SSC22-XII-03

Implementing the New CCSDS Housekeeping Data Compression Standard 124.0-B-1 (based on POCKET+) on OPS-SAT-1

David Evans, Georges Labrèche, Dominik Marszk, Sam Bammens
European Space Operations Center (ESOC), European Space Agency (ESA)
Darmstadt, Germany; +49 6151 902720
david.evans@esa.int

Miguel Hernández-Cabronero
Department of Information and Communications Engineering (dEIC), Universitat Autònoma de Barcelona (UAB)
Bellaterra, Spain; +34 935 811 861
miguel.hernandez@uab.cat

Vladimir Zelenevskiy
Telespazio Germany GmbH
Darmstadt, Germany; +49 615 182 570
vladimir.zelenevskiy@telespazio.de

Vasundhara Shiradhonkar
Terma GmbH
Darmstadt, Germany; +49 6151 860050
vash@terma.com

Milenko Starcik
VisionSpace Technologies GmbH
Darmstadt, Germany; +49 6151 6292270
milenko.starcik@visionspace.com

Maximilian Henkel
Institute of Communication Networks and Satellite Communications, Graz University of Technology
Graz, Austria; +43 316 873 7437
henkel@tugraz.at

ABSTRACT

The number of telemetry parameters available in a typical spacecraft is constantly increasing. At the same time, the bandwidth available to download all that information is rather static. Operators must therefore make hard choices between which parameters to downlink or not, in which different situations, and at which sampling rates. This trade-off is more problematic for missions with higher communication latency beyond LEO. Since 2009, The European Space Agency's European Space Operations Center (ESA/ESOC) has been promoting the compression of housekeeping telemetry as a solution to this problem. Most spacecraft housekeeping telemetry parameters compress extremely well if they are pre-processed correctly. Unfortunately, most spacecraft record telemetry packets in flat packet stores so accessing different packets within them is too CPU and memory intensive for flight computers. Using traditional compression schemes such as zip or tar are not compatible with the traditional "fire and forget" mode of operation i.e., occasional packet losses are expected. This would render entire compressed files unusable. ESOC invented an algorithm called POCKET+ to solve this problem. It is implemented using very low-level processor instructions such as OR, XOR, AND, etc. This means that it can run with low CPU usage and, more importantly, with a short execution time. It is designed to run fast enough to compress a stream of incoming packets as they are generated by the on-board packetiser. The output is a smaller stream of packets. The compressed packets can be handled by the on-board system in an identical fashion to the original larger uncompressed packets. Robustness with respect to the occasional packet loss is built into the protocol and does not require a back channel. In 2018, POCKET+ was proposed to the CCSDS data compression working group and after extensive research by other agencies the core idea has been

incorporated into a proposed new standard for "Robust Compression of Fixed Length Housekeeping Data." The second supporter for the mission is CNES, supported technically by the University of Barcelona (UAB). Both CNES and UAB have suggested changes that make POCKET+ even more powerful. POCKET+ is already flying on OPS-SAT, a 3U CubeSat launched by the European Space Agency on December 18th, 2019. The mission has updated the Onboard Software (OBSW) and ground control software to be compliant with the latest POCKET+ standard. The standard is set to be available for an ESA review. This paper describes the latest algorithm and how it is implemented on OPS-SAT, including how the same core software has been successfully deployed in two completely different scenarios/environments. One compresses files offline and then uses a transport protocol with a completeness guarantee; the other compresses a packet stream in real-time and uses the classic transport protocol where completeness is not guaranteed. The results show that compression ratios between eight and ten are usual for the OPS-SAT mission. Improvements made during the development of the planned CCSDS standard for "Robust Compression of Fixed Length Housekeeping Data" are also presented.

INTRODUCTION

The number of telemetry parameters available in a typical spacecraft is constantly increasing. At the same time, the bandwidth available to download all that information is usually bounded. Operators must make hard choices between which parameters to downlink or not, in which situations, and at which sampling rates.

To provide context, this paper first outlines the POCKET history, then it provides the summary of the basic principles of POCKET+. An overview is presented on the decompression and compression mechanisms of the Consultative Committee for Space Data Systems (CCSDS) 124.0-B-1 algorithm which is based on POCKET+. The general details of the OPS-SAT mission are described after which the usage of *CCSDS 124.0-B-1* on-board OPS-SAT is outlined in three different environments. Future work is presented for the algorithm, followed by a conclusion.

POCKET AND POCKET+ DEVELOPMENT

ESA/ESOC started promoting the compression of housekeeping telemetry in 2009. This is because they found that most spacecraft housekeeping telemetry packets compressed extremely well if they were pre-processed correctly. Several patented algorithms were produced that reached average compression ratios between 5 and 20 when tested with real stored housekeeping data (depending on the mission). One of those algorithms, called POCKET, can compress individual packets in only a few microseconds on representative flight hardware. This made it suitable for compressing real-time telemetry streams. A packet is generated, compressed into a smaller packet, and then either transmitted or stored for later transmission. On the ground segment, the small packet is intercepted, expanded back into the original packet, and passed to the mission control system with zero impact on the existing infrastructure. In 2012, the complete end-to-end chain was built and tested in an ESA contract with Spacebel SA of Belgium (this was flight tested on PROBA-2 in 2017 by QinetiQ Space Belgium). Also in 2012, the algorithm evolved into POCKET+ which has the advantage of being self-adapting i.e., it reacts to changes in the data behavior (e.g., change of system mode without ground intervention). In 2018, POCKET+ was proposed to the CCSDS data compression working group and after extensive research by other agencies the core idea has been incorporated into a proposed new standard for "Robust Compression of Fixed Length Housekeeping Data" [1, 2, 3, 4]. This greatly simplifies the ground processing required. It has been baselined for ESA's PROBA-3 mission currently in Phase D.

RECAP OF BASIC PRINCIPLES

To explain the advances, it is better to revisit the original POCKET algorithm as this shows the mechanism in its simplest form. This section provides a recap on the basic principles of POCKET and POCKET+.

POCKET

In POCKET, the ground employs statistical analysis of stored data to classify each bit position of each fixed-length housekeeping packet type into those bits that have a good chance of having the same value as the bit in the same position in the previous packet of that type and those that do not. The former are called predictable bits, the latter unpredictable bits. For each packet type the ground then loads a *mask* which defines which bits are predictable and unpredictable.

The compression process is described in Figure 1: (a) A mask is loaded which classifies all predictable bit positions with a zero value and all unpredictable positions with a one; (b) Input the new packet; (c) Check that each predictable bit has the same value as the previous packet and if not mark the position; (d) Record the positions of the predictable bits that changed value as a series of counters compressed using Run Length Coding (RLE); (e) Read all the values of the bits in unpredictable bit positions and append them; (f) Modify the original header indicating the original packet identifier and the length of the new variable length packet.

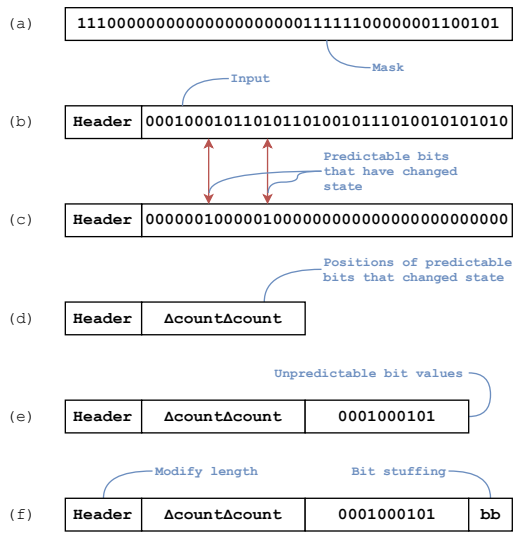

Figure 1: Basic principles of the POCKET method.

Once the ground has the first two pieces of information below, the decompressor is initialised. It is then relatively easy and quick to decompress the following compressed packets:

1. The last successfully decompressed packet of this type (can be past or future).
2. The mask used to compress this packet.
3. All the unpredictable bit values associated with this packet.

It can make a copy of the last successfully decompressed packet and by using the mask information, copy the current unpredictable bit values provided in the compressed packet into their correct positions.

This method provides effective compression when a subset of the bits in the packet tends to be unchanged from one packet to the next. This is often the case for housekeeping telemetry and some spacecraft payload data. The scheme works well on spacecraft housekeeping data and average compression ratios of 10 are achieved for many ESOC controlled spacecraft, e.g., ROSETTA or VENUS EXPRESS. Note that more complicated compression schemes can achieve better compression ratios but they require more CPU/memory to compress each packet or need a significant number of packets to be compressed together i.e. they cannot work on a real-time data flow.

POCKET+

Although POCKET performs well in a quasi-static situation, its performance deteriorates if the behaviors of the bits change with time, i.e., many bits that were predictable become unpredictable and vice versa. This results in a static mask not being optimal. To address this, the algorithm evolved into POCKET+ which updates the on-board mask automatically in the following way. If a bit in a predictable position changes value its position is immediately classified as unpredictable and if any bit in an unpredictable position has a value that remains constant during a certain number of input packets (called a tracking period) its position is classified as predictable.

The ground must apply these rules to update its mask every time it receives a compressed packet. Once done it can decompress the packet as before. However, this is effectively a delta update of the ground mask and is therefore not robust to data loss. To mitigate this, a mechanism to send a mask update which covers several previous updates as well as the last one was added. The number of mask updates included in each packet is referred to as the robustness level which specifies the maximum number of consecutive packets that can be lost without impacting the decompression. For instance, a robustness level of two means the mask update covers the last three iterations and so the decompressor can successfully decompress the packet if any of the previous three packets were correctly decoded. Of course, now there is a balance between an increase in robustness level and a corresponding decrease in compression performance. This must be decided on a case-by-case basis, but the robustness level can be altered for each iteration by setting the first of the following configuration parameters:

1. The requested robustness level, i.e., the maximum number of consecutive packets that can be lost without impacting the ability of the ground to decompress the next packet.
2. A request to start a new tracking period. This enables bit positions to change from unpredictable to predictable if they have not changed value since the last request to start a tracking period was received.
3. A request to include the uncompressed packet in the output rather than just the values of those bits that are classified as unpredictable.
4. A request to include the entire mask information in the output as well as the delta mask information. This is pre-processed and compressed using RLE beforehand.

Note that the configuration parameters 3) and 4) are only there to cover the case when the number of sequential compressed packets lost or corrupted is higher than the robustness level. In this case, ground mask synchronisation is lost, and the initialisation process must be repeated, i.e., the entire mask information must be sent to the ground and in some cases an uncompressed packet as well. It is also worth noting that any

compressed packets that have been received between the loss of ground mask synchronisation and initialisation can then be decompressed, i.e., decompression can work forwards and backwards in a time series of compressed packets.

CCSDS 124.0-B-1

The CCSDS standardisation process successfully generalised the POCKET+ method described in the previous section into CCSDS 124.0-B-1. The standard refers to a method for losslessly compressing a sequence of fixed-length input binary vectors into a sequence of variable-length output binary vectors rather than specifying they must be CCSDS packets. The standard imposes no requirements on the format of the input data except that the input vectors must be of fixed-length. Even the requirement that the bit length must be an integer number of bytes has been removed making it very generic. Thus, this paper will henceforth refer to vectors rather than packets.

Robust compression of fixed-length housekeeping data

Another improvement is that *CCSDS 124.0-B-1* telemeters the effective robustness level achieved in each output binary vector. Note that this can be significantly higher than the requested robustness level as it considers the cases when the mask did not change during an iteration. *CCSDS 124.0-B-1* limits the requested robustness level to a maximum of 7 while the effective robustness level has a maximum of 15. This comes at no extra coding cost apart from adding an extra bit to telemeter the larger value. The advantage of sending this information is that a decompressor can now check that any gaps in the output binary sequence are less than the effective robustness level and determine whether decompression can be reliably performed.

The working group's changes successfully allowed the decompressor on the ground to work only using the information in the compressed packets, i.e., there is no requirement for the ground to be configured beforehand. For example, the bit length of the input binary vector is included in each output binary vector that includes a copy of the input vector. This ensures that the compressed data is self-contained, and the decompressor does not need to manage this value externally. Finally, most user constraints on updating the configuration parameters have been removed so the user can update these parameters at any time except during initialisation.

The final compression mechanism is described in Figure 2. The generated output binary vector is made up of three parts: 1) the run-length encoded mask change position and value information; 2) the entire mask pre-processed and run length encoded (only if requested for re-initialisation); and 3) either a copy of the input binary vector preceded by its bit length or just the bit values of the unpredictable bits in the input binary vector.

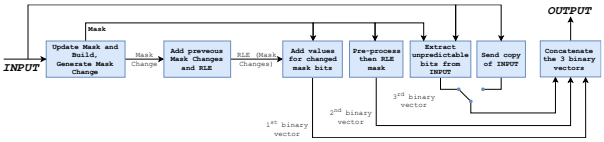

Figure 2: Compression mechanism of CCSDS 124.0-B-1

Decompression

One of the requirements of CCSDS standardisation is to perform a cross validation with another independent implementer from a different agency. This was achieved in partnership with the Universitat Autònoma de Barcelona (UAB), sponsored by CNES. This was relatively easy on the compression side but turned out to be challenging when cross validating the decompression process. Both UAB and ESA were surprised with the number of validity checks that could be applied to ensure that the compressed packets adhered to the *CCSDS 124.0-B-1* compression standard or whether the decompressed packets could be trusted. If these checks failed then the compressed packet must be declared as invalid. The following checks were identified:

1. The effective robustness level is not high enough to guarantee correct decompression, i.e., the last successfully decoded vector was received too long ago.
2. The initialisation process has not completed, i.e., the decompressor has not yet successfully decompressed a compressed vector or received a valid mask vector.
3. The decompression process does not terminate before the end of the compressed vector is reached, implying a corrupted or truncated vector.
4. There is a conflict between the mask information given in the delta mask counters and the absolute mask, implying a corrupted vector.
5. The absolute mask length or the delta mask length is longer than the bit length of the uncompressed packet, implying a corrupted vector.
6. The bit length of the uncompressed packet is outside the standardised limits (1 to 2^{16}-1).
7. The bit length of the uncompressed packet has changed. It is expected to remain the same.
8. The bit stuffing rules have not been respected. These rules are not stated in the standard but are extra fields that can be checked if used.

Even if a decompressed packet passes all these checks, it is not 100% guaranteed that the compressed vector is not corrupt. However, the number of possible checks give some measure of assurance. For instance, the checks cannot check the part of the compressed vector carrying the bit values of the unpredictable bits but on the other hand these values will be replaced in the subsequent vector. Another valid approach is to calculate a checksum on the original uncompressed vector and send this as part of the compressed vector. Then one could simply decompress the vector and then compare the checksum to check validity. In this case, the checks detailed above could be dropped.

OPS-SAT IMPLEMENTS CCSDS 124.0-B-1

The spacecraft was launched with Arianespace on a Soyuz from Kourou on December 18, 2019, following a one-day launch delay. OPS-SAT can be viewed as two satellites in one. A CubeSat satellite along with an ESA satellite flying an advanced communications module and a very powerful on-board computer. There are various peripherals (camera, GPS, advanced ADCS subsystem, etc.) and two payloads of opportunity. The CubeSat bus consists of an on-board computer called the NanoMind, a power subsystem, a UHF communications subsystem, and a basic ADCS subsystem. The mechanical architecture of the OPS-SAT is a 3U CubeSat structure with double folded deployable solar panels. It has a size of 10x10x30 cm (not including deployable) and a mass of approximately 4.8 kg. Two deployable solar array panels generate 30 W of electrical (peak) power. The Satellite Experimental Processing Platform (SEPP) is the heart of the OPS-SAT [5]. It is a powerful ALTERA Cyclone V system-on-chip (SoC) module with sufficient on-board memory to carry out advanced software and hardware experiments [6, 7, 8, 9]. It is the reconfigurable platform required on OPS-SAT on which all major experiments are processed. Both NanoMind and the SEPP communicate with the ground via the CCSDS engine which is effectively a CCSDS compliant packet and frame decoder/encoder. The space segment diagram is shown in Figure 4 of the Appendix.

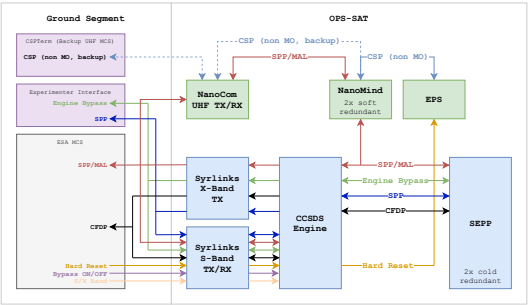

Figure 3: Interactions between ground and space systems.

The ground segment is centered around the European Mission Control Software, SCOS-2000, which has been modified to handle the new application-level interface CCSDS MO Services [10] and packets as well as *CCSDS 124.0-B-1 compressed packets*. File transfer based on CCSDS File Delivery Protocol (CFDP) is available to communicate with the SEPP. The interactions between space and ground segments are shown in Figure 3.

The implementation of POCKET+ on OPS-SAT is a great example of the flexibility of the *CCSDS 124.0-B-1* as it is deployed in three different places in totally different environments.

1. The input is a stream of just generated packets and the transport link assumes one-way communication only.
2. The input is a file of recorded packets and the transport uses two way communication to guarantee completeness.
3. The input is a file of recorded packets and the transport link assumes one-way communication only.

Classic packet stream-based implementation of CCSDS 124.0-B-1 in OBSW

The first implementation is within the On-Board Software (OBSW) running on the NanoMind computer. An embedded POCKET+ library API is called by the OBSW whenever one of a set of pre-specified housekeeping packet types is generated. The entire content of such a packet, including its CCSDS space packet header, is copied into an input buffer. This triggers the CCSDS 124.0-B-1 on-board algorithm which compresses the whole input buffer. The library adds a new 6-byte CCSDS space header with a specific Application Identifier (APID), the new packet length, a source sequence counter and finally a POCKET+ byte indicating which of one of the pre-specified housekeeping packet types was in the input buffer. The output is therefore a valid CCSDS space packet (according to CCSDS 133.0-B-2) and thus processed by the OBSW communication stack in an identical fashion to the uncompressed packets. SCOS then monitors the incoming packet stream and recognises compressed packets via their unique APID. These are then filtered off to a dedicated decompression module which recognises the original packet type from the POCKET+ byte value. The decompression module then reverses the CCSDS 124.0-B-1 compression using the associated mask and last decompressed packet saved for that packet type. It then runs a series of checks on the decompressed packet, see the section on CCSDS 124.0-B-1 decompression, and if these pass it reinserts the packet back into SCOS and processed as normal.

In this implementation, *CCSDS 124.0-B-1* is compressing a stream of packets as they are generated. The output is a stream of smaller packets that are then dealt with by the OBSW in the classical manner, i.e., sent in real-time or written to the packet store for later transmission. Note that the transport protocol used for transporting these packets to the ground assumes one-way communication. Error detection and correction is carried out at frame level so occasional transport frame and therefore packet loss, is expected. The *CCSDS 124.0-B-1* algorithm is therefore configured with different robustness levels to deal with this, depending on the scenario. Furthermore, counters are set which periodically send absolute mask information and/or uncompressed packets, refer to the section recapping the basic principles of POCKET+. Should the number of sequentially lost packets exceed the effective robustness level at any point then this periodic information is used to re-initialise the SCOS decompression module. Note that these counters are configured to send absolute mask information more frequently than uncompressed packets. This is because re-initialisation of the mask is always required if the number of sequentially lost packets exceeds the effective robustness level. However, a new uncompressed packet is not required if the period between it and the new compressed packet contains a maximum of one request to start a new tracking period, see the section recapping the basic principles of POCKET+. This is because the values of all the bits declared as predictable in the new compressed packet will not have changed value during that period.

File based implementation of CCSDS 124.0-B-1 in combination with CFDP transport

The second implementation is written as part of the on-board software running on the SEPP. In this case CCSDS packets are stored in files and the *CCSDS 124.0-B-1* algorithm is used to compress them. Note that processing files has certain advantages, for example there is no need for pre-specifying housekeeping packet types. The file processing ensures that all the packets in the file with a specific type are compressed together and that the first packet of that type is used to initialise the decompressor. In this case, the files are transported to the ground using the CFDP file transfer protocol which uses two-way communication to retransmit lost data blocks. Hence packet loss is not expected once a file is successfully downloaded. This has the advantage of less overheads e.g., no need for the additional information needed by *CCSDS 124.0-B-1* to guarantee resiliency against data loss events. The requested robustness is always set to zero and there is no need to send absolute mask information and/or uncompressed packets periodically, see the section recapping the basic principles of POCKET+.

On the ground, another application processes the file and reverses the *CCSDS 124.0-B-1* compression, resulting in the original packets. These are then played back into SCOS as if they were packets which had been stored at the ground station during a high-speed dump. Using *CCSDS 124.0-B-1* in combination with a file-based approach — as opposed to a packet streaming approach — has the advantage of less complication, e.g., no need to pre-specify packet types or identify them in real-time. In fact, the OPS-SAT flight control team (FCT) now uses this mechanism as a more convenient alternative to using the classic NanoMind packet store and dump approach. The FCT lets the OBSW transmit packets across the Controller Area Network (CAN) bus even outside ground coverage and have implemented an application on the SEPP that records all the CAN traffic in a file. This contains all the housekeeping packets and so it can easily be processed into the input files for the *CCSDS 124.0-B-1* compression described above.

The average compression ratios achieved on the file data and on the streamed data are 10.18 and 8.22 respectively. What is important is that the exact same core software is used in these two completely different environments. The user defined parameters within the algorithm itself entirely handle the difference.

File based implementation of CCSDS 124.0-B-1 in combination with UDP transport

The third implementation is identical to the previous section, but the file is transported to the ground using UDP rather than CFDP i.e., the completeness of the file cannot be guaranteed. This set-up exploits the advantages of file processing outlined above but it can be used in situations where only one way communication is available or practical e.g., long two-way light time delays, operations where uplink is not possible or desired etc. Due to the use of UDP, occasional file blocks losses are expected but the inbuilt robustness of the CCSDS 124.0-B-1 compression algorithm means this will only result in a partial loss of the information in the file. In order to ensure this, the *CCSDS 124.0-B-1* algorithm must be configured with a non-zero robustness level and counters must be set to send absolute mask information and/or uncompressed packets periodically, so there is a performance cost.

FUTURE WORK

An area of interest for future work is that it should be possible to retrieve partial but significant information from lost packets thanks to the *CCSDS 124.0-B-1* algorithm. As stated in the "*Classic packet stream-based implementation of CCSDS 124.0-B-1 in OBSW"* section, re-initialisation after gaps does not require a new uncompressed packet to be sent if the period between the

last decoded packet and the new compressed packet contains a maximum of one request to start a new tracking period, refer to the section recapping the basic principles of POCKET+. This is because the values of all the bits declared as predictable in the new compressed packet cannot have changed value during that period. That information can be used because when parameters are encoded as integers the lowest significant bits of the parameter are much more likely to be declared as unpredictable compared to the most significant bits. Therefore, in some cases, it should be possible to derive a *range* of possible values that those parameters had in those lost packets. This *range* might be completely acceptable for most housekeeping data uses e.g., limit checking, trend analysis, and reporting. This idea will be tested in a future OPS-SAT experiment.

CONCLUSIONS

The latest developments for the POCKET+ algorithm have been described including the improvements made during the development of the related *CCSDS 124.0-B-1* standard for "Robust Compression of Fixed-Length Housekeeping Data." The implementation on-board the ESA OPS-SAT mission has been described and the flexibility of the standard demonstrated, in that the same core software has been successfully deployed in three different scenarios/environments. One compresses files offline and then uses a transport protocol guaranteeing completeness; another compresses a packet stream in real-time and uses a transport protocol where completeness is not guaranteed, and the final implementation combines file processing on-board with a transport protocol where completeness is not guaranteed. Average compression ratios between eight and ten are usual for the OPS-SAT mission, depending on the data and method chosen.

REFERENCES

1. Evans, D., Martinez-Heras, J. A., & Timm, R. (2010). Housekeeping Data: Can You Afford Not to Compress It?. In *SpaceOps 2010 Conference Delivering on the Dream Hosted by NASA Marshall Space Flight Center and Organized by AIAA* (p. 2208).

2. Evans, D. (2012). Ten Times more information in your Real-Time TM. In *SpaceOps 2012* (p. 1275117).

3. Evans, D., Chattlain, O., & Vitulli, R (2014). Pocket Housekeeping Telemetry Compression Algorithm Successfully Passes End to End Tests. O*n-Board Payload Data Compression Workshop*.

4. Evans, D. J. (2016). Ops-sat: Operational concept for ESA's first mission dedicated to operational technology. In *14th International Conference on Space Operations* (p. 2354).

5. Evans, D., Labrèche, G., Mladenov, T., Marszk, D., Shiradhonkar, V., & Zelenevskiy, V. (2022). Agile Development and Rapid Prototyping in a Flying Mission with Open-Source Software Reuse On-Board the OPS-SAT Spacecraft. *AIAA SciTech Forum 2022*. https://doi.org/10.2514/6.2022-0648

6. Evans, D., Labrèche, G., Mladenov, T., Zelenevskiy, V., Marszk, D., & Shiradhonkar, V. (2022). OPS-SAT LEOP and Commissioning: Running a Nanosatellite Project in a Space Agency Context. *36th Annual Small Satellite Conference*.

7. Kacker, S., Meredith, A., Cahoy, K., & Labrèche, G. (2022). Machine Learning Image Processing Algorithms onboard OPS-SAT. *36th Annual Small Satellite Conference*.

8. Labrèche, G., Evans, D., Marszk, D., Mladenov, T., Shiradhonkar, V., Soto, T., & Zelenevskiy, V. (2022). OPS-SAT Spacecraft Autonomy with TensorFlow Lite, Unsupervised Learning, and Online Machine Learning. *2022 IEEE Aerospace Conference.*

9. Mladenov, T., Evans, D., & Zelenevskiy, V. (2022). Implementation of a GNU Radio-Based Search and Rescue Receiver on ESA's OPS-SAT Space Lab. *IEEE Aerospace and Electronic Systems Magazine*, vol. 37, no. 5, pp. 4-12, 1 May 2022. https://www.doi.org/10.1109/MAES.2022.3143875

10. Marszk, D., Evans, D., Mladenov, T., Labrèche, G., Zelenevskiy, V., & Shiradhonkar, V. (2022). MO Services and CFDP in Action on OPS-SAT. *36th Annual Small Satellite Conference*.

APPENDIX

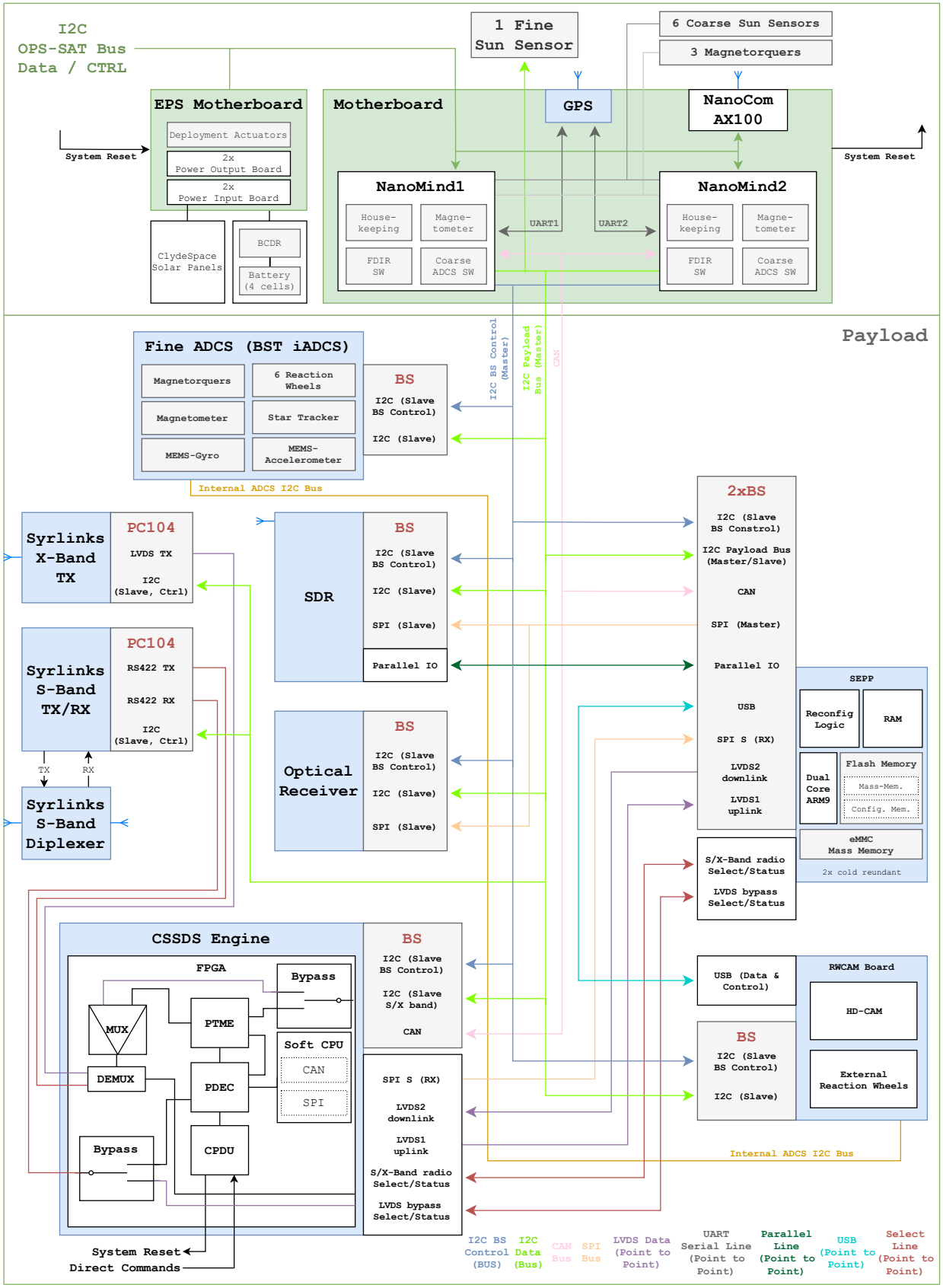

Figure 4: Space Segment System Diagram.