 2nd Edition, Addison-Wesley Professional, (2008).

[22] Seybold, D., Et al.: Experiences of models@ run-time with EMF and CDO, Proceedings of SLE 2016, Amsterdam, (2016)

[23] Luhunu, L., Syriani, E.: Comparison of the expressiveness and performance of template-based code generation tools, 10th ACM SIGPLAN International Conference, (2017)

[24] EGOS-MCS-S2K-ICD-0001 - SCOS-2000 Database Import ICD, Issue 7.1, (2018)

[25] Roques, P.: MBSE with the ARCADIA Method and the Capella Tool, (2016)

[26] Beck, T., Et al.: ProToS: Next Generation Procedure Tool Suite for Creation, Execution and Automation of Flight Control Procedures, 14th International Conference on Space Operations, Daejeon, (2016)