SPORT Mission
Observatory Control and Operations Manual

Rodolfo Wottrich
Aeronautics Institute of Technology (ITA), Brazil
rodolfow@ita.br

DRAFT of April 21, 2021

Approved by:

Luís Eduardo Vergueiro Loures da Costa	Date
Pedro Kukulka de Albuquerque	Date
Lidia Hissae Shibuya Sato	Date

Contents

1 Introduction

This document contains all required procedures and information for ground operations of the *Scintillation Prediction Observations Research Task* (SPORT) mission.

The SPORT mission is a heliophysics-focused investigation of the conditions under which ionospheric variability develops at low latitudes near the equator, leading to scintillation on RF signals. It was proposed to the Heliophysics Division at NASA HQ under the Heliophysics Technology and Instrument Development for Science program within ROSES, announcement number NNH16ZDA001N-HTIDS, and selected for funding in 2017.

The mission shall be fulfilled by an observation satellite (henceforth denominated *observatory*), orbiting the Earth at an altitude close to that of the International Space Station (ISS), and a network of Ground Stations (GSs) on Brazilian territory. The observatory is composed of spacecraft and payloads. The payloads are scientific instruments that make measurements of the ionospheric medium for later scientific analysis on the ground. The spacecraft is a 6U CubeSat [6] with subsystems to allow the instruments to carry out their measurements, for telemetry (TM) to be sent to the ground, and for telecommands (TCs) to be received from the ground for remote control of the observatory.

SPORT is a binational partnership between Brazil and the USA. The Brazilian partners are the Brazilian Space Agency (AEB), Aeronautics Institute of Technology (ITA), and National Institute for Space Research (INPE). The American partners are NASA Marshall Space Flight Center (MSFC), University of Texas at Dallas (UTD), The Aerospace Corporation, NASA Goddard Space Flight Center (GSFC), and Utah State University (USU). The USA are providing the payloads and the launch to orbit. On the Brazilian side, ITA are contributing the spacecraft and observatory integration and testing, and INPE are contributing observatory integration and testing facilities, GS networks, mission operations, and data management. The scientific data will be distributed from and archived at INPE/EMBRACE regional space-weather forecasting center in Brazil and mirrored at the NASA GSFC Space Physics Data Facility (SPDF).

This document is exclusively intended for the members of ITA, INPE, NASA, and USU who are directly involved in the SPORT mission.

The remainder of this document is organized as follows: Section 2 presents the observatory's high-level architecture, introducing the roles of each component, Section 3 describes how the spacecraft runs in a finite-state machine of operating modes, Section 4 describes all communication links and protocols for interaction with the observatory, Section 5 discusses how the spacecraft keeps and manages time information, Section 6 describes in detail how each individual component in the architecture operates, Section 7 discusses how operators should manage the spacecraft's memory resources, Section 8 describes all aspects of the observatory's Attitude Determination and Control Subsystem (ADCS), and Section 9 discusses the spacecraft's mechanisms to detect and mitigate faults. Additionally, Appendix A describes in detail the spacecraft's initialization routine and Appendices B through D extensively describe the formats of all types of uplink and downlink data.

2 Observatory Architecture

Figure 1 displays the high-level organization of components in the observatory's architecture.

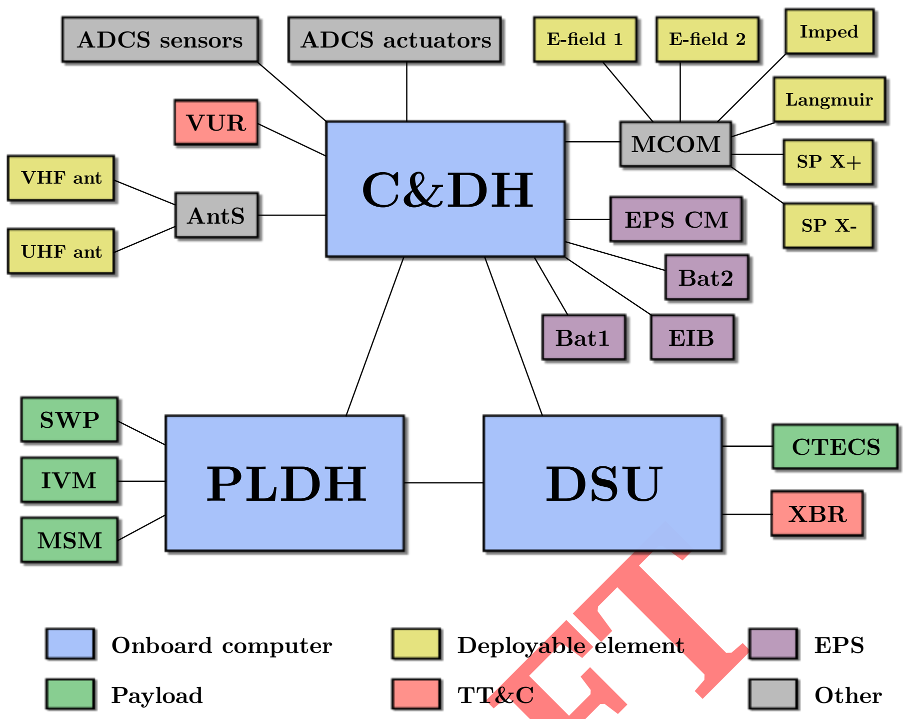


Figure 1: High-level observatory architecture.

Observatory by-products are denominated *mission data* and are composed of *science data* packets, *ancillary data* packets, and *operational data* packets.

Science data packets are produced by the observatory's four scientific payloads:

- NASA GSFC's *Miniaturized Science Magnetometer* (MSM): a precise magnetic field sensor;
- The Aerospace Corporation's *Compact Total Electron Content Sensor* (CTECS): a GPS receiver that tracks the occultation of GPS constellation satellites in the ionospheric horizon. Besides its science data production, it provides the spacecraft with GPS positioning data and precise time. It also produces a precise 1 Hz electrical signal, synchronized to GPS time to the milliseconds, that is distributed by the spacecraft to several components, to allow for time synchronization. Such signal is named *pulse-per-second* (PPS);
- USU's *Space Weather Probes* (SWP): a set of sensors that make measurements such as plasma temperature, plasma density, and electric field; *and*
- UTD's *Ion Velocity Meter* (IVM): a sensor that measures the velocity and direction of ions that collide with the observatory.

Ancillary data packets are produced by the spacecraft itself and include observatory position and attitude info, power info, and status of observatory components. Ancillary data are helpful in mission operations and in the analysis of science data.

The *Electrical Power Subsystem* (EPS) is the subsystem that conditions and distributes power to all components in the observatory. It is composed of two battery packs (Bat1 and Bat2) that are recharged when the solar panels receive sunlight and a control module (EPS CM) that allows the power to different components be switched on/off. Bat1, Bat2 and EPS CM are individual microcontrolled modules, manufactured by the company Clyde Space, that are also able to provide telemetry related to several EPS aspects, such as voltage and current of the available buses, switchable lines and batteries. EPS CM provides several power buses with different voltages and current capacities, which can be monitored by ground operators in ancillary data packets. To extend the functionality of this Clyde Space solution, the SPORT engineering team developed the *EPS Interface Board* (EIB), an in-house component that extends EPS' functionality by implementing extra switchable lines that allow more observatory components to be independently powered on/off.

The interface between the observatory and ground is known as *Telemetry, Tracking, and Command* (TT&C). TT&C is carried out through three radiofrequency channels (one uplink and two downlinks). The uplink operates on VHF, and its purpose is to allow ground operators to control the observatory with TCs. The first downlink operates on UHF. Its purpose is to allow the observatory to send TM that is useful for ground operations: beacons (periodically) and TC responses, including mission data samples, on demand. The onboard *VHF/UHF radio* (VUR) is TRXVU, manufactured by the company ISISpace.

According to data budget and orbital analyses, the observatory generates much more mission data daily than what the bandwidth of the UHF channel would allow to be transmitted to ground. Hence, the observatory has a second downlink channel that operates on X band. The onboard *X band radio* (XBR) is EWC27, manufactured by the company Syrlinks. All mission data packets shall be downloaded to ground through the X band downlink.

The observatory is capable of transmitting mission data packets to ground over both X band and UHF. However, the UHF channel is very limited and its usage for transmission of mission data should be reserved for functional verification in circumstances such as the mission's commissioning phase.

The observatory contains eight elements that shall be mechanically deployed after insertion into orbit. The first two are the VHF and UHF antennas, whose deployment is the responsibility of the ISISpace AntS board. The deployment of the other six is the responsibility of the custom *Monitoring and Control of Opening Mechanisms* (MCOM) board. Four of them, E-field 1, E-field 2, Imped, and Langmuir are sensor probes connected to the SWP and MSM instruments. The final two, SP X+ and SP X-, are solar panel arrays.

The spacecraft contains three main *onboard computers* (OBCs): *Control & Data Handler* (C&DH), *Payload Data Handler* (PLDH), and *Data Storage Unit* (DSU). Each has their particular function in the architecture and interfaces with different components.

PLDH interfaces with IVM, MSM, and SWP for retrieving science data and controlling the payloads' behaviour. Originally, during spacecraft development, CTECS would also interface with PLDH. However, DSU was eventually created due to technical reasons and the engineering team decided to isolate CTECS from PLDH. For situations in which CTECS' PPS is unavailable, PLDH also provides a hardware-generated PPS that serves as a placeholder for time synchronization (although not synchronized to GPS time). PLDH sends the payloads' science data to DSU over time as they are obtained. PLDH also sends ancillary data to C&DH periodically. Some of its behaviour can be controlled by ground operators through TCs that are routed from C&DH.

C&DH manages the spacecraft, orchestrating the work of all other components. C&DH:

- determines and controls the observatory's attitude;
- interfaces with attitude sensors for determination of attitude;
- interfaces with attitude actuators for controlling attitude;
- manages the status of all other components;
- interfaces with VUR for receiving TCs and sending TM;
- decodes TCs;
- is able to execute TCs immediately or schedule TCs for future execution;
- executes TCs that should be executed by itself;
- forwards TCs to PLDH and DSU that should be executed by them;
- receives science data from DSU upon ground demand;
- gathers packets of ancillary data coming from EPS, PLDH, DSU, and itself;
- sends ancillary data packets to DSU;
- sends TM to the ground: beacons, TC responses, mission data (depending on DSU);
- interfaces with EPS for turning components on/off and obtaining power-related EPS telemetry (battery health, power bus voltages, etc); *and*

- interfaces with AntS and MCOM for releasing the eight deployable mechanisms during spacecraft's initialization.

DSU's primary purpose is to store all mission data packets and, when adequate, interface with XBR to transmit them to ground. Also, for technical reasons, DSU is the OBC that interfaces with CTECS for retrieving its science data (rather than PLDH). Therefore, DSU forwards adequate CTECS mission data to C&DH and PLDH as required. DSU sends ancillary data to C&DH periodically. DSU also sends science data packets to C&DH for transmission over UHF upon ground request. Additionally, on an eventual (although improbable) CTECS malfunction, DSU is capable of reprogramming CTECS' firmware with the onboard copy it carries.

3 Operating Modes

NASA is responsible for delivering the observatory to the ISS, where it will be handled appropriately and placed in a mechanical deployer. The deployer will then eject the observatory into a low Earth orbit which will be similar in altitude to that of the ISS. Upon observatory deployment, the inhibition mechanisms that prevent the spacecraft from booting up will be removed and the batteries will power on EPS and C&DH.

The spacecraft runs in a finite-state machine of *operating modes*, as displayed by Figure 2.

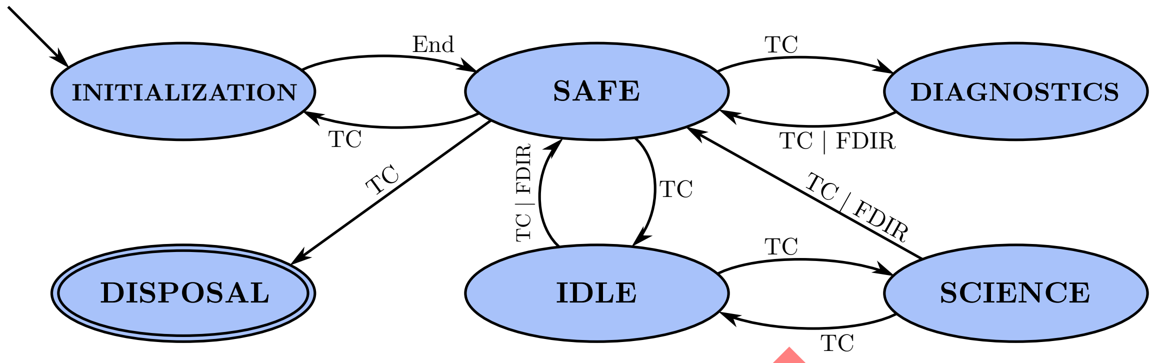


Figure 2: Spacecraft operating modes finite-state machine.

C&DH shall boot up and enter the *INITIALIZATION* operating mode. INITIALIZATION is composed of a routine executed by C&DH that is itself an FSM explained in Appendix A. The routine systematically and automatically prepares the spacecraft for operation, mainly by mechanically deploying eight moving elements. When the initialization routine ends, the spacecraft powers VUR on and switches to *SAFE*.

SAFE is the operating mode where minimal, safe operation occurs. It is accessed automatically after the INITIALIZATION routine ends, but also by TC request from *DIAGNOSTICS*, *IDLE* or *SCIENCE*, or because a fault in the observatory has been observed by the fault detection routines. In SAFE, only EPS, C&DH, and VUR are powered on. VUR sends beacons every 30 seconds. The spacecraft begins accepting TCs.

This is where the mission's commissioning procedures begin. They are not in the scope of this document.

The communication links between ground and observatory are explained and detailed in Section 4 and in the Appendices portion of this document. For now, note that TCs are selectively accepted by the spacecraft depending on the current operating mode. Refer to Appendix D).

During commissioning procedures, should operators determine that some part of the INITIALIZATION routine has been unsuccessful, the routine may be retried. INITIALIZATION may be accessed again from SAFE, by means of a TC. In this case, VUR is turned off and the observatory essentially soft-reboots. The routine in INITIALIZATION is executed again. However, after the routine has been executed the first time, the spacecraft will be aware of steps that have been previously successful and those will not be retried.

Commissioning procedures will require arbitrarily performing actions with components of the observatory (e.g. powering components on/off, enabling specific behaviours of payloads, or manually executing the steps in INITIALIZATION in case of anomalies). Arbitrary commanding of the observatory's behaviour is only possible in the *DIAGNOSTICS* operating mode. DIAGNOSTICS is the operating mode that accepts virtually every TC for purposes of commissioning and fault diagnostics. To switch the observatory from SAFE to DIAGNOSTICS, operators issue the corresponding TC.

Upon every operating mode transition, the spacecraft checks the *power matrix* to determine which components should be powered on in the new operating mode. If a component is on but should be off, it is powered off during the operating mode transition (and vice-versa). Under normal circumstances, EPS and C&DH are the only components that should never be powered off. Special circumstances exist in which this does not hold true (e.g. critical battery levels), but those are the only programatic exceptions to the power matrix.

The power matrix has a default configuration but operators are capable of modifying it over the course of the mission, according to unforeseen circumstances (such as payload malfunction). The only operating modes in which this is not possible are INITIALIZATION, as VUR is off and the spacecraft is soft-rebooting, and *DISPOSAL*, which represents the end of the mission and has strict regulations.

When commissioning procedures are over, and operators deem the observatory ready to commence operations, the observatory shall be switched back to SAFE mode, and subsequently, to IDLE mode. Observe that power matrix rules apply, and as such, PLDH, DSU and CTECS are powered on.

Before initiating scientific data collection, the spacecraft must provide suitable attitude for payload operation. In IDLE, C&DH begins executing the ADCS routines for attitude determination and control. Accurate attitude determination is dependent on CTECS-provided navigation data, hence CTECS is powered on in IDLE but remains in its initial *position, navigation, and time* (PNT) mode. IVM, MSM, and SWP are kept powered off.

When operators determine that adequate attitude parameters have been reached, the observatory shall be switched to SCIENCE mode. In SCIENCE, the remaining payloads are powered on and can be operated for scientific data collection. Payload operation is discussed in Section 6.

The operating mode transitions displayed in Figure 2 can be exercised at any time with specific TCs (observing operating mode acceptance constraints) or because of detected faults. In case a fault is detected, the spacecraft transitions automatically to SAFE. An example is failure in powering any component on or off during the execution of a power matrix rule in an operating mode transition. Fault detection is the subject of Section 9.

When the mission shall come to an end, operators shall switch the observatory to the DISPOSAL operating mode. DISPOSAL is the final operating mode, where all switchable power lines are powered off and the spacecraft ceases to perform any radiofrequency transmissions. The purpose of DISPOSAL is to prevent pollution of the radiofrequency spectrum after the intended mission period, according to International Telecommunication Union (ITU) regulations. The transition to DISPOSAL is irreversible and can only be performed from SAFE. In order to prevent an unwanted transition to DISPOSAL, operators need to perform a series of three steps to engage the transition: first, the transition must be armed, then two different transition TCs must be issued. The transition can be cancelled at any point before the last TC by disarming the transition.

4 Communication Links

All communication between ground and observatory is based on recommended standards from the *Consultative Committee for Space Data Systems* (CCSDS).

4.1 CCSDS Space Packets

Individual messages shall be encapsulated in a format denominated *Space Packet* (SP) [2], which is displayed in Figure 3.

	PACKET PRIMARY HEADER							PACKET DATA FIELD			
	VERSION NO.	PKT ID			PKT SEQ CTRL		PKT DATA LENGTH	SECONDARY HEADER	USER DATA FIELD		
		PKT TYPE	SEC HDR FLAG	APID	SEQ FLAGS	SEQ COUNT			TC ID	DATA	CHECKSUM
Length (bits)	3	1	1	11	2	14	16	0 / 64	0 / 8	Variable	16

Figure 3: Structure of CCSDS Space Packets.

SPs are variable-length structures composed of a *primary header* and a *packet data field*. The length of the primary header is fixed at 6 bytes. Its purpose is to identify the message and to declare the length of the packet data field, which is variable. The packet data field is where the actual message is contained. A message can be a TC (uplink) or TM (downlink).

Fields within SPs are always big endian (MSB first), and the first transmitted bit of every byte is the most significant one.

The first field in the primary header is *version number*. The contents of this field are fixed at 0b000 in all cases.

Next comes the *packet type* flag. This bit is 0 in TM SPs (downlink) and 1 in TC SPs (uplink).

The following bit is the *secondary header flag*. The *secondary header* is an optional field in SPs. A 1 in this flag represents the presence of the secondary header in a given SP. A 0 represents the abscence of the secondary header.

The next 11 bits, fulfilling the first two bytes of the primary header, represent the *application identifier* (APID). APID identifies which component is responsible for the SP's message. In TC SPs, this means which OBC is responsible for executing the TC. This allows C&DH to determine whether the command should be executed locally or routed to PLDH or DSU. In TM SPs, APID identifies which type of TM message is contained within the packet data field (which in turn identifies the component of origin).

The two following bytes compose the *sequence control*. According to the recommended standard [2]:

> "The sequence control field orders the Packets generated by the same APID, even though their order may be disturbed during transport from the origin to the destination, as well as supports the detection of missing packets within each APID."

The *sequence flags* allow both ground and observatory to segment large messages over multiple SPs and reconstruct them accordingly. The possible values of the sequence flags are:

- 0b11: an unsegmented message;
- 0b01: the first segment of a segmented message;
- 0b00: a continuation of a segmented message; *and*
- 0b10: the last segment of a segmented message.

Segmentation is a necessary feature because VUR can only receive uplink frames that are up to 200 bytes in length each and transmit downlink frames that are up to 235 bytes in length each.

In SPORT, however, the meaning of the *sequence count* field has been overloaded and differs from the recommended standard in two ways.

First, in unsegmented SPs, the sequence count is an unsigned integer that informs the global order of generation of SPs by each OBC. The first SP generated by each OBC has count zero and every subsequent one is incremented by one. Note that this differs slightly from the recommended standard, as the standard proposes a counter for each particular APID.

Second, in segmented SPs, the sequence count allows the reconstruction of large SPs by both ground and observatory by containing the order of segments of a segmented SP. Every segment starting from the second (count 1) is incremented sequentially. The first segment's count must be equal to the total count of expected segments (must be greater than 1). This is to allow the observatory to allocate enough memory beforehand in order to buffer the upcoming segments of a large TC.

TODO: segmented TM, what's the count?

The last field in the primary header is the *packet data length*. This field is an unsigned integer that informs how long the data field (rest of the SP) is, because the data field is of variable size. The data field must have at least one byte, therefore the value informed by the data length field is actually the length of the data field minus one. This allows the data field to be 1-65536 bytes in length.

The first field within the packet data field is the secondary header. The secondary header is present in a given SP if the secondary header flag is 1. In SPORT, the secondary header's purpose is to carry a time tag compatible with the format of GPS time. The format of GPS time is displayed in Figure 4.

	GPS TIME	
	WEEK	MS
Length (bits)	32	32
Format	uint32	uint32

Figure 4: Structure of GPS time tags.

GPS time is composed of a 32-bit *week* field and a 32-bit *ms* field. The week field indicates the unsigned integer number of weeks elapsed since the beginning of the current GPS epoch (which started on January 6, 1980). The ms field indicates the unsigned integer number of milliseconds elapsed since the beginning of the current week.

If a TC SP contains no secondary header, its execution shall be immediate. If it does contain the secondary header, the TC shall be scheduled for deferred execution at the time specified by the time tag. TM SPs always contain the secondary header, in order to indicate the moment of generation of the message by the spacecraft. Details on time keeping by the spacecraft are the subject of Section 5.

Next comes the *user data field* (not to be confused with the packet data field, in which it is contained).

If the SP represents a TC, first byte in the user data field is *TC ID*. The TC ID is an unsigned integer that identifies which TC the pertinent OBC should execute. The combination of APID and TC ID uniquely identifies an observatory TC.

The next field within the user data field is *data*. This field is variable in length. Its length is characterized by the length of the packet data field, as declared in the primary header, minus the length of the other fields present within the packet data field. In TCs, the data field can either be empty or contain parameters for the execution of the TC. Details regarding the expected format of each TC can be examined in Appendix D.

The last 16 bits of any SP shall always contain the checksum of the rest of the SP, for the purpose of validity verification. The 16-bit Cyclic Redundancy Check (CRC-16) error-detecting code used in SPORT is based on a standard by the European Cooperation for Space Standardization (ECSS) [5]. The algorithm is known as CRC-16-CCITT-FALSE and is characterized by the polynomial $h(x) = x^{16} + x^{12} + x^5 + 1$, or 0x11021. In this document, we shall henceforth refer to the algorithm simply as CRC-16. The SPORT engineering team shall provide a reference implementation of the algorithm employed.

Due to their variable-length nature, the spacecraft internally implements SPs as dynamically-allocated structures. This means that operators are responsible for monitoring the memory space of OBCs so that services such as TC queueing do not fail due to the lack of memory. Details are discussed in Section 7.

4.2 VHF Uplink

The characteristics of the physical layer of this channel are:

- Frequency: 149.775MHz;
- Modulation scheme: FSK-G3RUH; *and*
- Bitrate: 9600bps.

The VHF channel shall be used exclusively for telecommanding the observatory. All TCs shall be SPs.

4.2.1 Encryption

In order to prevent unauthorized control of the observatory, all TCs shall be encrypted with AES-CFB128 with a symmetric, 256-bit encryption key. The Advanced Encryption Standard (AES) is a widely used set of encryption algorithms.

The SPORT engineering team shall provide a reference implementation of the encryption algorithm using the mbed TLS software library [1].

AES-CFB128 uses the *Cipher Feedback* (CFB) block cipher mode of operation. The CFB mechanism is displayed in Figures 5 and 6.

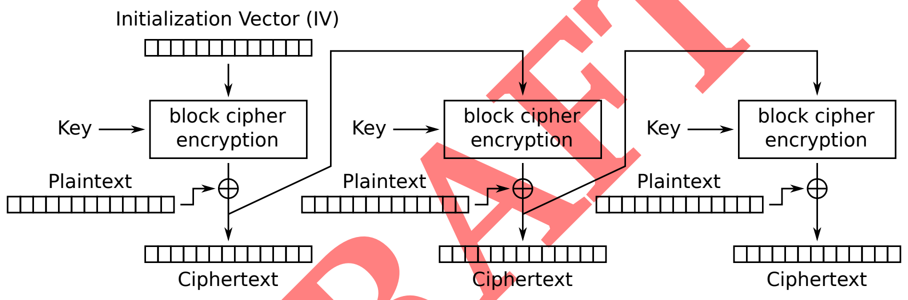


Figure 5: Cipher Feedback block cipher mode of operation (encryption). Adapted from [9].

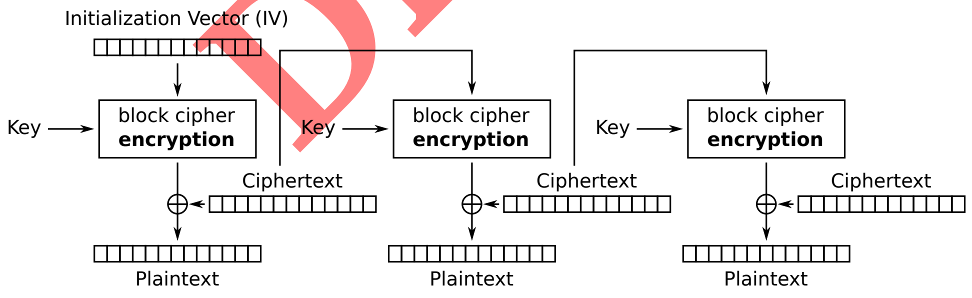


Figure 6: Cipher Feedback block cipher mode of operation (decryption). Adapted from [8].

In CFB, encryption and decryption are performed by the same core algorithm. Every step in CFB requires the output of the previous step as one of its inputs. Because of that, the first step requires a sequence of 128 bits denominated *initialization vector* (IV).

The union of an IV and an uplink SP is named *uplink frame*.

The IV must be random and unique for each uplink frame, otherwise the encryption effort may be broken by third parties even without the encryption key.

Uplink frames shall be encapsulated as the payload of AX.25 frames [7]. The handling of AX.25 frames is performed in hardware by VUR and transparent to C&DH, therefore not in the scope of this document. Figure 7 illustrates the high-level structure of uplink AX.25 frames.

	AX.25 FRAME		
	AX.25 HEADER	UPLINK FRAME	
		INITIALIZATION VECTOR	TC OR TC SEGMENT SP
Length (bits)	Not in scope	128	72-1472

Figure 7: High-level structure of an AX.25 uplink frame.

4.2.2 IV Uniqueness

Even with uplink encryption, the observatory may still be prone to a type of attacks called *replay attacks*. In replay attacks, a third party may be listening to and recording communications from ground. Even if they do not have access to the encryption key and do not even know what the encrypted TC is or what it does, they may still be able to replay the TC when the observatory is doing an overpass.

To prevent the observatory from accepting and executing the replayed TC, a countermeasure is in place that harnesses the use of random and unique IVs.

Every valid uplink frame has their IV hashed and kept in a mechanism with a probabilistic data structure. The idea is that if an uplink frame's IV has been seen recently, the frame is dropped silently. Not responding to the frame is a measure to discourage replay attacks.

The mechanism is an efficient way, in terms of computation, to reject replayed uplink frames. Its efficacy, however, is a tradeoff between the amount of resources allocated to the data structures and the amount of false positives that it causes. False positives cause a small portion of new uplink frames to be dropped. However, there are no false negatives, therefore no replayed frames among the most recent are accepted.

The parameters of the mechanism are fixed. It keeps track of the most recent 256 uplink frames (TBC). No replayed frames among the 256 most recent are accepted, but because of false positives, 1 every 256 new uplink frames are dropped on average.

Frames that are older than the most recent 256 can also be accepted, although frames scheduled for past times would be dropped anyway (an exception is discussed in Section 5). Because of that, we restrict most TCs so that they are only accepted if scheduled. TODO: to be implemented.

4.2.3 Encryption Key Exchange

SPORT's uplink encryption uses a symmetric key, which means that both ground and observatory shall possess the same key. Unlike asymmetric key pairs, symmetric keys are assumed to be kept entirely secret from third parties, thus excluding the need for extra authentication mechanisms.

The key in use by the observatory at a certain moment is called the *acting key*. For the case in which the acting key is deemed compromised, the observatory allows the on-orbit secure exchange of a new key with ground.

The key exchange mechanism is based on the Elliptic-Curve Diffie-Hellman protocol with ephemeral keys (ECDHE) and elliptical curve Curve25519. This protocol is designed to allow two parties to agree on a secret shared key over an unreliable medium, preventing third parties from discovering the secret key in the process.

At a high level, ECDHE's general steps are as follows:

1. Ground generates random 256-bit secret integer d_A.
2. Ground derives 256-bit public integer Q_A from d_A.
3. Ground transmits Q_A to spacecraft. Spacecraft now knows Q_A.
4. Spacecraft generates random 256-bit secret integer d_B.
5. Spacecraft derives 256-bit public integer Q_B from d_B.
6. Spacecraft transmits Q_B to ground. Ground now knows Q_B.
7. Spacecraft derives secret candidate key K from d_B and Q_A.
8. Ground derives same secret candidate key K from d_A and Q_B.

At this point, both sides agree on new key K, but the old key is still the acting key. Note that all this interchange follows the uplink rules, including encryption with the acting key.

In order to effectively switch keys, ground must send a TC to *commit* key K. After the execution of such TC, the candidate key becomes the new acting key and the old acting key is discarded. Uplink communications must now be encrypted with key K.

Alternatively, ground may choose to send a TC to discard the candidate key, and the old key remains as the acting key.

The SPORT engineering team shall provide a reference implementation of the key exchange mechanism using the mbed TLS software library [1].

Generation of the mission's first acting key shall be performed by a joint ITA/INPE commission pre-launch. The generation will use the same aforementioned mechanism.

Operators are also encouraged to periodically change the acting key in order to improve mission safety in general (akin to changing the password of a system with authentication). **The on-orbit key exchange must be perfomed with maximum caution, because there is NO RECOVERY MECHANISM in case the acting key is lost by ground**.

4.2.4 Uplink Handling (TC Enqueueing, Scheduling, Assembly)

The spacecraft periodically attempts to acquire and execute new TCs. It performs several steps to handle various possible situations in the uplink communication with ground. Figures 8 through 10 display how C&DH handles uplink frames and enqueue valid TCs for execution. The tool used to automatically generate the graphs sometimes does a poor job in organizing edges and labels, therefore we recommend zooming in for better visualization. All three Figures compose a single flowchart, whose representation has been split in order to reduce the complexity of representation. TODO: legend with shapes

Once per second, C&DH first attempts to handle an outstanding TC that is ready for execution, and in the absence of one, it then attempts to acquire a new uplink frame to handle.

Let us focus first on acquiring new uplink frames. C&DH queries VUR for a new uplink frame, and in case there is one, it performs a series of checks in order to determine if the uplink frame is valid.

A valid uplink frame is one that allows for successful decryption and that the spacecraft is able to identify as coming from the rightful operators. Thus it:

- contains a unique, random 16-byte IV;
- is at least 25 bytes long, IV included;
- has been encrypted with the acting encryption key;
- contains a consistent SP header (e.g. coherent sequence info, packet type flag = 1); *and*
- contains the correct CRC-16 checksum.

Nominally, the spacecraft shall respond to every valid uplink frame with at least one *standard response* indicating success or failure in handling the frame. The structure of the standard response is displayed in Appendix D.1. The standard response format, `TM_STD_001`, contains the TC's APID and ID and an error code that indicates either success in handling the SP or which failure occurred. In the Figures, the standard response of a TC is generated and transmitted to ground whenever the control flow reaches a parallelogram (slanted rectangle).

The only way for the spacecraft to determine whether the uplink frame comes from the rightful operators is by analyzing its structure after decryption, hence the header and CRC requirements. In other words, a frame that has been encrypted with the acting key but which makes no discernible sense is regarded by the spacecraft as invalid and a standard response is not generated. This measure intends to discourage third parties from attempting to communicate with the observatory.

After determining the newly-acquired uplink frame is valid, C&DH needs to decide what it should do with the frame's SP. First, unless the SP refers to a DSU TC or TC segment, the SP is sent to DSU for storage. The reason for storage is that communications from ground are operational data (which in turn are mission data). DSU SPs are not forwarded to DSU at this point because DSU TCs are eventually forwarded at the moment of execution anyway.

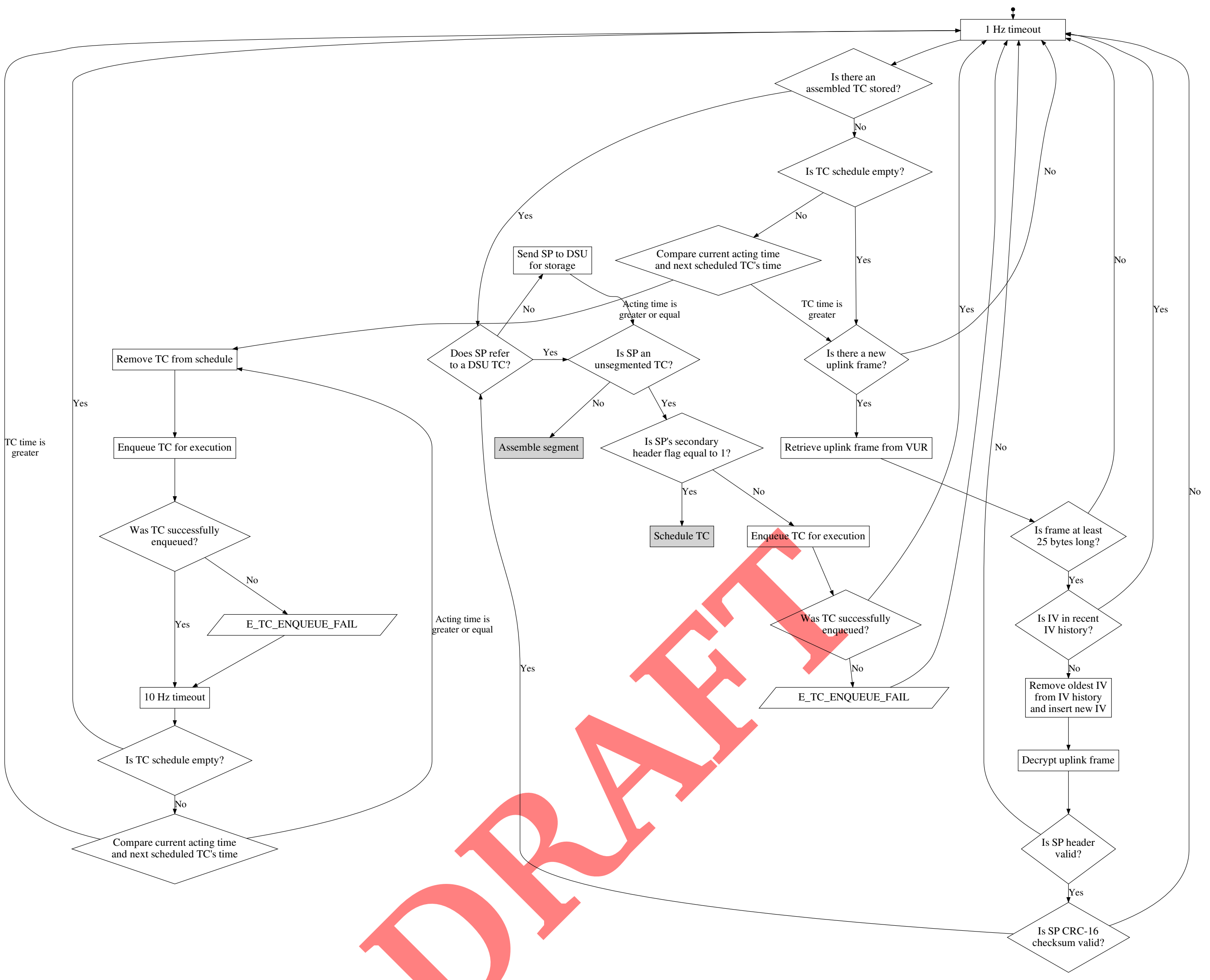

1 Hz timeout
Is there an assembled TC stored?
No
Is TC schedule empty?
No
Yes
Compare current acting time and next scheduled TC's time
Send SP to DSU for storage
Yes
Acting time is greater or equal
TC time is greater
No
Remove TC from schedule
Does SP refer to a DSU TC?
Yes
Is SP an unsegmented TC?
Is there a new uplink frame?
Yes
No
Yes
No
Yes
Assemble segment
Is SP's secondary header flag equal to 1?
Retrieve uplink frame from VUR
No
TC time is greater
Enqueue TC for execution
Yes
No
Was TC successfully enqueued?
Schedule TC
Enqueue TC for execution
Is frame at least 25 bytes long?
No
Yes
E_TC_ENQUEUE_FAIL
Acting time is greater or equal
Was TC successfully enqueued?
Is IV in recent IV history?
Yes
No
10 Hz timeout
E_TC_ENQUEUE_FAIL
Remove oldest IV from IV history and insert new IV
Is TC schedule empty?
Decrypt uplink frame
No
Compare current acting time and next scheduled TC's time
Is SP header valid?
Yes
Is SP CRC-16 checksum valid?
No

Figure 8: C&DH's flow of uplink frame handling (main).

Next, if the SP is a TC segment, C&DH attempts to incorporate it into the assembly of the larger TC. This will be discussed shortly.

Otherwise, if the SP presents a secondary header, that means the TC should be scheduled for deferred execution. This will also be discussed shortly.

Finally, if the SP falls under none of these cases, it is an unsegmented TC that should be executed immediately. C&DH then enqueues it for execution. Execution happens asynchronously in an independent software thread. The execution queue is an internal data structure that serializes the communication between threads in the onboard software. Note that C&DH may fail to enqueue the TC, for reasons such as the queue being full. In this case, a standard response is transmitted to ground comunicating the error (`E_TC_ENQUEUE_FAIL`) and the TC is discarded.

Scheduling

The decision process on scheduling an unsegmented TC for deferred execution is displayed in Figure 9. Here, C&DH attempts to include the TC in the *TC schedule*, which is a special ordered queue

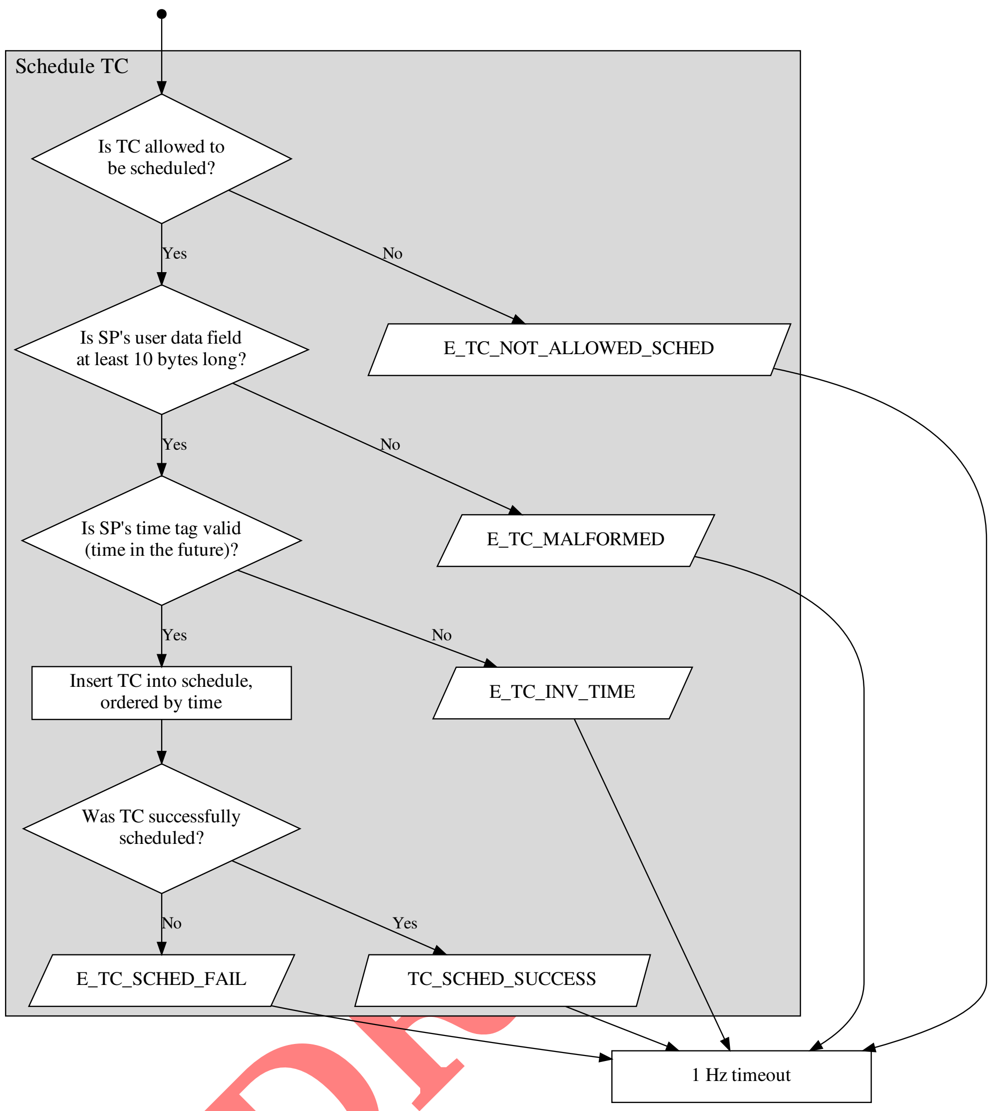

Schedule TC
Is TC allowed to be scheduled?
Yes
No
Is SP's user data field at least 10 bytes long?
E_TC_NOT_ALLOWED_SCHED
Yes
No
Is SP's time tag valid (time in the future)?
E_TC_MALFORMED
Yes
No
Insert TC into schedule, ordered by time
E_TC_INV_TIME
Was TC successfully scheduled?
No
Yes
E_TC_SCHED_FAIL
TC_SCHED_SUCCESS
1 Hz timeout

Figure 9: C&DH's flow of uplink frame handling (scheduling).

which is indexed based on time information. The time at which the TC must be executed is represented in the SP's secondary header, in the GPS time format displayed by Figure 4.

Not all TCs are allowed to be scheduled. Refer to Appendix D.2 for the listing of C&DH TCs, a subset of which cannot be scheduled.

If the SP's time tag is not a time in the future, it is not valid and thus the TC fails to be scheduled. In order to determine if the SP's time tag indeed refers to a time in the future, C&DH compares is to its *acting time*. Refer to Section 5 for details on time keeping.

If the time is in the future, C&DH attempts to include the TC in the TC schedule. The TC schedule is a dynamically-allocated queue, holding a variable quantity of TCs depending on memory availability. Thus, inclusion of a TC in the TC schedule is prone to failure.

In case of success, the TC is included in the proper position in the queue, ordered by time. Success in the inclusion causes the generation of a standard response with code `TC_SCHED_SUCCESS`.

The scheduled TC that should be executed the soonest is the head of the queue (the first element).

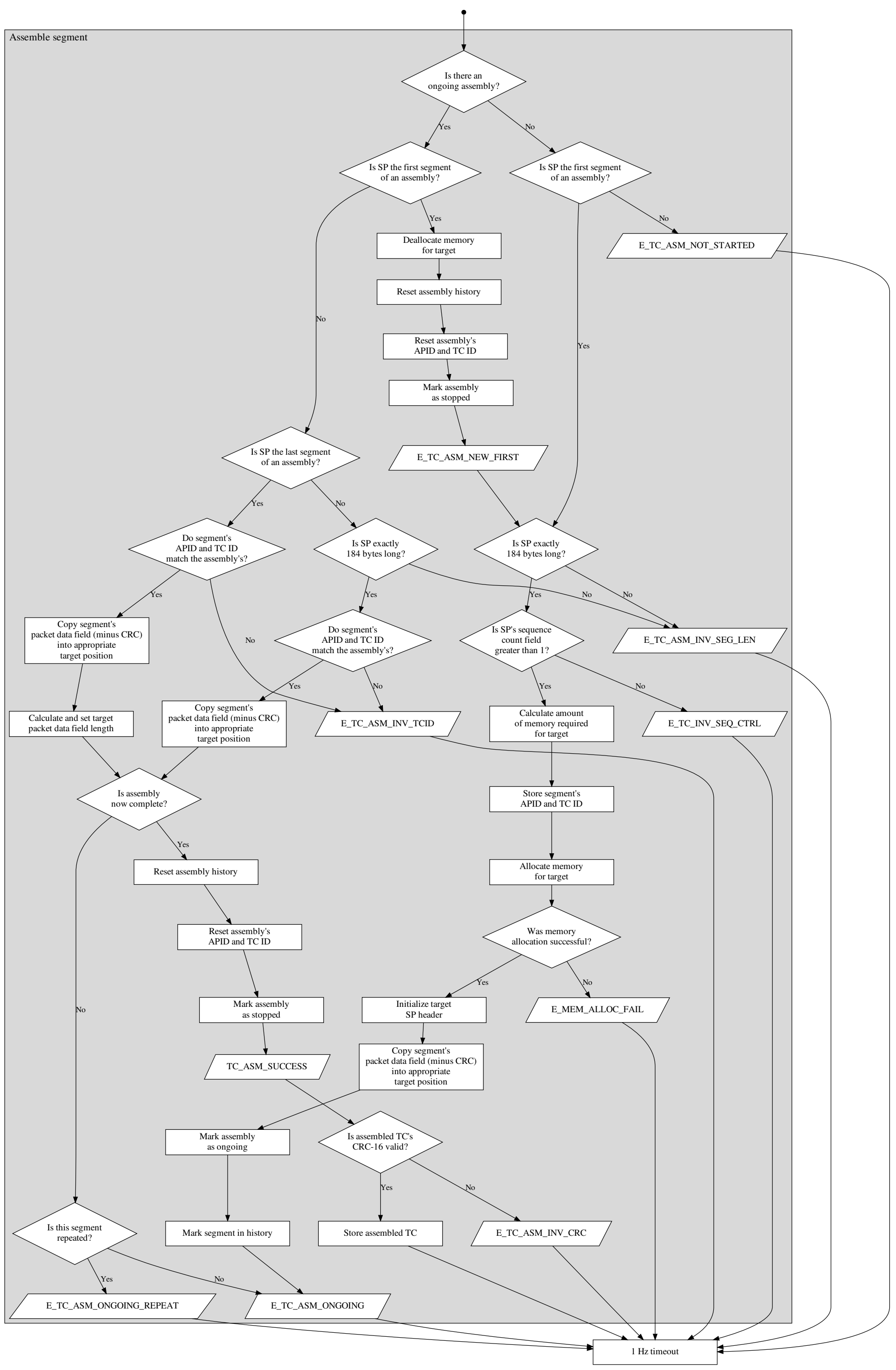

Assemble segment
Is there an ongoing assembly?
Yes
No
Is SP the first segment of an assembly?
Is SP the first segment of an assembly?
Yes
No
No
Yes
Deallocate memory for target
E_TC_ASM_NOT_STARTED
Reset assembly history
Reset assembly's APID and TC ID
Mark assembly as stopped
E_TC_ASM_NEW_FIRST
Is SP the last segment of an assembly?
Yes
No
Do segment's APID and TC ID match the assembly's?
Is SP exactly 184 bytes long?
Is SP exactly 184 bytes long?
Yes
No
Yes
No
Yes
No
Copy segment's packet data field (minus CRC) into appropriate target position
Do segment's APID and TC ID match the assembly's?
Is SP's sequence count field greater than 1?
E_TC_ASM_INV_SEG_LEN
Yes
No
Yes
No
Calculate and set target packet data field length
Copy segment's packet data field (minus CRC) into appropriate target position
E_TC_ASM_INV_TCID
Calculate amount of memory required for target
E_TC_INV_SEQ_CTRL
Is assembly now complete?
Store segment's APID and TC ID
Yes
Reset assembly history
Allocate memory for target
Reset assembly's APID and TC ID
Was memory allocation successful?
Yes
No
No
Mark assembly as stopped
Initialize target SP header
E_MEM_ALLOC_FAIL
TC_ASM_SUCCESS
Copy segment's packet data field (minus CRC) into appropriate target position
Mark assembly as ongoing
Is assembled TC's CRC-16 valid?
Yes
No
Is this segment repeated?
Mark segment in history
Store assembled TC
E_TC_ASM_INV_CRC
Yes
No
E_TC_ASM_ONGOING_REPEAT
E_TC_ASM_ONGOING
1 Hz timeout

Figure 10: C&DH's flow of uplink frame handling (assembly).

The TC schedule can be manipulated by operators via specific TCs. This includes querying the state of the schedule, removing one particular element, and resetting the queue. No two TCs with the exact same time tag are allowed in the TC schedule.

Referring back to Figure8, one of the first decisions in the handling of uplink frames is if the TC schedule is empty. If there is at least one TC in the schedule, before attempting to acquire a new frame from VUR, C&DH instead compares the time tag of the head of the schedule to its acting time. If the acting time is greater than or equal to the time specified by the TC, it is time for the TC to be executed. C&DH then diverts its control flow to a different path, in which no new uplink frames are acquired and TCs are enqueued for execution from the schedule.

One at a time, TCs are removed from the schedule and enqueued for execution. Here, instead of the usual 1 Hz timeout of the common case, TCs are executed at ten times the frequency. In each iteration, the next head of the schedule is checked for its turn to be executed. The regular flow is only resumed when either the schedule is empty or it is not yet time to execute the head of the schedule.

Assembly

Refer again to Figure8. After determining that an uplink frame is valid, C&DH may determine that its SP is a TC segment rather than an unsegmented TC. The assembly of segments is displayed in Figure 10.

The first step in processing a segment is checking whether an assembly is already in progress, because only one assembly is allowed at a time. Let us first explore the case in which no assembly is ongoing. In this case, only a first segment - a segment with sequence flags 0b01 - is accepted.

All uplink frames but the last in an assembly are expected to be exactly 200 bytes long (VUR's maximum uplink frame length). Given that each uplink frame is self-contained and encrypted, it must also contain an IV. Thus, all SPs of an assembly but the last one must be exactly 184 bytes long.

As described in section 4.1, a first segment's sequence count must contain the number of total segments in the assembly (first segment included), so that C&DH is capable of allocating the required memory.

If allocation goes successfully, then the target TC's SP header is initialized (with APID, flags, etc).

The assembly is marked as ongoing and the first segment is marked in the *assembly history*. The assembly history is a bitfield that keeps track of which segments have been received. This allows the spacecraft to receive the segments of an assembly out-of-order. Operators are be able to query the spacecraft for the ongoing assembly's history and retransmit any lost segments. Operators are also able to reset the assembly via a specific TC.

Success in starting the assembly is followed by a standard response with code `E_TC_ASM_ONGOING`. Failure in any of these steps causes the assembly not to be started and the first segment to be discarded.

If an assembly is indeed ongoing in the first step of processing the segment, then a new first segment causes the ongoing assembly to be cancelled and a standard response with code `E_TC_ASM_NEW_FIRST` to be transmitted. The same segment is immediately processed as if no assembly was ever ongoing. A second standard response is generated for the same segment and a new assembly is potentially started.

In processing continuation segments, C&DH copies the segment's packet data field (minus CRC) into the appropriate position in the target's packet data field. Processing an assembly's last segment is the same, except the precise packet data field length is calculated and set into the target.

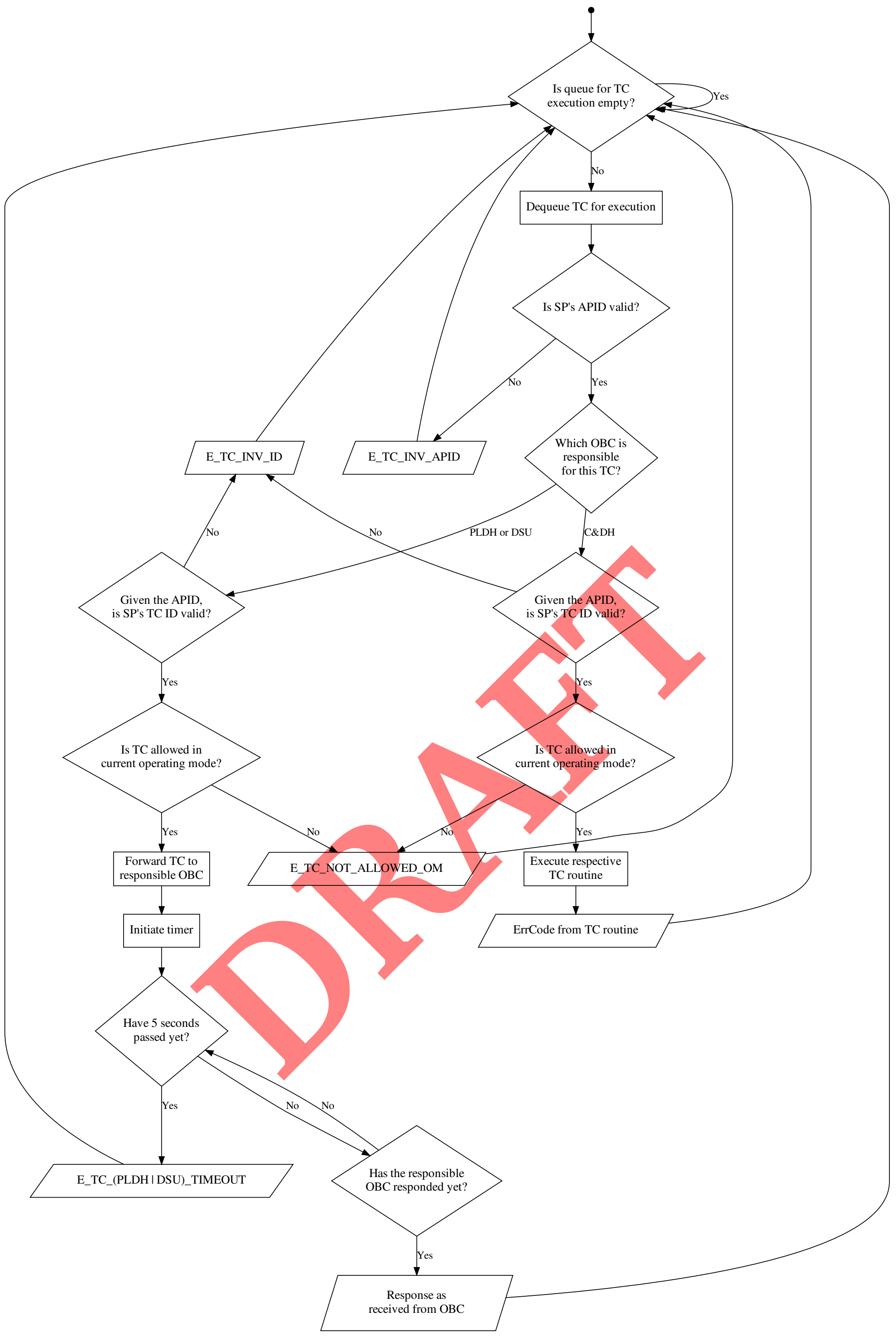
Because the assembly may be performed out-of-order, any continuation segments or the last segment of an assembly may cause the assembly to be completed. In this case, the assembly is marked as stopped, the history is reset, a standard response with code `TC_ASM_SUCCESS` is generated. The assembled, unsegmented TC's CRC-16 is checked and the TC is stored for processing in the next iteration of the uplink handling routine. In the next iteration of the routine, the TC is effectively handled as a regular unsegmented TC, being able to be scheduled in case it contains a secondary header.

While an assembly is ongoing, unsegmented TCs can be received and executed freely with no detriment to the assembly.

TODO: segmented TC figure.

TC Execution

Figure 11 displays the routine of the concurrent thread that executes enqueued TCs.



Is queue for TC execution empty?
Yes
No
Dequeue TC for execution
Is SP's APID valid?
No
Yes
E_TC_INV_ID
E_TC_INV_APID
Which OBC is responsible for this TC?
No
No
PLDH or DSU
C&DH
Given the APID, is SP's TC ID valid?
Given the APID, is SP's TC ID valid?
Yes
Yes
Is TC allowed in current operating mode?
Is TC allowed in current operating mode?
Yes
No
No
Yes
Forward TC to responsible OBC
E_TC_NOT_ALLOWED_OM
Execute respective TC routine
Initiate timer
ErrCode from TC routine
Have 5 seconds passed yet?
Yes
No
No
E_TC_(PLDH | DSU)_TIMEOUT
Has the responsible OBC responded yet?
Yes
Response as received from OBC

Figure 11: C&DH's flow of TC execution and routing.

The thread waits for an element to be enqueued on the execution queue by the uplink handler. When that happens, the executer dequeues it and first analyzes the TC's APID. The APID determines which OBC is responsible for the TC. Then, given the APID, C&DH determines if the TC ID is valid.

If the combination of APID and TC ID is indeed valid, then C&DH compares it against the operating mode *acceptance matrix*. The acceptance matrix is the specification of which TCs are allowed under which operating modes. A TC may be otherwise entirely correct, but not allowed to execute under the current operating mode. In this case, the TC is dropped and a standard response with code `E_TC_NOT_ALLOWED_OM` is generated.

In case it is allowed, C&DH's behaviour varies whether the TC falls under its responsibility or under any of PLDH's or DSU's. If it is a C&DH TC, C&DH simply executes the TC's respective routine and replies to ground with the error/success code generated by the TC's routine.

If it is a PLDH TC or DSU TC, C&DH forwards it to the respective OBC as-is and initiates a timer. The respective OBC then executes the TC and generates the standard response. The standard response is sent from the OBC to C&DH, which transmits it to ground as-is. Should the OBC not answer to C&DH with a response in up to 5 seconds, C&DH considers the TC execution a failure and generates itself the standar response with either code `E_TC_PLDH_TIMEOUT` or code `E_TC_DSU_TIMEOUT`.

Extensive listings and descriptions of every TC's semantics and the operating mode acceptance matrix can be found throughout Appendix D.

Standard and Specific Responses

As previously stated, every valid uplink frame shall be answered by the spacecraft with at least one standard response. Some TCs, such as those that request mission data samples, also cause a second, *specific response* besides the standard uplink response(s). The specific response shall be transmitted before the standard response, because the standard response may explain the reason of any experienced failures in the process of TC execution, including the generation and/or transmission of the specific response.

Unlike the standard response, the specific response may be of various formats and arbitrarily large. Therefore, it may be segmented for reassembly by ground following the same conventions used in the uplink.

If a TC segment successfully completes an assembly, a standard response is generated. If the resulting unsegmented TCs is scheduled for deferred execution, it causes the generation of another two standard response: one at scheduling time and one at execution time. Finally, depending on the TC, the specific response may be arbitrarily large. This results in multiple responses for a same valid uplink frame.

Descriptions of the standard response format, all possible error/success codes, and the format of every other possible response can also be found in Appendix D.

4.3 UHF Downlink

UHF is the main downlink channel for ground operators. This is the downlink channel used by the observatory to respond to uplink communications.

The characteristics of the physical layer of this channel are:

- Frequency: 400.86MHz;
- Modulation scheme: BPSK-G3RUH; *and*
- Bitrate: 9600bps.

Just like with the VHF uplink, all UHF downlink messages are AX.25 frames containing SPs. Unlike the VHF uplink, UHF comms are not encrypted, thus the AX.25 frames do not contain an IV. Henceforth in this document, we disregard the existence of AX.25, so all mentions to VHF/UHF comms are in terms of uplink frames and/or SPs.

All UHF TM SPs that are smaller than or equal to 235 bytes in total length shall be sent as single, unsegmented SPs. UHF TM SPs that are larger than 235 bytes in total length shall be segmented according to the conventions established in Sections 4.1 and 4.2 and shall be reassembled by ground.

C&DH has two telemetry modes: *discrete* and *continuous*. Discrete is the standard spacecraft mode, in which C&DH sends telemetry on UHF only upon contact from ground and at the moment of TC execution. For increased reliability in the UHF downlink, three copies of each downlink message are sent successively in discrete mode. TODO: VUR is janky, number and/or strategy TBC.

To satisfy a specific need from the attitude control operations team, continuous mode can be temporarily engaged. To do so, operators send a specific TC requesting the spacecraft to generate ancillary data packets of a particular format (or more) at a much higher frequency and send them continuously on UHF for a certain period. The result is that the team will be able to obtain much denser data samples to analyze and determine the appropriate atittude maneuvers that need to be executed by the spacecraft.

In continuous mode, packets of the specified type are sent only once each on UHF. Any other packets are sent three times as usual.

Continuous mode disengages automatically after the amount of time specified by the engagement TC or after a predetermined global maximum - which is derived from the longest Brazil overpass as shown by the results of the engineering team's orbital analyses.

4.4 CCSDS Transfer Frames

For reliable transmission of mission data to ground over the high-speed X band downlink channel, all scientific, ancillary, and operational data SPs gathered by the spacecraft are further encapsulated in a format denominated *Transfer Frame* (TF) [3], which is displayed in Figure 12.

	TF PRIMARY HEADER											TF TRAILER
	GLOBAL VIRTUAL CHANNEL ID					DATA FIELD STATUS						
	MASTER CHANNEL ID											
	VERSION NO.	S/C ID	VIRTUAL CHANNEL ID	OCF FLAG	GLOBAL VIRTUAL CHANNEL FRAME COUNT	SEC HDR FLAG	SYNC FLAG	PKT ORDER FLAG	SEGMENT LEN ID	FIRST HDR PTR	TF DATA FIELD	FRAME ERROR CONTROL FIELD
Length (bits)	2	10	3	1	16	1	1	1	2	11	16320	16
Contents	0	0	0	0	Sequential frame count %65536	0	0	0	0b11	Position (byte) of first SP hdr in Data Field	Space Packets	CRC-16

Figure 12: Structure of CCSDS Transfer Frames in SPORT.

TFs are fixed-length structures composed of a *primary header*, a *data field*, and a *trailer*. The length of the primary header is fixed at 6 bytes, and as with SPs, the primary header's function is to identify the message. However, unlike in SPs, the data field of TFs is of fixed length. In SPORT, such length is 2040 bytes. The data field shall contain a continuous sequence of mission data SPs generated by SPORT. The TF trailer contains solely the error control field. The error control field contains the CRC-16 checksum of the rest of the TF. This makes TFs a round 2048 bytes long.

As with SPs, fields within TFs are big endian (MSB first), and the first transmitted bit of every byte is the most significant one.

The first field in the primary header is *version number*. The contents of this field are fixed at 0b00 in all cases.

Next comes the *spacecraft ID* field, which is also fixed at 0x000 in all cases. The union of version number and spacecraft ID is denominated the *master channel ID* (MCID).

The contents of the *virtual channel ID* (VCID) field are also 0b000 in all cases. The union of master channel ID and virtual channel ID is denominated the *global virtual channel ID* (GVCID).

In accordance with CCSDS standards, a downlink channel may be shared by several missions and also by different applications in the same mission. TFs identified by the same MCID belong to the same mission, while TFs identified by the same GVCID belong to the same application in the same mission. In SPORT, there is only one virtual channel, therefore all TFs contain the same GVCID - which is composed entirely of zeros.

The next field is the *OCF flag*. It indicates the presence or absence of another field in the TF. Such field is never present in SPORT, therefore the flag is always 0.

In CCSDS standards, the next two fields would be *master channel frame count* and *virtual channel frame count*. Both would contain sequential counts of frames with the same MCID and VCID, respectively. The length of each field would be 8 bits, therefore the frame counts would be unsigned integers modulo 256 (which is 2^8). In SPORT, because there is only one GVCID, we simplify the frame count by merging the two fields into a single, 16-bit-wide *global virtual channel frame count*. The frame count is the sequential number of the frame as generated by SPORT, modulo 65536 (which is 2^{16}).

The last five fields in the primary header refer to the status of the frame's data field. In accordance with CCSDS standards, the contents of the first four are always 0b00011 in SPORT. The last, *first header pointer*, contains an unsigned integer that points to the position of the first byte of the header of the first SP contained in the data field (zero-indexed). If no SP starts in the data field, the first header pointer shall be set to 0b11111111111.

The data field shall contain the entire contents of mission data SPs gathered by SPORT. SPs are written sequentially in the data field in the order they are obtained by DSU, which is the OBC that encapsulates SPs into TFs. An SP may be entirely contained in a TF's data field, be split between two TFs, or even span multiple TFs. CCSDS standards predict the filling of "remaining space" in a TF's data field, but in SPORT, such provision is never utilized because all TFs compose the same continuous sequence of SPs.

Finally, the *TF trailer* contains the *frame error control field*, which contains the CRC-16 checksum of the rest of the TF.

4.5 X Band Downlink and Mission Data Retrieval

Due to the large amounts of mission data generated daily by the observatory, a second downlink channel of higher bandwidth exists which is the main channel for mission data downlink. This second downlink channel is on X band.

The characteristics of the physical layer of this channel are:

- Frequency: 8040.5MHz, 8052.0MHz, or 8083.5MHz (TBD);
- Bitrate: 20Mbit/s;
- Convolutional $7\frac{1}{2}$ Error Correction Code;
- Offset Quadrature Phase-Shift Keying (OQPSK) modulation; *and*
- 6th order Butterworth BT=0.5 filtering according to ECSS-E-ST-50-O5C (Nyquist filtering).

The process of downlink of SPORT mission data over X band is named *Mission Data Retrieval* (MDR).

Preparations for MDR happen at all times DSU is powered on. Upon acquisition of mission data originating from C&DH, PLDH, CTECS, or ground, DSU writes the SP's content in the data field of the TF that is currently being assembled.

This task is performed for every new acquired SP. Every time a TF is completely filled, it undergoes a process of preparation for persistent storage.

The first step in this process is *pseudo-randomizing* the entire TF [4]. Pseudo-randomizing is a method to ensure sufficient bit transition density in TFs for better channel reliability at high transfer rates. In the process of pseudo-randomizing, each byte of the TF is bitwise-XOR'ed with the corresponding byte (modulo 255) from a standard sequence. The standard sequence is generated by the polynomial $h(x) = x^8 + x^7 + x^5 + x^3 + 1$, or 0x1a9, with the sequence's first byte initialized to 0xff.

The ground must de-pseudo-randomize all pseudo-randomized TFs by applying the very same algorithm. The SPORT engineering team shall provide a reference implementation of the pseudo-randomizing algorithm.

The pseudo-randomized TF is then incorporated into a structure called *MDR unit*, which is represented in Figure 13.

	MDR UNIT	
	ATTACHED SYNC MARKER	PSEUDO-RANDOMIZED TF
Length (bits)	32	16384

Figure 13: Structure of SPORT's MDR unit.

The MDR unit is composed of a copy of the *Attached Sync Marker* (ASM) and a pseudo-randomized TF.

The ASM is a fixed sequence of four bytes used to synchronize the reception of data by ground and indicate where a new pseudo-randomized TF begins. The sequence is 0x1acffc1d (big endian).

Every time DSU fulfills a new MDR unit, the unit is stored persistently in a non-volatile medium, sequentially, in a circular file. The storage is addressable in terms of MDR units, with the first MDR unit being indexed as the zeroth unit. The onboard storage capacity is finite, fixed at approximately 29 GiB (nominally a 32GB microSD card). The max amount of stored MDR units at one time is a function of the length of a TF, and is precisely 15,174,713 units.

Before commanding the MDR process, operators may analyze the spacecraft's ancillary data to determine the status of DSU mission data storage. The status of storage is reported simply as the index of the most-recently-generated MDR unit in the circular file. The spacecraft only keeps track of what MDR unit is the latest, and always considers the mission data file as full. It is the ground's responsibility to keep track of what MDR units have been recently retrieved, and request the next desired range of MDR units.

The MDR process is initiated by ground operators with TCs. As with most other TCs, such TCs can be scheduled for timed execution.

The process takes two steps, preparation and transmission. The MDR preparation TCs cause DSU to read a range of MDR units from storage and keep a copy of them in memory. The MDR initiation TCs cause DSU to initiate the MDR transmission by interfacing with XBR and consuming the previously-prepared range of MDR units.

The preparation step does not remove any MDR units from the non-volatile storage. The mission data file is only written to when a new MDR unit is generated and stored. Therefore, apart from the very first time the file is filled with valid data, the MDR unit that follows the latest one in the circular file is actually the oldest one.

Likewise, the in-memory copy of MDR units is only written to on the execution of a new preparation TC. This means that the prepared MDR units remain ready for retransmission until the next preparation step takes place, although an MDR transmission does not require a new preparation step (unless DSU is coming from a fresh start). Actually, if a new preparation step overwrites fewer in-memory MDR units than had been previously prepared, the trailing units linger in memory and could theoretically be transmitted again alongside the new units.

The MDR preparation step is specified by a base MDR unit index and a number of MDR units. As an example, supppose the base index is 7,000 and the number of MDR units is 134,500. This means a copy of MDR units 7,000 through 141,499 will be made in DSU's memory, in preparation for transmission. Should the defined range overflow the highest possible index of stored MDR units, the buffer continues to be filled from index zero (thus characterizing the circular storage).

The length of an MDR transmission is limited by the length of the in-memory buffer used to contain the specified range of MDR units. In order to maximize transmission length and better utilize the (rather narrow) overpass times, a large buffer is desirable.

Due to technical reasons, DSU presents two (TODO: maybe more?) distinct MDR buffers named MDR1 and MDR2. MDR1 is TODO: number of MDR units long, amounting to approximately 512MiB. At XBR's transmission rate of 20Mbit/s, MDR1 transmissions may take up to approximately 215 consecutive seconds. MDR2 is TODO: number of MDR units long, amounting to approximately TODO. At XBR's transmission rate of 20Mbit/s, MDR2 transmissions may take up to approximately TODO consecutive seconds.

Each of MDR1 and MDR2 present their specific preparation and initiation TCs. MDR preparation and initiation TCs with a number of MDR units greater than the length of the respective buffer will fail as malformed TCs.

Operators can schedule MDR initiation TCs back-to-back in order to maximize overpass transmission time. Operators should keep in mind that the preparation process is not instantaneous, at least for large enough transfers. In fact, the separation between preparation and transmission has been designed to allow for operators to schedule the preparation throughout the orbit and have the data ready in memory for transmission on overpass. TODO: time estimates.

Operators are encouraged to plan MDR transfers such that the first MDR units in a new transfer overlap with the last ones in the previous transfer (in other words, so that a subrange of MDR units is retransmitted), to minimize data losses at the beginning and end of transfers/overpasses.

We estimate that DSU's non-volatile storage should be able to hold at least a week's worth of mission data, thus providing operators with plenty of room for planning MDR transfers weekly.

5 Time Keeping

An important responsibility of SPORT's spacecraft is keeping track of time for precise determination of the moment of generation of all mission data packets. This is accomplished by several mechanisms that are described in this section.

First, all time information is kept in the GPS time format displayed by Figure 4. Recapitulating: the format consists of two 32-bit unsigned integers that represent the week in the current GPS era and the number of milliseconds elapsed since the beginning of the week.

Each of C&DH, PLDH, and DSU keep their wall-clock time. This time is called the *free-running clock*. It is a measurement of the time elapsed since the CPU last booted. All three free-running clocks are naturally distinct from one another, as the OBCs boot in distinct moments.

The purposes of the free-running clock are twofold. First, it serves as the main time-keeping mechanism before the spacecraft acquires a valid GPS time tag. Second, it helps with correlating time-keeping drift in each OBC for more precise analysis of scientific data.

The spacecraft also keeps track of the real-world time as provided by the GPS constellation. This is possible through CTECS. When it is powered on, CTECS provides time information to the spacecraft in its TIMESYNC data log, once per second.

CTECS also provides the PPS, which is distributed to all OBCs and payloads by the spacecraft. The PPS is a pulse that synchronizes GPS time among components. The PPS is issued every second, in alignment with the round second. In other words, the PPS represents the precise moment of the start of a new second. The exact GPS time associated with a certain PPS is the time indicated by the next TIMESYNC log.

Because CTECS interfaces exclusively with DSU for its science output, GPS time info has to be relayed by the spacecraft to C&DH, PLDH, and the other payloads. Because of this, there is no one "central" GPS time in the spacecraft. Rather, the spacecraft's best effort in distributing this information across components results in each component having their view of the GPS time.

C&DH, PLDH, and DSU ancillary data shall contain each OBC's both free-running clock and GPS time view.

Each OBC's GPS time view will be invalid from the moment it boots until the reception of the first time information from CTECS that points to a time with precision other than UNKNOWN (as from TIMESYNC. Refer to Section 6.9). In the meantime, the time tag used as the secondary header in any mission data packets shall be the respective OBC's free-running clock. We say that, at this point, the free-running clock is the acting time.

When a valid GPS time is found by CTECS and informed to the rest of the spacecraft, GPS time becomes the acting time. Should the GPS time as informed by CTECS at any point roll back to UNKNOWN precision, or should at any point either CTECS or DSU become unavailable, the free-running clock shall become once again the acting time.

Time-keeping information in ancillary data comes with the information of what time is the acting time in the moment of data generation.

TODO: IV uniqueness mechanism and past scheduled frames. Acting time may still be valid if frames used GPS time and now the acting time is the free-running clock.

6 Component Operations

This section details operating procedures regarding SPORT's payloads and spacecraft components. For payloads, the focus of this section is purely to describe operation aspects. For further information, refer to each respective *Interface Control Document* (ICD) or manual.

6.1 EPS/Batteries/EIB

In SPORT, power conditioning and distribution is performed by a solution by AAC Clyde Space. Such solution is composed of two standalone 40Wh battery packs and a control module (CM), which in Clyde Space terminology *is* the EPS itself. Additionally, for extended functionality, the SPORT engineering team created an extra module named EPS Interface Board (EIB). For the scope of this document, however, we will name EPS the set of all four modules, as one subsystem. That said, understanding how SPORT's EPS works is a non-trivial task that we describe in this Section. Refer to Clyde Space documentation in order to better understand this section.

The EPS is enabled and initiates operation as soon as the *Remove Before Flight* (RBF) inhibitors are removed during the observatory's deployment procedures.

The solar panels on the external faces of the observatory generate current when exposed to sunlight. They act as inputs to the EPS' control module's *Battery Charge Regulators* (BCRs). The BCRs are circuits that each take the current generated by one solar panel array and regulate the voltage to levels compatible with the charge/discharge of the battery packs There are two types of BCR: SEPIC BCRs and Buck BCRs. The difference between them is essentially that SEPIC BCRs raise the voltage (from small solar arrays, that do not provide enough voltage) and Buck BCRs lower the voltage (from larger solar arrays, that provide more than the battery's voltage). SPORT's EPS has one SEPIC BCR and eight Buck BCRs.

The output of BCRs goes through ideal diodes for circuit protection. From the diodes, it goes to both the battery packs and the *Power Conditioning Modules* (PCMs). As with EPS' control module, the battery packs themselves are microcontrolled modules. The input current from BCRs' ideal diodes may recharge the batteries depending on the charging mode. Battery packs' functionality is discussed later. For now, it must be noted that the batteries feed BCRs with an *End of Charge* (EoC) signal to indicate whether the batteries have reached full charge.

The EoC signal allows the BCRs to switch charging modes, between *Maximum Power Point Tracking* (MPPT) mode and EoC mode. Whenever battery voltage is below the EoC voltage, the EPS is in MPPT mode.

In MPPT mode, the EPS recharges the batteries with constant current and tracks the maximum power point for each of the pairs of solar arrays. From the perspective of BCRs, the solar panels are arranged in pairs composed of panels on opposite sides of the observatory. Because of that, every time a solar array receives sunlight, its pair is in darkness. In MPPT, the EPS tracks which array is the dominant in each pair and uses that as the source of current. The EPS measures the dominant arrays every 2.5s.

In EoC mode, EoC voltage has been reached and the EPS increasingly reduces the current provided to the battery, moving away from the maximum power point. EoC mode tops up the batteries until full capacity, at which point the only power drawn from the arrays is the necessary for their own operation. Excess power is dissipated as heat.

Batteries provide unregulated power between 6.144V and 8.26V (nominal draw limit: 4.7A) in the VBAT bus.

TODO: PCMs, PDMs, LCL/protections, battery packs. TODO: Mapeamentos de BCRs/paineis, chaves, etc.

6.2 VUR

VUR is nominally powered on at all times the spacecraft is in SAFE, DIAGNOSTICS, IDLE, or SCIENCE. VUR may also be powered off when EPS shuts VUR's VBAT power bus off due to either over-current protection measures (unlikely) or battery under-voltage protection measures (possible).

VUR's VHF uplink can hold a maximum of 40 frames of up to 200 bytes each. VUR's UHF downlink can hold a maximum of 40 frames of up to 235 bytes each. In practice, no more than a couple frames should be needed at a time in either case. C&DH dequeues uplink frames periodically and does not enqueue downlink frames that quickly.

VUR is capable of sending static beacons periodically, but in nominal operation, C&DH shall dynamically re-program VUR beacons to transmit dynamic spacecraft status information.

6.3 C&DH

TODO: TIMESYNC.

As the spacecraft's main OBC, C&DH is nominally powered on at all times EPS is operating. After deployment, the only situations in which C&DH may be powered off are when EPS shuts C&DH's 3.3V power bus off due to either over-current protection measures (unlikely) or battery under-voltage protection measures (possible).

Upon C&DH power on, C&DH enters the INITIALIZATION operating mode, in which it systemically performs the deployment of deployable elements (as described in Section 3), and then automatically enters SAFE mode.

From then on, C&DH transparently interfaces with VUR and is able to receive TCs and transmit TM.

C&DH can execute TCs that fall under its responsibility, such as controlling its own status, providing data packet samples, attitude control, and other spacecraft-controlling duties. For the extensive listing of C&DH TCs, refer to Appendix D.2.

TCs that fall under the responsibility of either PLDH or DSU are forwarded to them as-is over the OBC' main communication bus. The respective OBC shall then execute the TC and answer with (at least) a standard TC response within 5 seconds. Should a response not be received by C&DH within this time limit, C&DH will itself generate a standard TC response with an error code that represents the fact that a response has not been received. Should a response be received normally, C&DH forwards it for UHF transmission as-is.

At all times C&DH is on, it periodically assembles the latest pieces of available ancillary data in their respective ancillary data packets. These pieces of data are gathered asynchronously and at different rates, because elements such as the EPS take different times to generate the data. Also, some pieces of ancillary data come from PLDH and DSU at a fixed frequency of 1Hz.

Ancillary data packets are sent immediately to DSU for storage when they are generated, even if DSU is off. Through DSU, the packets are later sent to ground over X band alongside science data packets. However, a copy of the latest of each type of ancillary data packet is always readily available for transmission over UHF upon demand. For more details about ancillary data packets, refer to Appendix C

In the SAFE, DIAGNOSTICS, IDLE, and SCIENCE operating modes, C&DH repeatedly attempts to handle a new TC at a frequency of 1Hz. A chain of events is performed in order for a TC to be handled.

TODO: TC handling chain of events description and diagram.

6.4 AntS

6.5 MCOM

6.6 PLDH

TODO: TIMESYNC.

PLDH is nominally powered on during the IDLE and SCIENCE operating modes (and may be powered on in DIAGNOSTICS). As with C&DH, PLDH can execute certain TCs that fall under its responsibility. Examples include queries about its own status and managing payloads. For the extensive listing of PLDH TCs, refer to Appendix D.3.

At all times PLDH is on, it automatically and periodically gathers ancillary data about its status and the payloads' and communicates them to C&DH, which in turn incorporates them into the respctive ancillary data packet. PLDH sends its ancillary data at 1Hz.

After powering PLDH on, no extra action is required from operators in order to have PLDH gather data from the payloads. At all times PLDH is on, it continosly queries IVM for data packets even when the payload is powered off. That is because PLDH is not aware of the payload's status before actually receiving any data, and power control comes from C&DH and EPS. Also, querying IVM packets is a purely time-based task, with no "packet ready" signal. PLDH is also always ready for querying MSM and SWP packets, although those tasks are based on "packet ready" interrupts which should not be raised unless the payloads are powered on.

Upon success on querying any data packet from IVM, MSM, or SWP, PLDH immediately encapsulates the data packet in the standard Space Packet format and automatically sends it to DSU. In case DSU is off, the data packet will be missed and discarded.

Following up on Section 5, providing the three payloads with time information is a responsibility of PLDH's. PLDH informs the payloads of its acting time periodically, regardless of whether the acting time is the free-running clock or GPS time. Just like with packet querying, PLDH attempts to inform the payloads of the current time at all times it is on. No extra action is required from operators in regards to payload time keeping.

When CTECS is on, it provides the spacecraft with the PPS signal. For times when CTECS is off or unavailable, PLDH hardware generates a secondary PPS signal. Such secondary PPS is not synchronized with GPS time in any form. It is just a rough fallback synchronization mechanism.

The selection between PPS sources is handled by PLDH and is requested by ground over TCs. The information of which PPS source is selected is available in ancillary data.

The last important operational aspect regarding PLDH is controlling the operation of IVM and SWP with scripts. Such scripts are variable-length sets of commands that can be fed by ground in order to change the operation of either payload. Payloads are operated at the ground's discretion. Please refer to Sections 6.10 and 6.12 for details on IVM and SWP operation, respectively. Naturally, the respective payload needs to be powered on in order to successfully be fed a script. MSM is simpler in terms of operation: no script feeding or operation modes are involved. Please refer to Section 6.11 for details on MSM operation.

6.7 DSU

Similar to PLDH, DSU is nominally powered on during the IDLE and SCIENCE operating modes (and of course may be powered on in DIAGNOSTICS). As with C&DH and PLDH, DSU can execute certain TCs that fall under its responsibility. Examples include queries about its own status and managing CTECS. For the extensive listing of DSU TCs, refer to Appendix D.4.

At all times DSU is on, it also automatically and periodically gathers ancillary data about its status, CTECS', and XBR's, and communicates them to C&DH, which in turn incorporates them into the respctive ancillary data packet. DSU sends its ancillary data at 1Hz.

Hardware-wise, DSU is SPORT's most complex OBC. It features a dual-core ARM-based CPU, running a complete Linux environment, and an FPGA fabric that interfaces with XBR to feed the X band downlink with a continuous 20 Mbit/s data stream.

Operation-wise, CTECS is similar to IVM and SWP in that the payload can be fed scripts that control its operation. Some CTECS scripts, however, need to be executed multiple times a day. Because of this, DSU features a complex CTECS script scheduler that executes the required scripts automatically on certain times of day. The scheduler can be configured by ground for customized CTECS operation, including adding/removing, enabling/disabling, and re-scheduling scripts. Some scripts are built into DSU software, but more can be uploaded from ground. TODO: CTECS script management.

After powering DSU on, no extra action is required from operators in order to have DSU gather data from CTECS or to persistently store mission data packets coming from C&DH and PLDH. At all times DSU is on, it listens for data coming from CTECS, even when the payload is powered off.

Upon successful reception of every CTECS data packet, known as *log*, DSU immediately encapsulates the log in the standard Space Packet format. Refer to Section 6.9 for details on CTECS data logs and CTECS operation.

Some CTECS data SPs are forwarded to C&DH for attitude control. TODO

TIMESYNC SPs are also broadcast to C&DH and PLDH for GPS time synchronization. In order to reduce bus contention, only one in every ten TIMESYNCs is broadcast. TIMESYNC SPs always refer to the most recent PPS. Based on that information, C&DH and PLDH adjust their GPS time views accordingly.

Following up on Section 4.5, every time DSU successfully encapsulates a CTECS data log into an SP or receives a mission data SP from C&DH or PLDH, DSU executes several processes in preparation for MDR. First, the current SP's contents are copied into the data field of one or more TFs. Every time a TF is completely filled (potentially several times even for a single SP), the TF is pseudo-randomized and a new MDR unit is assembled. The newly-assembled MDR unit is then stored persistently in DSU's non-volatile medium. All these processes are executed autonomously, with no interaction from ground. Operators then are able to control the execution of MDR as previously discussed.

6.8 XBR

6.9 CTECS

6.10 IVM

6.11 MSM

Out of the four scientifically payloads, MSM is the simplest to operate. MSM only has one operating mode, meaning

6.12 SWP

7 Memory Management

TODO

8 Attitude Determination and Control Subsystem (ADCS)

TODO

9 Fault Detection

TODO

10 References

[1] ARM Limited. mbed TLS software library. `https://tls.mbed.org/`. (Accessed 2021/02/23).

[2] Consultative Commmittee for Space Data Systems. Space Packet Protocol, CCSDS 133.0-B-2. `https://public.ccsds.org/Pubs/133x0b2e1.pdf`. (Accessed 2020/07/29).

[3] Consultative Commmittee for Space Data Systems. TM Space Data Link Protocol, CCSDS 132.0-B-2. `https://public.ccsds.org/Pubs/132x0b2.pdf`. (Accessed 2020/09/19).

[4] Consultative Commmittee for Space Data Systems. TM Synchronization and Channel Coding, CCSDS 131.0-B-3. `https://public.ccsds.org/Pubs/131x0b3e1.pdf`. (Accessed 2020/09/19).

[5] European Cooperation for Space Standardization. Space engineering: Ground systems and operations — Telemetry and telecommand packet utilization, ECSS-E-70-41A. `https://cwe.ccsds.org/moims/docs/Work%20Completed%20(Closed%20WGs)/Packet%20Utilization%20Standard%20Birds%20of%20a%20Feather/Meeting%20Materials/200909%20Background/ECSS-E-70-41A(30Jan2003).pdf`. (Accessed 2021/02/23).

[6] The CubeSat Program, Cal Poly SLO. 6U CubeSat Design Specification Rev. 1.0. `https://www.cubesat.org/s/6U_CDS_2018-06-07_rev_10.pdf`. (Accessed 2020/03/26).

[7] Tucson Amateur Packet Radio Corporation. AX.25 Link Access Protocol for Amateur Packet Radio. `https://www.tapr.org/pdf/AX25.2.2.pdf`. (Accessed 2020/08/01).

[8] Wikipedia. File:CFB decryption.svg. `https://en.wikipedia.org/wiki/File:CFB_decryption.svg`. (Accessed 2021/02/23).

[9] Wikipedia. File:CFB encryption.svg. `https://en.wikipedia.org/wiki/File:CFB_encryption.svg`. (Accessed 2021/02/23).

Acronyms

AEB Brazilian Space Agency. 3

AES Advanced Encryption Standard. 11

APID Application Identifier. 9

ASM Attached Sync Marker. 22

BCR Battery Charge Regulator. 25

C&DH Control & Data Handler (management and control OBC). 5

CCSDS Consultative Committee for Space Data Systems. 9

CFB Cipher Feedback. 11

CRC-16 16-bit Cyclic Redundancy Check. 10

CTECS Compact Total Electron Content Sensor (payload). 4

DSU Data Storage Unit (data storage and comms OBC). 5

ECDHE Elliptic-Curve Diffie-Hellman with ephemeral keys. 12

ECSS European Cooperation for Space Standardization. 10

EIB EPS Interface Board. 5

EoC End of Charge. 25

EPS Electrical Power Subsystem. 4

GPS Global Positioning System. 4

GS Ground Station. 3

GSFC NASA Goddard Space Flight Center. 3

ICD Interface Control Document. 25

INPE National Institute for Space Research. 3

ITA Aeronautics Institute of Technology. 3

IV initialization vector. 11

IVM Ion Velocity Meter (payload). 4

MCOM Monitoring and Control of Opening Mechanisms. 5
MDR mission data retrieval. 21
MPPT Maximum Power Point Tracking. 25
MSFC NASA Marshall Space Flight Center. 3
MSM Miniaturized Science Magnetometer (payload). 4
OBC onboard computer. 5
PCM Power Conditioning Module. 25
PLDH Payload Data Handler (payload OBC). 5
PPS pulse-per-second. 4
RBF Remove Before Flight. 25
SP Space Packet. 9
SPDF NASA GSFC Space Physics Data Facility. 3
SPORT Scintillation Prediction Observations Research Task. 3
SWP Space Weather Probes (payload). 4
TC telecommand. 3
TF Transfer Frame. 20
TM telemetry. 3
TT&C Telemetry, Tracking, and Command. 5
USU Utah State University. 3
UTD University of Texas at Dallas. 3
VUR VHF/UHF TT&C radio. 5
XBR X band TT&C radio. 5

Appendix A INITIALIZATION Operating Mode

INITIALIZATION is the spacecraft's operating mode when C&DH boots. Upon entering INITIALIZATION (by boot or TC), a routine is executed whose purpose is to systematically and safely perform the deployment of the observatory's eight deployable mechanisms.

The graphs in Figures 14 and 15 represent the finite-state machine (FSM) of INITIALIZATION's routine of deployments. They omit some edge labels for clarity, given that the tool used to generate the graphs automatically sometimes does a poor job in organizing edges and labels.

Both graphs compose a single FSM, whose representation has been split into two parts in order to reduce the complexity of representing the repeating pattern of deployment of each of the eight mechanisms.

The routine starts in state `init`, in which C&DH checks the history of past executions of the routine in its non-volatile storage. If it is the first time the routine is executed in the mission, the routine transitions to state `timeout_30m`. If the routine has been executed before but failed to complete the deployment of some element successfully, it transitions to state `status_determination`. If the routine has been executed before and is deemed complete, the routine ends by immediately transitioning to state `end`.

In `timeout_30m`, the spacecraft waits for 30 minutes before proceeding, in order to meet a mission requirement. During this time, the spacecraft does essentially nothing and, more importantly, does not transmit any data. After the timeout ends, the routine transitions to state `status_determination`.

`status_determination` is a state in which C&DH queries the status of the two boards responsible for performing the deployments (AntS and MCOM) and the status of deployment of each of the eight elements. With this information, `status_determination` acts as a serialization/ordering state, deciding what action to do next.

For each of the eight elements and the two boards, the routine keeps a status flag called *flag_ELEM* (with ELEM being one of: ant1, ant2, efield1, efield2, imped, langmuir, xplus, xminus, ants, mcom), first initialized to false. If set to true, the flag means "the tries to deploy this element have either been successful or have been exhausted, do not try any further". Analyzing these flags is essentially all `status_determination` does to determine the next action.

The order of deployment of the eight elements, and the respective board responsible for the element, are as follows:

1. (AntS) VHF antenna, aka ant1;
2. (AntS) UHF antenna, aka ant2;
3. (MCOM) E-field 1 probe, aka efield1;
4. (MCOM) E-field 2 probe, aka efield2;
5. (MCOM) Impedance probe, aka imped;
6. (MCOM) Langmuir probe, aka langmuir;
7. (MCOM) Solar panel, face X+, aka x_plus; *and*
8. (MCOM) Solar panel, face X-, aka x_minus.

Throughout the routine's execution, before doing any task that may negatively affect the level of battery charge, the level is checked in order to determine whether the batteries are healthy. This is done in every state that is called `battery_check_*`. The spacecraft deems the batteries healthy when their unregulated output voltage is 7V or higher. If the batteries are indeed healthy, the routine proceeds with the next logical state. Else, the state fails.

The routine keeps track of how many times each `battery_check_*` state has failed. If it has failed once or twice, then the routine transitions to state `timeout_2h`, which as the name suggests, halts all activity for two hours then transitions back to state `status_determination`. This is to allow the observatory to complete at least one entire orbit around the Earth, guaranteeing the incidence of sunlight that should recharge the batteries. If the state has failed three times, the routine gives up on trying that particular action and goes on to the next. This is also true to all other states that may fail.

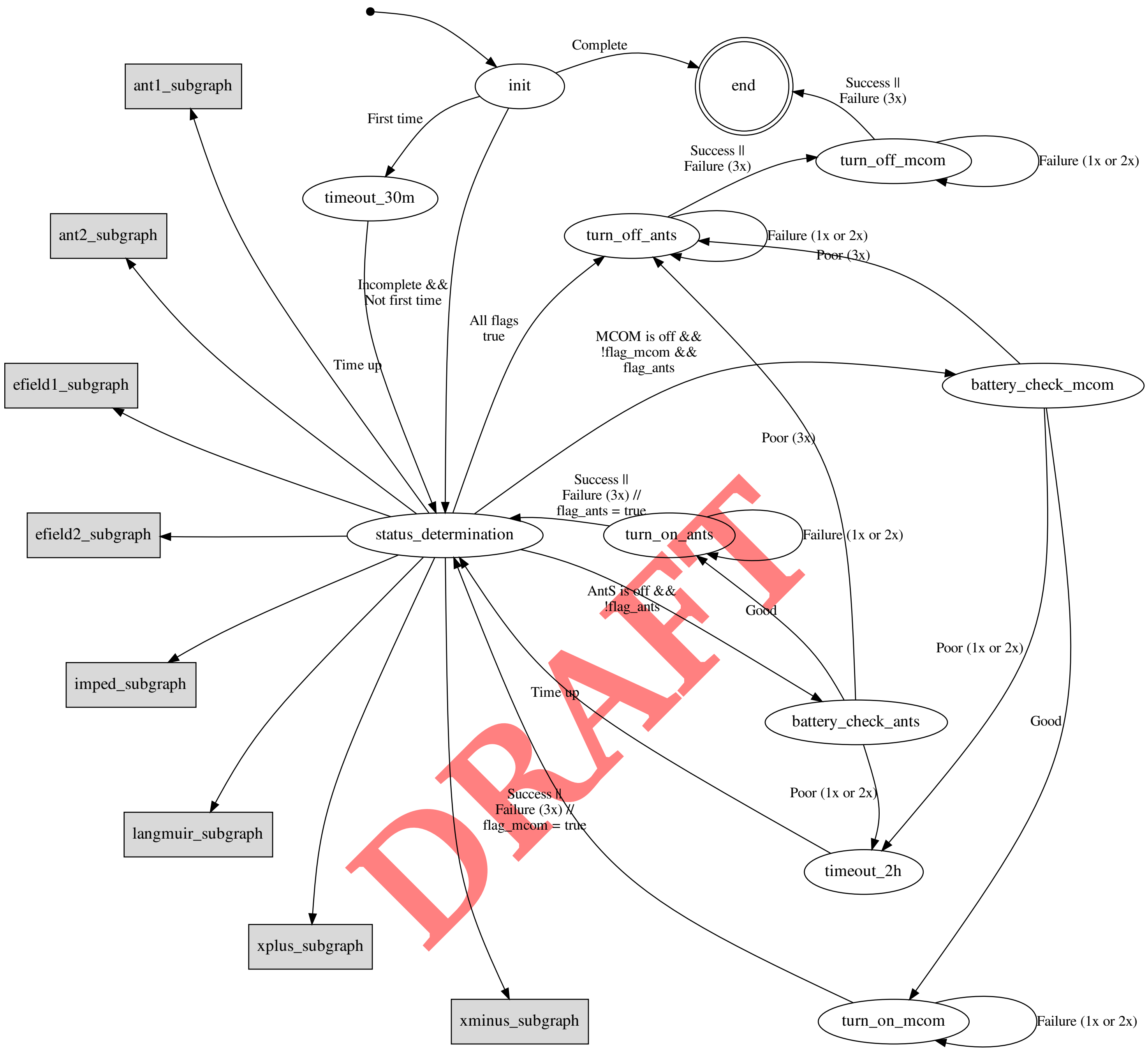


Figure 14: finite-state machine (FSM) of INITIALIZATION's routine (main).

During a nominal first execution of the routine, `status_determination` will first determine that AntS is off, which causes the transition to state `battery_check_ants`.

Success in `battery_check_ants` causes the transition to state `turn_on_ants`. If `turn_on_ants` fails three times, the routine proceeds to also turn off MCOM (just in case it is somehow on) and end.

`turn_on_ants` attempts to power AntS on. In case of success, the routine sets flag_ants to true and transitions back to `status_determination`. In case of failure, the state behaves similarly to

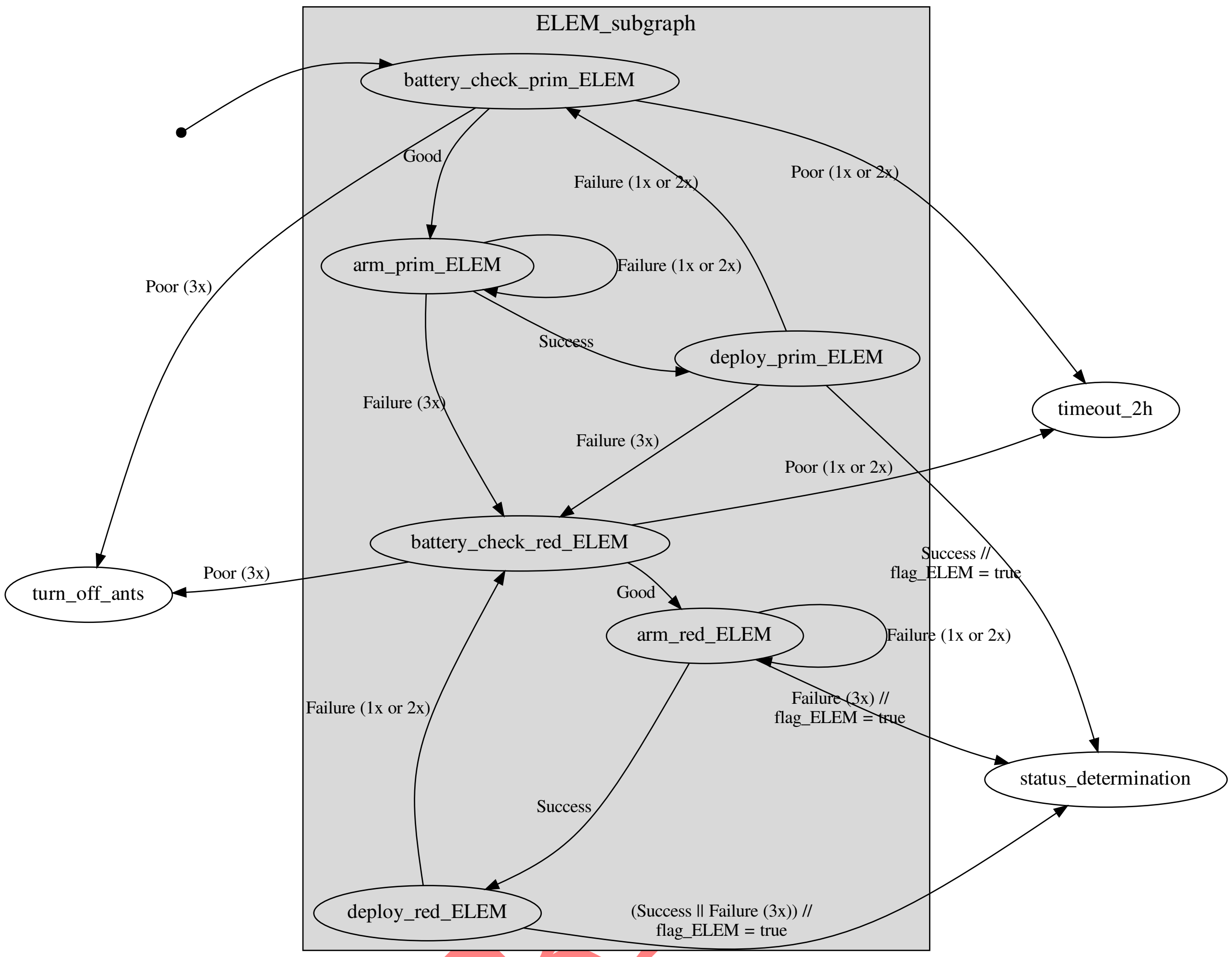


Figure 15: finite-state machine (FSM) of INITIALIZATION's routine (element subgraph).

the battery checking states, retrying twice before giving up. Either way, in transitioning back to `status_determination`, flag_ants is set to true so that nothing else depending on AntS is tried.

`status_determination` will now determine that MCOM is off, which causes the transition to state `battery_check_mcom`. The process of powering MCOM on is completely analogous to the steps previously taken with AntS.

In case of success, the next series of actions are meant to attempt the deployment of each of the eight elements. The attempts to deploy each element *ELEM* have been organized in subgraph ELEM_subgraph in Figure 15. This greatly reduces visual pollution in the FSM representation.

The execution of each subgraph follows a pattern. First, battery health is checked as usual in state `battery_check_prim_ELEM`. If it is poor the first and second times, the routine takes 2-hour timeouts. If it is poor a third time, it proceeds to turn both boards off (which subsequently leads to routine end). If it is good, transition to state `arm_prim_ELEM`.

`arm_prim_ELEM` attempts to *arm* the primary circuit in ELEM's respective board, which only then allows the deployment to be requested. In case of success, the routine transitions to `deploy_prim_ELEM`, which attempts to actually perform the deployment of ELEM. In case of successful deployment, the subgraph may be exited by setting flag_ELEM to true and transitioning to `status_determination`.

In case any `arm_prim_ELEM` or `deploy_prim_ELEM` fail after three tries, AntS and MCOM present redundant circuits that are tried instead. States `battery_check_red_ELEM`, `arm_red_ELEM`, and `deploy_red_ELEM` are functionally equivalent to the respective primary states, but attempting the deployment with the board's redundant circuitry. As usual, if everything fails again, the routine sets flag_ELEM and gives up trying to deploy ELEM.

After expanding each of the eight subgraphs, there are a total of 59 states in the initialization routine.

The subgraphs are identical for every element. There is, however, an extra mechanical constraint that only allows the subgraphs of x_plus and x_minus to be accessed from `status_determination` after successful deployment of efield1 and efield2, respectively. TODO: check correspondence.

It is estimated that a successful first execution of the initialization routine, with no battery recharge timeouts, may take around TODO: time. In the (highly unlikely) worst case, it is estimated that the routine may take up to approximately 36 hours and 30 minutes.

Appendix B Science Data Formats

TODO

Appendix C Ancillary Data Formats

Table 1 displays the structure of ancillary data packets. Ancillary data are separated into five types of packets, each of which containing information related to:

- C&DH;
- PLDH, IVM, MSM, SWP;
- DSU, CTECS, XBR;
- EPS; *and*
- ADCS.

		PACKET PRIMARY HEADER							PACKET DATA FIELD		
			PKT ID			PKT SEQ CTRL				USER DATA FIELD	
		VERSION NO.	PKT TYPE	SEC HDR FLAG	APID	SEQ FLAGS	SEQ COUNT	PKT DATA LENGTH	SECONDARY HEADER	DATA	CHECKSUM
	Length (bits)	3	1	1	11	2	14	16	64	Variable	16
Identification	**Name**										
TM_ANC_001	ANCILLARY C&DH packet	0	0	1	0x001	0x3	0	0x26	Acting time	DD_ANC_CDH	CRC-16
TM_ANC_002	ANCILLARY PLDH packet	0	0	1	0x002	0x3	0	0x7C	Acting time	DD_ANC_PLDH	CRC-16
TM_ANC_003	ANCILLARY DSU packet	0	0	1	0x003	0x3	0	0x63	Acting time	DD_ANC_DSU	CRC-16
TM_ANC_004	ANCILLARY EPS packet	0	0	1	0x004	0x3	0	0x1F	Acting time	DD_ANC_EPS	CRC-16
TM_ANC_005	ANCILLARY ADCS packet	0	0	1	0x005	0x3	0	0x61	Acting time	DD_ANC_ADCS	CRC-16

Table 1: listing of ancillary data packets.

Table 2 displays the structure of ADCS-related ancillary data.

Table 3 displays the structure of C&DH-related ancillary data.

Table 4 displays the structure of DSU- and XBR-related ancillary data.

Table 5 displays the structure of EPS-related ancillary data.

Table 6 displays the structure of PLDH- and payload-related ancillary data.

DD_ANC_ADCS		
Field	**Len (bits)**	**Format**
Observatory position.x	32	
Observatory position.y	32	
Observatory position.z	32	
Observatory velocity.x	32	
Observatory velocity.y	32	
Observatory velocity.z	32	
Observatory attitude.x	32	
Observatory attitude.y	32	
Observatory attitude.z	32	
ADCS magnetometer.x	16	
ADCS magnetometer.y	16	
ADCS magnetometer.z	16	
Ephemeris flag (propagated or interpolated/extrapolated)	2	bitfield
Attitude flag (propagated or interpolated/extrapolated)	2	bitfield
Star Tracker flag (if multiple ST used)	2	bitfield
Magnetic torque activation.b1	2	bitfield
Magnetic torque activation.b2	2	bitfield
Magnetic torque activation.b3	2	bitfield
ADCS mode	4	bitfield
Reaction wheel 1 speed	16	
Reaction wheel 1 current	16	
Reaction wheel 2 speed	16	
Reaction wheel 2 current	16	
Reaction wheel 3 speed	16	
Reaction wheel 3 current	16	
Reaction wheel 4 speed	16	
Reaction wheel 4 current	16	
Solar panel 1 maximum voltage	16	
Solar panel 1 current	16	
Solar panel 2 maximum voltage	16	
Solar panel 2 current	16	
Solar panel 3 maximum voltage	16	
Solar panel 3 current	16	
Solar panel 4 maximum voltage	16	
Solar panel 4 current	16	
Solar panel 5 maximum voltage	16	
Solar panel 5 current	16	
Solar panel 6 maximum voltage	16	
Solar panel 6 current	16	
Solar panel 7 maximum voltage	16	
Solar panel 7 current	16	
Total	**704**	

Table 2: listing of ADCS-related ancillary data.

DD_ANC_CDH		
Field	**Len (bits)**	**Format**
GPS time (C&DH view)	64	GPS TIME
C&DH free-running clock	64	GPS TIME
Acting time flag	1	bool
Current operating mode	7	uint7
Number of boots	32	uint32
Number of received telecommands	32	uint32
Number of successful telecommands	32	uint32
Number of unsuccessful telecommands	32	uint32
Number of telecommands in queue	32	uint32
Total	**296**	

Table 3: listing of C&DH-related ancillary data.

DD_ANC_DSU		
Field	**Len (bits)**	**Format**
GPS time (DSU view)	64	GPS TIME
DSU free-running clock	64	GPS TIME
Number of boots	32	uint32
Number of received telecommands	32	uint32
Number of successful telecommands	32	uint32
Number of unsuccessful telecommands	32	uint32
Science file front position	32	
Science file back position	32	
CTECS status	8	
Number of good CTECS ALMANAC logs	32	uint32
Number of corrupted CTECS ALMANAC logs	32	uint32
Number of good CTECS BESTXYZ logs	32	uint32
Number of corrupted CTECS BESTXYZ logs	32	uint32
Number of good CTECS DIAGNOSTIC logs	32	uint32
Number of corrupted CTECS DIAGNOSTIC logs	32	uint32
Number of good CTECS RANGEX logs	32	uint32
Number of corrupted CTECS RANGEX logs	32	uint32
Number of good CTECS RESPONSE logs	32	uint32
Number of corrupted CTECS RESPONSE logs	32	uint32
Number of good CTECS TIMESYNC logs	32	uint32
Number of corrupted CTECS TIMESYNC logs	32	uint32
XBR status	8	
Total	**720**	

Table 4: listing of DSU- and XBR-related ancillary data.

DD_ANC_EPS		
Field	**Len (bits)**	**Format**
On/off statuses	16	
VBAT voltage	16	
VBAT current	16	
3.3V EPS voltage	16	
3.3V EPS current	16	
3.3V main voltage	16	
3.3V main current	16	
5V voltage	16	
5V current	16	
12V voltage	16	
12V current	16	
Total	**176**	

Table 5: listing of EPS-related ancillary data.

DD_ANC_PLDH		
Field	**Len (bits)**	**Format**
GPS time (PLDH view)	64	GPS TIME
PLDH free-running clock	64	GPS TIME
Number of boots	32	uint32
Number of received telecommands	32	uint32
Number of successful telecommands	32	uint32
Number of unsuccessful telecommands	32	uint32
IVM status	8	
Number of successful IVM SCI polls	32	uint32
Number of successful IVM MAG polls	32	uint32
Number of unsuccessful IVM SCI polls	32	uint32
Number of unsuccessful IVM MAG polls	32	uint32
MSM status	8	
Number of successful MSM polls	32	uint32
Number of unsuccessful MSM polls	32	uint32
SWP status	8	
Number of successful SWP STATUS polls	32	uint32
Number of unsuccessful SWP STATUS polls	32	uint32
Number of successful SWP SCIENCE polls	32	uint32
Number of unsuccessful SWP SCIENCE polls	32	uint32
Number of successful SWP SLP_SWEEP polls	32	uint32
Number of unsuccessful SWP SLP_SWEEP polls	32	uint32
Number of successful SWP WAVE polls	32	uint32
Number of unsuccessful SWP WAVE polls	32	uint32
Number of successful SWP SIP_SWEEP polls	32	uint32
Number of unsuccessful SWP SIP_SWEEP polls	32	uint32
Number of successful SWP SIP_TRACK polls	32	uint32
Number of unsuccessful SWP SIP_TRACK polls	32	uint32
Number of successful SWP CONFIG polls	32	uint32
Number of unsuccessful SWP CONFIG polls	32	uint32
Total	**920**	

Table 6: listing of PLDH- and payload-related ancillary data.

Appendix D Operational Data Formats

D.1 Standard Uplink Response and ErrCode

The standard response format is displayed in Table 7.

		PACKET PRIMARY HEADER							PACKET DATA FIELD			
		VERSION NO.	PKT ID			PKT SEQ CTRL		PKT DATA LENGTH	SECONDARY HEADER	USER DATA FIELD		
			PKT TYPE	SEC HDR FLAG	APID	SEQ FLAGS	SEQ COUNT			TC ID + TC APID	DATA	CHECKSUM
	Length (bits)	3	1	1	11	2	14	16	64	24	16	16
Identification	**Name**											
TM_STD_001	Standard TC response	0	0	1	0x000	0x3	0	0xE	Acting time	Varies	ErrCode	CRC-16

Table 7: format of standard uplink responses.

The code that indicates success or error in the execution of TCs and is used in `TM_STD_001` is named *ErrCode* and is described in Table 8.

ErrCode		
Value	**Name**	**Description**
0x0000	SUCCESS	Unsegmented TC executed successfully.
0x0001	TC_SCHED_SUCCESS	Unsegmented TC scheduled successfully.
0x0002	TC_ASM_SUCCESS	Assembly completed successfully.
0x0003	TC_ASM_ONGOING	TC segment successfully incorporated into the assembly.
0x0004	TC_ASM_ONGOING_REPEAT	Repeated TC segment successfully incorporated into the assembly.
0x0005	E_TC_INV_APID	Invalid TC SP (invalid APID).
0x0006	E_TC_INV_ID	Invalid TC SP (invalid TC ID).
0x0007	E_TC_NOT_ALLOWED_OM	TC execution is not allowed in current operating mode.
0x0008	E_TC_NOT_ALLOWED_SCHED	Scheduling of unsegmented TC is not allowed (immediate execution only).
0x0009	E_TC_INV_TIME	Uplink TC with invalid secondary header (time in the past).
0x000A	E_TC_ENQUEUE_FAIL	Failure in enqueueing TC for execution/routing (internal queue).
0x000B	E_TC_SCHED_FAIL	Failure in inserting TC in schedule (data structure failure).
0x000C	E_TC_ASM_NOT_STARTED	Non-first TC segment received with no ongoing assembly.
0x000D	E_TC_ASM_INV_SEG_LEN	Invalid TC segment length (all but the last segment must be 184 bytes long).
0x000E	E_TC_ASM_INV_CRC	Assembled TC is invalid (incorrect CRC-16 checksum).
0x000F	E_TC_ASM_NEW_FIRST	New first segment received during an ongoing assembly.
0x0010	E_MEM_ALLOC_FAIL	Failure to allocate enough memory for target assembled TC.
0x0011	E_CRYPTO_FAIL	Failure in the execution of a cryptography-related function (e.g. key exchange).
0x0012	E_I2C_FAIL	Failure in internal I2C communications.
0x0013	E_OM_INV_TRANSITION	Invalid operating mode transition.
0x0014	E_OM_POWER_MATRIX_FAIL	Failure in power matrix transition.
0x0015	E_OM_DISPOSAL_FAIL	Invalid step in transition to DISPOSAL. Transition process is reset.
0x0016	E_TC_SCHED_REMOVAL	Failure in removing item from TC schedule (TC not in schedule).
0x0017	E_TC_DSU_TIMEOUT	Timeout in receiving response from DSU after forwarding DSU TC.
0x0018	E_TC_PLDH_TIMEOUT	Timeout in receiving response from PLDH after forwarding PLDH TC.
0x0019	E_TC_MALFORMED	TC is malformed (e.g. bad parameters).
0x001A	E_MD_PKT_UNAVAILABLE	Requested mission data packet sample is unavailable.

Table 8: ErrCode.

D.2 C&DH

Tables 9 through 11 display the structure of C&DH TCs, in accordance with the structure discussed in Sections 4.1 and 4.2. For each TC, Tables 9 through 11 also display the default operating mode acceptance entry, in accordance with Section TODO.

Table 12 displays the structure of C&DH TC-specific responses.

Table 13 displays the structure of the data field in C&DH TCs that require parameters.

Table 14 displays the structure of the data field in specific responses to C&DH TCs.

The following is a description of C&DH TCs:

- **TC_CDH_001 (get time)**: get current status of C&DH time keeping. TC takes no arguments.

The following is a description of C&DH TC responses:

- **TM_CDH_001 (get time)**: specific response to TC_CDH_001. Format:

		PACKET PRIMARY HEADER							PACKET DATA FIELD				OP MODE ACCEPTANCE					
		VERSION NO.	PKT ID			PKT SEQ CTRL		PKT DATA LENGTH	SECONDARY HEADER	USER DATA FIELD			INITIALIZATION	SAFE	DIAGNOSTICS	IDLE	SCIENCE	DISPOSAL
			PKT TYPE	SEC HDR FLAG	APID	SEQ FLAGS	SEQ COUNT			TC ID	DATA	CHECKSUM						
	Length (bits)	3	1	1	11	2	14	16	64	8	Varies	16						
Identification	**Name**																	
TC_CDH_001	C&DH: get time	0	1	0* / 1†	0x100	0x3	0	0x2* / 0xA†	-* / Time†	0x01	-	CRC-16	✘	✔	✔	✔	✔	✘
TC_CDH_002	C&DH: get last TC headers	0	1	0* / 1†	0x100	0x3	0	0x2* / 0xA†	-* / Time†	0x02	-	CRC-16	✘	✔	✔	✔	✔	✘
TC_CDH_003	C&DH: get last standard TC response	0	1	0* / 1†	0x100	0x3	0	0x2* / 0xA†	-* / Time†	0x03	-	CRC-16	✘	✔	✔	✔	✔	✘
TC_CDH_004	C&DH: dump TC hash history	0	1	0* / 1†	0x100	0x3	0	0x2* / 0xA†	-* / Time†	0x04	-	CRC-16	✘	✔	✔	✔	✔	✘
TC_CDH_005	C&DH: enable beacons	0	1	0* / 1†	0x100	0x3	0	0x2* / 0xA†	-* / Time†	0x05	-	CRC-16	✘	✔	✔	✔	✔	✘
TC_CDH_006	C&DH: disable beacons	0	1	0* / 1†	0x100	0x3	0	0x2* / 0xA†	-* / Time†	0x06	-	CRC-16	✘	✔	✔	✔	✔	✘
TC_CDH_007	C&DH: set beacon period	0	1	0* / 1†	0x100	0x3	0	0x3* / 0xB†	-* / Time†	0x07	DU_CDH_007	CRC-16	✘	✔	✔	✔	✔	✘
TC_CDH_008	C&DH: hardware reset	0	1	0	0x100	0x3	0	0x2	-	0x08	-	CRC-16	✘	✔	✘	✘	✘	✘
TC_CDH_009	C&DH: initiate crypto key exchange	0	1	0	0x100	0x3	0	0x22	-	0x09	DU_CDH_009	CRC-16	✘	✔	✔	✔	✔	✘
TC_CDH_010	C&DH: validate candidate crypto key	0	1	0	0x100	0x3	0	0x35	-	0x0A	DU_CDH_010	CRC-16	✘	✔	✔	✔	✔	✘
TC_CDH_011	C&DH: discard candidate crypto key	0	1	0	0x100	0x3	0	0x2	-	0x0B	-	CRC-16	✘	✔	✔	✔	✔	✘
TC_CDH_012	C&DH: commit candidate crypto key	0	1	0	0x100	0x3	0	0x2	-	0x0C	-	CRC-16	✘	✔	✔	✔	✔	✘
TC_CDH_013	TC schedule: dump	0	1	0	0x100	0x3	0	0x2	-	0x0D	-	CRC-16	✘	✔	✔	✔	✔	✘
TC_CDH_014	TC schedule: remove head	0	1	0	0x100	0x3	0	0x2	-	0x0E	-	CRC-16	✘	✔	✔	✔	✔	✘
TC_CDH_015	TC schedule: remove one	0	1	0	0x100	0x3	0	0xA	-	0x0F	DU_CDH_015	CRC-16	✘	✔	✔	✔	✔	✘
TC_CDH_016	TC schedule: remove all	0	1	0	0x100	0x3	0	0x2	-	0x10	-	CRC-16	✘	✔	✔	✔	✔	✘
TC_CDH_017	TC assembly: report history	0	1	0	0x100	0x3	0	0x2	-	0x11	-	CRC-16	✘	✔	✔	✔	✔	✘
TC_CDH_018	TC assembly: reset	0	1	0	0x100	0x3	0	0x2	-	0x12	-	CRC-16	✘	✔	✔	✔	✔	✘
TC_CDH_019	OM: report latest INITIALIZATION history	0	1	0	0x100	0x3	0	0x2	-	0x13	-	CRC-16	✘	✔	✘	✘	✘	✘
TC_CDH_020	OM: transition to INITIALIZATION	0	1	0	0x100	0x3	0	0x2	-	0x14	-	CRC-16	✘	✔	✘	✘	✘	✘
TC_CDH_021	OM: transition to SAFE	0	1	0	0x100	0x3	0	0x2	-	0x15	-	CRC-16	✘	✘	✔	✔	✔	✘
TC_CDH_022	OM: transition to DIAGNOSTICS	0	1	0	0x100	0x3	0	0x2	-	0x16	-	CRC-16	✘	✔	✘	✘	✘	✘
TC_CDH_023	OM: transition to IDLE	0	1	0	0x100	0x3	0	0x2	-	0x17	-	CRC-16	✘	✔	✘	✘	✔	✘
TC_CDH_024	OM: transition to SCIENCE	0	1	0	0x100	0x3	0	0x2	-	0x18	-	CRC-16	✘	✘	✘	✔	✘	✘
TC_CDH_025	OM: arm transition to DISPOSAL	0	1	0	0x100	0x3	0	0x2	-	0x19	-	CRC-16	✘	✔	✘	✘	✘	✘
TC_CDH_026	OM: disarm transition to DISPOSAL	0	1	0	0x100	0x3	0	0x2	-	0x1A	-	CRC-16	✘	✔	✘	✘	✘	✘
TC_CDH_027	OM: report DISPOSAL transition stage	0	1	0	0x100	0x3	0	0x2	-	0x1B	-	CRC-16	✘	✔	✘	✘	✘	✘
TC_CDH_028	OM: transition to DISPOSAL (2nd stage)	0	1	0	0x100	0x3	0	0x2	-	0x1C	-	CRC-16	✘	✔	✘	✘	✘	✘
TC_CDH_029	OM: transition to DISPOSAL (final stage)	0	1	0	0x100	0x3	0	0x2	-	0x1D	-	CRC-16	✘	✔	✘	✘	✘	✘
TC_CDH_030	OM: get power matrix	0	1	0	0x100	0x3	0	0x2	-	0x1E	-	CRC-16	✘	✔	✔	✔	✔	✘
TC_CDH_031	OM: set power matrix	0	1	0	0x100	0x3	0	0x8	-	0x1F	DU_CDH_031	CRC-16	✘	✔	✔	✔	✔	✘
TC_CDH_032	AD: set generation period of packet type	0	1	0* / 1†	0x100	0x3	0	0x4* / 0xC†	-* / Time†	0x20	DU_CDH_032	CRC-16	✘	✔	✔	✔	✔	✘
TC_CDH_033	AD: get generation period of packet type	0	1	0* / 1†	0x100	0x3	0	0x3* / 0xB†	-* / Time†	0x21	DU_CDH_033	CRC-16	✘	✔	✔	✔	✔	✘
TC_CDH_034	AD: get latest packet of type	0	1	0* / 1†	0x100	0x3	0	0x3* / 0xB†	-* / Time†	0x22	DU_CDH_034	CRC-16	✘	✔	✔	✔	✔	✘
TC_CDH_035	AD: engage continuous TM mode	0	1	0* / 1†	0x100	0x3	0	0x3* / 0xB†	-* / Time†	0x23	DU_CDH_035	CRC-16	✘	✔	✔	✔	✔	✘
TC_CDH_036	AD: disengage continuous TM mode	0	1	0* / 1†	0x100	0x3	0	0x2* / 0xA†	-* / Time†	0x24	-	CRC-16	✘	✔	✔	✔	✔	✘
TC_CDH_037	AD: engage selective data compression	0	1	0* / 1†	0x100	0x3	0	0x2* / 0xA†	-* / Time†	0x25	-	CRC-16	✘	✔	✔	✔	✔	✘
TC_CDH_038	AD: disengage selective data compression	0	1	0* / 1†	0x100	0x3	0	0x2* / 0xA†	-* / Time†	0x26	-	CRC-16	✘	✔	✔	✔	✔	✘
TC_CDH_039	EPS: power PLDH on	0	1	0* / 1†	0x100	0x3	0	0x2* / 0xA†	-* / Time†	0x27	-	CRC-16	✘	✘	✔	✘	✘	✘
TC_CDH_040	EPS: power PLDH off	0	1	0* / 1†	0x100	0x3	0	0x2* / 0xA†	-* / Time†	0x28	-	CRC-16	✘	✘	✔	✘	✘	✘
TC_CDH_041	EPS: power DSU on	0	1	0* / 1†	0x100	0x3	0	0x2* / 0xA†	-* / Time†	0x29	-	CRC-16	✘	✘	✔	✘	✘	✘
TC_CDH_042	EPS: power DSU off	0	1	0* / 1†	0x100	0x3	0	0x2* / 0xA†	-* / Time†	0x2A	-	CRC-16	✘	✘	✔	✘	✘	✘
TC_CDH_043	EPS: power CTECS on	0	1	0* / 1†	0x100	0x3	0	0x2* / 0xA†	-* / Time†	0x2B	-	CRC-16	✘	✘	✔	✘	✘	✘
TC_CDH_044	EPS: power CTECS off	0	1	0* / 1†	0x100	0x3	0	0x2* / 0xA†	-* / Time†	0x2C	-	CRC-16	✘	✘	✔	✘	✘	✘
TC_CDH_045	EPS: power IVM on	0	1	0* / 1†	0x100	0x3	0	0x2* / 0xA†	-* / Time†	0x2D	-	CRC-16	✘	✘	✔	✘	✘	✘
TC_CDH_046	EPS: power IVM off	0	1	0* / 1†	0x100	0x3	0	0x2* / 0xA†	-* / Time†	0x2E	-	CRC-16	✘	✘	✔	✘	✘	✘
TC_CDH_047	EPS: power MSM on	0	1	0* / 1†	0x100	0x3	0	0x2* / 0xA†	-* / Time†	0x2F	-	CRC-16	✘	✘	✔	✘	✘	✘
TC_CDH_048	EPS: power MSM off	0	1	0* / 1†	0x100	0x3	0	0x2* / 0xA†	-* / Time†	0x30	-	CRC-16	✘	✘	✔	✘	✘	✘
TC_CDH_049	EPS: power SWP on	0	1	0* / 1†	0x100	0x3	0	0x2* / 0xA†	-* / Time†	0x31	-	CRC-16	✘	✘	✔	✘	✘	✘
TC_CDH_050	EPS: power SWP off	0	1	0* / 1†	0x100	0x3	0	0x2* / 0xA†	-* / Time†	0x32	-	CRC-16	✘	✘	✔	✘	✘	✘
TC_CDH_051	EPS: power XBR on	0	1	0* / 1†	0x100	0x3	0	0x2* / 0xA†	-* / Time†	0x33	-	CRC-16	✘	✘	✔	✘	✘	✘
TC_CDH_052	EPS: power XBR off	0	1	0* / 1†	0x100	0x3	0	0x2* / 0xA†	-* / Time†	0x34	-	CRC-16	✘	✘	✔	✘	✘	✘
TC_CDH_053	EPS: power AntS on	0	1	0* / 1†	0x100	0x3	0	0x2* / 0xA†	-* / Time†	0x35	-	CRC-16	✘	✘	✔	✘	✘	✘
TC_CDH_054	EPS: power AntS off	0	1	0* / 1†	0x100	0x3	0	0x2* / 0xA†	-* / Time†	0x36	-	CRC-16	✘	✘	✔	✘	✘	✘
TC_CDH_055	EPS: power MCOM on	0	1	0* / 1†	0x100	0x3	0	0x2* / 0xA†	-* / Time†	0x37	-	CRC-16	✘	✘	✔	✘	✘	✘
TC_CDH_056	EPS: power MCOM off	0	1	0* / 1†	0x100	0x3	0	0x2* / 0xA†	-* / Time†	0x38	-	CRC-16	✘	✘	✔	✘	✘	✘
TC_CDH_057	EPS CM: get board status	0	1	0* / 1†	0x100	0x3	0	0x2* / 0xA†	-* / Time†	0x39	-	CRC-16	✘	✘	✔	✘	✘	✘
TC_CDH_058	EPS CM: get last error	0	1	0* / 1†	0x100	0x3	0	0x2* / 0xA†	-* / Time†	0x3A	-	CRC-16	✘	✘	✔	✘	✘	✘
TC_CDH_059	EPS CM: get firmware version	0	1	0* / 1†	0x100	0x3	0	0x2* / 0xA†	-* / Time†	0x3B	-	CRC-16	✘	✘	✔	✘	✘	✘
TC_CDH_060	EPS CM: get firmware checksum	0	1	0* / 1†	0x100	0x3	0	0x2* / 0xA†	-* / Time†	0x3C	-	CRC-16	✘	✘	✔	✘	✘	✘
TC_CDH_061	EPS CM: get firmware revision	0	1	0* / 1†	0x100	0x3	0	0x2* / 0xA†	-* / Time†	0x3D	-	CRC-16	✘	✘	✔	✘	✘	✘
TC_CDH_062	EPS CM: get telemetry	0	1	0* / 1†	0x100	0x3	0	TODO	-* / Time†	0x3E	TODO	CRC-16	✘	✘	✔	✘	✘	✘
TC_CDH_063	EPS CM: get comms watchdog period	0	1	0* / 1†	0x100	0x3	0	0x2* / 0xA†	-* / Time†	0x3F	-	CRC-16	✘	✘	✔	✘	✘	✘
TC_CDH_064	EPS CM: set comms watchdog period	0	1	0* / 1†	0x100	0x3	0	TODO	-* / Time†	0x40	TODO	CRC-16	✘	✘	✔	✘	✘	✘

Table 9: listing of C&DH TCs formats (part 1).

		PACKET PRIMARY HEADER							PACKET DATA FIELD				OP MODE ACCEPTANCE					
		VERSION NO.	PKT ID			PKT SEQ CTRL		PKT DATA LENGTH	SECONDARY HEADER	USER DATA FIELD			INITIALIZATION	SAFE	DIAGNOSTICS	IDLE	SCIENCE	DISPOSAL
			PKT TYPE	SEC HDR FLAG	APID	SEQ FLAGS	SEQ COUNT			TC ID	DATA	CHECKSUM						
	Length (bits)	3	1	1	11	2	14	16	64	8	Varies	16						
Identification	**Name**																	
TC_CDH_065	EPS CM: reset comms watchdog	0	1	0* / 1†	0x100	0x3	0	0x2* / 0xA†	-* / Time†	0x41	-	CRC-16	✘	✘	✓	✘	✘	✘
TC_CDH_066	EPS CM: get no. of brown-out resets	0	1	0* / 1†	0x100	0x3	0	0x2* / 0xA†	-* / Time†	0x42	-	CRC-16	✘	✘	✓	✘	✘	✘
TC_CDH_067	EPS CM: get no. of auto software resets	0	1	0* / 1†	0x100	0x3	0	0x2* / 0xA†	-* / Time†	0x43	-	CRC-16	✘	✘	✓	✘	✘	✘
TC_CDH_068	EPS CM: get no. of manual resets	0	1	0* / 1†	0x100	0x3	0	0x2* / 0xA†	-* / Time†	0x44	-	CRC-16	✘	✘	✓	✘	✘	✘
TC_CDH_069	EPS CM: get no. of comms watchdog resets	0	1	0* / 1†	0x100	0x3	0	0x2* / 0xA†	-* / Time†	0x45	-	CRC-16	✘	✘	✓	✘	✘	✘
TC_CDH_070	EPS CM: get actual state of all PDMs	0	1	0* / 1†	0x100	0x3	0	0x2* / 0xA†	-* / Time†	0x46	-	CRC-16	✘	✘	✓	✘	✘	✘
TC_CDH_071	EPS CM: get expected state of all PDMs	0	1	0* / 1†	0x100	0x3	0	0x2* / 0xA†	-* / Time†	0x47	-	CRC-16	✘	✘	✓	✘	✘	✘
TC_CDH_072	EPS CM: get initial state of all PDMs	0	1	0* / 1†	0x100	0x3	0	0x2* / 0xA†	-* / Time†	0x48	-	CRC-16	✘	✘	✓	✘	✘	✘
TC_CDH_073	EPS CM: switch PDM-N on	0	1	0* / 1†	0x100	0x3	0	TODO	-* / Time†	0x49	TODO	CRC-16	✘	✘	✓	✘	✘	✘
TC_CDH_074	EPS CM: switch PDM-N off	0	1	0* / 1†	0x100	0x3	0	TODO	-* / Time†	0x4A	TODO	CRC-16	✘	✘	✓	✘	✘	✘
TC_CDH_075	EPS CM: set PDM-N's timer limit	0	1	0* / 1†	0x100	0x3	0	TODO	-* / Time†	0x4B	TODO	CRC-16	✘	✘	✓	✘	✘	✘
TC_CDH_076	EPS CM: get PDM-N's timer limit	0	1	0* / 1†	0x100	0x3	0	TODO	-* / Time†	0x4C	TODO	CRC-16	✘	✘	✓	✘	✘	✘
TC_CDH_077	EPS CM: get PDM-N's current timer value	0	1	0* / 1†	0x100	0x3	0	TODO	-* / Time†	0x4D	TODO	CRC-16	✘	✘	✓	✘	✘	✘
TC_CDH_078	EPS CM: PCM reset	0	1	0* / 1†	0x100	0x3	0	TODO	-* / Time†	0x4E	TODO	CRC-16	✘	✘	✓	✘	✘	✘
TC_CDH_079	EPS CM: manual reset	0	1	0* / 1†	0x100	0x3	0	0x2* / 0xA†	-* / Time†	0x4F	-	CRC-16	✘	✘	✓	✘	✘	✘
TC_CDH_080	EPS Bat1: get board status	0	1	0* / 1†	0x100	0x3	0	0x2* / 0xA†	-* / Time†	0x50	-	CRC-16	✘	✘	✓	✘	✘	✘
TC_CDH_081	EPS Bat1: get last error	0	1	0* / 1†	0x100	0x3	0	0x2* / 0xA†	-* / Time†	0x51	-	CRC-16	✘	✘	✓	✘	✘	✘
TC_CDH_082	EPS Bat1: get firmware version	0	1	0* / 1†	0x100	0x3	0	0x2* / 0xA†	-* / Time†	0x52	-	CRC-16	✘	✘	✓	✘	✘	✘
TC_CDH_083	EPS Bat1: get firmware checksum	0	1	0* / 1†	0x100	0x3	0	0x2* / 0xA†	-* / Time†	0x53	-	CRC-16	✘	✘	✓	✘	✘	✘
TC_CDH_084	EPS Bat1: get telemetry	0	1	0* / 1†	0x100	0x3	0	TODO	-* / Time†	0x54	TODO	CRC-16	✘	✘	✓	✘	✘	✘
TC_CDH_085	EPS Bat1: get no. of brown-out resets	0	1	0* / 1†	0x100	0x3	0	0x2* / 0xA†	-* / Time†	0x55	-	CRC-16	✘	✘	✓	✘	✘	✘
TC_CDH_086	EPS Bat1: get no. of auto software resets	0	1	0* / 1†	0x100	0x3	0	0x2* / 0xA†	-* / Time†	0x56	-	CRC-16	✘	✘	✓	✘	✘	✘
TC_CDH_087	EPS Bat1: get no. of manual resets	0	1	0* / 1†	0x100	0x3	0	0x2* / 0xA†	-* / Time†	0x57	-	CRC-16	✘	✘	✓	✘	✘	✘
TC_CDH_088	EPS Bat1: get heater controller status	0	1	0* / 1†	0x100	0x3	0	0x2* / 0xA†	-* / Time†	0x58	-	CRC-16	✘	✘	✓	✘	✘	✘
TC_CDH_089	EPS Bat1: set heater controller status	0	1	0* / 1†	0x100	0x3	0	TODO	-* / Time†	0x59	TODO	CRC-16	✘	✘	✓	✘	✘	✘
TC_CDH_090	EPS Bat1: manual reset	0	1	0* / 1†	0x100	0x3	0	0x2* / 0xA†	-* / Time†	0x5A	-	CRC-16	✘	✘	✓	✘	✘	✘
TC_CDH_091	EPS Bat2: get board status	0	1	0* / 1†	0x100	0x3	0	0x2* / 0xA†	-* / Time†	0x5B	-	CRC-16	✘	✘	✓	✘	✘	✘
TC_CDH_092	EPS Bat2: get last error	0	1	0* / 1†	0x100	0x3	0	0x2* / 0xA†	-* / Time†	0x5C	-	CRC-16	✘	✘	✓	✘	✘	✘
TC_CDH_093	EPS Bat2: get firmware version	0	1	0* / 1†	0x100	0x3	0	0x2* / 0xA†	-* / Time†	0x5D	-	CRC-16	✘	✘	✓	✘	✘	✘
TC_CDH_094	EPS Bat2: get firmware checksum	0	1	0* / 1†	0x100	0x3	0	0x2* / 0xA†	-* / Time†	0x5E	-	CRC-16	✘	✘	✓	✘	✘	✘
TC_CDH_095	EPS Bat2: get telemetry	0	1	0* / 1†	0x100	0x3	0	TODO	-* / Time†	0x5F	TODO	CRC-16	✘	✘	✓	✘	✘	✘
TC_CDH_096	EPS Bat2: get no. of brown-out resets	0	1	0* / 1†	0x100	0x3	0	0x2* / 0xA†	-* / Time†	0x60	-	CRC-16	✘	✘	✓	✘	✘	✘
TC_CDH_097	EPS Bat2: get no. of auto software resets	0	1	0* / 1†	0x100	0x3	0	0x2* / 0xA†	-* / Time†	0x61	-	CRC-16	✘	✘	✓	✘	✘	✘
TC_CDH_098	EPS Bat2: get no. of manual resets	0	1	0* / 1†	0x100	0x3	0	0x2* / 0xA†	-* / Time†	0x62	-	CRC-16	✘	✘	✓	✘	✘	✘
TC_CDH_099	EPS Bat2: get heater controller status	0	1	0* / 1†	0x100	0x3	0	0x2* / 0xA†	-* / Time†	0x63	-	CRC-16	✘	✘	✓	✘	✘	✘
TC_CDH_100	EPS Bat2: set heater controller status	0	1	0* / 1†	0x100	0x3	0	TODO	-* / Time†	0x64	TODO	CRC-16	✘	✘	✓	✘	✘	✘
TC_CDH_101	EPS Bat2: manual reset	0	1	0* / 1†	0x100	0x3	0	0x2* / 0xA†	-* / Time†	0x65	-	CRC-16	✘	✘	✓	✘	✘	✘
TC_CDH_102	EIB: switch PDM-N on	0	1	0* / 1†	0x100	0x3	0	TODO	-* / Time†	0x66	TODO	CRC-16	✘	✘	✓	✘	✘	✘
TC_CDH_103	EIB: switch PDM-N off	0	1	0* / 1†	0x100	0x3	0	TODO	-* / Time†	0x67	TODO	CRC-16	✘	✘	✓	✘	✘	✘
TC_CDH_104	EIB: switch all 12V PDMs on	0	1	0* / 1†	0x100	0x3	0	0x2* / 0xA†	-* / Time†	0x68	-	CRC-16	✘	✘	✓	✘	✘	✘
TC_CDH_105	EIB: switch all 5V PDMs on	0	1	0* / 1†	0x100	0x3	0	0x2* / 0xA†	-* / Time†	0x69	-	CRC-16	✘	✘	✓	✘	✘	✘
TC_CDH_106	EIB: switch all 3.3V PDMs on	0	1	0* / 1†	0x100	0x3	0	0x2* / 0xA†	-* / Time†	0x6A	-	CRC-16	✘	✘	✓	✘	✘	✘
TC_CDH_107	EIB: switch all PDMs on	0	1	0* / 1†	0x100	0x3	0	0x2* / 0xA†	-* / Time†	0x6B	-	CRC-16	✘	✘	✓	✘	✘	✘
TC_CDH_108	EIB: software reset	0	1	0* / 1†	0x100	0x3	0	0x2* / 0xA†	-* / Time†	0x6C	-	CRC-16	✘	✘	✓	✘	✘	✘
TC_CDH_109	EIB: get available state info	0	1	0* / 1†	0x100	0x3	0	0x2* / 0xA†	-* / Time†	0x6D	-	CRC-16	✘	✘	✓	✘	✘	✘
TC_CDH_110	EIB: switch all 12V PDMs off	0	1	0* / 1†	0x100	0x3	0	0x2* / 0xA†	-* / Time†	0x6E	-	CRC-16	✘	✘	✓	✘	✘	✘
TC_CDH_111	EIB: switch all 5V PDMs off	0	1	0* / 1†	0x100	0x3	0	0x2* / 0xA†	-* / Time†	0x6F	-	CRC-16	✘	✘	✓	✘	✘	✘
TC_CDH_112	EIB: switch all 3.3V PDMs off	0	1	0* / 1†	0x100	0x3	0	0x2* / 0xA†	-* / Time†	0x70	-	CRC-16	✘	✘	✓	✘	✘	✘
TC_CDH_113	EIB: switch all PDMs off	0	1	0* / 1†	0x100	0x3	0	0x2* / 0xA†	-* / Time†	0x71	-	CRC-16	✘	✘	✓	✘	✘	✘
TC_CDH_114	EIB: get all thermistor readings	0	1	0* / 1†	0x100	0x3	0	0x2* / 0xA†	-* / Time†	0x72	-	CRC-16	✘	✘	✓	✘	✘	✘
TC_CDH_115	AntS: reset, prim	0	1	0* / 1†	0x100	0x3	0	0x2* / 0xA†	-* / Time†	0x73	-	CRC-16	✘	✘	✓	✘	✘	✘
TC_CDH_116	AntS: reset, red	0	1	0* / 1†	0x100	0x3	0	0x2* / 0xA†	-* / Time†	0x74	-	CRC-16	✘	✘	✓	✘	✘	✘
TC_CDH_117	AntS: arm, prim	0	1	0* / 1†	0x100	0x3	0	0x2* / 0xA†	-* / Time†	0x75	-	CRC-16	✘	✘	✓	✘	✘	✘
TC_CDH_118	AntS: arm, red	0	1	0* / 1†	0x100	0x3	0	0x2* / 0xA†	-* / Time†	0x76	-	CRC-16	✘	✘	✓	✘	✘	✘
TC_CDH_119	AntS: disarm, prim	0	1	0* / 1†	0x100	0x3	0	0x2* / 0xA†	-* / Time†	0x77	-	CRC-16	✘	✘	✓	✘	✘	✘
TC_CDH_120	AntS: disarm, red	0	1	0* / 1†	0x100	0x3	0	0x2* / 0xA†	-* / Time†	0x78	-	CRC-16	✘	✘	✓	✘	✘	✘
TC_CDH_121	AntS: deploy VHF ant., prim	0	1	0* / 1†	0x100	0x3	0	TODO	-* / Time†	0x79	TODO	CRC-16	✘	✘	✓	✘	✘	✘
TC_CDH_122	AntS: deploy VHF ant., red	0	1	0* / 1†	0x100	0x3	0	TODO	-* / Time†	0x7A	TODO	CRC-16	✘	✘	✓	✘	✘	✘
TC_CDH_123	AntS: deploy UHF ant., prim	0	1	0* / 1†	0x100	0x3	0	TODO	-* / Time†	0x7B	TODO	CRC-16	✘	✘	✓	✘	✘	✘
TC_CDH_124	AntS: deploy UHF ant., red	0	1	0* / 1†	0x100	0x3	0	TODO	-* / Time†	0x7C	TODO	CRC-16	✘	✘	✓	✘	✘	✘
TC_CDH_125	AntS: deploy all ants. sequentially, prim	0	1	0* / 1†	0x100	0x3	0	TODO	-* / Time†	0x7D	TODO	CRC-16	✘	✘	✓	✘	✘	✘
TC_CDH_126	AntS: deploy all ants. sequentially, red	0	1	0* / 1†	0x100	0x3	0	TODO	-* / Time†	0x7E	TODO	CRC-16	✘	✘	✓	✘	✘	✘
TC_CDH_127	AntS: measure system temperature, prim	0	1	0* / 1†	0x100	0x3	0	0x2* / 0xA†	-* / Time†	0x7F	-	CRC-16	✘	✘	✓	✘	✘	✘
TC_CDH_128	AntS: measure system temperature, red	0	1	0* / 1†	0x100	0x3	0	0x2* / 0xA†	-* / Time†	0x80	-	CRC-16	✘	✘	✓	✘	✘	✘

Table 10: listing of C&DH TCs formats (part 2).

		PACKET PRIMARY HEADER							PACKET DATA FIELD				OP MODE ACCEPTANCE					
			PKT ID			PKT SEQ CTRL				USER DATA FIELD								
Identification	Name	VERSION NO.	PKT TYPE	SEC HDR FLAG	APID	SEQ FLAGS	SEQ COUNT	PKT DATA LENGTH	SECONDARY HEADER	TC ID	DATA	CHECKSUM	INITIALIZATION	SAFE	DIAGNOSTICS	IDLE	SCIENCE	DISPOSAL
	Length (bits)	3	1	1	11	2	14	16	64	8	Varies	16						
TC_CDH_129	AntS: get deployment status, prim	0	1	0* / 1†	0x100	0x3	0	0x2* / 0xA†	-* / Time†	0x81	-	CRC-16	✗	✗	✓	✗	✗	✗
TC_CDH_130	AntS: get deployment status, red	0	1	0* / 1†	0x100	0x3	0	0x2* / 0xA†	-* / Time†	0x82	-	CRC-16	✗	✗	✓	✗	✗	✗
TC_CDH_131	AntS: get VHF ant. activation count, prim	0	1	0* / 1†	0x100	0x3	0	0x2* / 0xA†	-* / Time†	0x83	-	CRC-16	✗	✗	✓	✗	✗	✗
TC_CDH_132	AntS: get VHF ant. activation count, red	0	1	0* / 1†	0x100	0x3	0	0x2* / 0xA†	-* / Time†	0x84	-	CRC-16	✗	✗	✓	✗	✗	✗
TC_CDH_133	AntS: get UHF ant. activation count, prim	0	1	0* / 1†	0x100	0x3	0	0x2* / 0xA†	-* / Time†	0x85	-	CRC-16	✗	✗	✓	✗	✗	✗
TC_CDH_134	AntS: get UHF ant. activation count, red	0	1	0* / 1†	0x100	0x3	0	0x2* / 0xA†	-* / Time†	0x86	-	CRC-16	✗	✗	✓	✗	✗	✗
TC_CDH_135	AntS: get VHF ant. activation time, prim	0	1	0* / 1†	0x100	0x3	0	0x2* / 0xA†	-* / Time†	0x87	-	CRC-16	✗	✗	✓	✗	✗	✗
TC_CDH_136	AntS: get VHF ant. activation time, red	0	1	0* / 1†	0x100	0x3	0	0x2* / 0xA†	-* / Time†	0x88	-	CRC-16	✗	✗	✓	✗	✗	✗
TC_CDH_137	AntS: get UHF ant. activation time, prim	0	1	0* / 1†	0x100	0x3	0	0x2* / 0xA†	-* / Time†	0x89	-	CRC-16	✗	✗	✓	✗	✗	✗
TC_CDH_138	AntS: get UHF ant. activation time, red	0	1	0* / 1†	0x100	0x3	0	0x2* / 0xA†	-* / Time†	0x8A	-	CRC-16	✗	✗	✓	✗	✗	✗
TC_CDH_139	MCOM: arm booms, prim	0	1	0* / 1†	0x100	0x3	0	0x2* / 0xA†	-* / Time†	0x8B	-	CRC-16	✗	✗	✓	✗	✗	✗
TC_CDH_140	MCOM: arm booms, red	0	1	0* / 1†	0x100	0x3	0	0x2* / 0xA†	-* / Time†	0x8C	-	CRC-16	✗	✗	✓	✗	✗	✗
TC_CDH_141	MCOM: arm SP X plus, prim	0	1	0* / 1†	0x100	0x3	0	0x2* / 0xA†	-* / Time†	0x8D	-	CRC-16	✗	✗	✓	✗	✗	✗
TC_CDH_142	MCOM: arm SP X plus, red	0	1	0* / 1†	0x100	0x3	0	0x2* / 0xA†	-* / Time†	0x8E	-	CRC-16	✗	✗	✓	✗	✗	✗
TC_CDH_143	MCOM: arm SP X minus, prim	0	1	0* / 1†	0x100	0x3	0	0x2* / 0xA†	-* / Time†	0x8F	-	CRC-16	✗	✗	✓	✗	✗	✗
TC_CDH_144	MCOM: arm SP X minus, red	0	1	0* / 1†	0x100	0x3	0	0x2* / 0xA†	-* / Time†	0x90	-	CRC-16	✗	✗	✓	✗	✗	✗
TC_CDH_145	MCOM: disarm booms, prim	0	1	0* / 1†	0x100	0x3	0	0x2* / 0xA†	-* / Time†	0x91	-	CRC-16	✗	✗	✓	✗	✗	✗
TC_CDH_146	MCOM: disarm booms, red	0	1	0* / 1†	0x100	0x3	0	0x2* / 0xA†	-* / Time†	0x92	-	CRC-16	✗	✗	✓	✗	✗	✗
TC_CDH_147	MCOM: disarm SP X plus, prim	0	1	0* / 1†	0x100	0x3	0	0x2* / 0xA†	-* / Time†	0x93	-	CRC-16	✗	✗	✓	✗	✗	✗
TC_CDH_148	MCOM: disarm SP X plus, red	0	1	0* / 1†	0x100	0x3	0	0x2* / 0xA†	-* / Time†	0x94	-	CRC-16	✗	✗	✓	✗	✗	✗
TC_CDH_149	MCOM: disarm SP X minus, prim	0	1	0* / 1†	0x100	0x3	0	0x2* / 0xA†	-* / Time†	0x95	-	CRC-16	✗	✗	✓	✗	✗	✗
TC_CDH_150	MCOM: disarm SP X minus, red	0	1	0* / 1†	0x100	0x3	0	0x2* / 0xA†	-* / Time†	0x96	-	CRC-16	✗	✗	✓	✗	✗	✗
TC_CDH_151	MCOM: get arm status, prim	0	1	0* / 1†	0x100	0x3	0	0x2* / 0xA†	-* / Time†	0x97	-	CRC-16	✗	✗	✓	✗	✗	✗
TC_CDH_152	MCOM: get arm status, red	0	1	0* / 1†	0x100	0x3	0	0x2* / 0xA†	-* / Time†	0x98	-	CRC-16	✗	✗	✓	✗	✗	✗
TC_CDH_153	MCOM: soft reset, prim	0	1	0* / 1†	0x100	0x3	0	0x2* / 0xA†	-* / Time†	0x99	-	CRC-16	✗	✗	✓	✗	✗	✗
TC_CDH_154	MCOM: soft reset, red	0	1	0* / 1†	0x100	0x3	0	0x2* / 0xA†	-* / Time†	0x9A	-	CRC-16	✗	✗	✓	✗	✗	✗
TC_CDH_155	MCOM: get safe time, prim	0	1	0* / 1†	0x100	0x3	0	0x2* / 0xA†	-* / Time†	0x9B	-	CRC-16	✗	✗	✓	✗	✗	✗
TC_CDH_156	MCOM: get safe time, red	0	1	0* / 1†	0x100	0x3	0	0x2* / 0xA†	-* / Time†	0x9C	-	CRC-16	✗	✗	✓	✗	✗	✗
TC_CDH_157	MCOM: set safe time, prim	0	1	0* / 1†	0x100	0x3	0	TODO	TODO	0x9D	TODO	CRC-16	✗	✗	✓	✗	✗	✗
TC_CDH_158	MCOM: set safe time, red	0	1	0* / 1†	0x100	0x3	0	TODO	TODO	0x9E	TODO	CRC-16	✗	✗	✓	✗	✗	✗
TC_CDH_159	MCOM: cancel deployment, prim	0	1	0* / 1†	0x100	0x3	0	0x2* / 0xA†	-* / Time†	0x9F	-	CRC-16	✗	✗	✓	✗	✗	✗
TC_CDH_160	MCOM: cancel deployment, red	0	1	0* / 1†	0x100	0x3	0	0x2* / 0xA†	-* / Time†	0xA0	-	CRC-16	✗	✗	✓	✗	✗	✗
TC_CDH_161	MCOM: deploy element, prim	0	1	0* / 1†	0x100	0x3	0	TODO	-* / Time†	0xA1	TODO	CRC-16	✗	✗	✓	✗	✗	✗
TC_CDH_162	MCOM: deploy element, red	0	1	0* / 1†	0x100	0x3	0	TODO	-* / Time†	0xA2	TODO	CRC-16	✗	✗	✓	✗	✗	✗
TC_CDH_163	MCOM: get sensor status, prim	0	1	0* / 1†	0x100	0x3	0	0x2* / 0xA†	-* / Time†	0xA3	-	CRC-16	✗	✗	✓	✗	✗	✗
TC_CDH_164	MCOM: get sensor status, red	0	1	0* / 1†	0x100	0x3	0	0x2* / 0xA†	-* / Time†	0xA4	-	CRC-16	✗	✗	✓	✗	✗	✗
TC_CDH_165	MCOM: deploy element with override, prim	0	1	0* / 1†	0x100	0x3	0	TODO	-* / Time†	0xA5	TODO	CRC-16	✗	✗	✓	✗	✗	✗
TC_CDH_166	MCOM: deploy element with override, red	0	1	0* / 1†	0x100	0x3	0	TODO	-* / Time†	0xA6	TODO	CRC-16	✗	✗	✓	✗	✗	✗
TC_CDH_167	XBR: software reset	0	1	0* / 1†	0x100	0x3	0	0x2* / 0xA†	-* / Time†	0xA7	-	CRC-16	✗	✓	✓	✓	✓	✗

Table 11: listing of C&DH TCs formats (part 3).

		PACKET PRIMARY HEADER							PACKET DATA FIELD		
			PKT ID			PKT SEQ CTRL				USER DATA FIELD	
Identification	Name	VERSION NO.	PKT TYPE	SEC HDR FLAG	APID	SEQ FLAGS	SEQ COUNT	PKT DATA LENGTH	SECONDARY HEADER	DATA	CHECKSUM
	Length (bits)	3	1	1	11	2	14	16	64	Varies	16
TM_CDH_001	C&DH: get time	0	0	1	0x201	0x3	Varies	0x1A	Acting time	DD_CDH_001	CRC-16
TM_CDH_007	TC queue: dump	0	0	1	0x203	0x3	Varies	Varies	Acting time	DD_CDH_007	CRC-16
TM_CDH_011	TC assembly: report history	0	0	1	0x207	0x3	Varies		Acting time	DD_CDH_011	CRC-16
TM_CDH_021	OM: report DISPOSAL transition stage	0	0	1	0x214	0x3	Varies		Acting time	DD_CDH_021	CRC-16
TM_CDH_024	OM: get power matrix	0	0	1	0x218	0x3	Varies		Acting time	DD_CDH_024	CRC-16
	TODO										

Table 12: listing of C&DH TC-specific response (TM) formats.

DU_CDH_004		
Field	**Len (bits)**	**Format**
Diffie-Hellman public key	256	bitfield
Total	**256**	

DU_CDH_009		
Field	**Len (bits)**	**Format**
Time tag	64	GPS time
Total	**64**	

DU_CDH_025		
Field	**Len (bits)**	**Format**
Power matrix	48	bitfield
Total	**48**	

Table 13: listing of C&DH uplink data field formats.

DD_CDH_001		
Field	**Len (bits)**	**Format**
GPS time (C&DH view)	64	GPS time
Free-running clock	64	GPS time
Acting time flag	8	bool
Total	**136**	

DD_CDH_007		
Field	**Len (bits)**	**Format**
Entry’s time tag	64	GPS time
multiplied by N (number of entries)		
Total	**64N**	

DD_CDH_011		
Field	**Len (bits)**	**Format**
Entry’s time tag	TODO	GPS time
Total	**0**	

DD_CDH_021		
Field	**Len (bits)**	**Format**
DISPOSAL transition stage	8	bitfield
Total	**8**	

DD_CDH_024		
Field	**Len (bits)**	**Format**
Power matrix	48	bitfield
Total	**48**	

Table 14: listing of C&DH downlink data field formats.

D.3 PLDH

Table 15 displays the structure of PLDH TCs, in accordance with the structure discussed in Sections 4.1 and 4.2. For each TC, Table 15 also displays the default operating mode acceptance entry, in accordance with Section TODO.

		PACKET PRIMARY HEADER							PACKET DATA FIELD				OP MODE ACCEPTANCE					
			PKT ID			PKT SEQ CTRL				USER DATA FIELD								
		VERSION NO.	PKT TYPE	SEC HDR FLAG	APID	SEQ FLAGS	SEQ COUNT	PKT DATA LENGTH	SECONDARY HEADER	TC ID	DATA	CHECKSUM	INITIALIZATION	SAFE	DIAGNOSTICS	IDLE	SCIENCE	DISPOSAL
	Length (bits)	3	1	1	11	2	14	16	64	8	Varies	16						
Identification	**Name**																	
TC_PLD_001	PLDH: get time	0	1	0* / 1†	0x300	0x3	0	0x2* / 0xA†	-* / Time†	0x01	-	CRC-16	✗	✗	✓	✓	✓	✗
TC_PLD_002	PPS: set to CTECS	0	1	0* / 1†	0x300	0x3	0	0x2* / 0xA†	-* / Time†	0x02	-	CRC-16	✗	✗	✓	✓	✓	✗
TC_PLD_003	PPS: set to PLDH	0	1	0* / 1†	0x300	0x3	0	0x2* / 0xA†	-* / Time†	0x03	-	CRC-16	✗	✗	✓	✓	✓	✗
TC_PLD_004	IVM: feed script	0	1	0* / 1†	0x300	0x3	0	Varies	-* / Time†	0x04	IVM script	CRC-16	✗	✗	✓	✗	✓	✗
TC_PLD_005	SWP: feed script	0	1	0* / 1†	0x300	0x3	0	Varies	-* / Time†	0x05	SWP script	CRC-16	✗	✗	✓	✗	✓	✗

Table 15: listing of PLDH TCs.

Table 16 displays the structure of PLDH TC-specific responses.

		PACKET PRIMARY HEADER							PACKET DATA FIELD		
			PKT ID			PKT SEQ CTRL				USER DATA FIELD	
		VERSION NO.	PKT TYPE	SEC HDR FLAG	APID	SEQ FLAGS	SEQ COUNT	PKT DATA LENGTH	SECONDARY HEADER	DATA	CHECKSUM
	Length (bits)	3	1	1	11	2	14	16	64	Varies	16
Identification	**Name**										
TM_PLD_001	PLDH: get time	0	0	1	0x401	0x3	0	0x1A	Acting time	DD_PLD_001	CRC-16

Table 16: listing of PLDH TC-specific response (TM) formats.

The only PLDH TCs that require parameters are TC_PLD_004 and TC_PLD_005, which feed scripts to IVM and SWP, respectively. The scripts must be obtained from the instrument PIs and embedded into the user data field as is.

Table 17 displays the structure of the data field in specific responses to PLDH TCs.

DD_PLD_001		
Field	**Len (bits)**	**Format**
GPS time (PLDH view)	64	GPS time
Free-running clock	64	GPS time
Acting time flag	8	bool
Total	**136**	

Table 17: listing of PLDH downlink data field formats.

The following is a description of PLDH TCs:

- **TC_PLD_001 (LOREM IPSUM)**: LOREM IPSUM

The following is a description of PLDH TC responses:

- **TM_PLD_001 (LOREM IPSUM)**: LOREM IPSUM

D.4 DSU

Table 18 displays the structure of DSU TCs, in accordance with the structure discussed in Sections 4.1 and 4.2. For each TC, Table 18 also displays the default operating mode acceptance entry, in accordance with Section TODO.

		PACKET PRIMARY HEADER							PACKET DATA FIELD				OP MODE ACCEPTANCE					
			PKT ID			PKT SEQ CTRL				USER DATA FIELD								
		VERSION NO.	PKT TYPE	SEC HDR FLAG	APID	SEQ FLAGS	SEQ COUNT	PKT DATA LENGTH	SECONDARY HEADER	TC ID	DATA	CHECKSUM	INITIALIZATION	SAFE	DIAGNOSTICS	IDLE	SCIENCE	DISPOSAL
	Length (bits)	3	1	1	11	2	14	16	64	8	TODO	16						
Identification	**Name**																	
TC_DSU_001	DSU: get time	0	1	0* / 1†	0x700	0x3	0	0x2* / 0xA†	-* / Time†	0x01	-	CRC-16	✘	✘	✓	✓	✓	✘
TC_DSU_002	DSU: report MDR transfer status	0	1	0* / 1†	0x700	0x3	0	0x2* / 0xA†	-* / Time†	0x02			✘	✘	✓	✓	✓	✘
TC_DSU_003	DSU: prepare MDR transfer	0	1	0* / 1†	0x700	0x3	0	0xA* / 0x12†	-* / Time†	0x03	-	CRC-16	✘	✘	✓	✓	✓	✘
TC_DSU_004	DSU: initiate MDR transfer	0	1	0* / 1†	0x700	0x3	0	0x2* / 0xA†	-* / Time†	0x04	DU_DSU_004	CRC-16	✘	✘	✓	✓	✓	✘
TC_DSU_005	DSU: report latest MDR transfer	0	1	0* / 1†	0x700	0x3	0	0x2* / 0xA†	-* / Time†	0x05	-	CRC-16	✘	✘	✓	✓	✓	✘
TC_DSU_006	MD: get latest CTECS ALMANAC packet	0	1	0* / 1†	0x700	0x3	0	0x2* / 0xA†	-* / Time†	0x06	-	CRC-16	✘	✘	✓	✓	✓	✘
TC_DSU_007	MD: get latest CTECS BESTXYZ packet	0	1	0* / 1†	0x700	0x3	0	0x2* / 0xA†	-* / Time†	0x07	-	CRC-16	✘	✘	✓	✓	✓	✘
TC_DSU_008	MD: get latest CTECS DIAGNOSTIC packet	0	1	0* / 1†	0x700	0x3	0	0x2* / 0xA†	-* / Time†	0x08	-	CRC-16	✘	✘	✓	✓	✓	✘
TC_DSU_009	MD: get latest CTECS RANGEX packet	0	1	0* / 1†	0x700	0x3	0	0x2* / 0xA†	-* / Time†	0x09	-	CRC-16	✘	✘	✓	✓	✓	✘
TC_DSU_010	MD: get latest CTECS RESPONSE packet	0	1	0* / 1†	0x700	0x3	0	0x2* / 0xA†	-* / Time†	0x0A	-	CRC-16	✘	✘	✓	✓	✓	✘
TC_DSU_011	MD: get latest CTECS TIMESYNC packet	0	1	0* / 1†	0x700	0x3	0	0x2* / 0xA†	-* / Time†	0x0B	-	CRC-16	✘	✘	✓	✓	✓	✘
TC_DSU_012	MD: get latest IVM SCI packet	0	1	0* / 1†	0x700	0x3	0	0x2* / 0xA†	-* / Time†	0x0C	-	CRC-16	✘	✘	✓	✓	✓	✘
TC_DSU_013	MD: get latest IVM MAG packet	0	1	0* / 1†	0x700	0x3	0	0x2* / 0xA†	-* / Time†	0x0D	-	CRC-16	✘	✘	✓	✓	✓	✘
TC_DSU_014	MD: get latest MSM SCIENCE packet	0	1	0* / 1†	0x700	0x3	0	0x2* / 0xA†	-* / Time†	0x0E	-	CRC-16	✘	✘	✓	✓	✓	✘
TC_DSU_015	MD: get latest SWP STATUS packet	0	1	0* / 1†	0x700	0x3	0	0x2* / 0xA†	-* / Time†	0x0F	-	CRC-16	✘	✘	✓	✓	✓	✘
TC_DSU_016	MD: get latest SWP SCIENCE packet	0	1	0* / 1†	0x700	0x3	0	0x2* / 0xA†	-* / Time†	0x10	-	CRC-16	✘	✘	✓	✓	✓	✘
TC_DSU_017	MD: get latest SWP SLP_SWEEP packet	0	1	0* / 1†	0x700	0x3	0	0x2* / 0xA†	-* / Time†	0x11	-	CRC-16	✘	✘	✓	✓	✓	✘
TC_DSU_018	MD: get latest SWP WAVE packet	0	1	0* / 1†	0x700	0x3	0	0x2* / 0xA†	-* / Time†	0x12	-	CRC-16	✘	✘	✓	✓	✓	✘
TC_DSU_019	MD: get latest SWP SIP_SWEEP packet	0	1	0* / 1†	0x700	0x3	0	0x2* / 0xA†	-* / Time†	0x13	-	CRC-16	✘	✘	✓	✓	✓	✘
TC_DSU_020	MD: get latest SWP SIP_TRACK packet	0	1	0* / 1†	0x700	0x3	0	0x2* / 0xA†	-* / Time†	0x14	-	CRC-16	✘	✘	✓	✓	✓	✘
TC_DSU_021	MD: get latest SWP CONFIG packet	0	1	0* / 1†	0x700	0x3	0	0x2* / 0xA†	-* / Time†	0x15	-	CRC-16	✘	✘	✓	✓	✓	✘
TC_DSU_022	CTECS: enable script scheduler	0	1	0* / 1†	0x700	0x3	0	0x2* / 0xA†	-* / Time†	0x16	-	CRC-16	✘	✘	✓	✓	✓	✘
TC_DSU_023	CTECS: disable script scheduler	0	1	0* / 1†	0x700	0x3	0	0x2* / 0xA†	-* / Time†	0x17	-	CRC-16	✘	✘	✓	✓	✓	✘
TC_DSU_024	CTECS: set script scheduler	0	1	0* / 1†	0x700	0x3	0	TODO	-* / Time†	0x18	DU_DSU_024	CRC-16	✘	✘	✓	✓	✓	✘
TC_DSU_025	CTECS: list available scripts	0	1	0* / 1†	0x700	0x3	0	0x2* / 0xA†	-* / Time†	0x19	-	CRC-16	✘	✘	✓	✓	✓	✘
TC_DSU_026	CTECS: create new script	0	1	0* / 1†	0x700	0x3	0	TODO	-* / Time†	0x1A	DU_DSU_026	CRC-16	✘	✘	✓	✓	✓	✘
TC_DSU_027	CTECS: remove script	0	1	0* / 1†	0x700	0x3	0	TODO	-* / Time†	0x1B	DU_DSU_027	CRC-16	✘	✘	✓	✓	✓	✘
TC_DSU_028	CTECS: feed script PNTscript1v1.cmd	0	1	0* / 1†	0x700	0x3	0	0x2* / 0xA†	-* / Time†	0x1C	-	CRC-16	✘	✘	✓	✓	✓	✘
TC_DSU_029	CTECS: feed script TECscript1v1.cmd	0	1	0* / 1†	0x700	0x3	0	0x2* / 0xA†	-* / Time†	0x1D	-	CRC-16	✘	✘	✓	✓	✓	✘
TC_DSU_030	CTECS: feed script TECscript2v1.cmd	0	1	0* / 1†	0x700	0x3	0	0x2* / 0xA†	-* / Time†	0x1E	-	CRC-16	✘	✘	✓	✓	✓	✘
TC_DSU_031	CTECS: feed script S4script1v1.cmd	0	1	0* / 1†	0x700	0x3	0	0x2* / 0xA†	-* / Time†	0x1F	-	CRC-16	✘	✘	✓	✓	✓	✘
TC_DSU_032	CTECS: feed script S4script2v1.cmd	0	1	0* / 1†	0x700	0x3	0	0x2* / 0xA†	-* / Time†	0x20	-	CRC-16	✘	✘	✓	✓	✓	✘
TC_DSU_033	CTECS: feed script S4script3v1.cmd	0	1	0* / 1†	0x700	0x3	0	0x2* / 0xA†	-* / Time†	0x21	-	CRC-16	✘	✘	✓	✓	✓	✘
TC_DSU_034	CTECS: feed script S4script4v1.cmd	0	1	0* / 1†	0x700	0x3	0	0x2* / 0xA†	-* / Time†	0x22	-	CRC-16	✘	✘	✓	✓	✓	✘

Table 18: listing of DSU TCs.

Table 19 displays the structure of DSU TC-specific responses.

		PACKET PRIMARY HEADER							PACKET DATA FIELD		
			PKT ID			PKT SEQ CTRL				USER DATA FIELD	
		VERSION NO.	PKT TYPE	SEC HDR FLAG	APID	SEQ FLAGS	SEQ COUNT	PKT DATA LENGTH	SECONDARY HEADER	DATA	CHECKSUM
	Length (bits)	3	1	1	11	2	14	16	64	Varies	16
Identification	**Name**										
TM_DSU_001	DSU: get time	0	0	1	0x601	0x3	Varies	0x1A	Acting time	DD_DSU_001	CRC-16
TM_DSU_002	DSU: report MDR transfer status	0	0	1	0x602	0x3	Varies		Acting time	DD_DSU_002	CRC-16
TM_DSU_005	DSU: report latest MDR transfer	0	0	1	0x603	0x3	Varies		Acting time	DD_DSU_005	CRC-16
TM_DSU_025	CTECS: list available scripts	0	0	1	0x606	0x3	Varies		Acting time	DD_DSU_025	CRC-16

Table 19: listing of DSU TC-specific response (TM) formats.

Table 20 displays the structure of the data field in DSU TCs that require parameters.

DU_DSU_004		
Field	**Len (bits)**	**Format**
TODO	TODO	
Total	**0**	

DU_DSU_024		
Field	**Len (bits)**	**Format**
TODO	TODO	
Total	**0**	

DU_DSU_026		
Field	**Len (bits)**	**Format**
TODO	TODO	
Total	**0**	

DU_DSU_027		
Field	**Len (bits)**	**Format**
TODO	TODO	
Total	**0**	

Table 20: listing of DSU uplink data field formats.

Table 21 displays the structure of the data field in specific responses to DSU TCs.

DD_DSU_001		
Field	**Len (bits)**	**Format**
GPS time (DSU view)	64	GPS time
Free-running clock	64	GPS time
Acting time flag	8	bool
Total	**136**	

DD_DSU_002		
Field	**Len (bits)**	**Format**
TODO	TODO	
Total	**0**	

DD_DSU_005		
Field	**Len (bits)**	**Format**
TODO	TODO	
Total	**0**	

DD_DSU_025		
Field	**Len (bits)**	**Format**
TODO	TODO	
Total	**0**	

Table 21: listing of DSU downlink data field formats.

The following is a description of DSU TCs:

- **TC_DSU_001** (**LOREM IPSUM**): LOREM IPSUM

The following is a description of DSU TC responses:

- **TM_DSU_001** (**LOREM IPSUM**): LOREM IPSUM