Herschel PACS

DPU OBS
User Manual

Document Ref.: PACS-CR-UM-024

Issue: 3.4

Prepared by: Stefano Pezzuto Date: 8th June 2009

Approved by: Renato Orfei

Contents

Document Status Sheet

Document Title: DPU On Board Software User Manual			
Issue	**Revision**	**Date**	**Reason for change**
Draft 1		9th July 2002	First version for AVM acceptance test software version 1
Draft 2		31st July 2002	Changes in the code after the AVM acceptance test @ IFSI – 10/11 July 2002 software version 2
1	0	6th September 2002	First issue for the AVM delivery @ MPE – 9th September 2002
1	1	2nd October 2002	Upgraded during AVM delivery @ MPE – 9/20 September 2002 software version 3
1	2	19th December 2002	New version of the OBS software version 4
1	3	31st March 2003	New delivery of the AVM software version 5
1	4	8th August 2003	Software updates
1	5	15th September 2003	Changed 1355 interrupt handling software version 6
1	6	16th March 2004	New PS-ICD and new features in the OBS software version 7.01
1	7	15th September 2004	Software version 7.62
1	8	12th January 2005	RID from CGS
1	9	6th May 2005	Software version 8.00
2	0	18th April 2006	Software version 8.31 (FM)
2	1	20th July 2006	Software version 8.32
2	2	10th August 2006	Software version 8.33
2	3	28th September 2006	Software version 8.34
2	4	30th October 2006	Software version 8.36
2	5	4th December 2006	Software version 8.39
2	6	16th March 2007	Software version 8.45
2	7	5th April 2007	Preparation for DRB software version 8.46 - FM
2	8	20th April 2007	Software version 8.47
2	9	5th October 2007	Version 2.8 was issued by mistake without page numbering. This version is a copy of the previous one but with pages numbered
3	0	10th June 2008	Software version 9.00
3	1	23rd December 2008	Software version 9.01
3	2	4th February 2009	Software version 9.02
3	3	19th February 2009	Software version 9.03
3	4	8th June 2009	Software version 9.04

Document Change Records

Document Title: DPU On Board Software User Manual	
Document Reference Number: PACS-CR-UM-024	
Document Issue/Revision Number: 3/4	
Section	**Reason For Change**
1.2	Upgraded
1.5.1	New version of AD-8
1.5.2	New version of RD-3, RD-4, RD-5, RD-6 and RD-7
2.2	Upgraded
4.1	Upgraded
6.3.1	Upgraded for the changed DMC sequence (SCR-1551)
6.4	Changed OBCP#32 and DMC sequence #19 (SCR-1551)

ΩΣΠΕΡ ΣΑΡΜΑ ΕΙΚΗ ΚΕΧΥΜΕΝΩΝ Ο ΚΑΛΛΙΣΤΟΣ ΚΟΣΜΟΣ

1 Introduction

Heraclitus spent all his life in Efeso, maybe between 520 and 460 bC. About him nothing is known precisely. We are left with 125 fragments written, likely, not before 479/478. The meaning of them is not clear, some are interpreted in two, or more, completely opposite ways, others have the same beauty of the ZEN koan. We do not have an "user manual" for Heraclitus' fragments but we are still enlightened reading them after 2500 years.

This document is intended to leave no "gray area" in how the onboard software of the DPU inside the PACS instrument works. As a consequence, the reader should be able, hopefully, to completely understand the document content without the need to "interpret" it. The same day PACS is definitively switch-off, this document will be useless and forgotten.

1.1 Intended readership

The warm electronics of PACS, one of the three instruments of the Herschel satellite, consists of 4 computers: the DPU, the DEC and 2 SPU. This document explains how to operate the DPU OBS and since after the integration with the other subsystems the only way to operate PACS will be interacting directly with the DPU, anybody involved in PACS operations is a potential reader of this document and can benefit of reading it.

1.2 Applicability statement

This issue is upgraded to OBS Version 9.04. The software version ID is part of the DPU housekeeping sent in all the HK packets in every observing/operative mode (see Section 8.1).

1.3 Purpose

1.3.1 Purpose of the document

This document explains how to use the DPU OBSW: the reaction to each received TC and the generation of TM packets. Details are given to what happens inside the DPU when necessary to clarify the DPU behaviour.

1.3.2 Purpose of the software

The DPU is the only data interface of PACS with the spacecraft CDMS, so the main capabilities of the OBS is related with TC and TM handling. TC packets are received, checked, interpreted, translated into instrument instructions and sent to the specific subsystem, or internally processed. From the subsystems the DPU receives science data and all the HK values, which are used for monitoring the instrument behaviour.

A check is performed on some of the HK and if the corresponding critical values are reached, the OBS starts the pre-defined autonomy functions in order to prevent any damage to the instrument. Depending on the severity of the detected anomaly, the measurement could be stopped and/or DPU could ask the CDMS to enter some specific operative mode, or to switch off the instrument.

Furthermore, the OBS has to manage the uploading and downloading of part of the processor memory: this allows to upgrade the OBS as well as all the subsystems parameters tables.

The subsystems run their own programs: if a new image is required the OBS is in charge of receiving the memory load commands and of passing them to the appropriate subsystem.

1.4 How to use this document

The first sections of this manual are dedicated to the instrument switch-on: Section 2 gives a short description of boot SW and explains how to start the OBS, including the short functional test. Section 3 describes how to upload a new image, either using the boot SW or exploiting the patching facility of the OBS.

Section 4 describes the use of four OBCPs: activation of the 1355 links; writing the OBS in EEPROM; handover between boot SW and application SW in the subsystems; generation of dummy science data. Section 5 gives an overview of the autonomy functions, their description and when to activate them.

Section 6 is dedicated to the main three kinds (services) of telecommands: Service 1, through which the DPU reports if a telecommand has been accepted and executed; Service 8 which is used to send one single command to the subsystems, including the DPU itself; and Service 18 which is dedicated to OBCP, the main mechanism to command changes of PACS observing/operating modes and science observations.

The other services of AD–2 that are based on telecommands are described in Section 7, for instance memory management. Section 8 contains the description of the remaining services, those that are not directly based on telecommands (HK, events and science data transmission).

Section 9 describes for each service in the order followed inside AD–2, the precise structure of telecommand packets and of telemetry packets; this section is thought for expert people who has to code the TC and to interpret the TM packets (for instance to prepare MIB).

Appendix A contains a description of the test DPU can do on its own memory, while Appendix B gives some other useful informations (like memory architecture file and Makefile). Finally, in Appendix C the whole content of the HK packets is given.

1.5 Related documents

1.5.1 Applicable documents

Ref.	Name	Number/version/date
AD–1	Space engineering - Software - Part 1: Principles and requirements For the OBS development PACS adopts the standard given in this document tailored to be fully equivalent to RD–1.	ECSS-E-40 28 November 2003
AD–2	Herschel/PLANCK Packet Structure Interface Control Document	SCI-PT-ICD-07527 Issue 5.0. 20 July 2004
AD–3	DPU/ICU On Board Software Product Assurance Plan	IFSI/OBS/PL/2000-001 Issue 1.1. 2 April 2001
AD–4	DPU OBS Software Specification Document	PACS-CR-SR-013 Issue 3.1. 5 May 2006
AD–5	Interface Control Document DEC/MEC–DPU	PACS-CL-ID-003 Issue 3.5. 3 October 2003
AD–6	SPU High Level Software to DPU Interface Description	PACS-TW-ID-001 Issue 6.0. 13 December 2005
AD–7	PACS and LFI SPU_SUSW - DPU_ASW Protocol	PACS-IC-TN-001 Issue 01, Rev. A. 2 August 2001
AD–8	PACS Failure Detection Isolation and Recovery	PACS-ME-GP-002 Issue 1.3. 19 April 2008

1.5.2 Reference documents

Ref.	Name	Number/version/date
RD–1	Guide to applying the ESA software engineering standards to small software projects	BSSC(96)2 Issue 1. May 1996
RD–2	DPU/ICU Switch on Procedure	CNR.IFSI.2001.TR01 Draft 3. 21 March 2001
RD–3	PACS OBCPs and DMC Sequences	PACS-ME-LI-005 Issue 2.1. 8 June 2009
RD–4	DEC/MEC User Manual	PACS-CL-SR-002 Issue 4.5. 30 July 2008
RD–5	Herschel DPU/ICU Boot SW Telemetry/Telecommands Packets User Manual	HERS-GEN-MA-CGS-001 Issue 3. 5 May 2006
RD–6	FM Photometer Phocal Plane Unit User's Manual	SAp-PACS-MS-0616-06 Issue 1.0. 25 February 2009
RD–7	DPU OBSW Release notes	PACS-CR-TN-032 Issue 11. 4 June 2009

1.6 Conventions

- `Text in this font` is a message reported on the screen, for instance a SCOS2000 error/event message;
- A number like `0x...` is an hexadecimal number. To indicate a generic number letters from `a` to `f` are not used;
- this_text is the name of a function of the OBS;
- all the TC and TM packets have a checksum at their end, computed on the whole packet content. This checksum is always indicated with CRC. In some cases, when the packet transport memory data, an additional checksum is computed on these data only, and is indicated with crc;
- *P#i* is the i-th parameter, for instance of a TC; this font is also used to identify the value of a parameter, e.g. the length of the packet is *Length*;
- the symbol ↯ preceeds the description of the content of the TM report (1,8) in case the execution of a command fails.

1.7 Problem reporting instructions

An SPR can be submitted at the following address:
http://www.rssd.esa.int/herschel_webapps/servletsuite/ProblemReportServlet?area=pacs

In case of errors/anomalies, it would be very useful to know: if the DPU was in standalone or with the subsystems on; the values of the DPU HK; in case a telecommand had just been sent, all the TM packets of Service 1 generated by the DPU; if events have been observed just before and after the error/anomaly.

2 Overview

Once switched on, the DPU starts the execution of the boot SW: in nominal case (see next subsection) an event (5,1) is sent. To start the application SW send the command "Force Boot".

Once started, the OBS sends HK packets: the nominal one (every two seconds) and the additional one (see Section 8.1).

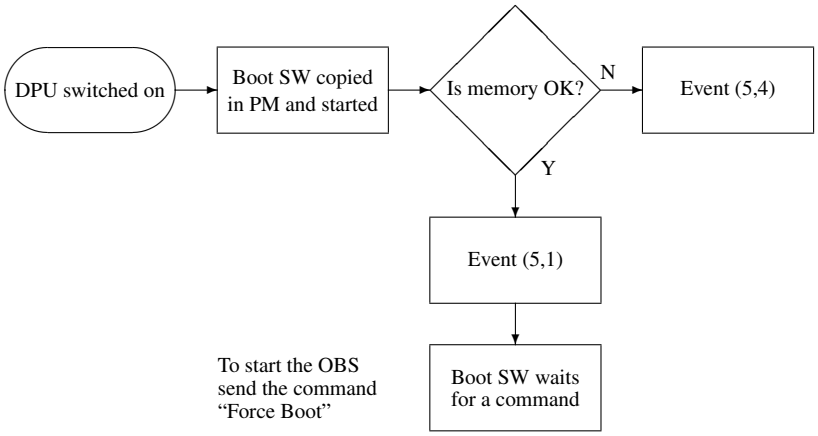
The 1355 links are not activated yet so that the DPU functions are: communications with the spacecraft, sampling of DPU HW and SW HK and their check, shipping of the HK packets. No communications is possible with the subsystems.

To activate a link, use the "Start 1355 link" procedure (see Section 4.2). Once a link is active, the DPU is ready to send commands to the subsystem attached to that link, and to receive packets from it. The kind of packets depends on the subsystem: acknowledgment of commands, HK and diagnostic HK from the DEC; acknowledgment of commands, HK and science data from the SPU.

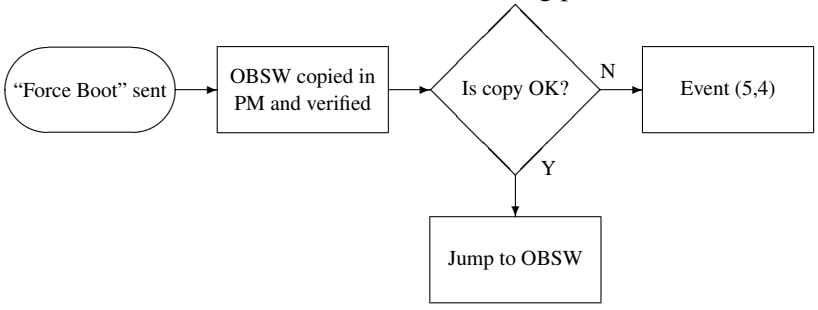
Some checks are performed on the status of the subsystems, in particular HW HK are checked against the nominal range. If a value is out of limit, an event is issued and, if defined, a corrective action (autonomy function) is started (see Section 5). However the autonomy functions must be enabled with a DPU command (see Section 6.3.3) otherwise the function is not started even if the corresponding HK value is out of limit. This is because after a transition from one state to another one (for instance from photometry to spectroscopy and viceversa) some HK values can be nominally out of limits and no action should be taken by DPU. As a consequence enabling/disabling autonomy functions should be considered part of the operations performed to put PACS in the required operative mode.

2.1 Handover between boot SW and OBSW

In the following picture the transition form boot SW to OBSW in nominal conditions is shown



If an error occurres an event (5,4) is generated[1]. Otherwise the event (5,1) is sent and the OBSW can be started with the command "Force Boot", see the following picture



Since the EEPROM is splitted in two partitions, the Force Boot command accepts one parameter: the partition from which the OBSW has to be copied. If no parameters are sent, the boot SW looks for an image in the first

[1] For the content of boot SW events see RD–5.

partition and, if not found, in the second one. If both partitions contain a corrupted image an event (5,4) is sent, otherwise the found OBSW is started

Important! Note that before copying the OBSW in PM, the boot SW controls the integrity of EEPROM checksums page by page. If a checksum is not correct, the boot SW reads the corresponding page in the other partition and if the checksum is correct the copying procedure goes on. In case the same version of the OBSW is written in both partitions this operation makes more robust the copying. On the other hand, if two different OBSW versions are written in the two partitions, then this procedure will make a "correct" but inconsistent copy of EEPROM in PM. However, before OBSW is started, boot SW will note that the overall checksum is not consistent and an event (5,4) will be generated.

2.2 Short functional test

Once the application program is started the DPU begins to generate the nominal (non prime) HK packets at a rate of 0.5 Hz, and the the additional HK packet at a rate of 0.1Hz.

Here is a list of expected values, or range of values, for the DPU HK in the nominal HK packet. Values are both in raw and in calibrated format

Name	Raw value	Calibrated value
DP_VOL_25P	[1832—2240]	2.5V±10%
DP_VOL_5P	[3048—3726]	5V±10%
DP_VOL_15P	[3048—3726]	+15V±10%
DP_VOL_15N	[3048—3726]	–15V±10%
DP_T	[313—3758]	[–40,+70]°C
DP_SPS_LINK	0	OFF
DP_SPL_LINK	0	OFF
DP_DMC_LINK	0	OFF
DP_SPUS_CMD	0	SS OFF
DP_SPUL_CMD	0	SS OFF
DP_DMC_CMD	0	SS OFF
DP_SPUS_HK	0	SS OFF
DP_SPUL_HK	0	SS OFF
DP_DMC_HK	0	SS OFF
DP_STATUS	0	0000000000
DP_WHICH_OBCP	63	NO_OBCP
DP_AF_STATUS	0x200400	Bit 22 and 11 (counting from 1) are 1
DP_WORK_LOAD	[200,400]	[2,4]%
DP_TM_RATE	4	NO PRIME
DP_SW_VERS_ID	9	9
DP_SW_SUBVERS_ID	4	4
DP_TC_LOST	0	0
DP_HK_LOST	0	0
DP_EVENT_LOST	0	0
DP_GEN_TM_LOST	0	0
DP_COM_REC_DPU	0	0
DP_COM_REJ_DPU	0	0
DP_COM_DMC	0	0
DP_COM_SPUS	0	0
DP_COM_SPUL	0	0

Beside the SW Version ID the OBSW is characterized also with the date and time at which the running image was compiled: to get this information send a Memory Dump command (see Section 7.1.2) with the

following parameters:

Memory ID	0x1600
Start Address	0
Length	21

The expected output is:
`4A 75 6E 20 20 38 20 32 30 30 39 00 31 30 3A 32 30 3A 35 31 00`
which corresponds to the ASCII representation of the string: *Jun 8 2009*\0*10:20:51*\0 (\0 is the end-of-string character).

A quick check on the correctness of the whole content of the PM can be done by sending these two Memory Check commands (see Section 7.1.3):

seg_init	
Memory ID	0x0100
Start Address	0x4000
Length	0x1551
Expected checkusm $\Longrightarrow$	0x302C
seg_pmco	
Memory ID	0x0100
Start Address	0x5551
Length	0xFFFF
Expected checkusm $\Longrightarrow$	0x69E1

3 Installation

This section is included in the release notes (see RD–7), given with each release of the OBS.

4 Operations

In this section it is explained how to execute particular operations based on OBCP whose general description is given in Section 6.4. An explanation of TM packets, especially for what concerns telecommand verification and execution can be find in Section 6.

4.1 Writing in EEPROM

The OBS is resident in EEPROM and copied in PM by the boot SW when the command Force Boot (see Section 2.1) is sent. After a new image has been uploaded with one of the two mechanism described in Section 3, the new OBSW is running until the DPU is switched off, then PM content is lost. To store permanently the OBSW in EEPROM it is not possible to use the memory load of service 6 described in Section 7.1.1 because this memory requires special operations and it is written in a format that depends on the boot SW. So, while the user can still use service 6 to dump and to check the content of the EEPROM, the DPU will deny any request to load memory. To save a new version of the software use this dedicated procedure whose ID is 20. The number of parameters is variable, between 5 (nominally) and 25, the maximum number for all OBCP. The meaning of the parameters is the following

1. Start Address: start of *seg_init* (see Appendix B.1), for version 9.04 type `0x4000`;
2. End Address: can be found in the `.MAP` file generated during the compilation of the code. For version 9.04 type `0x10CB4`;
3. Partition. Its value is 1 (primary partition) or 2 (secondary partition);
4. Safety parameter. This parameter has no operational meaning but it is used for safety reason: it must have the value `0x19660502` and must be sent every time this OBCP is started. It ensures that this procedure is not started by mistake. In case this value differs from the required number, the execution is stopped and a TM report (1,8) is generated (`Wrong EE PAR`);
5. Number of pages to avoid. This number tells the DPU how many EEPROM pages can not be written (for failure reasons). Nominally this number should be zero and since the maximum number of parameters is 25, not more than 20 pages can be given. If 0, there are no other parameters to write;
6. Index of first EEPROM page to avoid (if parameter #5 is at least 1);
7. Index of second EEPROM page to avoid (if parameter #5 is at least 2);

... and so on

As said, in nominal case parameter #5 is 0 so that only 5 parameters are required.

The procedure takes about 10/15 seconds to be completed. Once started DP_STATUS (see Table **DPU Global Status** on page 67) becomes `x11xxxxxx` and is set back to `x00xxxxxx` at the end.

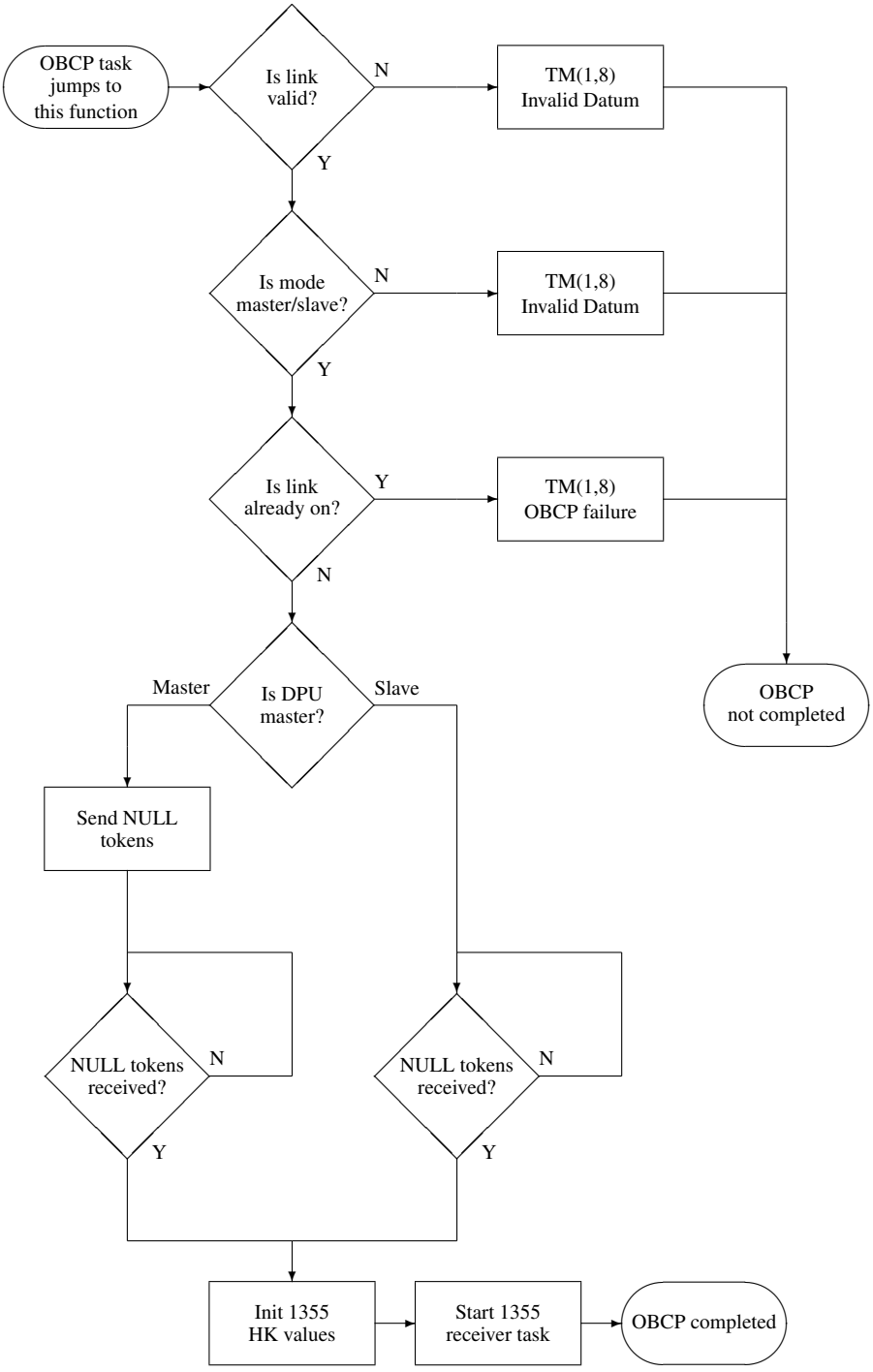
NOTE: the EEPROM has 256 usable pages or 128 pages per partition, 127 taking into account that one page is reserved for the interrupt table. Each page has 1024 words and begins with an header 7 words long, so that 1017 words can be used to write the image. Now, the EEPROM word is 32 bit while PM word is 48 bit, so that each page contains 1017×2/3=678 PM words. Since start address is `0x4000` **end address can not be greater than** `0x1905A`. The OBCP checks that partition, end address and index of EEPROM pages have compatible values, otherwise the OBCP is not executed.

4.2 How to start/restart 1355 links

The communications between the DPU with the subsystems is done via three 1355 links. When switched-on, the three links are off and must be activated with this procedure: while a link is off, neither it is possible to send commands to the subsystem attached to that link, nor the DPU can receive data. The ID of the procedure is 19, it accepts 2 parameters: the first identifies the link, the second defines the DPU as Master or Slave according to the following table

1st parameter	Value	2nd parameter
DEC	0	
Blue SPU	1	Master
Red SPU	2	Slave

Here is the sequence of operations



In mode Master the DPU starts sending NULL tokens through the link and waits until NULL tokens are received. In mode Slave the DPU waits until NULL tokens are received. No timeout mechanism is implemented, but the procedure can be stopped at any moment sending the command Stop OBCP (see Section 6.4).

In the following table it is shown the change in the 1355 HK status after the link is on. DEC link is taken as an example but the transition is exactly the same also for the two SPU:

Name	Before link is on		After link is on	
	Raw value	Calibrated value	Raw value	Calibrated value
DP_DMC_LINK	0	OFF	1	ON
DP_DMC_CMD	0	SS OFF	1	SS ENABLED
DP_DMC_HK	0	SS OFF	2	No NEW HK

- DP_DMC_LINK should remain ON at all times. One exception is during the handover between LLSW to HLSW (see next section);
- DP_DMC_CMD should remain SS ENABLED; it goes SS STOPPED if the DPU receives a NACK (see the scheme on page 84), in this case enable the link again with the command Set Function (see Section 6.3.3) before sending any command to the subsystems. After the handover between LLSW and HLSW the status returns to OFF until the link is started again;
- DP_DMC_HK becomes NEW HK once HK packets start arriving. Note that the LLSW of the three subunits does not send HK packets, so in this case No NEW HK is the nominal value. For SPU, during science data processing, the rate of sending HK packets to DPU is lower than nominal so also in this case the status No NEW HK is expected.

4.3 Start HLSW

The command used to start the application software inside the subunits requires special attention since the LLSW does not send any acknowledgment and the HLSW, once started, resets the 1355 link. For these reasons a specific procedure (ID 21) has been designed (see next figure).

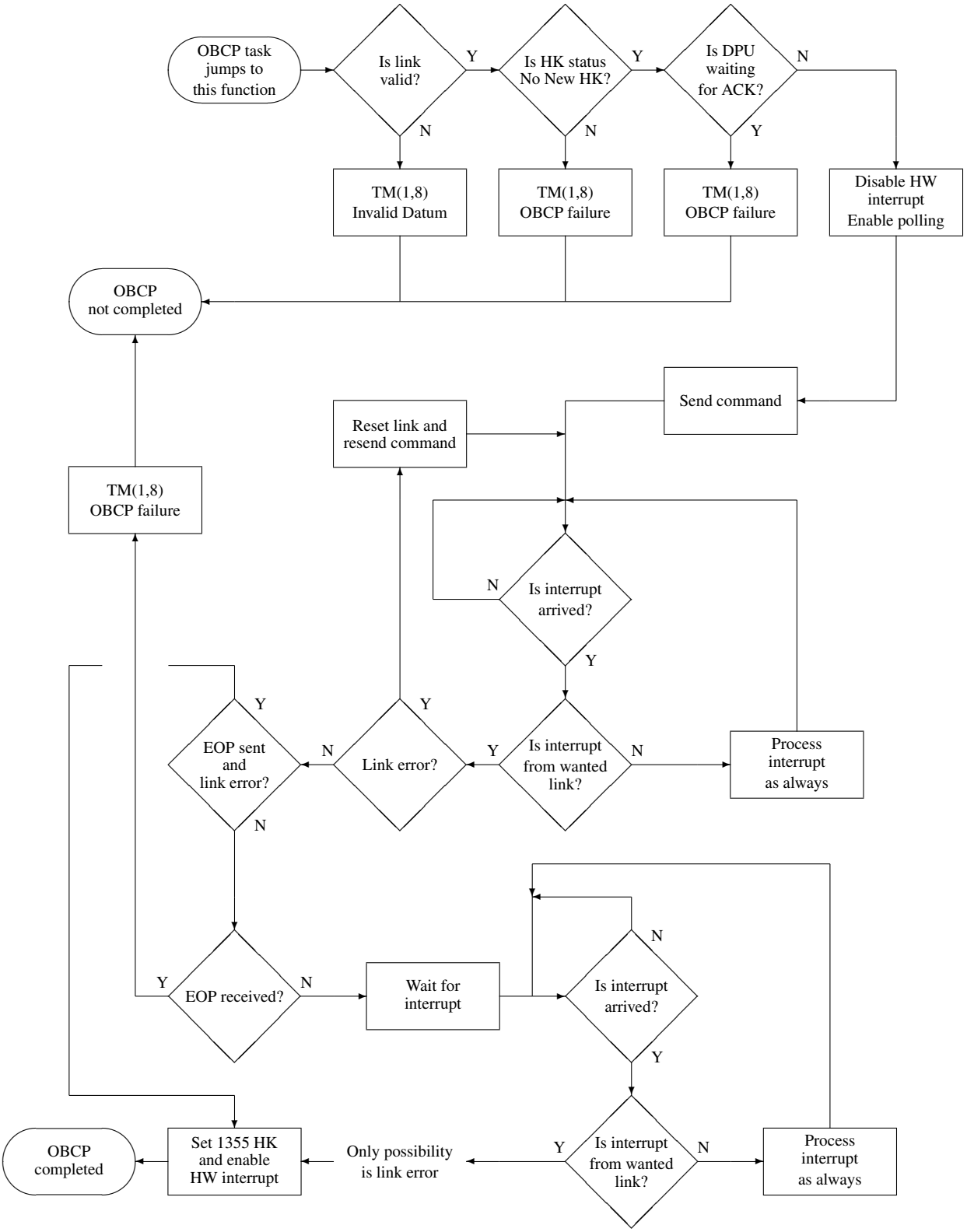
Three parameters are accepted but only one is directly used by the DPU, the first one which identifies the link: 0 for DEC, 1 for the blue SPU and 2 for the red SPU. Any other value makes the procedure issue an error (1,8) `Invalid DATA: Invalid DATUM` and exit. The other two parameters are ignored by the DPU and directly sent to the subsystem.

When the LLSW is running, the corresponding HK status is `NO NEW HK`. If the status is different, the procedure is aborted with the error (1,8) `Illegal STATUS: Not compl OBCP`. This TM report is also generated in case the DPU receives a packet from the subunit, since for this command no acknowledgment is expected. Another check done before sending the command is that DPU is not waiting an ACK for a previous command.

Once the HLSW starts, the link with the DPU is reset so that the disconnect error counter of the DPU HK is incremented and the link status is set to `OFF`. At this point, the communications with the subunit can be established again by mean of the procedure “Start 1355 link”.

4.4 Generate science dummy packets

This OBCP is intended for test purposes. The DPU generates dummy science packets at a commanded TM rate. The content of the packets is costant, an index running from 0 to 249. First parameter `P#1` gives the total duration of the procedure in seconds, while the second parameter `P#2` gives the number of packets per second to generate. In this way different buslists can be tested with the DPU in standalone configuration.


OBCP task jumps to this function
Is link valid?
Y
Is HK status No New HK?
Y
Is DPU waiting for ACK?
N
Y
N
N
TM(1,8) Invalid Datum
TM(1,8) OBCP failure
TM(1,8) OBCP failure
Disable HW interrupt Enable polling
OBCP not completed
Send command
Reset link and resend command
TM(1,8) OBCP failure
N
Is interrupt arrived?
Y
Y
Y
EOP sent and link error?
N
Link error?
Y
Is interrupt from wanted link?
N
Process interrupt as always
N
N
Y
EOP received?
N
Wait for interrupt
Is interrupt arrived?
Y
OBCP completed
Set 1355 HK and enable HW interrupt
Only possibility is link error
Y
Is interrupt from wanted link?
N
Process interrupt as always

5 Autonomy functions

Autonomy Functions (AF) execute actions taken autonomously by DPU in case of non nominal conditions. They are started on the base of out-of-limit (OOL) HK values or on internal conditions. What means OOL depends on the internal state of PACS and of its subsystems, for this reason **all the AF are disabled at start-up**. It is responsability of the user to enable (see Section 6.3.3) each function at the right moment. Exceptions are AF 11 and 22 (see table below): these functions are always enabled (but can be disabled like every other AF).

Note that in order to generate the event go_SAFE or SWITCH_OFF, a HK must stay out of limits for 3 consecutive HK packets. This counter is set to 0 again if a HK comes back in limit 1 or 2 packets later. This has nothing to do with the other counter defined for HK, i.e. the counter that avoids to generate an out-of-limit event if a certain timeout has not expired from the last out-of-limit event (counter that is anyway set to 0 at present).

Here is the table showing, for each AF, the ID and the HK that the function controls (BOL and HW SPU HK are numbered according to RD–4). Every function is explained in the next subsections.

ID	Name	Involved HK (ID and name)	
1	generate_event_SPU	DEC HK ID = 420, 421, 422	DMC_SPU_SWL_TEMP DMC_SPU_LWL_TEMP DMC_SPU_PS_TEMP
2	generate_event_DEC	DEC HK ID = 413, 414	DCDC_TEMP DSP_TEMP
3	monitor_stable_counter_DEC	DEC HK ID = 455, 456, 457, 458, 464	DM_SF_IND PM_SF_IND DM_DF_IND PM_DF_IND LAST_ERR_ID
4	monitor_counter_DEC	DEC HK ID = 232, 233	BLUE_ENC_PAC RED_ENC_PAC
5	monitor_counter_SPEC	DEC HK ID = 228, 229	DECB_REC_PAC DECR_REC_PAC
6	monitor_counter_PHOT	DEC HK ID = 234	BOL_REC_PAC
7	monitor_counter_SPS	SPS HK ID = 3	CIB
8	monitor_stable_counter_SPS	SPS HK ID = 14	MEM_CNTS
9	monitor_counter_SPL	SPL HK ID = 3	CIR
10	monitor_stable_counter_SPL	SPL HK ID = 14	MEM_CNTS
11	generate_event_DPU	DPU HK ID = 0, 1, 2, 3, 4	VOL_25_P_N VOL_5P_N VOL_15P_N VOL_15N_N T_N

12	generate_event_BOL_BIAS	BOL HK ID = 14, 21, 46, 53, 78, 85, 110, 117, 142, 149, 174, 181	VDDPROT_CLB1 VDDPROT_BUB1 VDDPROT_CLB2 VDDPROT_BUB2 VDDPROT_CLB3 VDDPROT_BUB3 VDDPROT_CLB4 VDDPROT_BUB4 VDDPROT_CLR1 VDDPROT_BUR1 VDDPROT_CLR2 VDDPROT_BUR2
13	generate_event_BOL_WE	BOL HK ID = 192, 193, 194, 195, 481, 482, 483, 484, 485, 486, 487	TEMP_BOLC_R_1 TEMP_BOLC_R_2 TEMP_BOLC_R_3 TEMP_BOLC_R_4 TEMP_BOLC_R_5 TEMP_BOLC_B_1 TEMP_BOLC_B_2 TEMP_BOLC_B_3 TEMP_BOLC_DAQ TEMP_PSU_1 TEMP_PSU_2
14	generate_event_BOL_FPU	BOL HK ID = 498, 499, 500	TEMP_EV TEMP_FPU1 TEMP_FPU1
15	generate_event_BOL_I_RO	BOL HK ID = 23, 24, 55, 56, 87, 88, 119, 120, 151, 152, 183, 184	I_VSS_B_1 I_VSS_BU_B_1 I_VSS_B_2 I_VSS_BU_B_2 I_VSS_B_3 I_VSS_BU_B_3 I_VSS_B_4 I_VSS_BU_B_4 I_VSS_R_1 I_VSS_BU_R_1 I_VSS_R_2 I_VSS_BU_R_2
16	generate_event_BOL_I_Heater	BOL HK ID = 502, 503	HEAT_SP_SWT HEAT_EV_SWT

17	generate_event_pwr	BOL HK ID = 28, 29, 30, 60, 61, 62, 92, 93, 94, 124, 125, 126, 156, 157, 158, 188, 189, 190, 506, 507, 508	BC_PWR_ANA_P_1 BC_PWR_ANA_N_1 BC_PWR_DIG_1 BC_PWR_ANA_P_2 BC_PWR_ANA_N_2 BC_PWR_DIG_2 BC_PWR_ANA_P_3 BC_PWR_ANA_N_3 BC_PWR_DIG_3 BC_PWR_ANA_P_4 BC_PWR_ANA_N_4 BC_PWR_DIG_4 BC_PWR_ANA_P_5 BC_PWR_ANA_N_5 BC_PWR_DIG_5 BC_PWR_ANA_P_6 BC_PWR_ANA_N_6 BC_PWR_DIG_6 BC_PWR_ANA_P_7 BC_PWR_ANA_N_7 BC_PWR_DIG_7
18	generate_event_BOL_I_SP2	BOL HK ID = 501	HEATER_SP
19	generate_event_BOL_I_FPU	BOL HK ID = 504	HEATER_FPU
20	generate_event_DEC_SPC	DEC HK ID = 284, 352	DECB_DCDC_T3 DECR_DCDC_T1
21	generate_event_BOL_I_SP1	DEC HK ID = 501	HEATER_SP
22	verify_DMC_chksum	DEC HK ID = 212	DMC_CHECKSUM
23	1355_link_lost	DPU HK ID = 5, 6, 7	DPU_SPS_LINK DPU_SPL_LINK DPU_DMC_LINK

5.1 Counters that should stay stable: monitor_stable_counter_SPS (ID = 8), monitor_stable_counter_SPL (ID = 10)

These functions control that subsystems counters that should remain constant, eg counters of memory failures, are not incremented. A new value different from the last received causes the event `COUNTER Error` (see Section 8.2.1–10) to be generated, reporting the previous as well as the new value. The latter value is then used as new reference value.

Enable these functions at the end of the PACS Switch-on procedure.

5.1.1 monitor_stable_counter_DEC (ID = 3)

This AF is similar to the previous two already described. The only difference concerns the HK `LAST_ERR_ID` which is not initialized to 0 but to the last value received at the moment the AF is enabled. The reason for this different behaviour is that at switch-on it may be possible that this counter is incremented because of the activation procedure.

Enable this function at the end of the PACS Switch-on procedure.

5.2 Counters that should be incremented

These functions control that subsystems counters that should be regularly incremented do not remain constant. On reception of a new HK packet from a subsystem the DPU controls that the value of the counter is different from the previous one; if not so the event `COUNTER Error` (see Section 8.2.1–10) is generated. The content of the packet is the same as for counters that should not be incremented; however, in this case the old and the new values reported in the event are the same.

5.2.1 monitor_counter_SPS (ID = 7), monitor_counter_SPL (ID = 9)

These counters are related to the SPU HK CIB/CIR that are incremented for each new HK. In case a counter is not incremented the event `COUNTER Error` (see Section 8.2.1–10) is generated.

Enable these functions at the end of the PACS Switch-on procedure.

5.2.2 monitor_counter_DEC (ID = 4)

AF related to the two counters of packets sent from DEC to both SPU.

Currently there is no nominal instrument status when these two counters are expected to be regularly incremented, so this function should not be used.

5.2.3 monitor_counter_SPEC (ID = 5)

AF related to the two counters of packets sent from both DEC to MEC (probably this is the only section of this user manual where the difference between DEC and MEC is important).

Enable this AF once the instrument is set in spectroscopy observing mode. Disable otherwise. **This AF is disabled during the execution of the safe mode OBCP's**.

5.2.4 monitor_counter_PHOT (ID = 6)

Similar to the previous one but for the counter of packets sent from BOLC to MEC.

Enable this AF at the end of the PACS Switch-on procedure. **Note that this AF generates also the event `NOMINAL OFF` after the `COUNTER Error`**.

5.3 HW HK of DEC, SPU and DPU: generate_event_SPU (ID = 1), generate_event_DEC (ID = 2) and generate_event_DPU (ID = 11)

These functions are associated to the HW HK of SPU (temperatures) DEC (temperatures) and DPU (voltages and temperature). The allowed range is reported in the following table for DEC and SPU, and in Section 8.1 for DPU.

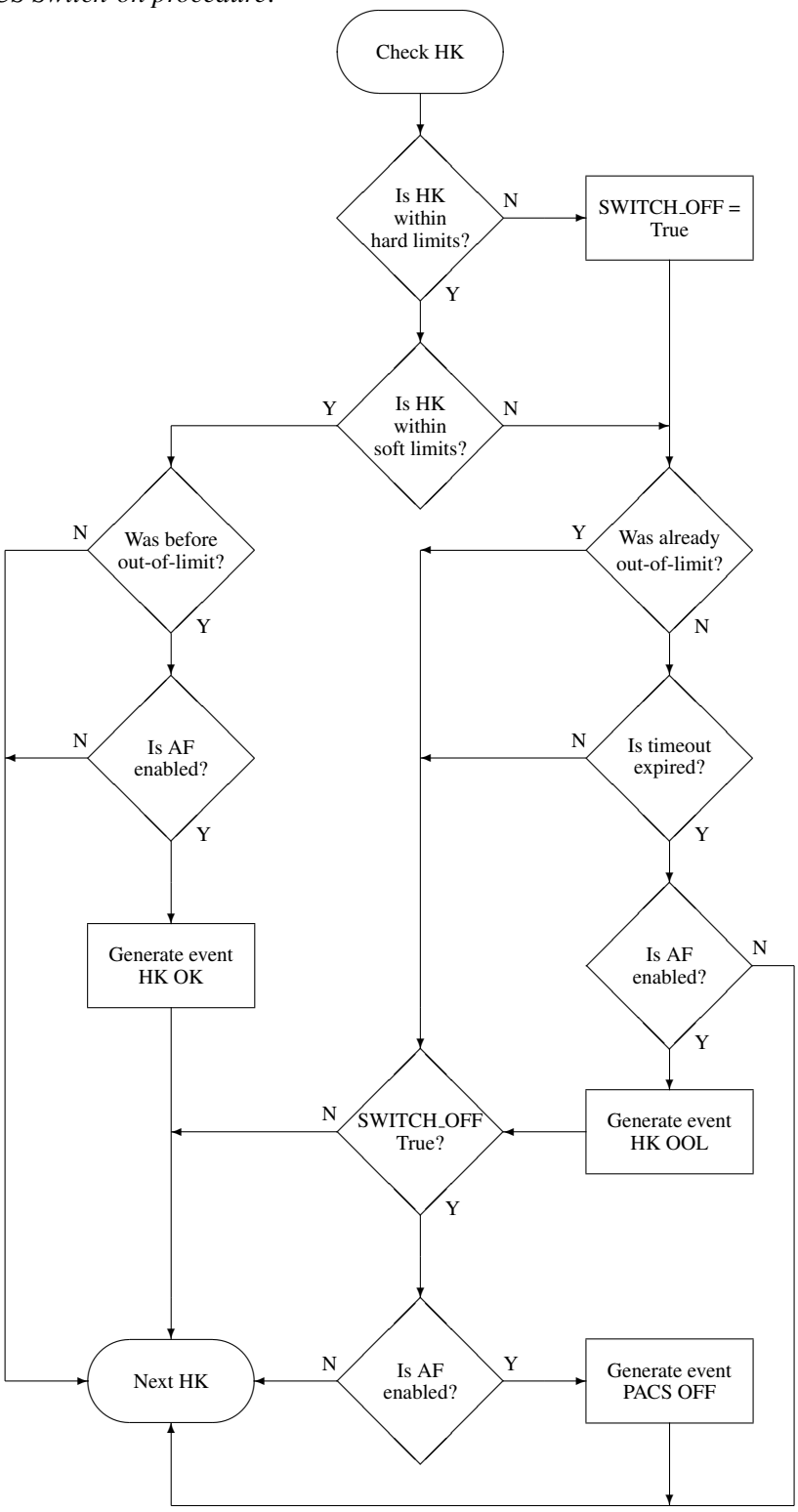
ID	Name	Allowed variability range	
		Soft limits	Hard limits
413	DCDC_TEMP	0x8E77—0xFEE1	0x8001—0xFF32
414	DSP_TEMP	0x8001—0xFE95	0x8001—0xFF33
420	SPU_SWL_TEMP	0x8001—0xFE95	0x8001—0xFF33
421	SPU_LWL_TEMP	0x8001—0xFE95	0x8001—0xFF33
422	SPU_PS_TEMP	0x8001—0xFE95	0x8001—0xFF33

If a HK goes outside the soft limits but is still within the hard limits an event (5,1) "HK SOFT" is generated (see next figure; note that the timeout mechanism is implemented but currently no HK uses it); once the HK comes back in limit an event (5,1) "HK OK" is generated. If hard limits are violated then **event**

PACS_NOMINAL_OFF is sent to the satellite.

Since the event `NOMINAL OFF` does not have parameters, the event HK SOFT is also generated in case of transition in-limit→out-of-limit.

AF 11 (DPU) is enabled by default (it should never be disabled). Enable AF 1 and 2 at the end of the PACS Switch-on procedure.



5.3.1 HW HK of DEC: generate_event_DEC_SPC (ID = 20)

This function follows the same rules as generate_event_DEC (see above) but should be used only once PACS is in SPEC mode: the allowed range is reported in the following table

ID	Name	Allowed variability range	
		Soft limits	Hard limits
284	DMC_DECB_DCDC_T3	0x8E77—0xFEE1	0x8001—0xFF32
352	DMC_DECR_DCDC_T1	0x8E77—0xFEE1	0x8001—0xFF32

Enable this AF once the instrument is set in spectroscopy observing mode. Disable otherwise. **This AF is disabled during the execution of the safe mode OBCP's**.

5.4 generate_event_BOL_BIAS (ID = 12)

This AF is associated to the bias voltages of the bolometers. If the AF is enabled and one or more biases goes outside the range given in RD–6, event GO_SAFE is generated.

Enable this AF once the instrument is set in photometry observing mode. Disable otherwise. **This AF is disabled during the execution of the safe mode OBCPs**.

5.5 generate_event_BOL_WE (ID = 13)

In case one of the temperatures monitored with this HK is outside the range [–30,+55] °C and the AF is enabled, the event `PACS NOMINAL OFF` is generated. See RD–6 for the raw values[2].

Enable this AF at the end of the PACS Switch-on procedure.

5.6 generate_event_BOL_FPU (ID = 14)

This function controls the three temperatures `TEMP-EV`, `TEMP-FPU1` and `TEMP-FPU2`: if one or more are out the range [260,320] mK the event `HK DEC SOFT` is generated; if the temperature is above 400 mK the event `GO SAFE` is also raised. See RD–6 for the raw values.

Enable this AF once the instrument is set in photometry observing mode. Disable otherwise. **This AF is disabled during the execution of the safe mode OBCPs**.

5.7 generate_event_BOL_I_RO (ID = 15)

This function is associated with the currents of the readout circuits. It controls that the value of the currents of the readout circuit are $-5 \leq I(\mu\mathrm{A}) \leq +5$ and those of the buffer units are $-2.5 \leq I(\mathrm{mA}) \leq +2.5$. If one value is outside its allowed range, the event `NOMINAL Off` is generated. See RD–6 for the raw values.

Enable this AF once the instrument is set in photometry observing mode. Disable otherwise. **This AF is disabled during the execution of the safe mode OBCP's**.

5.8 generate_event_BOL_I_Heater (ID = 16)

This function controls that the currents `HEATER-SP-SWITCH` and `HEATER-EV-SWITCH` are below 1.5 mA, otherwise the event `NOMINAL Off` is generated. See RD–6 for the raw values.

Enable this AF at the end of the PACS Switch-on procedure.

[2]Note that the lower limit is set to -30 °C according to SCR-1259, and not to -15 °C as reported in RD–6.

5.9 generate_event_pwr (ID = 17)

Function that controls the voltages on power lines. These values have soft limits (if violated a warning event is raised) as well as hard limits (if violated the event `NOMINAL Off` is sent). See RD–6 for the raw values.

Enable this AF once the instrument is set in photometry observing mode. Disable otherwise. **This AF is disabled during the execution of the safe mode OBCPs**.

5.10 generate_event_BOL_I_SP2 (ID = 18) and generate_event_BOL_I_SP1 (ID = 21)

These functions monitor the HK that reports the current running in the sorption pump heater; the first function (ID = 18) controls that the current is less than 30 mA; the second (ID = 21) controls that the absolute value of the current is less than 20 μA. In both cases, if the limits are violated the event `NOMINAL Off` is sent. See RD–6 for the raw values. Note that only one of the two functions should be enabled but in case AF#21 has higher priority than AF#18.

Enable AF#18 at the beginning of cooler recycling and disable once recycling is completed; **this AF is disabled during the execution of the safe mode OBCP's**. *Enable AF#21 at the end of the PACS Switch-on procedure and disable at the beginning of cooler recycling;* **this AF is enabled during the execution of the safe mode OBCP's**. See also Section 5.14.1.

5.11 generate_event_BOL_I_FPU (ID = 19)

Function that controls the current absorbed inside the FPU heater. If the value is greater than 75 μA the event `NOMINAL Off` is sent. See RD–6 for the raw values.

Enable this AF at the end of the PACS Switch-on procedure.

5.12 verify_DMC_chksum (ID = 22)

HK entry number 212 contains a checksum to verify the integrity of HK packet sent from DMC to DPU. For each packet DPU computes the checksum, excluding entry 212, and compares the result with the value of HK 212. If the two values are not equal the packet is not accepted and an event is generated (see Section 8.2.1–2). This AF is enabled by default and the user should never disabled it. However in case the DPU interfaces with a DMC model that runs an old version of the OBSW, before the checksum was implemented, this AF should be disabled before starting the link with DMC.

5.13 1355_link_lost (ID = 23)

In case one 1355 link is lost for an electrical error the DPU is physically disconnected from the corresponding susbsystem. In this case there is nothing to do but switch off the instrument. This AF generates such an event in case one link is lost.

Enable this AF at the end of the PACS Switch-on procedure.

5.14 Summary of AF activations

ID		After switch on	In photometry	In spectroscopy	During cooler recycling	go_SAFE OBCPs
1	GENERATE_EVENT_SPU	ON	ON	ON		X
2	GENERATE_EVENT_DEC	ON	ON	ON		X
3	MONITOR_STABLE_DEC	ON	ON	ON		X
4	MONITOR_COUNTER_DEC[1]	OFF	OFF	OFF		X
5	MONITOR_COUNTER_SPEC	OFF	OFF	ON		OFF
6	MONITOR_COUNTER_PHOT	ON	ON	ON		X
7	MONITOR_COUNTER_SPS	ON	ON	ON		X
8	MONITOR_STABLE_SPS	ON	ON	ON		X
9	MONITOR_COUNTER_SPL	ON	ON	ON		X
10	MONITOR_STABLE_SPL	ON	ON	ON		X
11	GENERATE_EVENT_DPU[2]	ON	ON	ON		X
12	EVENT_BOL_POLARIZATION	OFF	ON	OFF		OFF
13	EVENT_BOL_TEMP_WE	ON	ON	ON		X
14	EVENT_BOL_TEMP_FPU	OFF	ON	OFF		OFF
15	EVENT_BOL_CURRENT_RO	OFF	ON	OFF		OFF
16	EVENT_BOL_CURRENT_HEAT	ON	ON	ON		X
17	GENERATE_EVENT_PWR	OFF	ON	OFF		OFF
18	EVENT_BOL_CURRENT_SP2[3]	OFF	OFF	OFF	ON	OFF
19	EVENT_BOL_CURRENT_FPU	ON	ON	ON		X
20	GENERATE_EVENT_DEC_SPC	OFF	OFF	ON		OFF
21	EVENT_BOL_CURRENT_SP1[3]	ON	ON	ON	OFF	ON
22	VERIFY_CHECKSUM[4]	ON	ON	ON		X
23	1355_LINK_LOST	ON	ON	ON		X

[1]: this function is never used
[2]: enabled autonomously by DPU OBSW, do not change
[3]: see the following note on cooler recycling
[4]: enabled autonomously by DPU OBSW, see Section 5.2

This table shows when the user has to enable/disable the autonomy functions. All functions are OFF when DPU OBSW is started, with the exceptions of GENERATE_EVENT_DPU and VERIFY_CHECKSUM already discussed. The DPU change the status of an AF on reception of a specific telecommand, the only exception being during the execution of OBCPs go_SAFE: these OBCPs change the status of the AF that appeares in red or green in the corresponding column, an X means that the status is not changed after the execution of these two OBCPs.

5.14.1 NOTES ON COOLER RECYCLING

At the end of switch on sequence the funcion EVENT_BOL_CURRENT_SP1 is enabled. To start cooler recycling do these operations in the specified order: **changing this order can put PACS in an uncontrolled status**

1. Disable AF 21 (EVENT_BOL_CURRENT_SP1);
2. Enable AF 18 (EVENT_BOL_CURRENT_SP2);
3. Execute the cooler recycling procedure;
4. Disable AF 18 (EVENT_BOL_CURRENT_SP2);
5. Enable AF 21 (EVENT_BOL_CURRENT_SP1).

Step 5 must be executed only once the current measured in the BOLC HK HEATER_SP has decreased to an absolute value less than $2{\cdot}10^{-5}$A.

5.15 How to change the HK limits

It is not foreseen that the user changes the range of validity of the HK, and for this reason no specific command has been implemented to change that range. However, it may be necessary during test to change the limits and to this aim ordinary memory load commands can do the job.

The address in memory corresponding to the HK whose identifier is i, is given by this formula

$$\text{address} = \text{Start\_address} + 6 * i + \text{offset}$$

where:

$$\text{Start\_address} \begin{cases} \texttt{0xBF2} & \text{for DPU} \\ \texttt{0xD6C} & \text{for red SPU} \\ \texttt{0xDE3} & \text{for blue SPU} \\ \texttt{0xE5A} & \text{for DEC} \end{cases} \qquad \text{offset} \begin{cases} 0 & \text{upper hard limit} \\ 1 & \text{lower hard limit} \\ 2 & \text{upper soft limit} \\ 3 & \text{lower soft limit} \end{cases}$$

5.16 Example 1

We want to change the upper hard limit of DPU HK DPU_VOL_15P. The ID is read from Appendix C, taking into account the DPU_VOL_25_P has ID 0, ie subtract 4 from the ID of Appendix C. So the ID is 6–4 or 2. Then

$$\text{address} = \texttt{0xBF2} + 6 * 2 + 0 = \texttt{0xBFE}$$

5.17 Example 2

We want to change the lower soft limit of BOL HK VDDPROT_CLB1. From RD–4 we see that the ID is 14 so that

$$\text{address} = \texttt{0xE5A} + 6 * 14 + 3 = \texttt{0xEB1}$$

6 Services 1, 8 and 18

In the next sections all the commands accepted by the DPU are presented. Since the commands are received from the satellite through the 1553 interface they must be encapsulated in one telecommand (TC) structured according to AD–2. The TC's are divided in types and subtypes, each type defining a service (for instance memory management) and each subtype defining a specific action (for instance memory load). In the same way, all the data sent out from the DPU must follow the prescriptions given in AD–2 for the telemetry (TM) packets.

The first subsection explains the telecommand verification service because this service is common to all telecommands. Then service 8 is discussed, through which specific actions are requested to the DPU or to the subsystems. This section ends with Service 18, OBCP management, the main service used for PACS commanding. All the other services are presented in the next two sections.

6.1 Telecommand verification: TM(1,1) and (1,2)

Each time the DPU receives a TC packet from the spacecraft, the procedure shown in the next figure is started to test the consistency of the packet with the ESA standard, as described in AD–2. If the checks are all passed and the acknowledgment bit is set then a TM (1,1) packet is sent; otherwise, once a check is not passed a TM (1,2) is always sent to the satellite. In both cases the TM packet reports in the application data field two words copied from the first two words of the TC packet header, in this way the accepted/rejected telecommand can be identified. In case of error the TM report (1,2) contains also a failure code along with one or two parameters that better explain the problem found

Failure Code	Symbolic Name	1st parameter	2nd parameter
0	`Illegal APID`	the wrong APID	0
1	`Invalid LENGTH`	length of the packet	number of received bytes
2	`Incorrect CRC`	received CRC	computed CRC
3	`Illegal TYPE`	the invalid type	the received (type,subtype)
4	`Illeg SUBTYPE`	the invalid subtype	the received (type,subtype)

To check the packet length the OBSW reads the number of bytes written in the Packet Length field of the telecommand: this number plus 7 (see AD–2) is "length of the packet" and should be equal to "Number of received bytes" which is the number of bytes the DPU has received from the satellite.

6.1.1 Telecommand execution started: TM(1,3)

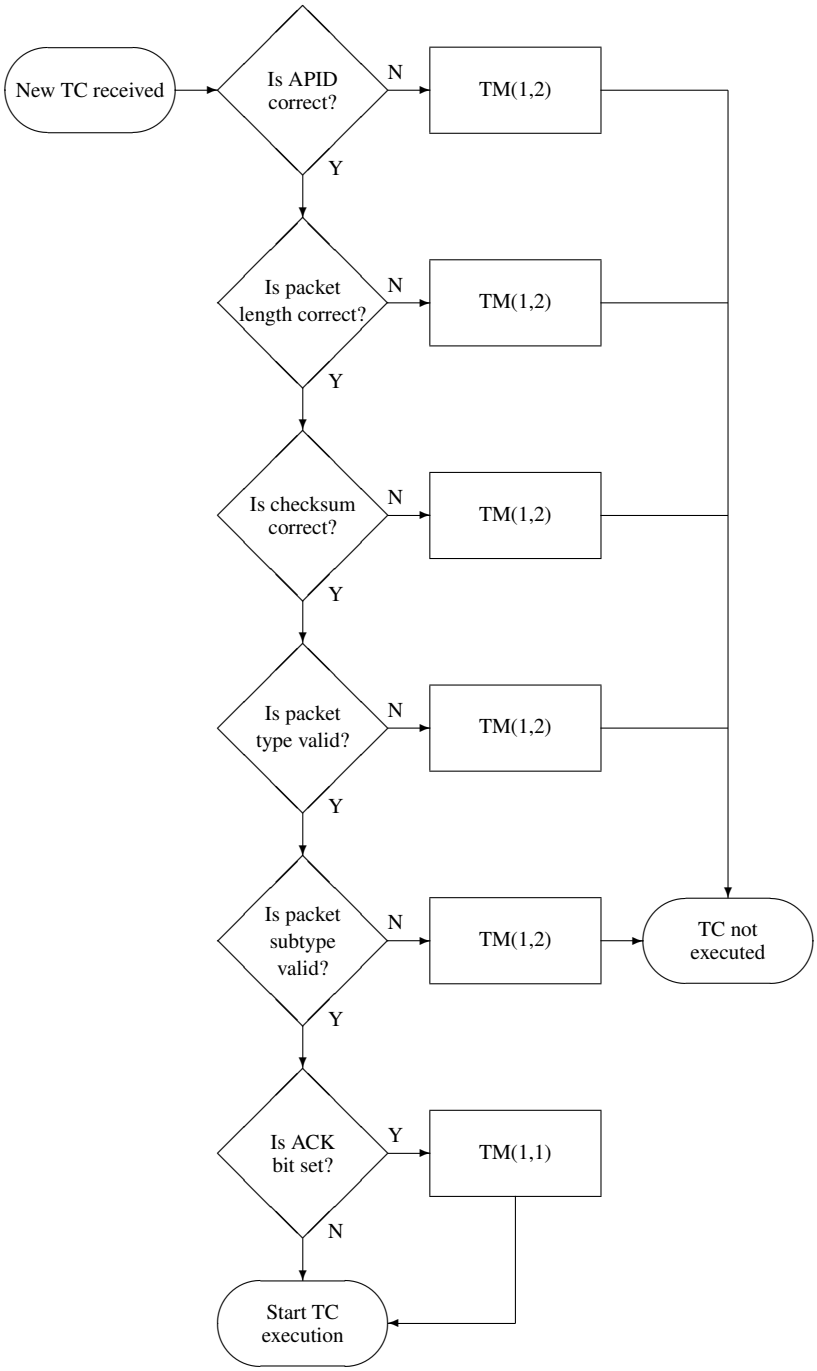
This TM report is only used when an OBCP is started. It contains the copy of the first two words of the packet header of the TC received. The generation of this report can be disabled (see Service 14).

6.1.2 Telecommand execution completed: TM(1,7)

This TM report is generated in case of successfull completion of the following telecommands:

1. Memory Load (see Section 7.1.1): sent after one memory load command for DPU or a susbsystem has been executed;
2. Check PM (see Section 6.3.13): sent if the checksum on the specified PM area is equal to the checksum written in the telecommand.
3. Load OBCP (see Section 6.4.1): sent after the last load telecommand has been executed;
4. Start OBCP (see Section 6.4.3): sent after the OBCP has been succesfully executed;

The content of the packet is the same as in case of (1,1) or (1,3) reports. The generation of this report can be disabled (see Service 14).



6.1.3 Telecommand execution failure: TM(1,8)

This TM report is used when the execution of a TC failed, its generation can be disabled (see Service 14). The packet contains the first two fields copied from the the packet header of the TC received. Then there are the failure code (in the following table the value and the symbolic name of this code are reported), the error code and a parameter:

Failure Code		
5	`Invalid DATA`	the telecommand contains one or more data which are out of the allowed range. This error corresponds to the Failure-Code 5 of the AD–2 for the Service (1,2)
16	`Illegal STATUS`	the command is incompatible with the DPU status, for instance the command is for a subsystem whose status is off
17	`Resource FAIL`	the command can not be executed for incompatibility with the DPU resources, for instance not enough space in DEC sequence buffer when loading a new sequence
Error Code		this code depends on the specific service. For instance, to see the errors related to the OBCP service see Section 6.4;
Parameter		as before, the meaning of this parameter depends on the specific service. It is reported along with the error code messages described at the end of each TC and higlighted with the symbol ϟ

6.1.4 TC—TM reference table

Here is a table that shows the combinations (type,subtype) of each TC accepted by the DPU and the telemetry (TM) packets associated. This list is a subset of all the services allowed in AD–2. If a TM packet is not associated to a TC packet this means that it is generated autonomously by the DPU. TM (1,1) and TM (1,2) are not reported.

			Report on completion	
Name	TC	TM	Success	Failure
HK reports	—	(3,25)		
Event report	—	(5,1)		
Exception report	—	(5,2)		
Error report	—	(5,4)		
Memory load	(6,2)	—	(1,7)	(1,8)
Memory dump	(6,5)		(6,6)	(1,8)
Memory check	(6,9)		(6,10)	(1,8)
Start function	(8,1)			
Stop function	(8,2)			
Perform activity	(8,4)		(1,7)[3]	(1,8)
Status of function	(8,5)			
Time verification	(9,7)		(9,9)	
Enable TM packet	(14,1)			
Disable TM packet	(14,2)			
Report TM packet	(14,3)		(14,4)	
Test Connection	(17,1)		(17,2)	
Load OBCP	(18,1)		(1,7)	(1,8)
Delete OBCP	(18,2)			(1,8)
Start OBCP	(18,3)	(1,3)[4]	(1,7)	(1,8)
Stop OBCP	(18,4)			(1,8)

[3] Only for particular DPU commands.
[4] Sent just before start of execution.

Name	TC	TM	Report on completion Success	Failure
Suspend OBCP	(18,5)			(1,8)
Resume OBCP	(18,6)			(1,8)
Communicate parameters	(18,7)			(1,8)
List of OBCP	(18,8)		(18,9)	
List of active OBCP	(18,10)		(18,11)	(1,8)
OBCP status	(18,12)		(18,13)	(1,8)
Science data A	—	(21,1)		
Science data B	—	(21,2)		
Science diagnostic data	—	(21,3)		

6.2 Sending atomic commands to subsystems (Service 8)

Service 8 (Function management in AD–2) is used to send atomic commands to the DPU or to the subsystems. All the other subtypes of this service ((8,1), (8,2) and (8,5)) are ignored, i.e. after the telecommand verification nothing happens in the DPU. Each subsystem is identified through an ID (Function ID) according to this table

Subsystem	Function ID		Subsystem	Function ID	
DPU	100	0x64	SPU (blue)	101	0x65
DEC	103	0x67	SPU (red)	102	0x66

6.2.1 Perform activity: TC(8,4)

P#1	0xSSII
P#2	SID
P#3	1st parameter
P#n	last parameter

The first word of the TC contains the function ID (0xSS) and the activity ID (0xII); the second word is the SID, ie the number of parameters that follow. SID is 4 for WRITE commands. The DPU checks that the values of Function ID and SID are allowed values. If so, the command is sent to the subsystem or executed inside the DPU. If the command is not executed by the subsystem, an event and a TM report (1,8) give the cause of the failure.

Warning: the DPU does not check that the TC packet does contain a number of parameters compatible with the SID.

6.2.1–1 TM (1,8) — Invalid DATA: Invalid FUN-ID

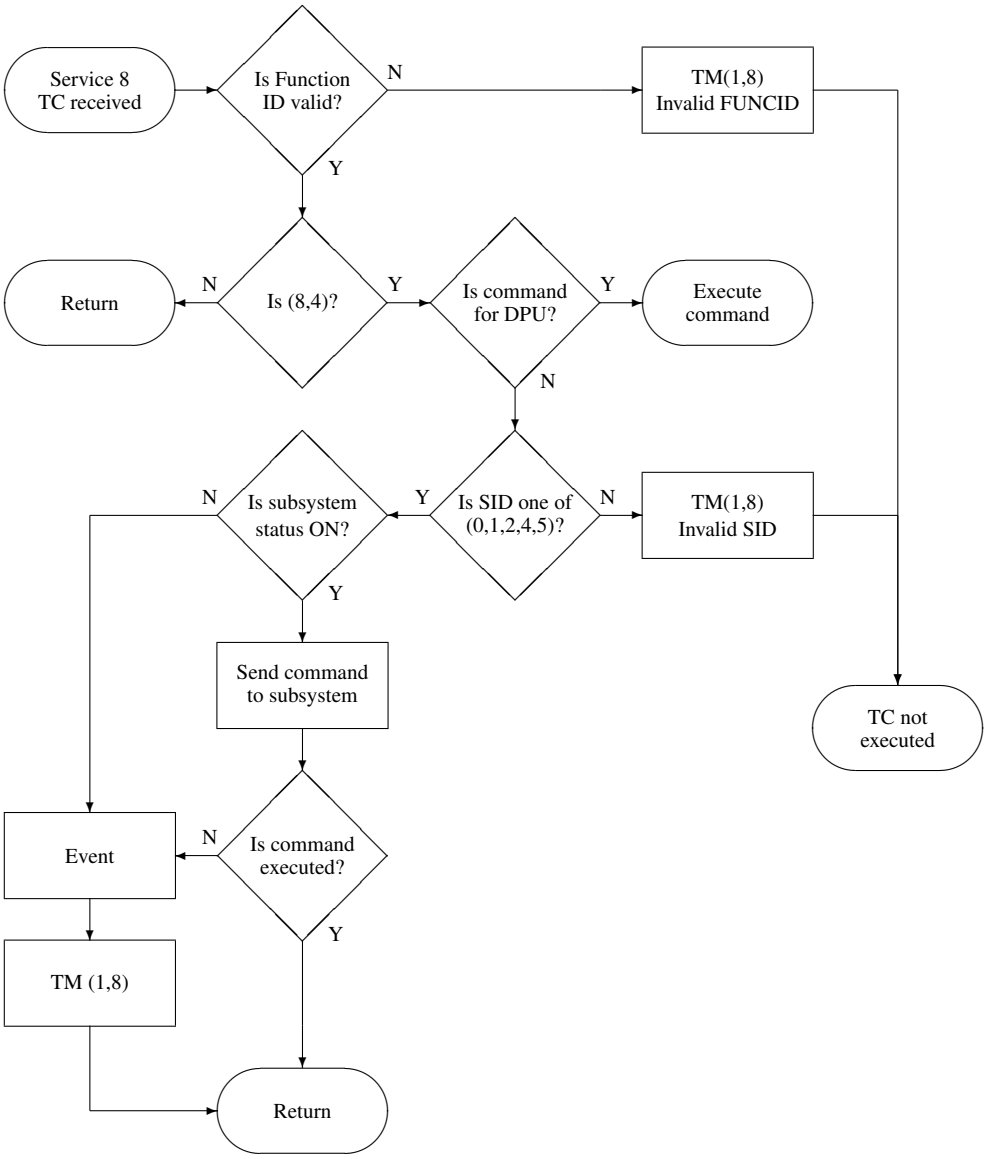
0xSS, the Function ID, does not have a valid value (see Table 6.2 on page 33).

↯ Parameter reported in the TM (1,8) report: 0xSS (the Function ID extracted from *P#1*)

6.2.1–2 TM (1,8) — Invalid DATA: Invalid SID

An illegal SID has been used. Valid values are 0, 1, 2, 5 for *TRIGGER* commands; 4 for *WRITE* commands.

↯ Parameter reported in the TM (1,8) report: SID (copy of *P#2*)



6.2.1–3 TM (1,8) — `Illegal STATUS: UNIT STOPPED`

Before sending the command the DPU reads the HK DP_xxx_CMD, for instance DP_DMC_CMD if the command is for DEC. If the value is 0 (link not yet started) or 3 (link dead after a parity/disconnect error) the command is not sent and this TM report is generated. Otherwise DPU checks the three DP_xxx_CMD and if one of them is 2 (a NACK has been previously received) the command is not sent. In the latter case the status must be set to `ON` using the DPU command "Set function". The event `SS Stopped` is also raised.

↯ Parameter reported in the TM (1,8) report: the subsystem (0 DEC, 1 SPS, 2 SPL)

6.2.1–4 TM (1,8) — `Resource FAIL: UNIT STOPPED`

If the subsystem reacts to the command with a negative acknowledgment the DPU raises the event `NACK`, sets the HK DP_xxx_CMD to 2 (`STOPPED`) and finally sends this (1,8) report. From now on no command can be sent to any subsystem until the DPU command "Set function" is sent.

↯ Parameter reported in the TM (1,8) report: the subsystem (0 DEC, 1 SPS, 2 SPL)

6.2.1–5 TM (1,8) — `Resource FAIL: FUNCT TIMEOUT`

The DPU did not receive from the 1355 interface the "EOP sent" interrupt within 2 msec. This condition could be either due to a hardware problem, or to an incorrect timing of the software. See also the event `1355 Timeout`.

↯ Parameter reported in the TM (1,8) report: the subsystem (0 DEC, 1 SPS, 2 SPL)

6.2.1–6 TM (1,8) — `Invalid DATA: Inv COMMAND`

The DPU received the command to start the application SW of one subsystem. This command can not be executed directly but requires a specific OBCP (see Section 4.3).

↯ Parameter reported in the TM (1,8) report: `0xSS` (the Function ID extracted from *P#1*)

6.2.1–7 TM (1,8) — `Illegal STATUS: FUNK LINK USED`

The DPU received a new command while it was still waiting the acknowledgment for a previous command.

↯ Parameter reported in the TM (1,8) report: the subsystem (0 DEC, 1 SPS, 2 SPL)

6.3 Sending commands to DPU: TC(8,4)

DPU commands are encapsulated in TC(8,4) like commands for subsystems, and recognized on the base of the activity ID which is part of the first parameter of the TC packet (the list of ID and SID are in the table on page 82). If the ID does not correspond to a valid DPU command the following error packet, common to all commands, is generated

6.3.0–8 TM (1,8) — `Invalid DATA: Invalid ACT-ID`

Activity ID not recognized.

↯ Parameter reported in the TM (1,8) report: `0xII` (the Activity ID extracted from *P#1*)

6.3.1 Upgrade Sequence, Delete Sequence, Add Sequence

These three commands are used to maintain the DEC sequences stored in the DPU. The available space is 1500 words, the 19 sequences (out of 32 allowed) available at start-up and reported in the table on page 44 occupy 1492 words. Upgrade and Delete operate on existing sequences, while Add is used to upload a new sequence. After the execution of these commands, the available space may change.

Note that the only way to verify the correct execution of these commands is through a memory dump: the start address for DEC sequences is `0x5BD`

	Upgrade Sequence	Delete Sequence	Add Sequence	
P#1	`0x6401`	`0x6402`	`0x6403`	
P#2	4	1	4	
P#3	Seq. ID	Seq. ID	Seq. ID	
P#4	*N*	—	*N*	Number of 32 bits data words
P#5	1st word	—	1st word	
P#6	2nd word	—	2nd word	
P#(4+N)	last word	—	last word	
P#(5+N)	crc	—	crc	Checksum of the data words

6.3.1–1 TM (1,8) — `Invalid DATA: Invalid SEQ-ID`

P#3 is 0 or larger than 32.

↯ Parameter reported in the TM (1,8) report: the invalid sequence ID (copy of *P#3*)

6.3.1–2 TM (1,8) — `Illegal STATUS: Invalid SEQ-ID`

P#3 corresponds, for the command “Add Sequence”, to an existing sequence; for the commands “Upgrade Sequence” and “Delete Sequence” to a non existent sequence.

↯ Parameter reported in the TM (1,8) report: the invalid sequence ID (copy of *P#3*).

6.3.1–3 TM (1,8) — `Invalid DATA: Invalid CRC`

The crc computed by the DPU is not equal to *P#(5+N)* (only for commands “Upgrade Sequence” and “Add Sequence”).

↯ Parameter reported in the TM (1,8) report: the crc computed by the DPU.

6.3.1–4 TM (1,8) — `Resource FAIL: SEQ NO Space`

The number of free words available in memory is not enough to store the new sequence sent with the command “Upgrade Sequence” or “Add Sequence”.

↯ Parameter reported in the TM (1,8) report: the number of available words in the DPU sequences array.

6.3.2 Set HK list

Select one of the three pre-defined HK packets: spectroscopy, photometry or non-prime (default at start-up); the second parameter enables the transmission of science data from SPU

After the command has been sent the DPU HK DP_TM_RATE should be one of SPEC, PHOT or NO-PRIME according to *P#3*; if *P#4* is 1 DP_STATUS becomes xxxxx11xx, if 2 becomes xxxxx01xx, if 3 it is xxxxx10xx. When *P#3* is 4, DP_STATUS is xxxxx00xx independently on *P#4*

P#1	`0x6404`
P#2	1
P#3	1 for spectroscopy; 2 for photometry; 4 for non-prime (default at start-up)
P#4	1 to enable transmission of science data for both SPU; 2 to enable science data stream from blue SPU only; 3 to enable science data stream from red SPU only. In non-prime mode, this parameter is ignored and science packets are disabled

6.3.2–1 TM (1,8) — `Invalid DATA: Invalid PARAM`

P#3 is not recognized (allowed values: 1,2,4).

↯ Parameter reported in the TM (1,8) report: the wrong parameter (copy of *P#3*).

6.3.2–2 TM (1,8) — `Invalid Data:  Invalid ARRAY`

P#4 is not not in the range 1—3.

↯ Parameter reported in the TM (1,8) report: the wrong parameter (copy of *P#4*)

6.3.3 Set function

Enable or disable a function. Note that the meaning of this command is different if the function ID is in the range 1—99 or in the range 100—103.

P#1	`0x6406`
P#2	2
P#3	Function ID
P#4	0 to disable, 1 to enable, any other number has no effect

• **Function ID in the range 1 — 99**: this command is used to enable/disable an autonomy function; can be sent at any time but note that activating an AF at the wrong moment can make DPU react to a condition that for a specifing operative mode could be nominal (for instance enabling an AF that checks the temperature of a subunit that at that moment is off). Also, disabling an AF at the wrong moment makes DPU not react to anomalous conditions. The list of autonomy functions is reported in Section 5.

Once the command is sent a bit in the DPU HK DP_AF_STATUS (see Section 8.1) is set/reset, for instance if the AF with ID 1 is currently disabled then DP_AF_STATUS has the value `xxx...xxx0`; after sending the command to enable this function the HK will have the value `xxx...xxx1`. Note that if ID is greater than 24 the command is ignored.

• **Function ID in the range 100 — 103**: in this case the function to enable/disable is related to the subunits according to the table on page 33: 100 is for the DPU itself, its status is always enabled and can not be modified, so in this case the command is ignored; 101 is the blue SPU; 102 is the red SPU; 103 is for DEC. This command should be sent after the HK DP_xxx_CMD became `SS_STOPPED` in consequence of a command execution failure. This command can also be used to set the DP_xxx_CMD from `SS_ENABLED` to `SS_STOPPED`, in this way disabling the whole mission timeline. It is not clear why an user should want to do that!

Note: if the link has not been started yet (see Section 4.2) the command is ignored; enabling an enabled function, or disabling a disabled function, has no consequence.

6.3.3–1 TM (1,8) — `Invalid DATA: Invalid FUN-ID`

P#3 is 0 or greater than 103.

↯ Parameter reported in the TM (1,8) report: the function ID (copy of *P#3*).

6.3.4 Force execution of autonomy function

An autonomy function is called when a non nominal condition is met, for instance a HK value out of range. This command forces the DPU to start the function, as long as the function is enabled. At start-up all the functions are disabled (see previous command for the function ID).

This command is potentially dangerous! Use it only after approval by IFSI.

P#1	`0x6405`
P#2	1
P#3	Function ID (any number from 1 to 99)

6.3.4–1 TM (1,8) — `Invalid DATA: Invalid AF`

P#3 outside the range 1—99.

↯ Parameter reported in the TM (1,8) report: the wrong parameter (copy of *P#3*).

6.3.4–2 TM (1,8) — `Illegal STATUS: Invalid AF`

The requested function is not enabled.

↯ Parameter reported in the TM (1,8) report: the function ID (copy of *P#3*).

6.3.5 Warm reset

This command performs a warm reset, which means that the OBSW starts again its execution from the beginning like after the transition from boot SW; as a consequence the 1355 chip is reset and the links, if on, are lost. Note that due to a HW problem in the **AVM1!** the execution of this command may set the DPU in a "frozen" state: if no telemetry is generated after this command is sent, push the **CPU!** reset button.

P#1	`0x6407`
P#2	0

6.3.6 Jump to boot software

This command is similar to a warm reset but the interrupt table used by the boot software is copied in PM before the reset. This means that the boot SW starts again its execution and, for instance, a new image can be uploaded, see Section 3. Note that due to a HW problem in the **AVM1!** the execution of this command may set the DPU in a "frozen" state: if no telemetry is generated after this command is sent, push the **CPU!** reset button.

P#1	`0x6409`
P#2	0

6.3.7 Send time to DEC

Computes the internal DPU time, in the format specified in AD–2, and sends it to DEC with the command *WRITE TIME* (see RD–4). Note that since a command is sent to DEC all the possible errors reported for Service (8,4) can occur.

P#1	`0x6408`
P#2	0

6.3.7–1 TM (1,8) — `Illegal STATUS: UNIT STOPPED`

If the HK DP_DMC_CMD is 0 or 3, or any of the three DP_xxx_CMD is 2 the DPU does not send the *WRITE TIME* command. In the latter case the status must be set to `ON` using the "Set function" command. The event `SS Stopped` is also raised.

↯ Parameter reported in the TM (1,8) report: 0 (DEC link)

6.3.7–2 TM (1,8) — `Resource FAIL: UNIT STOPPED`

If DEC reacts to the command with a negative acknowledgment, the DPU raises the event `NACK`, sets DP_DMC_CMD status to `STOPPED` and sends this (1,8) report.

↯ Parameter reported in the TM (1,8) report: 0 (DEC link)

6.3.7–3 TM (1,8) — `Resource FAIL: FUNCT TIMEOUT`

The DPU did not receive from the 1355 interface the "EOP sent" interrupt within 2 msec. See also the event `1355 Timeout`.

↯ Parameter reported in the TM (1,8) report: 0 (DEC link)

6.3.7–4 TM (1,8) — `Illegal STATUS: FUNK LINK USED`

The DPU tried to send the *WRITE TIME* command while it was still waiting the acknowledgment for a previous command.

↯ Parameter reported in the TM (1,8) report: 0 (DEC link)

6.3.8 Set buslist

Enable/disable the burst mode for the 1553 interface protocol. When the burst mode is enabled the HK DP_STATUS becomes `xxx1xxxxx`. Enabling the burst mode while it is already enabled and viceversa has no consequence.

P#1	`0x640A`
P#2	1
P#3	0 disables the burst mode, any other number enables the burst mode

To enter the burst mode when the DPU is connected to the CDMS simulator execute these steps:

1. Select the burst mode buslist in the CDMS
2. Send the command Set buslist with *P#3* set to 1

To leave burst mode:

1. Send the command Set buslist with *P#3* set to 0
2. Select the nominal buslist in the CDMS

In general **burst mode should be started first on satellite side, and disabled first on DPU side.**

6.3.9 Reset 1355

This command resets the SMCS332 chip that handles the 1355 interface. If ON the links are lost and no communications is possible until the links are started again.

P#1	`0x640B`
P#2	0

6.3.10 Reset 1553

This command resets the chip that handles the 1553 interface. It has no consequence for the communications with the CDMS simulator

P#1	`0x640D`
P#2	0

6.3.11 Enter Test Mode

This command is used to enter/exit the test mode; when in test mode the DPU fills in the HK packets with running numbers starting from 1 for each subunits. This mode is intended to check consistency between MIB and OBS. If subsystems are ON their HK are in any case checked against nominal values

P#1	`0x640C`
P#2	1
P#3	0 to exit Test Mode, any other number to activate Test Mode

6.3.12 Copy OBSW (patching)

This command allows to patch the OBSW. Here patching means not only uploading part of the OBSW (a real patch) but also the upload of the full image.

P#1	`0x640E`
P#2	3
P#3	Direction (1 or 2, see below)
P#4	Start Address
P#5	Number of Words

To patch the OBSW it is necessary to executes the following steps while to upload the full image the first step is not necessary.

Step #1: send COPY OBSW command with Direction 1. The content of PM from address 0 to address (Number of Words - 1) is copied in high memory to address Start Address; Number of Words is the length of the OBSW. These two parameters should be specified as part of the OBSW delivery;

Step #2: send all the memory load commands that have been delivered;

Step #3: send COPY OBSW command with Direction 2. All the interrupts are disabled and the new OBSW is copied to low memory from address Start Address. If Step #1 has been executed Start Address must be unchanged; Number of Words is the length of the new OBSW and it should be specified as part of the OBSW delivery.

Step #4 (done automatically by the OBSW): the DPU makes a warm reset and the SW starts again. Once HK packets are delivered the SW Version ID should be the new value.

Note that between Step #2 and Step #3 the command Check PM (see next command) may be sent to be sure that the new image is correct.

6.3.12–1 TM (1,8) — `Invalid DATA: Invalid PARAM`

Number of words is not valid. In case Direction is 1 this means that Number of words is either zero or greater than (Start Address - 50). In case Direction is 2 the error is due to Number of words equal to zero; or $\geq$ (`0x80000` - Start Address); or $\geq$ (Start Address - 50). 50 is the length of a small portion of code that makes the actual copy.

↯ Parameter reported in the TM (1,8) report: *P#5* (Number of words)

6.3.13 Check PM

This command performs a checksum on a specified PM area and compares the result with the value written in the command. If the computed checksum is the same a TM packet (1,7) is reported, while if the two values are different a TM packet (1,8) and the event (5,2) `PM FAILURE` are generated; in both cases the two checksums are reported.

The difference with respect to memory check command is that in this case the DPU can autonomously discover a problem (HW failure or bit flip) in PM.

P#1	`0x640F`
P#2	3
P#3	Start Address
P#4	End Address
P#5	Expected Checksum

6.3.13–1 TM (1,8) — `Illegal STATUS: Invalid CRC`

The computed checksum is not equal to *P#5*. An error, either a bitflip or a HW failure, is present in PM. The consequences are unpredictable and the following actions should be taken: **set PACS in SAFE mode and make a dump of PM memory!**

↯ Parameter reported in the TM (1,8) report: MSB is *P#5*, the crc written in the TC; LSB is the checksum computed by the DPU

6.4 How to handle procedures

Procedures are functions written in C language and compiled as part of the SW image. Each procedure is completely defined by three parameters: the ID, in the range 1—50; the number of parameters, hereinafter NoP, in the range 0—25; and the status: 0 (`STOPPED`), 1 (`ACTIVE`), 2 (`SUSPENDED`) and 3 (`DELETED`)[5].

The procedures available and the number of parameters they accept are:

Name	ID	NoP	Sequence	NoP
Bolometer transition to IDLE state	1	0	NO	
Time Synchronization III	2	0	NO	
Fixed-Fixed Chopped Photometry	3	12	YES	8
Chopped Photometry	4	14	YES	9
Chopped Photometry with Dither	5	15	YES	9
“Freeze Frame” Chopping Photometry	6	9	YES	4
Staring Photometry for Line Scans	7	5	YES	1
Grating Line Scan Chopped	8	22	YES	12
Grating Line Scan Chopped with Dither	9	23	YES	12
Photometry Calibration I	10	11	YES	6
Photometry Calibration II	11	15	YES	10
Photometry Calibration III	12	13	YES	8
Internal Calibration Spectroscopy	13	19	YES	9
Chopper Up-Down Scan Photometry	14	13	YES	6
DMC Test Mode	15	3	NO	
Switch Spectroscopy to Photometry	16	1	NO	
Enter SAFE mode 2	17	0	NO	
Chopper Up-Down Scan Spectroscopy	18	17	YES	6

[5] Also `NOT EXISTING` in SCOS2000 messages.

Start 1355 link	19	2	NO	
Write in EEPROM	20	25	NO	
Start HLSW	21	3	NO	
Wavelength Switch Grating	22	20	YES	11
Grating Line Scan Chopped 2 Fast	23	21	YES	11
Enter SAFE mode	24	0	NO	
Time Synchronization I	25	0	NO	
Time Synchronization II	26	0	NO	
Grating Line Scan Chopped 2	27	21	YES	11
Grating Line Scan Without Chopping	28	20	YES	11
Generate dummy science packets	29	2	NO	
SPU test mode spec	30	2	NO	
SPU test mode phot	31	2	NO	
Wavelength Switch Grating 2	32	25	YES	15
OBMO	33	0	NO	
ACWE	34	1	NO	
Grating Line Scan Chopped 3	35	21	YES	11

The description of these procedures can be found in RD–3, with the exception of # 19 (see Section 4.2), 20 (Section 4.1), 21 (Section 4.3) and 29 (Section 4.4). The difference between Procedure #17 and 24 is that the former does not switch off the calibration sources while the latter does

The column Sequence says if the procedure uses a DEC sequence, whose ID is passed as parameter to the procedure; the sequences themselves require a number of parameters, given in the last column. The available sequences are:

Name	ID	NoP	Length
Two or Three Position Chopping Photometry	1	9	72
Two or Three Position Chopping Photometry with Dither	2	9	72
Staring Photometry	3	1	18
Freeze Frame Chopping Photometry	4	4	30
Chopping on Internal Calibration Sources I	5	6	44
Chopping on Internal Calibration Sources II	6	10	96
Chopping on Internal Calibration Sources III	7	8	54
Grating Line Scan with Two or Three Position Chopping	8	12	142
Grating Line Scan with Two or Three Position Chopping with Dither	9	12	142
Line Observation with Wavelength Switching	10	11	112
Chopped Grating Scan on Internal Calibration Sources	11	9	74
Grating Line Scan with Two Position Chopping	12	11	110
Grating Line Scan without Chopping	13	11	82
Fixed-Fixed Chopping	14	8	100
Chopper Up-Down Scan Photometry	15	6	38
Chopper Up-Down Scan Spectroscopy	16	6	38
Grating Line Scan with Two Position Chopping Fast	17	11	88
Line Observation with Wavelength Switching 2	18	15	72
Grating Line Scan with Two Position Chopping (on-off-off-on)	19	11	108

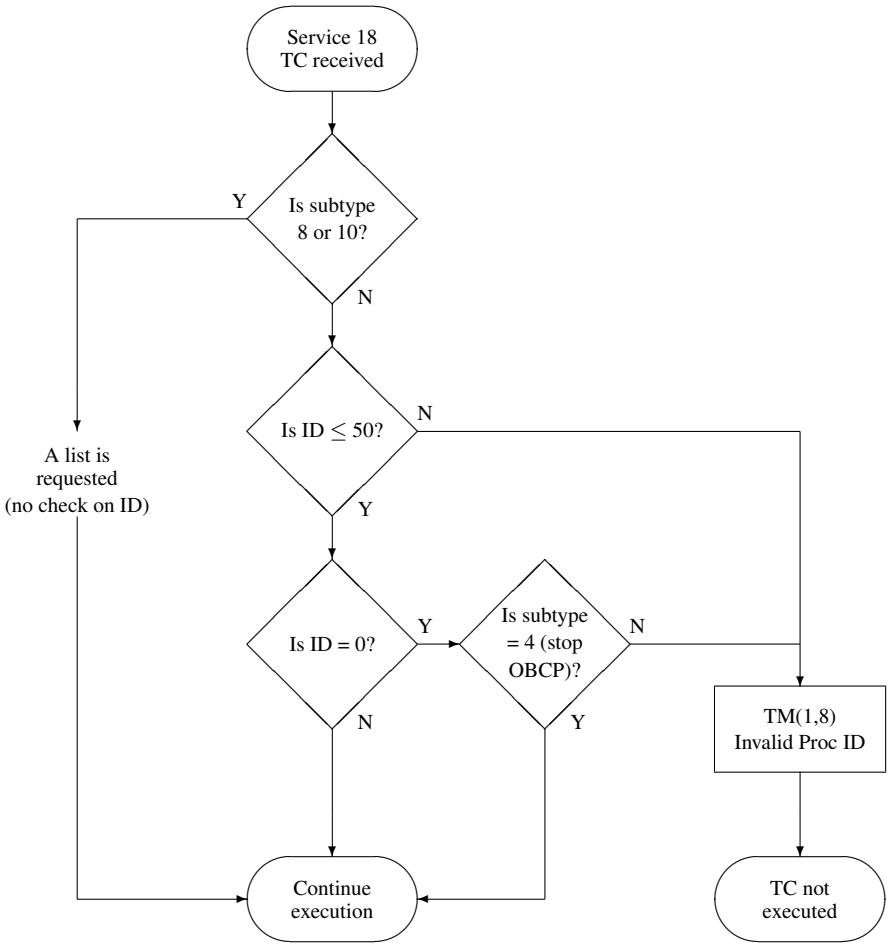
The status of existing procedures at boot is `STOPPED`; for all other procedures ($35 \leq ID \leq 50$) the status is `DELETED`. In the following table it is shown the influence of the status on the service, as well as the status transition if the command is executed. If the combination status/command makes the OBS issue a TM packet

(1,8), the error code is given.

Service	Status				Status Transition
	STOPPED	ACTIVE	SUSPENDED	DELETED	
Load	Executed	0x1206	Executed	Executed	$\Longrightarrow$ STOPPED
Delete	Executed	Ignored	Ignored	Ignored	STOPPED $\Longrightarrow$ DELETED
Start	Executed	Ignored	Ignored	0x1202	STOPPED $\Longrightarrow$ ACTIVE $\Longrightarrow$ STOPPED
Stop	Ignored	Executed	Executed	Ignored	$\Longrightarrow$ STOPPED
Suspend	Ignored	Executed	Ignored	Ignored	ACTIVE $\Longrightarrow$ SUSPENDED
Resume	Ignored	Ignored	Executed	Ignored	SUSPENDED $\Longrightarrow$ ACTIVE
Comm par	Executed	Executed	Executed	Ignored	Status is not changed

At boot all the parameters are set to 0. The first time a procedure is called all its parameters must be set, otherwise the OBS uses 0. The new values are stored and can be used at the next call. If one or more parameters are not passed, the last values sent are used. All parameters are 32 bits.

The checks common to all telecommands for this service are reported in the next figure



6.4.1 Loading a procedure: TC(18,1)

This service is used to upload a new procedure to which the ID 50 is assigned by default. A TC packet contains up to 242 bytes of application data, of which 4 are for the header (see AD–2), 2 for the CRC, 4 for the first two parameters of the command, so that there are 232 bytes available, or 38 PM words: a procedure longer than 38 assembler instructions must be splitted in segments, one within each TC.

P#1	Procedure ID (ignored, always set to 50)
P#2	Segment ID
P#3	*Length* in bytes (i.e. 6 times the number of words, less or equal to 228. When *P#2* is `0xFF`, it must contain 0)
P#3	16 MSb of the 1st word
P#4	16 intermediate bits of the 1st word
P#5	16 LSb of the 1st word
	...
P#(Length)	16 MSb of the last word
P#(Length+1)	16 intermediate bits of the last word
P#(Length+2)	16 LSb of the last word

If the procedure is short such that its code fits in one single telecommand, then Segment ID is set to 0 and after successfull completion a TM (1,7) is generated; if n telecommands are necessary the Segment ID runs from 1 to n. After succesfull load of the n-th segment, send another TC setting this field to `0xFF`. On reception of this last telecommand the DPU will report a TM (1,7). If loading i-th segment fails it is possible to upload again all the telecommands or to start uploading from i-th segment. In other words, if part of the uploading fails, the already loaded code remains in memory.

A new procedure can be uploaded only if the status associated to the procedure 50 is 3 (`DELETED`). If an OBCP has been alredy loaded, send the command "delete OBCP" (18,2) before starting the upload. Once the upload is completed the status is set to 0 (`STOPPED`).

Once the procedure is uploaded, service (18,7) must be used in order to communicate to the OBS how many parameters the procedure needs. Afterwards the procedure can be started. Once the instrument is switched off, this new code is lost since it is not foreseen to change OBCP permanently with service (18,1). If the new OBCP is accepted its code should be then part of the OBS through a new release.

The whole sequence of uploading an OBCP is shown in the next figure

6.4.1–1 TM (1,8) — `Illegal STATUS: Illegal LOAD`

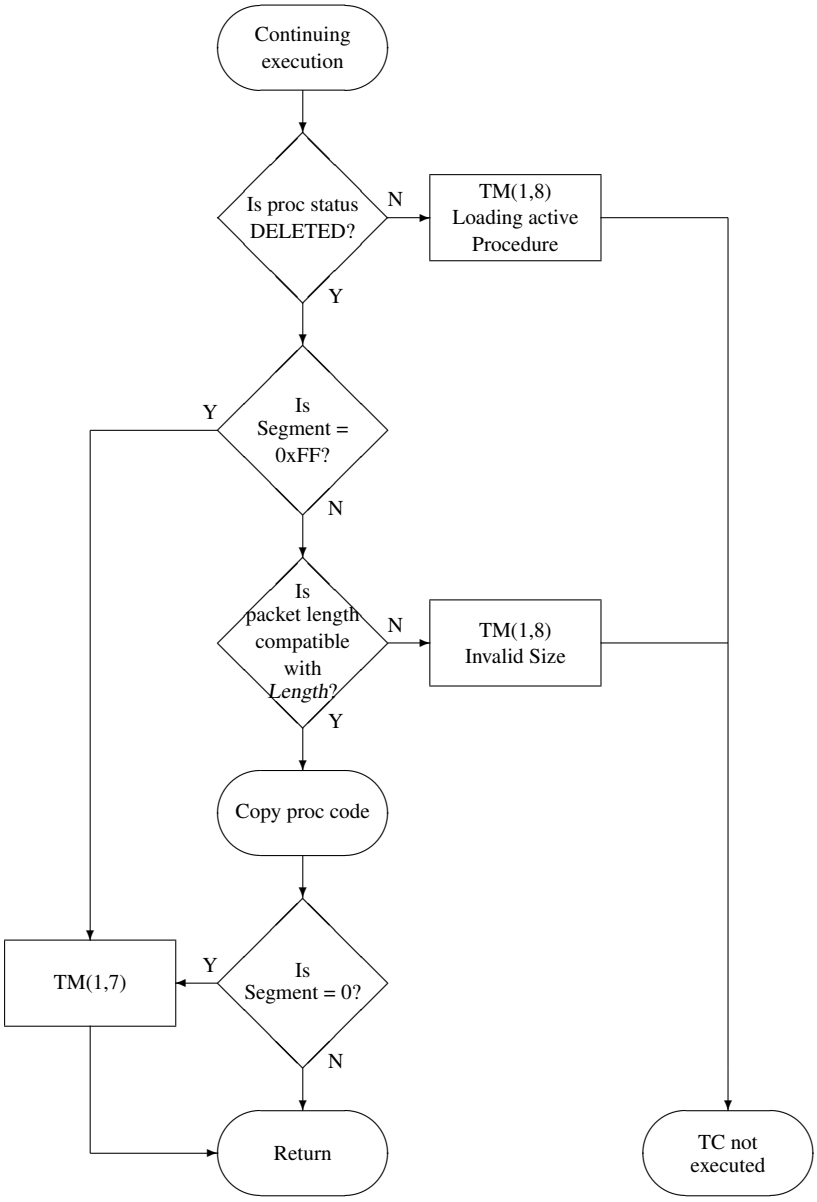
The status of the procedure is not 3 (`DELETED`). This happens if an old procedure has already been loaded and it has not been deleted yet. Send command "Delete Procedure".

↯ Parameter reported in the TM (1,8) report: the ID of the running procedure (always 50 in this case).

6.4.1–2 TM (1,8) — `Invalid DATA: OBCP INV Size`

Parameter *P#3* (*Length*) is inconsistent with the packet length as written in the TC header.

↯ Parameter reported in the TM (1,8) report: the length (copy of *P#3*).



6.4.2 Deleting a procedure: TC(18,2)

The request is ignored if the status is `ACTIVE` or `SUSPENDED`.

P#1	Procedure ID

To delete a procedure, the DPU simply changes OBCP status. However the code remains in memory and after a switch-off/on, the procedure is again runnable.

6.4.2–1 TM (1,8) — `Invalid DATA: Invalid PROCID`

P#1 is outside the range 1—50.

↯ Parameter reported in the TM (1,8) report: the procedure ID (copy of *P#1*).

6.4.3 Starting a procedure: TC(18,3)

The request is ignored if the status is `ACTIVE` or `SUSPENDED`. It is executed if the status is `STOPPED`. Once started the status is set to `ACTIVE`.

P#1	Procedure ID
P#2	n = number of uploaded parameters ($\leq$ NoP)
P#3	ID of the 1st parameter
P#4	1st parameter
	...
*P#(n*2)+1*	ID of the last parameter
*P#(n*2)+2*	last parameter

At startup all the parameters are set to 0. The set of values sent with this command is then stored and when the procedure is called again, it is not necessary to upload the unchanged parameters. As a consequence, it is possible to set *P#2* to 0.

Once the procedure is started a TM report (1,3) is issued. On completion a TM packet (1,7) is generated if the procedure has been succesfully completed, otherwise a (1,8) report is transmitted.

6.4.3–1 TM (1,8) — `Invalid DATA: Invalid PROCID`

P#1 is outside the range 1—50.

↯ Parameter reported in the TM (1,8) report: the procedure ID (copy of *P#1*).

6.4.3–2 TM (1,8) — `Illegal STATUS: Start DEL OBCP`

The user tried to start a not existing OBCP.

↯ Parameter reported in the TM (1,8) report: the procedure ID (copy of *P#1*).

6.4.3–3 TM (1,8) — `Illegal STATUS: Running OBCP`

Start command sent while another procedure is already running (if the same OBCP is running the request is ignored).

↯ Parameter reported in the TM (1,8) report: the ID of the running procedure.

6.4.3–4 TM (1,8) — `Invalid DATA: Too Much PARAM`

P#2, the number of parameters contained in the TC (18,3), is greater than the total number of parameters (NoP) accepted by that procedure.

↯ Parameter reported in the TM (1,8) report: *P#2*.

6.4.3–5 TM (1,8) — `Invalid DATA: Illegal PAR-ID`

One of the parameters ID is zero or is greater than NoP (for instance the user sent the 7th parameter to an OBCP that accepts only 6 parameters).

↯ Parameter reported in the TM (1,8) report: the illegal parameter ID.

6.4.3–6 TM (1,8) — `Invalid DATA: WRONG SEQ ID`

The sequence ID sent as part of the OBCP parameters is wrong: it corresponds to a not existing sequence (ID>32) or a not defined sequence (ID$\leq$32 but length is 0).

↯ Parameter reported in the TM (1,8) report: the illegal sequence ID.

6.4.3–7 TM (1,8) — `Invalid DATA: Wrong EE PAR`

The procedure #20 is used to copy the application program in EEPROM. The fourth parameter does not have an operational meaning but is for safety reason, to avoid that this procedure is started writing 20 by mistake instead of the intended ID (if this happens, the EEPROM would be corrupted). The user is then forced to set the fourth parameter to a special value (`0x19660502`) which the DPU resets to zero, i.e. to a non valid value, before starting the procedure. Every time this procedure is called without setting the fourth parameter to the expected value, this error is generated.

↯ Parameter reported in the TM (1,8) report: the fourth parameter.

6.4.3–8 TM (1,8) — `Invalid DATA: Invalid DATUM`

This error occurs when the value of a parameter is incorrect (out of range). It is used in the procedures “Start 1355 link” (ID #19), “Write in EEPROM” (ID #20) and ”Start SPU HLSW” (ID #21), see Section 4.

↯ Parameter reported in the TM (1,8) report: the ID of the incorrect parameter.

6.4.3–9 TM (1,8) — `Illegal STATUS: Not compl OBCP`

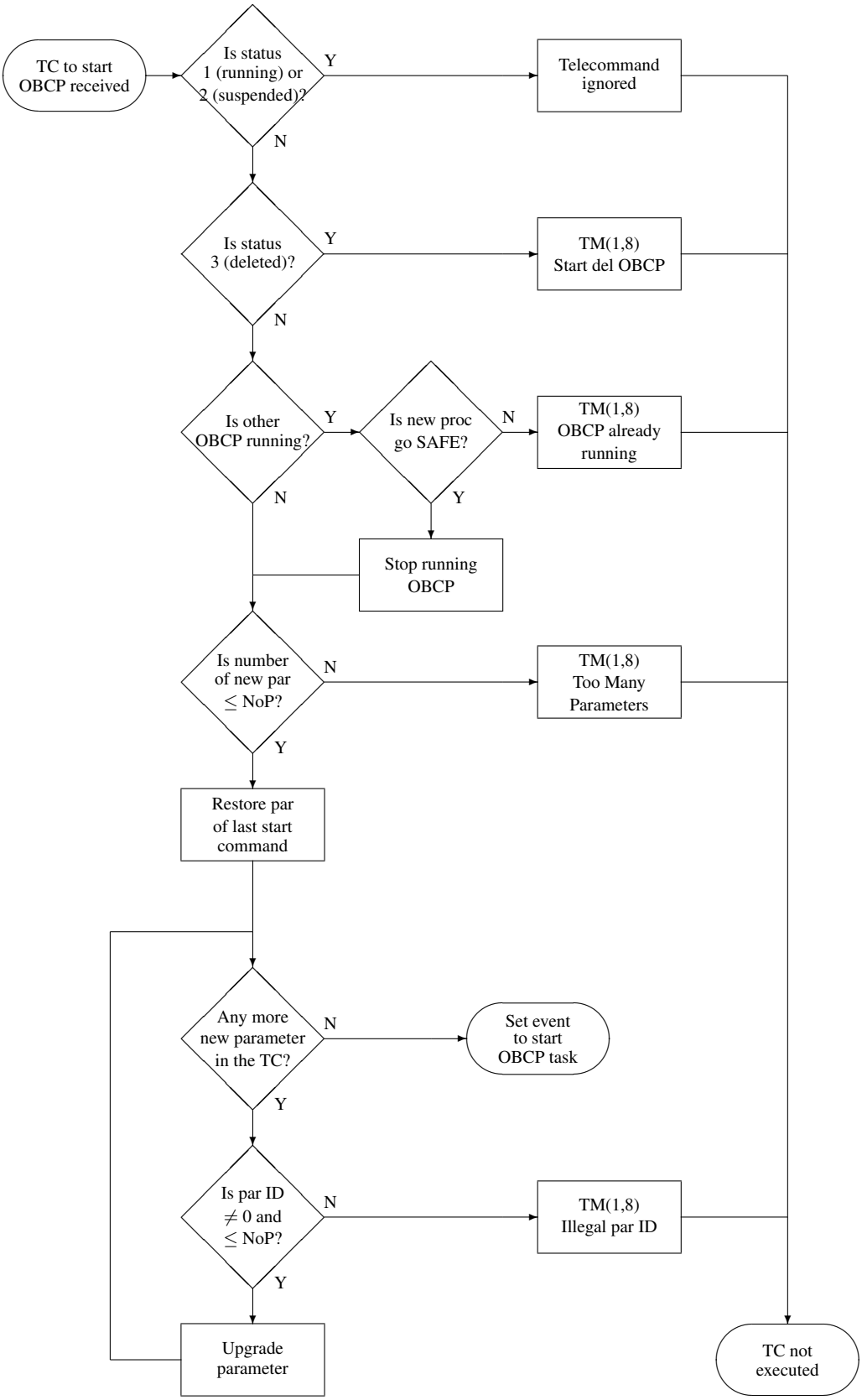
This is a generic error reported when the execution of a procedure was stopped before its end. Usually this happens because one command for a subsystem was not acknowldged. The failure reason can be found by looking at the events generated (see the communications scheme on page 84). It is also used in the procedure ”Start SPU HLSW” if the SPU status is not `NO NEW HK` or an acknowledgment is received from the SPU (see Section 4.3).

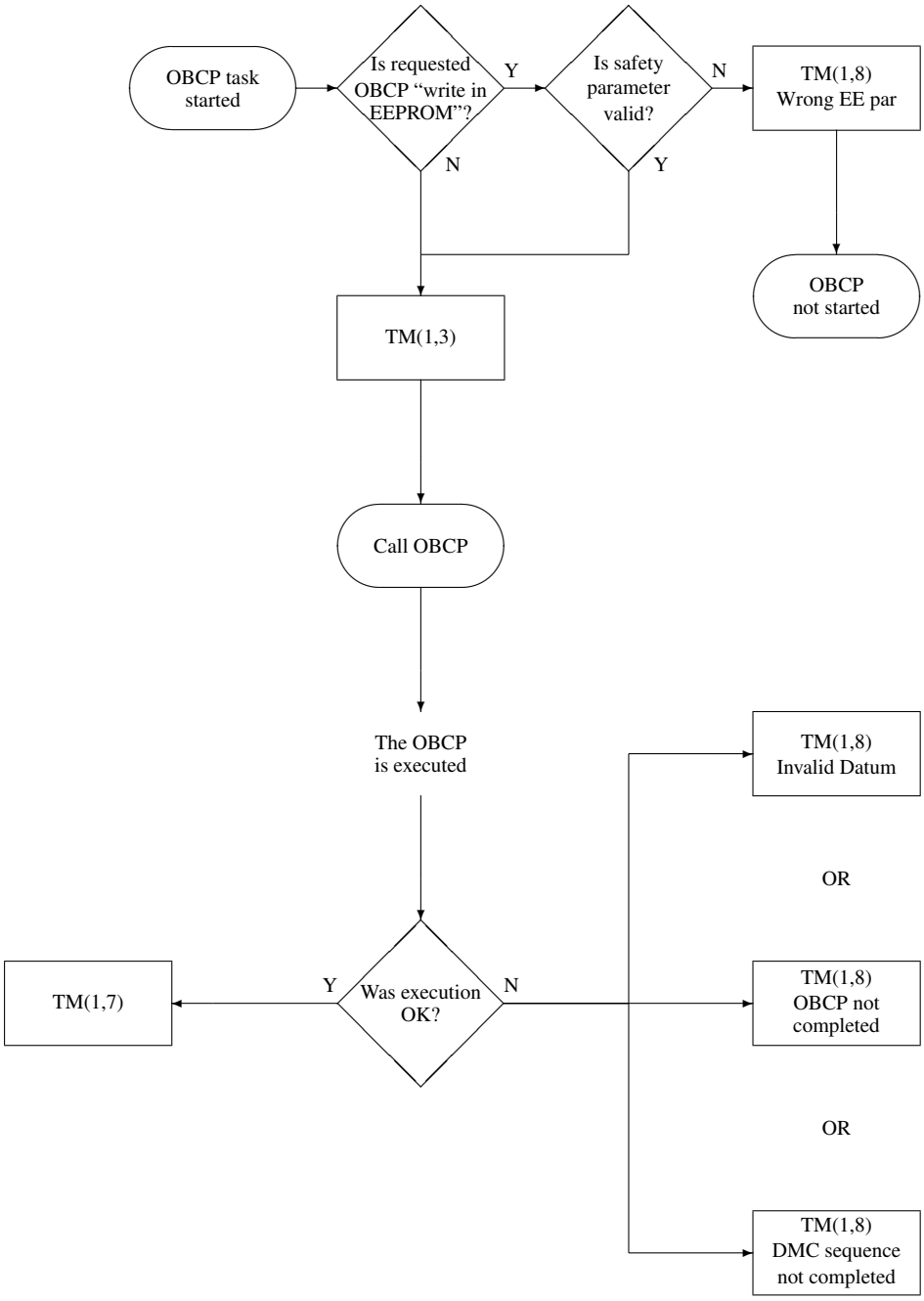
↯ Parameter reported in the TM (1,8) report: 2 (it is an internal error code and is meaningless for the user).

6.4.3–10 TM (1,8) — `Illegal STATUS: SEQ NOT Compl`

A DEC sequence has not been completed, according to the DEC HK sequence status, after the commanded waiting time has elapsed. Beside the (1,8) the DPU generates also the event `SEQ NOT Compl`.

↯ Parameter reported in the TM (1,8) report: the content of the DEC sequence status (see RD–4).


TC to start OBCP received
Is status 1 (running) or 2 (suspended)?
Y
Telecommand ignored
N
Is status 3 (deleted)?
Y
TM(1,8) Start del OBCP
N
Is other OBCP running?
Y
Is new proc go SAFE?
N
TM(1,8) OBCP already running
N
Y
Stop running OBCP
Is number of new par ≤ NoP?
N
TM(1,8) Too Many Parameters
Y
Restore par of last start command
Any more new parameter in the TC?
N
Set event to start OBCP task
Y
Is par ID ≠ 0 and ≤ NoP?
N
TM(1,8) Illegal par ID
Y
Upgrade parameter
TC not executed


OBCP task started
Is requested OBCP "write in EEPROM"?
Y
N
Is safety parameter valid?
N
Y
TM(1,8) Wrong EE par
OBCP not started
TM(1,3)
Call OBCP
The OBCP is executed
TM(1,8) Invalid Datum
OR
TM(1,8) OBCP not completed
OR
TM(1,8) DMC sequence not completed
Was execution OK?
Y
N
TM(1,7)

6.4.4 Stopping a procedure: TC(18,4)

The request is ignored if the status is `STOPPED` or `DELETED`. Once the command is executed, the status is set to `STOPPED`. This telecommand can be used also if the user does not know the ID of the running OBCP: in this case just set ID to 0.

P#1	Procedure ID

6.4.4–1 TM (1,8) — `Invalid DATA: Invalid PROCID`

P#1 is greater than 50.

↯ Parameter reported in the TM (1,8) report: the procedure ID (copy of *P#1*).

6.4.5 Suspend a procedure: TC(18,4)

The request is ignored if the status is `STOPPED` or `SUSPENDED`. Once the command is executed, the status is set to `SUSPENDED`.

P#1	Procedure ID
P#2	Step ID

6.4.5–1 TM (1,8) — `Invalid DATA: Invalid PROCID`

P#1 is outside the range 1—50.

↯ Parameter reported in the TM (1,8) report: the procedure ID (copy of *P#1*).

6.4.5–2 TM (1,8) — `Resource FAIL: SUSP TIMEOUT`

To suspend a procedure the DPU waits that the required step has been reached. If the step is not reached after 500 msec this error is reported and the procedure status is not changed. Currently all the procedures have defined only the starting step (0) which never changes. Any call to this service with a step different from 0 will then cause this error message.

↯ Parameter reported in the TM (1,8) report: the step ID (copy of *P#2*).

6.4.6 Resume a procedure: TC(18,6)

The request is ignored if the status is `STOPPED` or `ACTIVE`. Once the command is executed, the status is set to `ACTIVE`

P#1	Procedure ID

6.4.6–1 TM (1,8) — `Invalid DATA: Invalid PROCID`

P#1 is outside the range 1—50.

↯ Parameter reported in the TM (1,8) report: the procedure ID (copy of *P#1*).

6.4.7 Communicate parameters to a procedure: TC(18,7)

The request is ignored if the procedure does not exist (status equal to `DELETED`).

P#1	Procedure ID
P#2	n = number of uploaded parameters ($\leq$ NoP for an existing procedure; may be NoP+1 if *P#1* is 50)
P#3	ID of the 1st parameter
P#4	1st parameter
	...
*P#[(n*2)+1]*	ID of the last parameter
*P#[(n*2)+2]*	last parameter

6.4.7–1 Example: Communicate parameters to an existing procedure

We want to communicate the new values for 7th and 8th parameters of OBCP 12:

P#1	12
P#2	2
P#3	8
P#4	xxxx (new value of 8th parameter)
P#5	7
P#6	xxxx (new value of 7th parameter)

Note that the parameters are not required to be ordered.

6.4.7–2 Example: Communicate parameters to an uploaded procedure (1)

We have uploaded a new procedure that accepts 5 parameters. First we have to communicate to DPU how many parameters this OBCP accepts. Suppose also that all the parameters but the fourth are 0. Then we can use this command

P#1	50 (always for a new OBCP)
P#2	2
P#3	0
P#4	5 (ID 0 means that *P#4*, 5 in this case, is the maximum number of parameters for this OBCP)
P#5	4
P#6	xxxx (new value of 4th parameter)

6.4.7–3 Example: Communicate parameters to an uploaded procedure (2)

We have uploaded a new procedure that does not need parameters.

P#1	50 (always for a new OBCP)
P#2	1
P#3	0
P#4	0 (this OBCP does not have parameters)

6.4.7–4 TM (1,8) — `Invalid DATA: Invalid PROCID`

P#1 is outside the range 1—50.

↯ Parameter reported in the TM (1,8) report: the procedure ID (copy of *P#1*).

6.4.7–5 TM (1,8) — `Invalid DATA: Too Much PARAM`

This error occurs when on of these conditions is true:

1. *P#1* is not 50 AND *P#2* is greater than NoP;
2. *P#1* is 50 AND *P#3* is zero AND *P#4* is greater than 25 (in this case *P#4* is the number of parameters that the new procedure accepts, but this number is limited by SW design to 25);
3. *P#1* is 50 AND *P#3* is not zero AND *P#2* is greater than NoP (since *P#3* is not zero the DPU assumes that the number of parameters for this OBCP has already been sent, so that *P#2* must be less or equal to NoP).

↯ Parameter reported in the TM (1,8) report: *P#2* for cases 1) and 3); *P#4* for case 2).

6.4.7–6 TM (1,8) — `Invalid DATA: Illegal PAR-ID`

One of the parameters ID is zero or is greater than NoP.

↯ Parameter reported in the TM (1,8) report: the illegal parameter ID.

6.4.8 Report the list of existing procedures: TC(18,8) and TM(18,9)

On reception of this command the DPU generates a TM report (18,9), giving the number of stored procedures followed by their ID. At start-up the number is 34 and the ID runs from 1 to 34.

6.4.9 Report the list of active procedures: TC(18,10) and TM(18,11)

On reception of this command the DPU generates a TM report (18,11) giving the number, at most 1, and the procedure ID of the OBCP under execution. If no procedure is running number is zero.

6.4.10 Report OBCP status: TC(18,12) and TM(18,13)

P#1	Procedure ID

On reception of this command the DPU generates a TM report (18,13) giving the procedure ID, the Step ID with the procedure status, the number of parameters the procedure accepts and their values. The step is always zero while the status can be: 0 (`STOPPED`), 1 (`ACTIVE`), 2 (`SUSPENDED`), 3 (`DELETED`). At start-up the first 34 procedures are `STOPPED`, the remaining 16 are `DELETED`.

6.4.10–1 TM (1,8) — `Invalid DATA: Invalid PROCID`

P#1 is outside the range 1—50.

↯ Parameter reported in the TM (1,8) report: the procedure ID (copy of *P#1*).

7 Memory and Time management; Test service; Packet transmission control

In this section all the remaining services used inside PACS are presented with the exceptions of events, HK and science data that are considered separately in the next section.

7.1 Memory management

This service allows to load, dump and check memory. All the three commands require a memory ID, a start address and a length. The memory ID, which will be indicated with `0xii` in this section, is coded in 8 bits according to the following table

Subsystem ID (3 bits)	Memory type (1 bit)	Block (4 bits)	Explanation
$b_1b_2b_3$	b_4	$b_5b_6b_7b_8$	$b_1b_2b_3$ = 000b : command for DPU 001b : command for DEC 010b : command for blue SPU 011b : command for red SPU 100b : command for blue and red SPU (not supported for DUMP command)
			b_4 = 0b : PRAM (48 bit per word) 1b : DRAM (32 bit per word)
			$b_5b_6b_7b_8$ = 0000b : PROM 0001b : RAM 0010b : Extended RAM 0011b : EEPROM 0100b : SMCS DRAM 0101b : 1553 DRAM 0110b : DRAM mapped in PRAM

The PROM content is not actually visible to the OBS. However after the DPU is switched on, the PROM SW is copied in PM at addresses below `0x1555` and there it remains until DPU is switched off. These addresses are then considered as PROM. RAM is either PM or DM; the other blocks are mapped in DM with the exception of the last one (DRAM mapped in PRAM) which is mapped in PM from address `0x7BC00`.

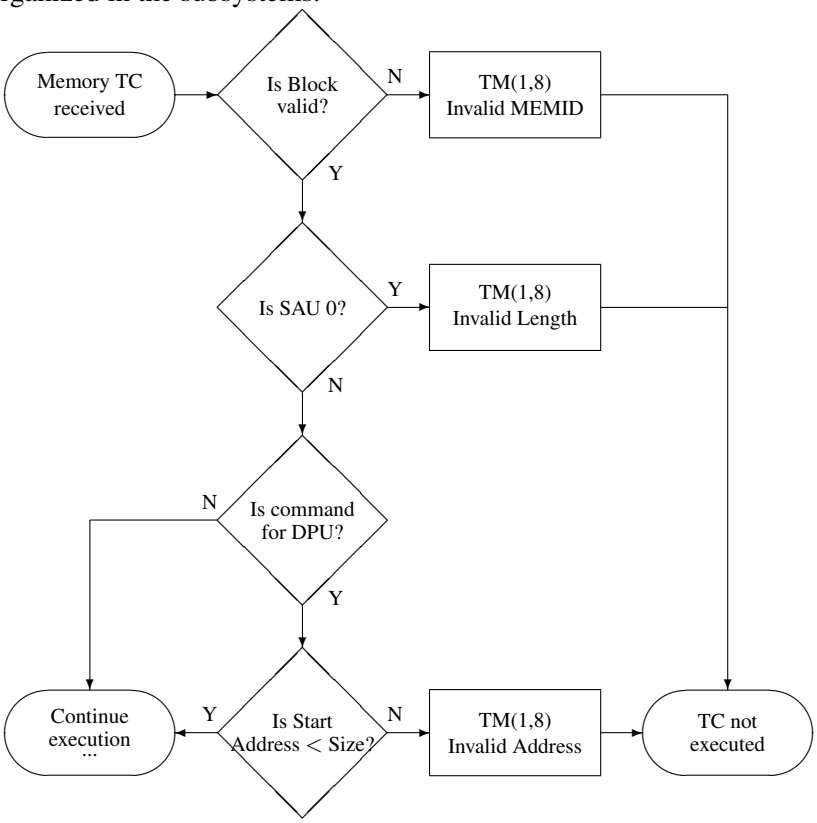
For the start address 24 bits are used; for compatibility with AD–2 which reserves only 16 bits for this field, a generic address `0xTTtttt` is logically splitted in 8 MSb (`0xTT`) and 16 LSb (`0xtttt`). For addresses larger than 24 bits an offset may be required according to the subsystem memory map. For the DPU the used offsets are

Block	Offset	Max. Length
PROM	0x00000000	0x1555
RAM (DM)	0x00000000	0x7FFFF
RAM (PM)	0x00000000	0x7BBFF
Extended RAM	0x84000000	0x6F
EEPROM	0x80000000	0x3FFFF
SMCS DRAM	0x40000000	0x1FFF
1553 DRAM	0x8F000000	0xFFFFFF
DM in PM	0x0007BC00	0x4400

The Extended RAM is used to address the SMCS registers while the SMCS DRAM corresponds to the 1355 interface dual port RAM. In the 1553 DRAM both the chip registers and the dual port RAM of the interface are mapped. The PROM and the EEPROM can only be dumped or checked (the application program can be copied into EEPROM using the procedure described in Section 4.1).

The length is expressed in SAU (Smallest Addressable Unit): 1 SAU corresponds to 6 bytes for PRAM words or 4 bytes for DRAM. Since the telecommand packet is organized in 16 bits words, 1 PM SAU is written in 3 words while 1 DM SAU is splitted in 2 words.

The controls common to all memory commands are shown in the figure below. Note that if the command is for a subsystem the only checks performed are on the 4 bits of Block (they are common throughout PACS) and on SAU being different from zero. No other checks are possible since DPU does not know how memory is organized in the subsystems.



7.1.1 Memory Load: TC(6,2)

P#1	`0xiiTT` (Memory ID and Start address, see the previous text)
P#2	`0xtttt` (Start address, see the previous text)
P#3	*N* (number of memory words in SAU)
for DRAM	
P#4	16 MSb of the 1st word
P#5	16 LSb of the 1st word
...	
P#[2x(N+1)]	16 MSb of the last word
P#[2x(N+2)]	16 LSb of the last word
P#[2x(N+3)]	crc
for PRAM	
P#4	16 MSb of the 1st word
P#5	16 intermediate bits of the 1st word

P#6	16 LSb of the 1st word
...	
P#[(3xN)+1]	16 MSb of the last word
P#[(3xN)+2]	16 intermediate bits of the last word
P#[(3xN)+3]	16 LSb of the last word
P#[(3xN)+4]	crc

Here are two examples to upload the word `0x123456789ABC` at address `0x46789` in PRAM and the word `0x12345678` at address `0x59876` in DRAM (DPU assumes little endian convention). The complete application data fields, without data field header and packet checksum, are

Program Memory		Data Memory	
Memory ID	`0x0104`	Memory ID	`0x1105`
Start address	`0x6789`	Start address	`0x9876`
Length	`0x0001`	Length	`0x0001`
Datum	`0x1234`	Datum	`0x1234`
Datum	`0x5678`	Datum	`0x5678`
Datum	`0x9ABC`	crc	`0x30EC`
crc	`0xA840`		

The maximum dimension of the application data field of a TC is 242 bytes, of which 4 are used for the data header and 2 for the CRC; 8 bytes are used for the memory ID, start address, length and crc. So that the maximum length for a memory load in DRAM is 57 words, while for PRAM is 38. If the TC is inserted in the mission timeline, its data field is reduced by 20 bytes, so that the maximum length is 52 or 34 words, for the two memory types. At the end of a successfull memory load the DPU reports a TM (1,7) packet. The full sequence of operations and checks performed are reported in the next picture

7.1.1–1 TM (1,8) — `Invalid DATA: Invalid MEMID`

`0xii` is not recognized as a valid memory ID. Note that since this parameter contains many informations, there are three possible sources for this error: Subsystem ID is wrong; Block is wrong; the block is not writeable (PROM or EEPROM).

↯ Parameter reported in the TM (1,8) report: `0xii` (memory ID extracted from *P#1*).

7.1.1–2 TM (1,8) — `Invalid DATA: Inv MEMLENGTH`

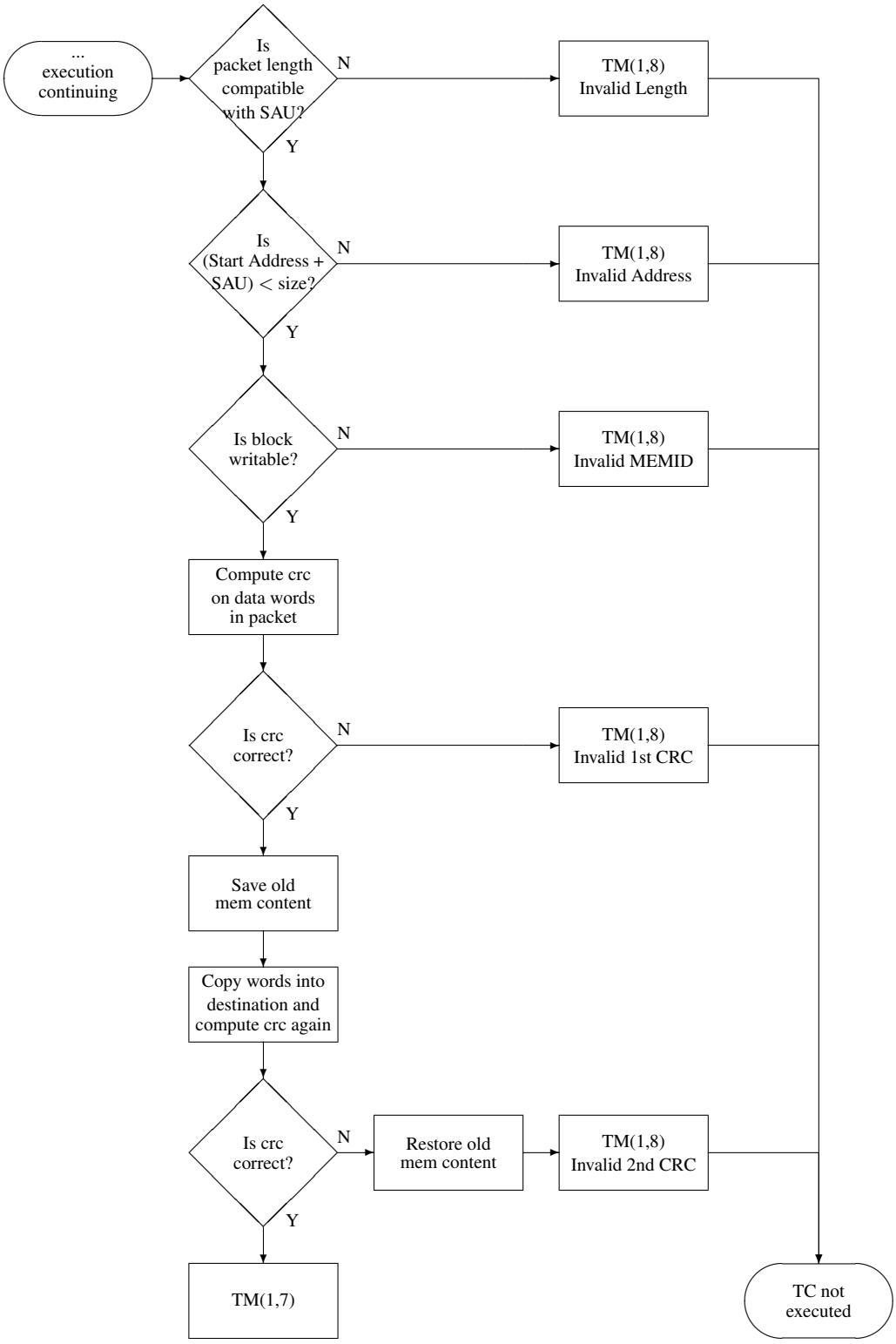
N, number of data words in SAU, is 0 or incompatible with the total packet length, ie (packet_length - 13)≠x**N* where x is 4 or 6 for DRAM or PRAM, respectively.

↯ Parameter reported in the TM (1,8) report: *P#3*.

7.1.1–3 TM (1,8) — `Invalid DATA: Inv ADDRESS`

`0xTTtttt` is not a valid start address, either because it is larger than “Offset + Max. Length” (see the second table of Section 7.1) or because the number of words are incompatible with the memory block size.

↯ Parameter reported in the TM (1,8) report: `0xTTtttt` where `0xTT` is extracted from *P#1* and `0xtttt` is the content of *P#2*.



7.1.1–4 TM (1,8) — `Invalid DATA: Inv CRC 1 CHK`

The checksum computed on the data words to be loaded in memory is not equal to the checksum (crc) reported in the TC packet.

↯ Parameter reported in the TM (1,8) report: the crc computed by the DPU.

7.1.1–5 TM (1,8) — `Resource FAIL: Inv CRC 2 CHK`

After copying the data into the final destination in memory, the DPU computes again the checksum on the written data. If this checksum is not equal to the crc written in the TC, this error is reported.

↯ Parameter reported in the TM (1,8) report: the crc computed by the DPU.

7.1.2 Memory Dump: TC(6,5) and TM(6,6)

P#1	`0xiiTT` (Memory ID and Start address, see above)
P#2	`0xtttt` (Start address, see above)
P#3	*Length* (in SAU)

On reception of this command the DPU generates one or more memory dump packets (6,6). The application data field of a TM packet can contain up to 1018 bytes. The data field header plus the packet checksum (CRC) need 12 bytes. Then we have the memory ID, the start address, the length and the checksum crc of the data, 8 bytes. In the end, the space left is 998 bytes: a single TM (6,6) packet can carry up to 166 or 249 words of PRAM or DRAM, respectively. In both cases the used bytes are 996. If *Length* has its largest value (65535) then 395/264 TM packets are generated for PRAM/DRAM, respectively.

The memory words are splitted in 16 bits words, in little endian mode. If, for instance, the DPU is required to dump 508 words of DRAM, with starting address `0x4FE14`, the following TM (6,6) packets are delivered:

Content of the 1st packet

P#1	`0x1104`
P#2	`0xFE14`
P#3	`0x00F9`
P#4	16 MSb of word at address `0x4FE14`
P#5	16 LSb of word at address `0x4FE14`
P#500	16 MSb of word at address `0x4FF0C`
P#501	16 LSb of word at address `0x4FF0C`
P#502	crc of the first 249 dumped words

Content of the 2nd packet

P#1	`0x1104`
P#2	`0xFF0D`
P#3	`0x00F9`
P#4	16 MSb of word at address `0x4FF0D`
P#5	16 LSb of word at address `0x4FF0D`
P#500	16 MSb of word at address `0x50005`
P#501	16 LSb of word at address `0x50005`
P#502	crc of the second 249 dumped words

Content of the 3rd packet	
P#1	0x1105
P#2	0x0006
P#3	0x000A
P#4	16 MSb of word at address 0x50006
P#5	16 LSb of word at address 0x50006
P#22	16 MSb of word at address 0x5000F
P#23	16 LSb of word at address 0x5000F
P#24	crc of the last 10 dumped words

7.1.2–1 TM (1,8) — Invalid DATA: Invalid MEMID

0xii is not recognized as a valid memory ID. Note that since this parameter contains many informations, there are two possible sources for this error: Subsystem ID is wrong (remember that for DUMP command 4 is not supported); Block is wrong.

↯ Parameter reported in the TM (1,8) report: 0xii (memory ID extracted from *P#1*).

7.1.2–2 TM (1,8) — Invalid DATA: Inv MEMLENGTH

N, number of data words in SAU, is 0.

↯ Parameter reported in the TM (1,8) report: *P#3*.

7.1.2–3 TM (1,8) — Invalid DATA: Inv ADDRESS

0xTTtttt is not a valid start address, either because it is larger than "Offset + Max. Length" (see the second table of Section 7.1) or because the number of words to dump are incompatible with the memory block size.

↯ Parameter reported in the TM (1,8) report: 0xTTtttt where 0xTT is extracted from *P#1* and 0xtttt is the content of *P#2*.

7.1.3 Memory Check: TC(6,9) and TM(6,10)

P#1	0xiiTT (Memory ID and Start address, see above)
P#2	0xtttt (Start address, see above)
P#3	*Length* (in SAU)

On reception of this command the DPU generates one memory check packet (6,10) with this content

P#1	copy of *P#1* contained in the TC
P#2	copy of *P#2* contained in the TC
P#3	copy of *P#3* contained in the TC
P#4	the computed crc

7.1.3–1 TM (1,8) — Invalid DATA: Invalid MEMID

0xii is not recognized as a valid memory ID. Note that since this parameter contains many informations, there are two possible sources for this error: Subsystem ID is wrong; Block is wrong.

↯ Parameter reported in the TM (1,8) report: 0xii (memory ID extracted from *P#1*).

7.1.3–2 TM (1,8) — `Invalid DATA: Inv MEMLENGTH`

N, number of data words in SAU, is 0.

↯ Parameter reported in the TM (1,8) report: *P#3*.

7.1.3–3 TM (1,8) — `Invalid DATA: Inv ADDRESS`

`0xTTtttt` is not a valid start address, either because it is larger than "Offset + Max. Length" (see the second table of Section 7.1) or because the number of words to check are incompatible with the memory block size.

↯ Parameter reported in the TM (1,8) report: `0xTTtttt` where `0xTT` is extracted from *P#1* and `0xtttt` is the content of *P#2*.

7.1.4 Memory commands for subsystems

If the memory command is for a subsystem the DPU (see next figure) controls that the subsystem ID is one of the allowed values (see Section 7.1). Then the data are sent according to the respective ICD: the decoding/coding for loading/dumping of a 32 bits or 48 bits word is responsability of the subsystem. Note that:

1. in case of memory load, if the DPU receives a positive acknowledgment a TM report (1,7) is issued, otherwise an event is generated (see page 84);
2. in case the subsystem ID is 4, the required memory command is first sent to the SPS; then, only in case of positive acknowledgment, to the SPL;
3. subsystem ID 4 is not supported for DUMP command and causes a TM(1,8) Invalid MEMID;
4. the length of an itermediate DUMP packet is not verified, ie the DPU assumes that it contains the largest allowed number of DM/PM words. If the length is shorter the DPU will write in the TM packet what found in the 1355 Dual Port RAM; if the packet is longer it will be truncated;
5. if a subsystem sends too many DUMP packets service (6,6) is disabled!

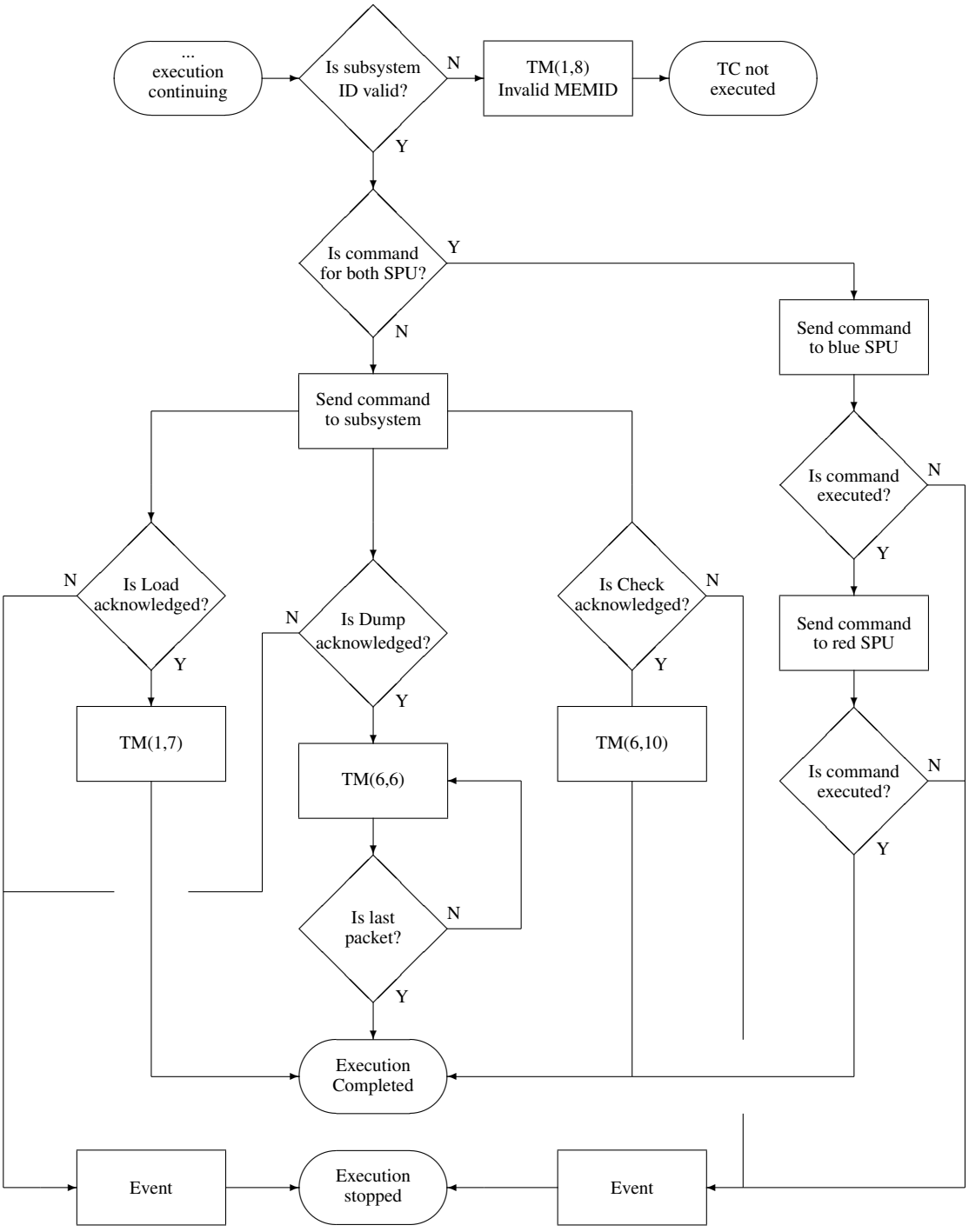
7.2 Time management

The DPU receives the time from the bus controller which distributes the sync signal every second. This (absolute) time is stored in an internal variable. When the time information is necessary, the DPU computes the fraction of seconds elapsed from the last sync signal and adds this quantity to the absolute time. The internal time is driven by the 20MHz clock and stored in a 32 bit register which then wraps around every ∼214 seconds: *this is the largest time interval over which the DPU can live without sync from the satellite while still providing the correct time*.

The time written in the TM packets is computed before the packet is buffered in a queue. The time at which the packet is sent to the CDMS is unpredictable, depending on the bus list, on the operative mode (prime, no prime), and on other TM packets pending.

7.2.1 Time verification: TC(9,7) and TM(9,9)

The only command accepted for this service is the Enable Time Verification (with no parameters). When this command is received, the DPU reports its time in a TM report (9,9), where the time is written in 48 bits: the first 32 bits are the seconds, while the LSB contains the fraction of seconds in 1/65536 seconds. This packet should be generated at the moment of next sync but since it is impossible for DPU to send a packet at a well defined moment, as said before, the time written in (9,9) is just the absolute time at the last sync plus 1 second.


...
execution
continuing
Is subsystem
ID valid?
N
TM(1,8)
Invalid MEMID
TC not
executed
Y
Is command
for both SPU?
Y
N
Send command
to blue SPU
Send command
to subsystem
Is command
executed?
N
Y
N
Is Load
acknowledged?
Y
N
Is Dump
acknowledged?
Y
N
Is Check
acknowledged?
Y
Send command
to red SPU
TM(1,7)
TM(6,6)
TM(6,10)
Is command
executed?
N
Y
Is last
packet?
N
Y
Execution
Completed
Event
Execution
stopped
Event

7.3 Packet transmission control

This service is used to enable or disable the trasmission to the CDMS of TM packets of a certain type/subtype/SID. This is the complete list of TM packets generated by the DPU and the status at startup: ✓ enabled; □ disabled; ☑ enabled (can not be disabled)

Name	Type	Subtype	SID	Status
TC ACCEPTANCE REPORT - SUCCESS	1	1		☑
TC ACCEPTANCE REPORT - FAILURE	1	2		☑
TC EXECUTION REPORT - STARTED	1	3		✓
TC EXECUTION REPORT - ENDED	1	7		✓
TC EXECUTION REPORT - FAILURE	1	8		✓
HK PACKET - SPEC	3	25	1	✓
HK PACKET - PHOT	3	25	2	✓
HK PACKET - NON PRIME	3	25	3	✓
HK ESSENTIAL PACKET	3	25	4	✓
EVENT REPORT	5	1	All	✓
EXCEPTION REPORT	5	2	All	✓
ERROR REPORT	5	4	All	✓
MEMORY DUMP	6	6		✓
MEMORY CHECK	6	10		✓
TIME VERIFICATION REPORT	9	9		✓
ENABLED TM PACKETS REPORT	14	4		✓
CONNECTION TEST REPORT	17	2		✓
OBCP LIST REPORT	18	9		✓
OBCP ACTIVE LIST REPORT	18	12		✓
OBCP STATUS REPORT	18	14		✓
SCIENCE SPEC BLUE	21	1	1	□
SCIENCE SPEC RED	21	1	2	□
SCIENCE PHOT BLUE	21	2	1	□
SCIENCE PHOT RED	21	2	2	□
DIAGNOSTIC DATA	21	3	0	✓

7.3.1 Enabling TM packets: TC(14,1)

This command allows to enable one or more TM packets that are currently disabled.

P#1	*N* (Number of packets to enable)
P#2	Type of first packet
P#3	Subtype of first packet
P#4	Packet-ID of the first packet
	...
P#(3xN)-1	Type of the last packet
P#(3xN)	Subtype of the last packet
P#(3xN)+1	Packet-ID of the last packet

Concerning the Packet-ID:

- for service 5 (Events) it is the Event ID (see Section 8.2)
- for services 3 (HK reporting) and 21 (Science data) it is the SID
- for all other services it is 0

Note that in the first two cases where the field Packet-ID is used, setting this parameter to 0 has a special meaning: it means to enable all the combinations (type,subtype). For instance, if type is 5, subtype 1 and Packet-ID is n, only the corresponding event will be enabled; if Packet-ID is 0, all events (5,1) will be enabled.

No error code is associated with this command. If a non existent combination of Type/Subtype/Packet-ID is sent, the DPU ignores the request. Enabling an already enabled packet has no consequence.

The science packets can be enabled with this command, however this should be avoided, otherwise the bit in the DPU HK **DP_STATUS** that signals which array is enabled (blue or red) is no longer meaningful. Use always the "Set HK list" command.

7.3.1–1 Example: enabling SPEC HK packet

P#1	1
P#2	3
P#3	25
P#4	1

7.3.2 Disabling TM packets: TC(14,2)

This command allows to disable one or more TM packets that are currently enabled.

P#1	*N* (Number of packets to disable)
P#2	Type of first packet
P#3	Subtype of first packet
P#4	Packet-ID of the first packet
	...
P#(3xN)-1	Type of the last packet
P#(3xN)	Subtype of the last packet
P#(3xN)+1	Packet-ID of the last packet

The Packet-ID has the same meaning as for the previous service (14,1).

No error code is associated with this command. If a non existent combination of Type/Subtype/Packet-ID is sent, the DPU ignores the request. Disabling an already disabled packet has no consequence. Any request to disable (1,1) or (1,2) is ignored.

The science packets can be disabled with this command, however this should not be done, otherwise the bit in the DPU HK **DP-STATUS** that signals which array is enabled (blue or red) is no longer meaningful. Use always the "Set HK list" command.

7.3.2–1 Example: disabling all event packets (5,1)

P#1	1
P#2	5
P#3	1
P#4	0

7.3.3 Reporting the list of enabled TM packets: TC(14,3) and TM(14,4)

When the DPU receives this TC, a TM report (14,4) is generated with the list of all the enabled TM packets. The list starts with the number of enabled packets (46 at start-up, of which 25 are events). This number can not be smaller than 3 because the TM reports (1,1) and (1,2) are always enabled, and to see this list the report (14,4) must be enabled. If a certain packet does not make use of the SID, the value 0 is reported.

This command does not accept any parameter.

7.4 Test service

7.4.1 Connection test: TC(17,1) and TM(17,2)

This command has no parameters. On its reception the DPU generates a TM report (17,2) with no data. Actually the DPU status can be already checked by the generation of the acceptance report (1,1). If the acceptance report is received but the (17,2) is not, then verify that this TM packet is enabled.

8 HK, events and science data

8.1 Housekeeping report and check of instrument health

The status of PACS is checked every two seconds by the DPU on the base of the values of HW measurements or SW status variables.

There are four blocks of HK: one for the DPU, discussed below, one for the DEC, which also contains the HK of the bolometers as well as the HW HK of the SPU, and one for each SPU, the blue and the red, which contain SW parameters only. The OBS generates the DPU HK and then checks the values coming from the subsystems. If no new values are available the last available values are packed.

The check done on the SW HK is described in the respective ICD (see AD–5 and AD–6). If their value is not as expected by the DPU, an event is raised. When an HW measurements goes out of limit, an event is raised and, if defined, an autonomy function is started. When the value comes back in limit a second event is raised (see the scheme reported in Section 5).

DPU HK

Hardware Measurements

Name	Calibration $0 \leq d \leq 4095$	Description	Soft limits $\pm 10\%$	Hard limits $\pm 20\%$
DP_VOL_25	$V = d \times 0.0012279$	2.5 V reference voltage	[1832—2240]	[1628—2444]
DP_VOL_5	$V = d \times 0.0014763$	5 V analogue channel	[3048—3726]	[2709—4065]
DP_VOL_15	$V = d \times 0.0044279$	+15 V analogue channel	[3048—3726]	[2709—4065]
DP_VOL_15	$V = -d \times 0.0044279$	–15 V analogue channel	[3048—3726]	[2709—4065]
DP_T	$T(°\mathrm{C}) =$ $d \times 0.0319254 - 50$	temperature sensor	[313—3758] [–40,+70]°C	[1—4071] [–50,+80]°C

Subsystems Status

Name	Value	Description
DP_SPS_LINK **DP_SPL_LINK** **DP_DMC_LINK**	0, 1	Status of 1355 interface. 0: link not started yet or stopped after an error 1: link active
DP_SPUS_CMD **DP_SPUL_CMD** **DP_DMC_CMD**	0, 1, 2, 3	0: link not yet started 1: link on (nominal condition) 2: error occurred after a command has been sent (NACK) 3: link lost after communications error (parity/disconnect)
DP_SPUS_HK **DP_SPUL_HK** **DP_DMC_HK**	0, 1, 2, 3	0: link not yet started 1: new HK packet received (nominal condition) 2: HK packet not received during the last 2 seconds 3: HK packets not received for the last 10 seconds

DPU Global Status

Name	Description
DP_WHICH_OBCP	A number between 1 and 50 is the ID of the running OBCP. 63 means that no OBCP is running
DP_WORK_LOAD	Workload in percentage of the CPU averaged over the last second. One unit corresponds to 0.01%

DPU Global Status (cnt.)

Name	Description	
DP_TM_RATE	1: spectroscopy observing mode 2: photometry observing mode 4: no prime operative mode	
DP_SW_VERS_ID	SW version identifier (first 4 bits of HK 38 in Appendix C)	
DP_SW_SUBVERS_ID	SW subversion identifier (last 7 bits of HK 38 in Appendix C)	
DP_AF_STATUS	24 bit signaling if the corresponding AF is enabled (1) or disabled (0). At switch-on AF#11 and 22 are enabled	
DP_STATUS	x x x x x x x x x x | | | | | | | | | |->	0/1 correspond to nominal/redundant DPU
	| | | | | | | | |--->	1 if TM science data buffer is full
	| | | | | | | |----->	1 if transmission of science from blue SPU is enabled
	| | | | | | |------->	1 if transmission of science from red SPU is enabled
	| | | | | |--------->	0/1 correspond to channel A/B of 1553
	| | | | |----------->	1 if DPU is in burst mode
	| | | |------------->	1 if a procedure is under execution
	| | |--------------->	1 if EEPROM can be written
	| |----------------->	1 if DPU is in Test Mode
	| ------------------>	SPARE

DPU Task Status

The following HK report the status of the OBS tasks (see AD–4). They correspond to HK with ID in the range [20—29] in the HK table of Appendix C. The values are coded according to the following scheme

0: task running
1: task stopped (initial value before the task is started)
2: task aborted
3: task sleeping (waiting on a timer)
4: task waiting on event
5: task waiting on FIFO
6: task waiting on semaphore
7: task waiting on resource
8: task in an unknown status. **Warning: this value signals an abnormal condition inside the DPU**

Task Name	Possible values							
	0	1	2	3	4	5	6	7
DP_DEC_LINK	✓	✓	✓		✓			✓
DP_OBCP_MANAGER	✓			✓	✓			✓
DP_CONTROLLER						✓		
DP_SPUS_LINK	✓	✓	✓		✓			✓
DP_INIT						✓		
DP_1355_HANDLER							✓	
DP_HK_MONITOR	✓							
DP_SPUL_LINK	✓	✓	✓		✓			✓
DP_1553_HANDLER					✓			
DP_IRQ3_TASK	✓							

1355 Interface Status

Name	Description
DP_DEC_LINK_PE **DP_DEC_LINK_DE** **DP_SPUS_LINK_PE** **DP_SPUS_LINK_DE** **DP_SPUL_LINK_PE** **DP_SPUL_LINK_DE**	These counters are incremented every time an error occurres in one of the 1355 links. According to the 1355 protocol, if the DPU detects a parity error the link is stopped and the other side detects a disconnect error; on the contrary, if the other unit detects a parity error, the link is stopped and the DPU reports a disconnect error. When the DPU sends the command to start the application software of one of the three links, the unit resets the 1355 interface so in this case the disconnect error is the nominal condition (see Sections 4.2 and 4.3)
DP_COM_DMC **DP_COM_SPUS** **DP_COM_SPUL**	After a command is sent to a subunit, the LSB of the corresponding counter is incremented if the DPU has received a positive acknowledgement, while the MSB is incremented if the DPU has received a negative acknowledgement

1553 Interface Status

Name	Description
DP_COM_REC_DPU	This counter is incremented every time the DPU receives a TC
DP_COM_REJ_DPU	The LSB of this counter is incremented every time the DPU sends a TM packet (1,2); the MSB is incremented when the DPU sends a TM packet (1,8)
DP_TC_LOST	This counter is incremented in case the DPU receives a TC while the telecommand buffer is full (its size corresponds to one single telecommand). This overflow happens if the DPU receives a new telecommand i while telecommand $i-1$ has been copied in the buffer and telecommand $i-2$ is still under execution. At a rate of 2 telecommands/second, this scenario appears very unlikely
DP_HK_LOST **DP_EVENT_LOST** **DP_GEN_TM_LOST**	In the DPU memory three areas are availabe to buffer TM packets if necessary. One buffer (64 packets) is for HK, nominal and additional; one is for the events (32 packets) and one is for all the other kind of packets (400 packets). The delivery of the packets is prioritized: event packets first, then HK and then all the others. If the event or HK buffer is full the corresponding counter is incremented. If a generic packet is lost the counter is incremented and the event `BUFFER FULL` is raised

8.1.1 Housekeeping parameter report: TM(3,25)

The DPU generates an HK packet every two seconds. There are three pre-defined packets: one for spectroscopy, one for photometry and one for non-prime mode. The corrispondent SID are 1, 2 and 3. The delivery to the spacecraft of this nominal HK packet can be disabled[6] (see Section 7.3).

At start-up the DPU is set by default in non-prime mode. The HK packet should be changed when the DPU makes a transition to a different observing/operative mode. This transition is usually done inside an OBCP and in this case the change of HK packet happens as part of the procedure. The HK packet can also be changed using the DPU command "Set HK list" (see Section 6.3.2).

This service is also used to deliver an additional HK packet generated every ten seconds (named Essential HK packet in AD–2): it has the same content of the non-prime packet but a different APID (1150 instead of 1152) and SID (4 instead of 3) and is delivered by the DPU in all the observing/operative mode.

[6]Disabling the transmission of the HK packet has no consequence on the check of the health of the instrument done by the DPU on the base of the HK values.

The complete content of the HK packets is reported in Appendix C. SID and the total packet length in bytes are

Mode	SID	Length
Spectroscopy	1	834
Photometry	2	886
Non Prime	3	388
Essential	4	388

Remember that in the length field of any TM packet the written number corresponds to (Length - 7), which means that, for instance, in spectroscopy the length is reported as 827 (or `0x33B`). The largest TM rate allocated for the essential HK is 330 bit/s: a length of 388 bytes corresponds to 388*8=3104 bit or 310.4 bit/s (1 packet every 10 seconds).

8.2 Events

Through this service the DPU informs the satellite of non-nominal conditions occurred in all the monitored subsystems of PACS. Three kind of reports can be generated: event report (5,1), in case something occurs that does not require an external action to be taken; exception report (5,2), in case recovery procedures are required to be executed by the spacecraft; error/alarm report (5,4), which informs of anomalous conditions that can be corrected only by ground.

The content of the packet is the following

P#1	Event ID
P#2	SID
P#3	OBSID
P#4	BBID
P#5	Event Sequence Counter
P#6—P#n	Parameters

In the following table there is the list of events, their ID and SID

Event	ID	Severity	SID
`NO 1355 ACK`	1	(5,1)	5
`WRONG DMC CHKSUM`	2	(5,1)	5
`NACK`	3	(5,1)	6
`GO SAFE`	4	(5,2)	0
Spare	5		
`POWER CYCLE`	6	(5,2)	0
`SS Stopped`	7	(5,1)	3
`DUMP too words`	8	(5,1)	5
`SEQ NOT Compl`	9	(5,1)	4
`SPUL DEAD`	10	(5,1)	0
`PM FAILURE`	11	(5,2)	1
`SCIENCE LOST`	12	(5,1)	5
`IMMEDIATE OFF`	13	(5,2)	0
`SPUS DEAD`	14	(5,1)	0
`COUNTER Error`	15	(5,1)	8
`DM FAILURE`	16	(5,4)	0xFF
Spare	17		

HK DPU SOFT	18	(5,1)	2
HK DPU OK	19	(5,1)	3
DEC DEAD	20	(5,1)	0
Spare	21		
HK DEC SOFT	22	(5,1)	2
HK DEC OK	23	(5,1)	3
Spare	24		
PACS NOMINAL OFF	25	(5,2)	0
Spare	26		
BUFFER FULL	27	(5,1)	1
Unexp 1355 ACK	28	(5,1)	5
Spare	29		
1355 Read ERR	30	(5,1)	8
1355 Timeout	31	(5,1)	3

The SID identifies the number and type of the parameters *P#6—P#n* according to the following table. As always, a 32 bits parameter is written with the 2 LSB following the 2 MSB

SID	P6	P7	P8	P9	P10	P11	*Length*
0				no parameter			25
1	2 octets	2 octets					29
2	2 octets	4 octets					31
3	2 octets						27
4	4 octets						29
5	2 octets	4 octets	4 octets				35
6	2 octets	4 octets	4 octets	4 octets	4 octets		43
7	2 octets	2 octets	4 octets	4 octets	4 octets	4 octets	45
8	2 octets	4 octets	4 octets	2 octets			37
0xFF	variable length, *P#6* (16 bits) gives the number of 32 bits parameters that follow						

OBSID and BBID are parameters set by ground people, the value is changed with dedicated DEC commands. Event Sequence Counter is a counter incremented for each event packet released: there are three separate counters, one for each subtype (5,1), (5,2) and (5,4). Here follows the description of each event.

8.2.1 Event report: (5,1)

8.2.1–1 NO 1355 ACK

No acknowledgement received within 200 msec after a command has been sent to a subsystem. Reported parameters:

1 the subsystem ID (0 DEC, 1 SPS, 2 SPL)
2 the first word of the command sent
3 the second word of the command sent

Note that for the LLSW commands “Load ASW” and “Perform a Reset”, no ack is the nominal condition so that this event is not reported. These commands are considered successfully sent after a disconnect error is detected on the 1355 link.

8.2.1–2 WRONG DMC CHKSUM

DMC HK packet is corrupted. If this happens the HK values are not used for determining the status of the instrument, i.e. no AF is started. Reported parameters:

1 internal status: 1, 2, 3 (see below)
2 received checksum
3 computed checksum

1 means that the packet has been received corrupted from DMC; 2 and 3 mean that the corruption occurred inside DPU memory. If the checksum verification is not to be performed (for instance because an old version of DMC OBSW is running), the user has to disable AF#22.

8.2.1–3 NACK

A negative acknowledgment from a subsystem has been received. Reported parameters:

1 the subsystem (0 DEC, 1 SPS, 2 SPL)
2 the first word of the command sent
3 the second word of the command sent
4 the first word of the received acknowledgment
5 the second word of the received acknowledgment

The reason for the negative acknowledgment is not analyzed by the DPU. The subsystems have their own error codes which are reported as 4th and 5th paramenters in the packet event. Note that after such an event occurres on one link, the commanding of all the subsystems is disabled.

Action: send the command “Set Function” with 1st parameter equal to the subsystem ID and 2nd parameter equal to 1.

8.2.1–4 SS Stopped

The user sent to the DPU a command for the subsystem xxx but: 1) the HK DP_xxx_CMD is 0 (link not yet started) or 3 (status after a parity/disconnect error); or 2) one or more HK DP_SPUS_CMD, DP_SPUL_CMD, DP_DMC_CMD are 2 (a NACK has been previously received). Reported parameters:

1 the function ID (101 SPS, 102 SPL, 103 DEC)

Condition 1 is checked only for the subsystem for which the command is intended, condition 2 is checked for all the three links.

Action: in case DP_xxx_CMD is 0 the user should start the link before trying to send a command to that subsystem; if DP_xxx_CMD is 3 the 1355 chip should be reset and all the three links should be started again; if one or more of the three DP_xxx_CMD is 2 send the command “Set Function” with 1st parameter equal to the subsystem ID and 2nd parameter equal to 1.

8.2.1–5 DUMP too words

When the DPU receives a DUMP command for a subsystem, it computes how many memory words are expected. If more words are received, this event is reported and the TM (6,6) (memory dump) is disabled (see Section 4.14.1 to enable it again). Reported parameters:

1 the subsystem (0 DEC, 1 SPS, 2 SPL)
2 the first word of the last received packet
3 the second word of the last received packet

Action: none foreseen on DPU side.

8.2.1–6 SEQ NOT Compl

Some OBCP make use of DEC sequences which are started when the DPU sends to DEC the command "Start Sequence". After this command the DPU waits a certain time, one of the OBCP parameters, and then verifies in the DEC HK if the sequence has been completed. If not this event is generated, the procedure is aborted and a TC report (1,8) is issued. Reported parameters:

1 the DMC_SEQ_STATUS as read in the DEC HK packet

Note: it is possible that during the (very short) interval between the reading of DMC_SEQ_STATUS in the OBCP and the generation of the event packet a new set of HK from DEC arrived, in which the sequencer status has changed. In turn, this means that the DMC_SEQ_STATUS written in the packet may be not the same read during the OBCP execution.

Action: none.

8.2.1–7 SPUL DEAD

This event is generated by the HK monitoring task if a new HK packet has not been received from the red SPU over the last 10 seconds. The DPU HK DP_SPUL_HK is then set to DEAD. This report does not contain parameters.

Note: this event is only informal and has no consequence at all on the activities of the instrument. The subsystem link remains active, commands can be sent, data received. If a new HK packet is received it is processed. However, it signals that something wrong may have occurred in the subsystem.

Action: none.

8.2.1–8 SCIENCE LOST

The science frames generated by the SPU are splitted in 1 or more packets. When the total number of packets is greater than 1, the DPU expects to receive the packets in a regular incrementing sequence: first packet, second packet and so on. If after packet i the DPU does not receive packet $i + 1$ this event is generated. Reported parameters:

1 the subsystem (1 SPS, 2 SPL)
2 the last PIX value (see AD–6)
3 expected packet (2 MSB), received packet (2 LSB)

Note: if the frame has been received in the correct order the last packet contains the remaining science data words but if one packet is lost, the length of the last packet is always set to largest allowed value (1024 bytes).

Action: none. Note that after the command STOP_REDUCTION_COMPRESSION this event is nominal.

8.2.1–9 SPUS DEAD

This event is generated by the HK monitoring task if a new HK packet has not been received from the blue SPU over the last 10 seconds. The DPU HK DP_SPUS_HK is then set to DEAD. This report does not contain parameters.

Note: this event is only informal and has no consequence at all on the activities of the instrument. The subsystem link remains active, commands can be sent, data received. If a new HK packet is received it is processed. However, it signals that something wrong may have occurred in the subsystem.

Action: none.

8.2.1–10 COUNTER Error

The DPU performs a check on some subsystems counters. There are two types of counters: for the first type the nominal condition is that the counter is not incremented (eg counters of memory failure); the second type of counters are constantly incremented (eg number of packets sent from BOLC and received from DEC). This event is then generated when, for the former type, the new value is different from the last one or, for the latter type, when the new value is equal to the last received. Reported parameters:

1 the subsystem (0 DEC, 1 SPS, 2 SPL)
2 the old counter content
3 the new counter value
4 the index of the counter in the subsystem HK packet

8.2.1–11 HK DPU SOFT

The DPU HW HK are sampled and checked against the nominal range every 2 seconds. If the value is out of soft limits this report is generated. Reported parameters:

1 the ID of the HK (0 = DP_VOL_25P; 1 = DP_VOL_5P; 2 = DP_VOL_15P; 3 = DP_VOL_15N)
2 the HK value (in digital units)

Note: the event marks the transition in-limits→out-of-limits. If during next checks the HK is still out of limits, the event is not generated.

8.2.1–12 HK DPU OK

A HK value previously found out-of-limits is now back in-limits. Reported parameters:

1 the ID of the HK (see above)

8.2.1–13 DEC DEAD

This event is generated by the HK monitoring task if a new HK packet has not been received from DEC over the last 10 seconds. The DPU HK DP_DMC_HK is then set to `DEAD`. This report does not contain parameters.

Note: this event is only informal and has no consequence at all on the activities of the instrument. The subsystem link remains active, commands can be sent, data received. If a new HK packet is received it is processed. However, it signals that something wrong may have occurred in the subsystem.

Action: none.

8.2.1–14 HK DEC SOFT

The DEC HW HK are received and checked against the nominal range every 2 seconds. If the value is out of soft limits this report is generated. Reported parameters:

1 the ID of the HK (see RD–4)
2 the HK value (in digital units)

Note that the event marks the transition in-limits→out-of-limits. If during next checks the HK is still out of limits, the event is not generated.

8.2.1–15 HK DEC OK

A HK value previously found out-of-limits is now back in-limits. Reported parameters:

1 the ID of the HK (see RD–4)

8.2.1–16 BUFFER FULL

Since the generation of TM packets inside the DPU is not synchronous with the packets transmission to the satellite, the packets are buffered. Events and HK have their own buffer, all the other packets are stored in a third buffer that can contain at most 400 packets. If this buffer is full, likely for a too high data rate from SPU, this event is generated and DPU disables the transmission of science packets to the spacecraft until its internal buffer is at least 25% free. Reported parameters:

1 the packet ID of the first lost packet (1st word of the packet header according to AD–2)
2 the sequence counter of the first lost packet (2nd word of the packet header according to AD–2)

8.2.1–17 Unexp 1355 ACK

An acknowledgment has been received by the DPU without sending a command. Reported parameters:

1 the 1355 link (0 DEC, 1 SPS, 2 SPL)
2 the first word of the received acknowledgment
3 the second word of the received acknowledgment

Note: in case a subsystem sends an ACK after more than 200 msec, the DPU reacts reporting first the event `NO 1355 ACK` and then the event `Unexp 1355 ACK`.

8.2.1–18 1355 Read ERR

DEC has sent: a nominal HK packet whose length does not correspond to the expected length; or a diagnostic HK packet whose length is larger than the maximum allowed size (250 words). Reported parameters:

1 0 (the 1355 link corresponding to DEC)
2 the first word (the header) of the packet (`0x00870000` for nominal HK packets, `0x00880000` for diagnostic HK packets)
3 for nominal HK packets the expected length; for diagnostic HK packets the number of words written in the packet received from DEC
4 for nominal HK packets the number of words written in the packet received from DEC; for diagnostic HK packets this parameter is 250, the largest number allowed for compatibility with AD–2

8.2.1–19 1355 Timeout

After the DPU sends a command to a subsystem, it waits for the “EOP sent” interrupt from the 1355 interface. The largest packet that can be sent, according to the ICD with the subsystems, is 512 words, which requires less than 2 msec at 10MHz. If after 2 msec the interrupt has not yet been received, the DPU generates this event report and the command is considered not sent. Reported parameters:

1 the 1355 link (0 DEC, 1 SPS, 2 SPL)

Action: this event signals an anomaly at 1355 chip level. Send the command “Reset 1355” and restart the links.

8.2.2 Exception report: (5,2)

For these events see also AD–8.

8.2.2–1 GO SAFE

This event is sent when DPU wants the satellite command the execution of the SAFE OBCP, usually after a critical HK value goes out-of-limit. It does not contain parameters.

8.2.2–2 POWER CYCLE

This event is not currently used.

8.2.2–3 PM FAILURE

Beside Service (6,9) (Memory check) to verifiy the integrity of memory content, there is also a DPU specific command (see Section 6.3.13) that contains the PM memory area to check and the expected result. If the resulting checksum is not what is written in the telecommand, the DPU generates a TM (1,8) and this event. Reported parameters:

1 the crc written in the telecommand packet
2 the computed checksum

Action: it is suggested to dump the same DPU memory area specified in the telecommand.

8.2.2–4 IMMEDIATE OFF

This event is currently not used.

8.2.2–5 PACS NOMINAL OFF

This event is generated when one HK goes out of hard limits and then it is necessary to switch off PACS (see autonomy functions description in Section 5).

8.2.3 Error/alarm report: (5,4)

8.2.3–1 DM FAILURE

Every second an interrupt routine checks the integrity of the DM. The old content is saved and a writing/reading of the memory cell is performed. If the read content is not equal to the written value the memory cell is damaged. This routine verifies 6 cells per second which means that the full DM is checked in about 24 hours. The HK monitoring task controlls every two seconds if some cell is damaged and in case generates this event. This packet has a variable length. Reported parameters:

1 the number of damaged cells
2 address of first damged cell
n address of the last damaged cell

Action: it depends on the address of the memory cell(s). It may not require any action, or it may be necessary to switch to redundant unit.

8.3 Science Data Transfer

This service is used to report science data packets received from the SPU, and diagnostic HK data from DEC. Remember that with the exception of diagnostic HK data, the TM packets of service 21 are disabled at start-up and must be enabled either with the command “Set HK list” (Section 6.3.2, preferred way) or with Service 14 (Section 7.3, not recommended).

For science data the structure of the packets is the following

P#1	SID
P#2	Counter
P#3	Number of packets
P#4—P#n	Data

Each science frame is splitted in a number of packets given by *P#3*; Counter is incremented in each packet, from 1 to *P#3*. The complete structure of the packets is shown in Section 9.10.1 where also the use of segmentation flag is reported; here are the SID (note the change of APID between nominal and redundant unit)

APID			
Nom.	Red.	SID	
0x48A	0x48B	1	Science data from blue SPU
488	489	2	Science data from red SPU
486	487	0	Diagnostic HK packet

The last packet of a set of science packets has variable length according to the size of the science frame. **If, for any reason, the computed length of the last packet is greater than the maximum TM packet size, the length is set to the maximum allowed value (1017 bytes) and** `0x40` **is added to the SID**.

8.3.1 Nominal Science Data Report: TM(21,1)

This service is used to transmit spectroscopic data. The packet structure follows the content given at the beginning of this section. The last packet has variable length, depending on the total length of the science frame. However, in burst mode the length of the last packet is always set to 1024 bytes as requested by AD–2.

8.3.2 Science Type B Data Report: TM(21,2)

This service is used to transmit photometric data. All the fields have the same meaning of the previous service.

8.3.3 Diagnostic Science Report: TM(21,3)

For diagnostic purposes DEC can be commanded to generate specific HK values at a rate higher than the nominal one (2 seconds). These values are transmitted to ground using this service. The first word, the SID, is always 0. Then all the data follow. The length of this packet is variable unless PACS is in burst mode. In this case the packets have fixed length (1024 bytes) like all the other science data packets. For a description of the content of the packets see RD–4.

9 Reference

In this section the packets structure is reported for each service, both for telecommands and telemetry packets. If necessary, additional informations are given on how the commands are internally processed by the DPU.

According to AD–2, a TC packet has the following structure (each cell is a binary digit, each word is 16 bits, here and in all other examples it is assumed that the LSb of the TC ACK field is set to 1; actually this bit is always checked)

Packet Header															
TC Packet ID	0	0	0	1	1	`0x480` or `0x481` (the APID)									
TC Packet Sequence Control	1	1	*Counter*												
TC Packet Length	*Length*														
Data Header															
Packet Type	0	0	0	0	0	0	0	1	`0xTT`						
Packet Subtype	`0xSS`								0	0	0	0	0	0	0	0
Application Data															
1st word	Any 16 bits number														
	...														
k-th word	Any 16 bits number (can be a crc)														
TC Packet Checksum	CRC														

where $k = \frac{Length+1}{2} - 3$. The total packet length, in bytes, is *Length* + 7. *Counter* is a number which is only used by the DPU when preparing packets of type (1,x), othwerise it is ignored. The CRC is the checksum computed on the whole packet, including the header; crc is a checksum computed, for some packets, on a subset of the application data.

For the TM packets the APID can take on different values (see next Table). The structure is

Packet Header															
TM Packet ID	0	0	0	0	1	APID									
TM Packet Sequence Control	1	1	*Counter*												
TM Packet Length	*Length*														
Data Header															
Packet Type	0	0	0	0	0	0	0	0	`0xTT`						
Packet Subtype	`0xSS`								0	0	0	0	0	0	0	0
2 MSB of seconds	`0xtttt`														
2 LSB of seconds	`tttt`														
Fractions of seconds	`ssss`														
Application Data															
1st word	Any 16 bits number														
	...														
k-th word	Any 16 bits number (can be a crc)														
TM Packet Checksum	CRC														

where $k = \frac{Length+1}{2} - 6$. The total packet length, in bytes, is *Length* + 7. *Counter* is a field incremented by one for each packet, and separately for each APID, delivered by the DPU. Note also that for science packets the first two bits before *Counter* can differ from `11` (see Section 9.10.1). CRC and crc are defined as above. The last three words of the data header are the time in the format specified in AD–2: `0xttttttttt` is the 32 bits counter of seconds, while `0xssss` is the fraction in units of 1/65536 of seconds.

The following APID are used by PACS (N for nominal and R for redundant unit)

N	R	
0x480	0x481	All generic TM packets; additional HK packet
482	483	Periodic HK packets
484	485	Not used
48A	48B	Science data from blue SPU
488	489	Science data from red SPU
486	487	Diagnostic HK packet

Throughout this section, only APID for nominal unit shall be used in the examples.

9.1 Telecommand Verification Service

9.1.1 Telecommand Acceptance Report – Success: TM(1,1)

If the packet passes the checks performed on reception (see the figure on page 31), a TM report (1,1) is immediately generated with this structure

0	0	0	0	1	0x480										
1	1	*Counter*													
15															
0	0	0	0	0	0	0	0	0x01							
0x01								0	0	0	0	0	0	0	0
0xtttt															
tttt															
ssss															
TC Packet ID															
TC Packet Sequence Control															
CRC															

9.1.2 Telecommand Acceptance Report – Failure: TM(1,2)

If a check is not passed a TM report (1,2) is immediately generated

0	0	0	0	1	0x480										
1	1	*Counter*													
21															
0	0	0	0	0	0	0	0	0x01							
0x02								0	0	0	0	0	0	0	0
0xtttt															
tttt															
ssss															
TC Packet ID															
TC Packet Sequence Control															
Failure Code															
0xpppp															
0xqqqq															
CRC															

where the Failure Code is taken from AD–2. For 0xpppp and 0xqqqq see the table reported on page 30.

9.1.3 Telecommand execution started: TM(1,3)

This report is issued by the DPU when a procedure is started, its structure is the same as for Service (1,1).

9.1.4 Telecommand execution completed: TM(1,7)

This report packet has the same structure of Service (1,1). It is used after successful completion of an OBCP (Section 6.4.3); after loading the last segment of an OBCP (Section 6.4.1); at the end of a memory load (Section 7.1.1); after successful completion of Check PM command (Section 6.3.13).

9.1.5 Telecommand execution failure

This report is used either when a command can not be executed or its execution failed. The packet structure is the following:

0	0	0	0	1	0x480										
1	1	*Counter*													
23															
0	0	0	0	0	0	0	0	0x01							
0x08								0	0	0	0	0	0	0	0
0xtttt															
tttt															
ssss															
TC Packet ID															
TC Packet Sequence Control															
Failure Code															
Error Code															
0xpppp															
pppp															
CRC															

The Failure Code can be (see also Section 6.1.3): 5 (`Illegal DATA`), 16 (`Illegal STATUS`) or 17 (`Resource FAIL`). Error Code and the 32 bits parameter depend on the specific service: each (1,8) is discussed in the specific subsection.

9.2 Housekeeping & Diagnostic Data Reporting

As explained in Section 8.1, PACS uses two HK reports: the nominal, sent every two seconds, and the additional, sent every ten seconds. At start-up the default mode is non prime.

9.2.1 Housekeeping parameter report: TM(3,25)

0	0	0	0	1	0x482										
1	1	*Counter*													
Length															
0	0	0	0	0	0	0	0	0x03							
0x19								0	0	0	0	0	0	0	0
0xtttt															
tttt															
ssss															
SID															
OBSID 0xpppp															
pppp															
BBID 0xqqqq															
qqqq															
DPU HK															
Red SPU HK															
Blue SPU HK															
DEC HK (without OBSID and BBID)															
CRC															

OBSID and BBID are extracted from DEC HK; note that from DPU HK on, the values are written as a stream of bit. The complete content of the packets is reported in Appendix C.

9.2.1–1 Additional housekeeping parameter report: TM(3,25)

This packet is the exact copy of the non-prime mode, but with APID `0x480` and SID 4. It is sent every ten seconds

0	0	0	0	1	`0x480`										
1	1	*Counter*													
`0x17D`															
0	0	0	0	0	0	0	0	`0x03`							
`0x19`								0	0	0	0	0	0	0	0
`0xtttt`															
`tttt`															
`ssss`															
4															
The remaining part is like the nominal non-prime HK packet															

9.3 Event Reporting: TM(5,1), (5,2) and (5,4)

0	0	0	0	1	`0x480`										
1	1	*Counter*													
Length															
0	0	0	0	0	0	0	0	`0x05`							
`0xSS`								0	0	0	0	0	0	0	0
`0xtttt`															
`tttt`															
`ssss`															
Event ID															
SID															
OBSID `0xpppp`															
`pppp`															
BBID `0xqqqq`															
`qqqq`															
x	x	Event Counter													
A number of parameters depending on SID															
CRC															

`0xSS` is one of the three subtypes defined in AD–2: Event Report `0xSS=1`; Exception Report `0xSS=2` and Error/Alarm Report `0xSS=4`. The Event Counter field (14 bits) is different, each subtype having its own, and is incremented every time a packet is generated: `xx` is `01` for (5,1), `10` for (5,2) and `11` for (5,4). In Section 8.2 the list of ID, SID and the description of each event are reported.

9.4 Memory Management: service 6

The content of telecommands and telemetry packets are in Section 7.1; here the error codes for TM(1,8) are given

Error message	Failure Code	Error Code	Parameter
`Invalid MEMID`	5	18	`000000ii`
`Inv ADDRESS`	5	19	`00aaaaaa`
`Inv MEMLENGTH`	5	20	`00001lll`
`Inv CRC 1 CHK`	5	21	`0000rrrr`
`Inv CRC 2 CHK`	17	27	`0000rrrr`

`ii` is the memory ID as shown in the table on page 55; `aaaaaa` is the 24 bits address; `llll` is the length in SAU as given in the TC; `rrrr` is the crc computed by the DPU.

9.5 Function Management

9.5.1 Perform activity: TC(8,4)

A *TRIGGER* command for a subsystem has the following structure (32 bits parameters)

0	0	0	1	1	0x480										
1	1	*Counter*													
9 + SID*4															
0	0	0	0	0	0	0	1	0x08							
0x04								0	0	0	0	0	0	0	0
0xSSII															
SID = 0				SID = 1				SID = 2				SID = 5			
CRC				0xpppp				0xpppp				0xpppp			
				pppp				pppp				pppp			
				CRC				0xpppp				0xpppp			
								pppp				pppp			
								CRC				0xpppp			
												pppp			
												0xpppp			
												pppp			
												0xpppp			
												pppp			
												CRC			

`0xII` is the activity ID: for the subsystems it is reported in RD–4 and AD–6. The DPU does not check that there is consistency between the TC packet length and the SID: **if the packet has less parameters than expected on the base of the SID the subsystem receives meaningless numbers**. `0xSS` is the function ID used to identify the subsystem, according to the table on page 33

Activity ID's for the DPU commands are here reported. Note that the trigger commands (those with SID different from 4) use 16 bit parameters with the exceptions of "Copy OBSW" and "Check PM": the former accepts two 32 bit parameters and one 16 bit parameter, the latter uses one 16 bit parameter and two 32 bit parameters

Command	Activity ID	SID
Upgrade Sequence	1	4
Delete Sequence	2	1
Add Sequence	3	4
Set HK list	4	2
Force execution of autonomy function	5	1
Set function	6	2
Warm reset	7	0
Send time to DEC	8	0
Jump to boot software	9	0
Set buslist	10	1
Reset 1355 interface	11	0
Enter Test Mode	12	1
Reset 1553 interface	13	0
Copy OBSW (patching)	14	3
Check PM	15	3

For the commands that accept 16 bit parameters the structure of the packet is

0	0	0	1	1	0x480										
1	1	*Counter*													
9 + SID*2															
0	0	0	0	0	0	0	1	0x08							
0x04								0	0	0	0	0	0	0	0
0x64II															
SID = 0								SID = 1							SID = 2
CRC								0xpppp							0xpppp
								CRC							0xpppp
															CRC

For *WRITE* commands the SID is always 4: for a subsystem the packet is so structured

0	0	0	1	1	0x480										
1	1	*Counter*													
15 + *Length**4															
0	0	0	0	0	0	0	1	0x08							
0x04								0	0	0	0	0	0	0	0
0xSSII															
4															
Write ID															
Length															
0xpppp															
pppp															
...															
0xpppp															
pppp															
crc ((2**Length*+5)th word)															
CRC ((2**Length*+6)th word)															

crc is the checksum computed on all the 0xpppppppp words (from 5th word to (2**Length*+4)th word). As before, the DPU does not check that the packet length is consistent with *Length*, nor it checks the crc. If the command is for the DPU then 0xSS=0x64. **Both for DPU and subsystems, SID=4 is reserved for *WRITE* commands.**

In case the command is for a subsystem, these are the errors code in case of a TM(1,8):

Error message	Failure Code	Error Code	Parameter
Invalid FUN-ID	5	0x0801	000000SS
Invalid SID	5	0x0803	SID
UNIT STOPPED	17	0x080A	0000000z
FUNCT TIMEOUT	17	0x080C	0000000z
Inv COMMAND	5	0x080D	000000SS
FUNK LINK USED	16	0x080E	0000000z

SS is the function ID and, like SID, is copied from the TC packet; z is 0 for DEC, 1 for the blue SPS and 2 for the red SPL.

For the DPU commands these are the possible errors:

Error message	Failure Code	Error Code	Parameter
Invalid AF	5	0x0802	0000pppp
Invalid CRC	5	0x0804	crc
SEQ NO Space	17	0x0805	$\leq$1500
Invalid ACT-ID	5	0x0806	000000II
Invalid SEQ-ID	5	0x0807	0000pppp
Invalid SEQ-ID	16	0x0807	0000pppp
Invalid PARAM	5	0x0808	0000pppp
FUNCT STOPPED	16	0x0809	0000pppp
UNIT STOPPED	16	0x080A	00000000
Invalid ARRAY	5	0x080B	0000pppp
FUNCT TIMEOUT	17	0x080C	00000000
FUNK LINK USED	16	0x080E	00000000

0xpppp is the copy of the received parameter; II is the activity ID and, like crc, is copied from the TC packet. The Failure Code for the error Invalid SEQ-ID is 5 (Illegal DATA) if the ID of the sequence is greater than 32; or 16 (Illegal STATUS) if the ID corresponds to a non existent sequence for a delete/upgrade command, or to an existing sequence for an add command. The error Invalid PARAM has different meanings depending on the command that generates it, see Section 6.3.

9.5.1–1 Communications mechanism

Before sending a command to a subsystem the DPU makes some checks on the HK status associated to the subsystems and only afterwards the command is sent. The following sheme is followed

	Report (5,1)	DPU action
a command has been previously sent and the DPU is still waiting the ACK		TM(1,8) `FUNK LINK USED`
the link is not active	`SS Stopped`	TM(1,8) `UNIT STOPPED`
one of the three links has previously replied to a command with a NACK	`SS Stopped`	TM(1,8) `UNIT STOPPED`
the command is "Start high level SW"		TM(1,8) `Inv COMMAND`
The command is finally sent		
the DPU does not receive the "end of packet sent" interrupt after 2 msec (this might be a problem on the other link)	`1355 Timeout`	TM(1,8) `FUNCT TIMEOUT`
the command has been sent but no acknowledgment has been received within 200 msec	`NO 1355 ACK`	TM(1,8) `UNIT STOPPED`; the HK DP_xxx_CMD is set to `STOPPED`: commands can not be sent to any subsystem but all the packets are received and processed. To restart the link use the DPU command"Set function" (see Section 6.3.3)
the correct PACK is received		
a NACK is received (the DPU does not interpret the packet, i.e. any packet header different from the expected PACK is treated as NACK)	`NACK`	Same as case `NO 1355 ACK`

9.6 Time Management

The DPU time is synchronized with the satellite time every second using SA 8R (see Appendix 9 of AD–2). When DPU receives the sync the satellite time is stored. At (almost) the same moment the internal time, as measured by the Virtuoso high frequency timer, in turn based on the 21020 clock@20 MHz, is saved. When DPU needs to know the time in between two sync signals, the Virtuoso timer is read again and the previous reading is subtracted. This offset is then added to the absolute time received at the last sync signal. When the OBS is started the seconds are set to the value `0x80000000`, i.e. the MSb is 1, until the first sync signal is received.

The timestamp written in the TM packets is computed before copying the packet in one of the TM buffers. The actual time of the packet delivery to the spacecraft depends on the priority of the packet, on the packets queue as well as on the CDMS adopted bus profile.

At the frequency of 20 MHz, the Virtuoso 32 bit high resolution timer wraps around after ∼214 seconds. This is the largest time interval over which the DPU time can be computed correctly without receiving the sync from the satellite. After this interval the time is still written in the packets but its relative as well as absolute value is incorrect.

9.6.1 Enable Time Verification: TC(9,7) and TM(9,9)

0	0	0	1	1	0x480										
1	1	*Counter*													
5															
0	0	0	0	0	0	0	1	0x09							
0x07								0	0	0	0	0	0	0	0
CRC															

When the DPU receives this TC, a TM report (9,9) is generated. According to the AD–2, the application data field of this report contains the internal time at the next sync sygnal.

Suppose that the time received at the last sync signal is `0xttttttttt.rrrr`. Then the DPU assumes that at the next sync signal the new time will be `0xuuuuuuuu.rrrr` or `0xttttttttt.rrrr` plus 1 second (it is expected, but not necessary, that the fractional part `rrrr` is always 0).

0	0	0	0	1	0x480										
1	1	*Counter*													
17															
0	0	0	0	0	0	0	0	0x09							
0x09								0	0	0	0	0	0	0	0
0xtttt															
tttt															
qqqq															
0xuuuu															
uuuu															
0000															
CRC															

It has been assumed that the DPU did receive a sync in the previous second. If one or more sync are lost the DPU will still wright 1 second plus the last time received which in this case is not correct (but the time stamp of the packet is still correct as long as a sync has been received not more than 214 seconds before).

9.7 Packet Transmission Control

9.7.1 Enable Telemetry Packets: TC(14,1)

9.7.2 Disable Telemetry Packets: TC(14,2)

These two commands have the same packet structure, with only the subtype different

0	0	0	1	1	0x480										
1	1	*Counter*													
7 + *N**4															
0	0	0	0	0	0	0	1	0x0E							
0x01 or 0x02								0	0	0	0	0	0	0	0
N															
0xTTSS															
Packet-ID															
...															
0xTTSS															
Packet-ID															
CRC															

`TT` and `SS` are the type and the subtype of the packets. If a certain combination `TT`/`SS`/Packet-ID is not valid for the DPU, or it corresponds to one of the packets that can not be disabled (see Section 7.3), the next one is processed. The meaning of Packet-ID has been detailed in Section 7.3.1. No TM packet is foreseen for these services.

The science packets can be enabled/disabled with this command, however this should not be done, otherwise the bit in the DPU HK **DP-STATUS** that signals which array is enabled (blue or red) is no longer meaningful. Use always the "Set HK list" command (Section 6.3.2).

9.7.3 Report Enabled Telemetry Packets: TC(14,3) and TM(14,4)

0	0	0	1	1	0x480										
1	1	*Counter*													
5															
0	0	0	0	0	0	0	1	0x0E							
0x03								0	0	0	0	0	0	0	0
CRC															

Once the DPU receives this command a TM report (14,4) is prepared with the list of all enabled TM packets

0	0	0	0	1	0x480										
1	1	*Counter*													
13 + (*N**4)															
0	0	0	0	0	0	0	0	0x0E							
0x04								0	0	0	0	0	0	0	0
0xtttt															
tttt															
ssss															
N (≥ 4)															
0x0101															
0															
0x0102															
0															
...															
0x0319															
4															
...															
0x0E04															
0															
...															
CRC															

Note that *N* can not be smaller than 3 because 2 packets are always enabled (see Section 7.3), and the report (14,4) itself must be enabled, otherwise it is not reported. At start-up N is 46 and becomes 48 when in spectroscopy or photometry observing mode with both blue and read arrays enabled (see the "Set HK list" command in Section 6.3.2).

9.8 Test Service: TC(17,1) and TM(17,2)

9.8.1 Perform Connection Test

0	0	0	1	1	0x480										
1	1	*Counter*													
5															
0	0	0	0	0	0	0	1	0x11							
0x01								0	0	0	0	0	0	0	0
CRC															

To this TC the DPU reacts sending the TM report (17,2)

0	0	0	0	1	0x480										
1	1	*Counter*													
11															
0	0	0	0	0	0	0	0	0x11							
0x02								0	0	0	0	0	0	0	0
0xtttt															
tttt															
ssss															
CRC															

9.9 On-board Control Procedures

When the DPU receives a TC (18,x), with the exception of (18,8) and (18,10), first checks if the procedure ID, the first data word of the command, is 1$\leq$ ID $\leq$50 (but note that ID=0 is valid for Stop OBCP command, see Section 9.9.4): if not, the execution is stopped and the following TM (1,8) report is generated

Error message	Failure Code	Error Code	Parameter
`Invalid PROCID`	5	`0x1201`	ID

If the check is OK, the execution depends on the service required and on the procedure status (one of `STOPPED`, `ACTIVE`, `SUSPENDED`, `DELETED`).

9.9.1 Load Procedure: TC(18,1)

a) 1 single TC

One TC contains up to 38 PM words, or 228 bytes. If the procedure is shorter or equal to 38 words, one packet is enough. This is its structure

0	0	0	1	1	0x480										
1	1	*Counter*													
9 + *Length*															
0	0	0	0	0	0	0	1	0x12							
0x01								0	0	0	0	0	0	0	0
50															
0								*Length* (in bytes)							
0xpppp															
pppp															
pppp															
...															
0xpppp															
pppp															
pppp															
CRC															

b) n TC

To load a procedure longer than 38 words n+1 TC are required. The first n have the following structure

0	0	0	1	1	0x480										
1	1	*Counter*													
9 + *Length*															
0	0	0	0	0	0	0	1	0x12							
0x01								0	0	0	0	0	0	0	0
50															
1≤i≤n								*Length* (in bytes)							
0xpppp															
pppp															
pppp															
...															
0xpppp															
pppp															
pppp															
CRC															

The last TC signals to the DPU that the loading is completed

0	0	0	1	1	0x480										
1	1	*Counter*													
9															
0	0	0	0	0	0	0	1	0x12							
0x01								0	0	0	0	0	0	0	0
50															
0xFF								0							
CRC															

In all cases *Length* is in bytes, or 6 times the number of PM words. On completion, a TM report (1,7) is issued.

The DPU checks if a procedure with ID 50 exists and is on execution. If so the error 0x1206 is reported, otherwise it is checked that *Length* is consistent with the TC packet length. If also this check is passed the content of the TC packet is copied in the appropriate memory zone. The procedure status is set to STOPPED after all the segments have been loaded (or after the first if one TC is enough).

Error message	Failure Code	Error Code	Parameter
`Illegal LOAD`	16	`0x1206`	*ID*
`OBCP INV Size`	5	`0x120E`	*Length*

9.9.2 Deleting a procedure: TC(18,2)

0	0	0	1	1	`0x480`										
1	1	*Counter*													
7															
0	0	0	0	0	0	0	1	`0x12`							
`0x02`								0	0	0	0	0	0	0	0
ID															
CRC															

For this command the only possible error is an invalid ID, described at the beginning of this section.

9.9.3 Starting a procedure: TC(18,3)

0	0	0	1	1	`0x480`										
1	1	*Counter*													
9 + *N**6															
0	0	0	0	0	0	0	1	`0x12`							
`0x03`								0	0	0	0	0	0	0	0
ID															
N															
Parameter ID															
`0xpppp`															
`pppp`															
...															
Parameter ID															
`0xpppp`															
`pppp`															
CRC															

If the status of the procedure is `DELETED`, the execution is stopped and the error `Start DEL OBCP` is generated. If the status is `STOPPED`, the DPU checks whether another procedure is already running. If so the error `Running OBCP` is reported unless the procedure to start is one of the two "Enter SAFE mode": in this case the running procedure is stopped and the new one is started.

For procedures resident in memory, the DPU knows the number of parameters NoP. For a new procedure, NoP is communicated by using service (18,7) (see Section 6.4.7). The DPU checks that $0 \leq N \leq$NoP, and if this condition is not met the error `Too Much PARAM` is generated. If $N <$NoP, for the missing parameters the DPU uses the values sent with the last (18,3) or (18,7) TC packet. At start-up all the parameters are set to 0. If the j-th parameter is sent, this new value overwrights the previous one and becomes the new default value. If a procedure is called a second time with all the parameters unchanged, it is possible to set *N* equal to zero. The ID of the parameters does not need to follow a special order but the correspondance (parameter ID)/(parameter value) must be correct. Parameters are 32 bits.

If for a parameter the ID is greater than NoP, the error `Illegal PAR-ID` is generated: note that if this is not the first parameter of the TC packet, the previous parameters have been already changed. For instance, suppose that a procedure requires three parameters. The procedure is started the first time with this TC (only

the data field is shown)

0	0	0	0	0	0	0	1	0x12							
0x03								0	0	0	0	0	0	0	0
ID															
3															
1															
0x0000															
0005															
2															
0x0000															
0006															
3															
0x0000															
0007															
CRC															

Then a second start is sent with this TC

0	0	0	0	0	0	0	1	0x12							
0x03								0	0	0	0	0	0	0	0
ID															
3															
1															
0x0000															
0064															
22															
0x0000															
0065															
3															
0x0000															
0066															
CRC															

When the DPU processes the ID of the second parameter finds that 22 is not allowed, so the execution is stopped. Now, if we call again this procedure without parameters, the DPU will use the following set of values for the three parameters: 100 (i.e. `0x64`), 6 and 7.

If the parameters are correct, the task which handles procedure starts its execution. It checks the procedure ID and if it corresponds to “Write in EEPROM” then the third parameter must be `0x19660502`, otherwise the error `Wrong EE PAR` is reported and the procedure is not started (see Section 4.1). Eventually, the procedure is started, its status is set to `ACTIVE` and a TM report (1,3) is issued. Since this task has a lower priority all the other DPU activities are not affected by its execution.

On exit, the status is set to `STOPPED`. If the execution has been succesfully completed, a TM report (1,7) is issued; if not a TM report (1,8) is generated.

All the possible errors for this command are here summarized (see also Section 6.4.3)

Error message	Failure Code	Error Code	Parameter
`Start DEL OBCP`	16	`0x1202`	ID
`Running OBCP`	16	`0x1204`	ID of active procedure
`Too Much PARAM`	5	`0x1205`	*N*
`Illegal PAR-ID`	5	`0x1207`	the wrong parameter ID
`WRONG SEQ ID`	5	`0x1208`	the wrong sequence ID
`Wrong EE PAR`	5	`0x1209`	`pppppppp`
`Not compl OBCP`	16	`0x120A`	an internal OBS number
`SEQ NOT Compl`	16	`0x120B`	`yyyyyyyy`
`Invalid DATUM`	5	`0x120C`	an internal OBS number

`pppppppp` is the third parameter passed to the "Write in EEPROM" proc; `yyyyyyyy` is the content of the DEC sequence status HK (see RD–4).

9.9.4 Stopping a procedure: TC(18,4)

0	0	0	1	1	`0x480`										
1	1	*Counter*													
7															
0	0	0	0	0	0	0	1	`0x12`							
`0x04`								0	0	0	0	0	0	0	0
ID															
CRC															

For this command the only possible error is the invalid ID, described at the beginning of this section. If ID=0 then the DPU stops any running OBCP, if any.

9.9.5 Suspend a procedure: TC(18,5)

0	0	0	1	1	`0x480`										
1	1	*Counter*													
9															
0	0	0	0	0	0	0	1	`0x12`							
`0x05`								0	0	0	0	0	0	0	0
ID															
Step ID															
CRC															

The only Step ID valid is 0, otherwise the DPU will always report a TM (1,8)

Error message	Failure Code	Error Code	Parameter
`SUSP TIMEOUT`	17	`0x1203`	Step ID

9.9.6 Resume a procedure: TC(18,6)

0	0	0	1	1	0x480										
1	1	*Counter*													
7															
0	0	0	0	0	0	0	1	0x12							
0x06								0	0	0	0	0	0	0	0
ID															
CRC															

For this command the only possible error is the invalid ID, described at the beginning of this section.

9.9.7 Communicate parameters to a procedure: TC(18,7)

Besides the obvious purpose to communicate the value of the parameters to an existing procedure, this service can also be used to inform the DPU about the NoP of a newly uploaded procedure. NoP must be $\leq$25, the largest allowed NoP, otherwise the error `Too Much PARAM` is generated. The TC has the following structure (the two columns refer to the case ID is different from 50 or is just 50)

0	0	0	1	1	0x480										
1	1	*Counter*													
9 + *N**6															
0	0	0	0	0	0	0	1	0x12							
0x07								0	0	0	0	0	0	0	0
ID (any but 50)								50							
N								*N*							
Parameter ID								0							
0xpppp								0							
pppp								NoP							
...								...							
Parameter ID								Parameter ID							
0xpppp								0xpppp							
pppp								pppp							
CRC															

In the general case (left column) *N* must always be $\leq$NoP and the parameter ID can not be zero. When *N* is 50 (right column) only the first parameter ID can be zero; to communicate only the NoP just put *N*=1, first parameter ID = 0 and parameter value = NoP, otherwise put *N*=NoP+1 and write all the parameters values.

The checks performed by the DPU are the same as for service (18,3), but with the following differences: the command is executed also if another procedure is running; the procedure is not started; if the procedure status is `DELETED` the command is ignored but no TM report (1,8) is issued. As a consequence, the possible errors for this command are a subset of the errors of service (18,3)

Error message	Failure Code	Error Code	Parameter
`Too Much PARAM`	5	`0x1205`	*N* or NoP
`Illegal PAR-ID`	5	`0x1207`	the wrong parameter ID

9.9.8 Report the list of existing procedures: TC(18,8) and TM(18,9)

0	0	0	1	1	0x480							
1	1	*Counter*										
5												
0	0	0	0	0	0	0	1	0x12				
0x08								0	0	0	0	0	0	0	0
CRC															

When the DPU receives this command, generates a report (18,9) with the identifiers of all existing procedures, i.e. all procedures whose status is not DELETED

0	0	0	0	1	0x480							
1	1	*Counter*										
13 + (*N**2)												
0	0	0	0	0	0	0	0	0x12				
0x09								0	0	0	0	0	0	0	0
0xtttt															
tttt															
ssss															
N (34 at start-up)															
ID of the first existing procedure															
...															
ID of the last existing procedure															
CRC															

9.9.9 Report the list of active procedures: TC(18,10) and TM(18,11)

0	0	0	1	1	0x480							
1	1	*Counter*										
5												
0	0	0	0	0	0	0	1	0x12				
0x0A								0	0	0	0	0	0	0	0
CRC															

When the DPU receives this command, generates a report (18,11) with the identifiers of the active procedure.

0	0	0	0	1	0x480							
1	1	*Counter*										
15												
0	0	0	0	0	0	0	0	0x12				
0x0B								0	0	0	0	0	0	0	0
0xtttt															
tttt															
ssss															
1															
ID of the only active procedure															
CRC															

If no procedure is running the report (18,11) has the following structure

0	0	0	0	1	0x480										
1	1	*Counter*													
13															
0	0	0	0	0	0	0	0	0x12							
0x0B								0	0	0	0	0	0	0	0
0xtttt															
tttt															
ssss															
0															
CRC															

9.9.10 Report OBCP status: TC(18,12) and TM(18,13)

0	0	0	1	1	0x480										
1	1	*Counter*													
7															
0	0	0	0	0	0	0	1	0x12							
0x0C								0	0	0	0	0	0	0	0
ID															
CRC															

On reception of this TC, a TM report (18,13) is generated (if the procedure ID is valid)

0	0	0	0	1	0x480										
1	1	*Counter*													
17 + (NoP*6)															
0	0	0	0	0	0	0	0	0x12							
0x0D								0	0	0	0	0	0	0	0
0xtttt															
tttt															
ssss															
ID															
Step ID								s							
NoP															
Parameter ID															
0xpppp															
pppp															
...															
Parameter ID															
0xpppp															
pppp															
CRC															

Step ID is the 8 bits counter (always 0); the other 8 bits defines the procedure status

Status	s
STOPPED	0
ACTIVE	1
SUSPENDED	2
DELETED	3

9.10 Science Data Transfer

The content of the TM packets of this service can be found in AD–6 for science data and in RD–4 for the diagnostic HK data. The DPU simply selects the subtype and the SID according to the following table

Data	APID Nom.	APID Red.	TM packet	SID
Spec blue array	0x48A	0x48B	(21,1)	1
Spec red array	488	489	(21,1)	2
Phot blue array	48A	48B	(21,2)	1
Phot red array	488	489	(21,2)	2
DEC diagnostic HK	486	487	(21,3)	0

Remember that the (21,1) and (21,2) reports are disabled at start-up, and are enabled with the command "Set HK list" (Section6.3.2). Also, a SID in the form 0x4i in the last packet means that its length, computed on the base of the header of the first packet, is greater than the maximum allowed length, so that it has been fixed to the maximum value (1017).

9.10.1 Nominal Science Data Report: TM(21,1)

This service is used to send spectroscopic data. SPU sends science data in the format specified in AD–6. Each science frame is splitted in one or more packets and the content is shown below.

9.10.1–1 Example: Science data in one packet

	0	0	0	0	1	0x48A for blue or 0x488 for red SPU										
Seg. flag →	1	1	*Counter*													
	Length															
	0	0	0	0	0	0	0	0	0x15							
	0x01								0	0	0	0	0	0	0	0
	0xtttt															
	tttt															
	ssss															
	1 for blue or 2 for red SPU															
	1 (Packet Counter)															
	1 (Total number of Packets)															
	7 words of 32bits for the header															
	0xssss															
	ssss															
	...															
	0xssss															
	ssss															
	CRC															

where 0xsssssssss are the science data words. *Length* is 49 bytes plus (CDHS+SCIS)*4 (see AD–6). **In burst mode** *Length* **is always 1017, according to AD–2**.

9.10.1–2 Example: Science data in two packets

First packet

Seg. flag →																
0	0	0	0	1	0x48A for blue or 0x488 for red SPU											
0	1	*Counter*														
1017																
0	0	0	0	0	0	0	0	0x15								
0x01								0	0	0	0	0	0	0	0	
0xtttt																
tttt																
ssss																
1 for blue or 2 for red SPU																
1 (Packet Counter)																
2 (Total number of Packets)																
7 words of 32bits for the header																
0xssss																
ssss																
...																
0xssss																
ssss																
CRC																

Second packet

Seg. flag →																
0	0	0	0	1	0x48A for blue or 0x488 for red SPU											
1	0	*Counter*														
Length																
0	0	0	0	0	0	0	0	0x15								
0x01								0	0	0	0	0	0	0	0	
0xtttt																
tttt																
ssss																
1 for blue or 2 for red SPU																
2 (Packet Counter)																
2 (Total number of Packets)																
0xssss																
ssss																
...																
0xssss																
ssss																
CRC																

Length is (CDHS+SCIS)*4 minus 955 bytes (see AD–6). **In burst mode** *Length* **is always 1017, according to AD–2**.

9.10.1–3 Example: Science data in three or more packets

First packet

Seg. flag →						
0	0	0	0	1	0x48A for blue or 0x488 for red SPU	
0	1	*Counter*				
1017						
0	0	0	0	0	0	0	0	0x15	
0x01								0	0	0	0	0	0	0	0
0xtttt															
tttt															
ssss															
1 for blue or 2 for red SPU															
1 (Packet Counter)															
n (Total number of Packets)															
7 words of 32bits for the header															
0xssss															
ssss															
...															
0xssss															
ssss															
CRC															

Second and intermediate packets

Seg. flag →						
0	0	0	0	1	0x48A for blue or 0x488 for red SPU	
0	0	*Counter*				
1017						
0	0	0	0	0	0	0	0	0x15	
0x01								0	0	0	0	0	0	0	0
0xtttt															
tttt															
ssss															
1 for blue or 2 for red SPU															
2–(n-1) (Packet Counter)															
n (Total number of Packets)															
0xssss															
ssss															
...															
0xssss															
ssss															
CRC															

Last packet

Seg. flag →						
0	0	0	0	1	0x48A for blue or 0x488 for red SPU	
1	0	*Counter*				
Length						
0	0	0	0	0	0	0	0	0x15	
0x01								0	0	0	0	0	0	0	0
0xtttt															
tttt															
ssss															
1 for blue or 2 for red SPU															
n (Packet Counter)															
n (Total number of Packets)															
0xssss															
ssss															
...															
0xssss															
ssss															
CRC															

Length is 45 bytes plus (CDHS+SCIS)*4 minus 1000*(n-1) (see AD–6). **In burst mode** *Length* **is always 1017, according to AD–2**.

9.10.2 Science Type B Data Report: TM(21,2)

This service is used to send photometric data. The packet format is the same as in the spectroscopic data, the only difference being the subtype of the packets.

9.10.3 Diagnostic Science Report: TM(21,3)

This service is used to transport the diagnostic HK values from DEC.

0	0	0	0	1	0x486										
1	1	*Counter*													
Length															
0	0	0	0	0	0	0	0	0x15							
0x03								0	0	0	0	0	0	0	0
0xtttt															
tttt															
ssss															
0															
0xhhhh															
hhhh															
...															
0xhhhh															
hhhh															
CRC															

where 0xhhhhhhhh are the 32bits words received from DEC. *Length* is 13 bytes plus 4 times the length reported in the diagnostic HK packet (see AD–5). **In burst mode** *Length* **is always 1017, according to AD–2**.

A Test on DPU memory

There are two possible failures of memory, we could name them "HW" and "SW" failures: the latter cause bitflips in memory cells and can, at least partially, be corrected by additional hardware (EDAC) if present (this is not the case of DPU). The former can not be corrected because are caused by hardware breakage.

As discussed in Section 7.1 the DPU has different memories but here we are interested only in "internal" memory, RAM in the table on page 55. The PRAM has a constant content (the only exception being the interrupt table which is however changed only during initialization of OBS), the DRAM does not, so in this case we can only check if there is a permanent (HW) failure.

To check PM the command "Check PM" is used as explained in Section 6.3.13. The user gives the start address, the end address and the expected checksum. The DPU computes the checksum and compares the result with the value written in the telecommand. If the two values are equal a TM(1,7) is generated, otherwise a TM(1,8) and an event (5,2) are reported. The rationale of this telecommand is that it can be sent periodically by the satellite to verify that no event is reported while the ordinary memory check command does not allow an automatic verification of the computed result. In case the event is reported the autonomy function has not been decided yet. A possibility is to set the instrument in SAFE mode, make a dump of DPU PRAM and then switch off the instrument

The case of data memory is more complex. The content of DM changes in time and it is not possible to foresee the checksum as in the previous case. So for DM only permanent failures can be detected. To this aim the DPU makes use of the timer interrupt (IRQ3) driven by the external FPGA. During the initialization of the OBSW the timer period is set to 1 second so that every second the DPU receives this interrupt which has the highest priority. The associated interrupt service routine checks the content of 6 memory cells by reading and saving the original content. This value is negated (bit's 1 are set to 0 and viceversa) and the result is written and then read in the memory cell. If the written and the read values are equal the check is passed, otherwise the address is written in a buffer. In any case the original content is written back. The number 6 has been chosen because in this way the 524288 memory cells can be checked in ∼87382 seconds, or about 1 day. The routine is executed, if there are no failures, in $\sim 10\mu$s. Note that this value can be changed with a memory load command, writing at address `0x5BB` in DM the number of words per second you want to check. During emergency this number can be as high as 50000 so that the whole DM can be tested in 10 seconds (but then some 1553 interrupts may be lost).

Every two seconds the HK monitoring task wakes up and, beside all the checks, the last address verified is checked against the previous value. If different the IRQ3 task status is set to running. Then the content of the failed addresses buffer is read and if some new values have been written the DPU generates the event (5,4) `DM FAILURE` (see Section 8.2.3–1) containing the addresses of the failed cells. The buffer can contain up to 32 addresses so it could be overflown if the rate of cells tested is larger than 16 per seconds (but such a possibility would imply a "catastrophic" condition for the DPU).

B Other useful informations

B.1 The architecture file

The architecture file is used by the 21020 compiler to know in which memory segment the code and the data have to be put.

```
.system PACSDPU;
.processor = ADSP21020;
!============================================================ Interrupt table
.segment /pm /ram   /begin=0x000000 /end=0x0000FF    seg_rth;
!==================PM
.segment /pm /ram         /begin=0x004000 /end=0x005550    seg_init;
.segment /pm /ram         /begin=0x005551 /end=0x07BBFF    seg_pmco;
.segment /pm /ram         /begin=0x07BC00 /end=0x07BF9F    seg_pmda;
!==================DM
.segment /dm /ram   /begin=0x00000000 /end=0x0003FFFF                seg_dmda;
.segment /dm /ram   /begin=0x00040000 /end=0x0004FFFF /cstack   seg_stak;
.segment /dm /ram   /begin=0x00050000 /end=0x0007FFFF /cheap    heap1;
!==================Mapped Memory
.segment /dm /ram   /begin=0x40000000 /end=0x40001FFF     1355_IF;
.segment /dm /ram   /begin=0x80000000 /end=0x8003FFFF     EEPROM;
.segment /dm /port  /begin=0x81000000 /end=0x81FFFFFF     Timer;
.segment /dm /port  /begin=0x82000000 /end=0x82FFFFFF     watchdog;
.segment /dm /port  /begin=0x83000000 /end=0x83FFFFFF     Int_mng;
.segment /dm /ram   /begin=0x84000000 /end=0x84FFFFFF     SMCS_reg;
.segment /dm /ram   /begin=0x88000000 /end=0x8FFFFFFF     Bus_IF;
!============================================================================
!Bank Description
!the PM bank1 is not mounted
.bank /pm0 /wtstates=0 /wtmode=internal /begin=0x000000;
!.bank /pm1 /wtstates=0 /wtmode=internal /begin=0x800000;
!
! DM bank 0 is used for data storing
! DM bank 1 is reserved for Mezzanine IF and it is not used
! DM bank 2 is reserved for IEEE 1355
! DM bank 3 is reserved for the following register and Device
!                              EEPROM, Interval Timer, Watchdog, Interrupt Manager
!                              SMCS332 register, 32 bit bus interface
.bank /dm0 /wtstates=1 /wtmode=internal /begin=0x00000000;
.bank /dm1 /wtstates=1 /wtmode=both     /begin=0x20000000;
.bank /dm2 /wtstates=1 /wtmode=internal /begin=0x40000000;
.bank /dm3 /wtstates=1 /wtmode=both     /begin=0x80000000;
!============================================================================
.endsys;
!***** end of file *****
```

B.2 Building a new image and uploading

Here is the `Makefile` to compile OBS. It is used by VIRTUOSO (Version 4.1 R2.04) and uses the ADSP-21000 Family Development Tools 3.3 by Analog Devices. The output consists of two files: `pacs.exe` is the OBS image; `pacs.fil` is image dump and contains the ASCII representation of the image. It can also be used to find how many PM words are used.

```
KERNEL=SP
SOURCEPATH = C:\Virtuoso\Adi21020\Rev33\Sigma\MyProj\dpu_pacs

all: PACS.exe
include base

ACHFILE = pacs.ach
ASFLAGS = -DADI21020 -D__ADSP21020__ -DPACS_CODE -DOBSCODE=PACS_CODE

INCLUDE_DIR = C:\Virtuoso\ADI21020\inc

INC1355 = LT_1355.h \
          spwdef.H

INC1553 = 1553_def.h \
            ivar1553.h \
            init1553.h \
            conf1553.h \
            MilDef.h    \
            MilErr.h    \
            MilConf.h   \
            MilIrq.h    \
            MilInit.h   \
            Milmem.h    \
            MilRt.h

INC_EEPROM = Eprm.h    \
             pmload.h

INC_HK = LT_HKdef.h \
         HK_def.h

SEQUENCES = DmcCmd.h \
                SEQ_BUFF.h

CCFLAGS = -Wall -c -no-CO -I$(INCLUDE_DIR) -I$(SOURCEPATH) -I. -a pacs.ach -DADI21020 -DVIRTUOSO
@echo $(CCFLAGS) > cflags

Eprm.o: Eprm.c cflags NODE1.h $(INC_EEPROM) MM_MISC.h
        $(CC) @cflags -o $@ $<

L4_FUNC.o: L4_FUNC.c cflags $(INC1355) LT_FUNC.h LT_OBCP.h LT_TMdef.h MM_lib.h MM_21020.h $(INC_HK) DmcCmd.h LT_MEM.h Inttab.h
        $(CC) @cflags -o $@ $<
```

```
L4_LIB.o: L4_LIB.c cflags LT_TMdef.h $(INC_HK)
	$(CC) @cflags -o $@ $<

L4_MEM.o: L4_MEM.c cflags LT_TMdef.h LT_MEM.h MM_LIB.h MM_21020.h $(INC1355)
	$(CC) @cflags -o $@ $<

L4_OBCP.o: L4_OBCP.c cflags LT_TMdef.h LT_OBCP.h $(INC_HK) $(INC1355) MM_21020.h
	$(CC) @cflags -o $@ $<

L5_D_AUT.o: L5_D_AUT.c cflags LT_TMdef.h $(INC_HK) LT_FUNC.h LT_1355.h MM_21020.h
	$(CC) @cflags -o $@ $<

L9_BOL_P.o: L9_BOL_P.c cflags $(INC1355) LT_OBCP.h DmcCmd.h LT_FUNC.h
	 $(CC) @cflags -O2 -o $@ $<

L9_EEPRM.o: L9_EEPRM.c cflags $(INC_EEPROM) LT_OBCP.h  MM_21020.h $(INC_HK)
	$(CC) @cflags -o $@ $<

L9_GRATP.o: L9_GRATP.c cflags $(INC1355) $(INC_HK) LT_OBCP.h DmcCmd.h LT_TMdef.h
	 $(CC) @cflags -O2 -o $@ $<

L9_MISC.o: L9_MISC.c cflags $(INC1355) LT_OBCP.h DmcCmd.h SPUCmd.h LT_TMdef.h
	 $(CC) @cflags -o $@ $<

L9_P1355.o: L9_P1355.c cflags $(INC1355) LT_OBCP.h $(INC_HK) LT_TMdef.h NODE1.h
	$(CC) @cflags -o $@ $<

L9_PHOTC.o: L9_PHOTC.c cflags $(INC1355) $(INC_HK) LT_OBCP.h DmcCmd.h LT_TMdef.h
	$(CC) @cflags -O2 -o $@ $<

L9_PHOTP.o: L9_PHOTP.c cflags $(INC1355) $(INC_HK) LT_OBCP.h DmcCmd.h LT_TMdef.h
	$(CC) @cflags -O2 -o $@ $<

L9_SPCMD.o: L9_SPCMD.c cflags SPUCmd.h $(INC1355) $(INC_HK) LT_OBCP.h LT_TMdef.h
	$(CC) @cflags -o $@ $<

L9_SPECC.o: L9_SPECC.c cflags $(INC1355) $(INC_HK) LT_OBCP.h DmcCmd.h LT_TMdef.h
	$(CC) @cflags -O2 -o $@ $<

L9_SWITC.o: L9_SWITC.c cflags $(INC1355) $(INC_HK) LT_OBCP.h DmcCmd.h SPUCmd.h LT_FUNC.h LT_TMdef.h
	$(CC) @cflags -o $@ $<
```

```
L9_newOB.o: L9_newOB.c cflags
	$(CC) @cflags -o $@ $<

LT_1355.o: LT_1355.c cflags $(INC1355) LT_TMdef.h $(INC_HK) DmcCmd.h MM_LIB.h MM_21020.h NODE1.h
	$(CC) @cflags -o $@ $<

LT_FUNC.o: LT_FUNC.c cflags LT_TMdef.h $(INC_HK) LT_FUNC.h
	$(CC) @cflags -o $@ $<

LT_INIT.o: LT_INIT.c cflags $(INC_HK) $(INC1355) NODE1.h LT_MEM.h
	$(CC) @cflags -o $@ $<

LT_upTMb.o: LT_upTMb.c cflags LT_TMdef.h MM_21020.h MM_MISC.h $(INC1553) $(INC_HK) NODE1.h
	$(CC) @cflags -o $@ $<

MM_MISC.o: MM_MISC.c cflags MM_MISC.h
	$(CC) @cflags -O2 -o $@ $<

MM_crc.o: MM_crc.c cflags MM_crc.h
	$(CC) @cflags -O2 -o $@ $<

MM_lib.o: MM_lib.c cflags MM_crc.h MM_21020.h MM_lib.h
	$(CC) @cflags -o $@ $<

MilConf.o: MilConf.c cflags $(INC1553)
	$(CC) @cflags -o $@ $<

MilInit.o: MilInit.c cflags $(INC1553)
	$(CC) @cflags -o $@ $<

MilIrq.o: MilIrq.c cflags $(INC1553)
	$(CC) @cflags -o $@ $<

MilRt.o: MilRt.c cflags $(INC1553)
	$(CC) @cflags -o $@ $<

Milmem.o: Milmem.c cflags $(INC1553)
	$(CC) @cflags -o $@ $<

NODE1.o: NODE1.c cflags NODE1.h
	$(CC) @cflags -o $@ $<

T1_INIT.o: T1_INIT.c cflags NODE1.h $(INC_HK) LT_TMdef.h $(INC1355) $(INC1553) DmcCmd.h LT_FUNC.h T1_INIT.h LT_MEM.h $(SEQUENCES)
```

```
	$(CC) @cflags -o $@ $<

T2TMTCIF.o: T2TMTCIF.c cflags $(INC1553) LT_TMdef.h NODE1.h init1553.h $(INC_HK)
	$(CC) @cflags -o $@ $<

T3IRQ1SV.o: T3IRQ1SV.c cflags $(INC1355) LT_TMdef.h $(INC_HK) MM_21020.h NODE1.h
	$(CC) @cflags -o $@ $<

T4CNTRLR.o: T4CNTRLR.c cflags LT_TMdef.h LT_OBCP.h MM_21020.h $(INC1553) NODE1.h
	$(CC) @cflags -o $@ $<

T5_HKMON.o: T5_HKMON.c cflags LT_MEM.h LT_TMdef.h $(INC_HK) $(INC1355) LT_OBCP.h LT_FUNC.h $(INC1553) MM_21020.h NODE1.h
	$(CC) @cflags -o $@ $<

T6_MECRX.o: T6_MECRX.c cflags $(INC1355) LT_TMdef.h MM_21020.h NODE1.h
	$(CC) @cflags -o $@ $<

T7_SPSRX.o: T7_SPSRX.c cflags $(INC1355) LT_TMdef.h $(INC_HK) MM_21020.h NODE1.h
	$(CC) @cflags -o $@ $<

T8_SPLRX.o: T8_SPLRX.c cflags $(INC1355) LT_TMdef.h $(INC_HK) MM_21020.h NODE1.h
	$(CC) @cflags -o $@ $<

T9_OBCP.o: T9_OBCP.c cflags LT_TMdef.h $(INC_HK) LT_OBCP.h LT_FUNC.h DmcCmd.h MM_21020.h NODE1.h
	$(CC) @cflags -o $@ $<

init1553.o: init1553.c cflags $(INC1553) LT_TMdef.h
	$(CC) @cflags -o $@ $<

isr1553.o: isr1553.c cflags $(INC1553) LT_TMdef.h MM_21020.h
	$(CC) @cflags -o $@ $<

util1553.o: util1553.c cflags $(INC1553) MM_21020.h LT_TMdef.h
	$(CC) @cflags -o $@ $<

OBCP.a: L9_newOB.c L9_newOB.o
	lib21k -c $@ L9_newOB.o

PACS.exe: $(DEPENDENCIES) $(ACHFILE) pacs1.lnk NODE1.o OBCP.a
	$(LN) -i pacs1.lnk -o pacs.out -a $(ACHFILE) -m
	$(RM) T1_INIT.o
	mem21k -o PACS.exe pacs.out -a pacs.ach
	cdump PACS.exe > pacs.fil
```

The used compiler is the gcc21020 provided by Analog Devices version 3.3, a modified version of the GNU compiler. The compiler flags mean

- `-Wall` print all the warning messages
- `-c` do not link the object files
- `-no-CO` mem21k is to be used
- `-Ixxx` search in directory xxx for header files
- `-a pacs.ach` use pacs.ach as architecture file
- `-DADI21020` define ADI21020=1 (used for assembly files)
- `-DVIRTUOSO` define VIRTUOSO=1 (required by the 1553 drivers)

The linker flags mean

- `-i pacs1.lnk` link the files listed in pacs1.lnk
- `-o pacs.out` write the output in pacs.out
- `-a $(ACHFILE)` use $(ACHFILE), or pacs.ach (see the top of makefile), as architecture file
- `-m` mem21k is to be used

Output files are saved in the current directory, the one from which "make" command is run. Object files are saved with the exception of `T1_INIT.O` which is always removed after linked. The source file contains date and time of the computer at the time it is compiled. Deleting `T1_INIT.O` forces the compiler to always generate the object file, in this way date and time of compilation are always contained in the executable. A map file is always generated, as well as the ASCII representation of the executable (`pacs.fil`).

To generate the telecommands for uploading a new image, a specific tool written by CGS is required. Its description is out of the scope of this document, here only the steps to follow are given.

It is assumed that the program, named `tcgen` resides in the directory `C:\local_path\tcgen\Debug`. This program writes the output in the subdirectory `Debug\result` that must be created before the program is run. The image `Test1.exe` is copied in the directory `C:\local_path\tcgen`. In the `Debug` directory the batch `tcgencom.bat` is used, containing the following line command: `tcgen -i ..\segfile.txt -p ..\pagefile.txt -f ..\Test1.exe -a 0x480 -o result\DmPage -m 0 > out` `segfile.txt` is an ASCII file containing these three text lines: `seg_rth seg_init` and `seg_pmco`. `pagfile.txt` is also an ASCII file containing the page to avoids. Currently it contains 9 text lines with the numbers from 2 to 10. On program completion, all the generated TC are available in the directory `Debug\result`.

B.3 List of files

The list of files is here reported; for every new release the complete source code is saved in the PACS official repository in Leuven at address http://cvs.ster.kuleuven.be/ under the directory `obsw/dpu/` (password protected site, if a copy is requested contact the PACS Project Office at `ppo@mpe.mpg.de`)

1553_def.h	L9_MISC.c	LT_OBCP.h	MilInit.c	T1_INIT.h	irq2.s
DUMMY.c	L9_P1355.c	LT_TMdef.h	MilInit.h	T2TMTCIF.c	isr.s
DmcCmd.h	L9_PHOTC.c	LT_upTMb.c	MilIrq.c	T3IRQ1SV.c	isr1553.c
Eprm.c	L9_PHOTP.c	MM_21020.h	MilIrq.h	T4CNTRLR.c	ivar1553.h
Eprm.h	L9_SPCMD.c	MM_21020.s	MilRt.c	T5_HKMON.c	makefile
HK_def.h	L9_SPECC.c	MM_MISC.c	MilRt.h	T6_MECRX.c	pacs.ach
Inttab.h	L9_SWITC.c	MM_MISC.h	Milmem.c	T7_SPSRX.c	pmload.h
L4_FUNC.c	L9_newOB.c	MM_crc.c	Milmem.h	T8_SPLRX.c	pmload.s
L4_LIB.c	LT_1355.c	MM_crc.h	NODE1.c	T9_OBCP.c	spwdef.H
L4_MEM.c	LT_1355.h	MM_lib.c	NODE1.h	allnodes.h	util1553.c
L4_OBCP.c	LT_FUNC.c	MM_lib.h	PACS.NLI	base	
L5_D_AUT.c	LT_FUNC.h	MilConf.c	PACSApp.vpf	changelog.txt	

```
L9_BOL_P.c  LT_HKdef.h  MilConf.h   SEQ_BUFF.h   conf1553.h
L9_EEPRM.c  LT_INIT.c   MilDef.h    SPUCmd.h     init1553.c
L9_GRATP.c  LT_MEM.h    MilErr.h    T1_INIT.c    init1553.h
```

At IFSI the files are saved daily on an internal repository at address http://cvs.ifsi-roma.inaf.it/ under directory `/usr/local/cvs-rep/PACS_V2/code/`. Note that also this site is password protected.

B.4 Potential problems

- If a command is sent to DPU with service (8,4) during the execution of an OBCP which is sending a command to a subsystem, and both (8,4) command execution and OBCP command execution fail, the counter of TM (1,8) may be incremented by one instead of two;
- On reception of a TC (9,7), the corresponding TM (9,9) may be generated after a new sync is received. As a consequence TM (9,9) may be wrong by one second;
- **During execution of OBCP 21 do not send commands for the subsystems**;
- For test purposes, an AF can be started using a specific DPU command. However, the same AF could be started by the HK monitoring task if the corresponding HK values is out of range. No check is done that this condition does not occur;
- DPU does not check that a DEC sequence is modified/deleted during the execution of an OBCP wich uses it. Avoid this condition;
- The dump command can require more than one TM packet. If a new dump command for a subsystem is received by the DPU before the last dump packet has been sent, unpredictable side effects may occur.

C Content of the HK packets

The HK are written one after each other in the packet, without padding bits. In the following table all the HK are listed giving their name and size (in bit). OBSID and BBID are part of the DEC HK packet (between DMC_LABEL and DMC_TIME_1) but are written in the first fields of the TM packet.

No.	Name	Size	Spec	Phot	Non prime
Begin of Application Data Field					
1	SID	16	✓	✓	✓
2	DMC_OBSID (from DEC HK)	32	✓	✓	✓
3	DMC_BBID (from DEC HK)	32	✓	✓	✓
		Total	80	80	80
DPU HK (begin)					
4	DPU_VOL_25_P_N	12	✓	✓	✓
5	DPU_VOL_5P_N	12	✓	✓	✓
6	DPU_VOL_15P_N	12	✓	✓	✓
7	DPU_VOL_15N_N	12	✓	✓	✓
8	DPU_T_N	12	✓	✓	✓
9	DPU_SPS_LINK	1	✓	✓	✓
10	DPU_SPL_LINK	1	✓	✓	✓
11	DPU_DMC_LINK	1	✓	✓	✓
12	DPU_SPS_CMD	2	✓	✓	✓
13	DPU_SPL_CMD	2	✓	✓	✓
14	DPU_DMC_CMD	2	✓	✓	✓

15	DPU_SPS_HK	2	✓	✓	✓
16	DPU_SPL_HK	2	✓	✓	✓
17	DPU_DMC_HK	2	✓	✓	✓
18	DPU_STATUS	10	✓	✓	✓
19	DPU_WHICH_OBCP	6	✓	✓	✓
20	DPU_AF_STATUS	24	✓	✓	✓
21	DPU_MUMON_STATUS	3	✓	✓	✓
22	DPU_ANSWEREDPRAYERS_STATUS	3	✓	✓	✓
23	DPU_ISIDE_STATUS	3	✓	✓	✓
24	DPU_HUNAHPU_STATUS	3	✓	✓	✓
25	DPU_FRANCESCO_STATUS	3	✓	✓	✓
26	DPU_GINEVRA_STATUS	3	✓	✓	✓
27	DPU_MACGIG_STATUS	3	✓	✓	✓
28	DPU_IXBALAMQUE_STATUS	3	✓	✓	✓
29	DPU_THOTH_STATUS	3	✓	✓	✓
30	DPU_DMCHECK_STATUS	1	✓	✓	✓
31	DPU_DEC_LINK_PE	5	✓	✓	✓
32	DPU_DEC_LINK_DE	5	✓	✓	✓
33	DPU_SPS_LINK_PE	5	✓	✓	✓
34	DPU_SPS_LINK_DE	5	✓	✓	✓
35	DPU_SPL_LINK_PE	5	✓	✓	✓
36	DPU_SPL_LINK_DE	5	✓	✓	✓
37	DPU_WORKLOAD	10	✓	✓	✓
38	DPU_TM_RATE	8	✓	✓	✓
39	DPU_SW_VERS_ID	11	✓	✓	✓
40	DPU_TC_LOST	16	✓	✓	✓
41	DPU_HK_LOST	16	✓	✓	✓
42	DPU_EVENT_LOST	16	✓	✓	✓
43	DPU_GEN_TM_LOST	16	✓	✓	✓
44	DPU_COMMANDS_REC_DPU	16	✓	✓	✓
45	DPU_COMMANDS_REJ_DPU	16	✓	✓	✓
46	DPU_COMMANDS_DMC	16	✓	✓	✓
47	DPU_COMMANDS_SPS	16	✓	✓	✓
48	DPU_COMMANDS_SPL	16	✓	✓	✓
DPU HK (end)		DPU Total	346	346	346
		Packet Total	426	426	426
Red SPU HK (begin)					
49	SPU_OBSID	32	✓	✓	
50	SPU_PIXRB	32	✓	✓	✓
51	SPU_CIRB	16	✓	✓	✓
52	SPU_REAL	16	✓	✓	
53	SPU_SATURATION_FLAG	8	✓	✓	
54	SPU_SAMP_CORR	24	✓	✓	
55	SPU_N_RAMPS	16	✓	✓	
56	SPU_WORKLOAD	16	✓	✓	✓
57	SPU_DMC_LINK_STATUS	16	✓	✓	✓
58	SPU_INTEG_RAMPS	8	✓	✓	
59	SPU_VID	8	✓	✓	✓
60	SPU_RCX	16	✓	✓	
61	SPU_DMC_ERROR	8	✓	✓	✓

62	SPU_MEM_CNTS	16	✓	✓	✓
63	SPU_SPARE_1	16	✓	✓	
64	SPU_LLC_ERROR	16	✓	✓	
65	SPU_PAR_MONITOR	16	✓	✓	
Red SPU HK (end)		Red SPU Total	280	280	112
		Packet Total	706	706	538
Blue SPU HK (begin)					
66	SPU_OBSID	32	✓	✓	
67	SPU_PIXRB	32	✓	✓	✓
68	SPU_CIRB	16	✓	✓	✓
69	SPU_REAL	16	✓	✓	
70	SPU_SATURATION_FLAG	8	✓	✓	
71	SPU_SAMP_CORR	24	✓	✓	
72	SPU_N_RAMPS	16	✓	✓	
73	SPU_WORKLOAD	16	✓	✓	✓
74	SPU_DMC_LINK_STATUS	16	✓	✓	✓
75	SPU_INTEG_RAMPS	8	✓	✓	
76	SPU_VID	8	✓	✓	✓
77	SPU_RCX	16	✓	✓	
78	SPU_DMC_ERROR	8	✓	✓	✓
79	SPU_MEM_CNTS	16	✓	✓	✓
80	SPU_SPARE_1	16	✓	✓	
81	SPU_LLC_ERROR	16	✓	✓	
82	SPU_PAR_MONITOR	16	✓	✓	
Blue SPU HK (end)		Blue SPU Total	280	280	112
		Packet Total	986	986	650
DEC HK (Bolometers section 1st block— begin)					
83	BF1B_VH_B_1	16		✓	
84	BF1B_VL_B_1	16		✓	
85	BF1B_VRL_B_1	16		✓	
86	BF1B_VINJ_B_1	16		✓	
87	BF1B_HEATER_B_1	16		✓	
88	BF1B_VDL_B_1	16		✓	
89	BF1B_VSS_B_1	16		✓	
90	BF1B_VGL_B_1	16		✓	
91	BF1B_CKRLH_B_1	16		✓	
92	BF1B_CKRLL_B_1	16		✓	
93	BF1B_VDECXH_B_1	16		✓	
94	BF1B_VDECXL_B_1	16		✓	
95	BF1B_VSMSH_B_1	16		✓	
96	BF1B_VSMSL_B_1	16		✓	
97	BF1B_VDDPROT_CLB1	16		✓	
98	BF1B_GND_BU_B_1	16		✓	
99	BF1B_VDD_B_1	16		✓	
100	BF1B_VGG_B_1	16		✓	
101	BF1B_VSS_BU_B_1	16		✓	
102	BF1B_VDL_BU_B_1	16		✓	
103	BF1B_VGL_BU_B_1	16		✓	
104	BF1B_VDDPROT_BUB1	16		✓	
105	I_HEATER_B_1	16		✓	✓

106	I_VSS_B_1	16		✓	
107	I_VSS_BU_B_1	16		✓	
108	VH_BLIND_B_1	16		✓	
109	CKTRIL_REF_B_1	16		✓	
110	BC_PWR_ANA_P_1	16	✓	✓	
111	BC_PWR_ANA_N_1	16	✓	✓	
112	BC_PWR_DIG_1	16	✓	✓	
113	BF2B_VH_B_2	16		✓	
114	BF2B_VL_B_2	16		✓	
115	BF2B_VRL_B_2	16		✓	
116	BF2B_VINJ_B_2	16		✓	
117	BF2B_HEATER_B_2	16		✓	
118	BF2B_VDL_B_2	16		✓	
119	BF2B_VSS_B_2	16		✓	
120	BF2B_VGL_B_2	16		✓	
121	BF2B_CKRLH_B_2	16		✓	
122	BF2B_CKRLL_B_2	16		✓	
123	BF2B_VDECXH_B_2	16		✓	
124	BF2B_VDECXL_B_2	16		✓	
125	BF2B_VSMSH_B_2	16		✓	
126	BF2B_VSMSL_B_2	16		✓	
127	BF2B_VDDPROT_CLB2	16		✓	
128	BF2B_GND_BU_B_2	16		✓	
129	BF2B_VDD_B_2	16		✓	
130	BF2B_VGG_B_2	16		✓	
131	BF2B_VSS_BU_B_2	16		✓	
132	BF2B_VDL_BU_B_2	16		✓	
133	BF2B_VGL_BU_B_2	16		✓	
134	BF2B_VDDPROT_BUB2	16		✓	
135	I_HEATER_B_2	16		✓	✓
136	I_VSS_B_2	16		✓	
137	I_VSS_BU_B_2	16		✓	
138	VH_BLIND_B_2	16		✓	
139	CKTRIL_REF_B_2	16		✓	
140	BC_PWR_ANA_P_2	16	✓	✓	
141	BC_PWR_ANA_N_2	16	✓	✓	
142	BC_PWR_DIG_2	16	✓	✓	
143	BF3B_VH_B_3	16		✓	
144	BF3B_VL_B_3	16		✓	
145	BF3B_VRL_B_3	16		✓	
146	BF3B_VINJ_B_3	16		✓	
147	BF3B_HEATER_B_3	16		✓	
148	BF3B_VDL_B_3	16		✓	
149	BF3B_VSS_B_3	16		✓	
150	BF3B_VGL_B_3	16		✓	
151	BF3B_CKRLH_B_3	16		✓	
152	BF3B_CKRLL_B_3	16		✓	
153	BF3B_VDECXH_B_3	16		✓	
154	BF3B_VDECXL_B_3	16		✓	
155	BF3B_VSMSH_B_3	16		✓	
156	BF3B_VSMSL_B_3	16		✓	

157	BF3B_VDDPROT_CLB3	16		✓	
158	BF3B_GND_BU_B_3	16		✓	
159	BF3B_VDD_B_3	16		✓	
160	BF3B_VGG_B_3	16		✓	
161	BF3B_VSS_BU_B_3	16		✓	
162	BF3B_VDL_BU_B_3	16		✓	
163	BF3B_VGL_BU_B_3	16		✓	
164	BF3B_VDDPROT_BUB3	16		✓	
165	I_HEATER_B_3	16		✓	✓
166	I_VSS_B_3	16		✓	
167	I_VSS_BU_B_3	16		✓	
168	VH_BLIND_B_3	16		✓	
169	CKTRIL_REF_B_3	16		✓	
170	BC_PWR_ANA_P_3	16	✓	✓	
171	BC_PWR_ANA_N_3	16	✓	✓	
172	BC_PWR_DIG_3	16	✓	✓	
173	BF4B_VH_B_4	16		✓	
174	BF4B_VL_B_4	16		✓	
175	BF4B_VRL_B_4	16		✓	
176	BF4B_VINJ_B_4	16		✓	
177	BF4B_HEATER_B_4	16		✓	
178	BF4B_VDL_B_4	16		✓	
179	BF4B_VSS_B_4	16		✓	
180	BF4B_VGL_B_4	16		✓	
181	BF4B_CKRLH_B_4	16		✓	
182	BF4B_CKRLL_B_4	16		✓	
183	BF4B_VDECXH_B_4	16		✓	
184	BF4B_VDECXL_B_4	16		✓	
185	BF4B_VSMSH_B_4	16		✓	
186	BF4B_VSMSL_B_4	16		✓	
187	BF4B_VDDPROT_CLB4	16		✓	
188	BF4B_GND_BU_B_4	16		✓	
189	BF4B_VDD_B_4	16		✓	
190	BF4B_VGG_B_4	16		✓	
191	BF4B_VSS_BU_B_4	16		✓	
192	BF4B_VDL_BU_B_4	16		✓	
193	BF4B_VGL_BU_B_4	16		✓	
194	BF4B_VDDPROT_BUB4	16		✓	
195	I_HEATER_B_4	16		✓	✓
196	I_VSS_B_4	16		✓	
197	I_VSS_BU_B_4	16		✓	
198	VH_BLIND_B_4	16		✓	
199	CKTRIL_REF_B_4	16		✓	
200	BC_PWR_ANA_P_4	16	✓	✓	
201	BC_PWR_ANA_N_4	16	✓	✓	
202	BC_PWR_DIG_4	16	✓	✓	
203	BF1R_VH_R_1	16		✓	
204	BF1R_VL_R_1	16		✓	
205	BF1R_VRL_R_1	16		✓	
206	BF1R_VINJ_R_1	16		✓	
207	BF1R_HEATER_R_1	16		✓	

208	BF1R_VDL_R_1	16		✓	
209	BF1R_VSS_R_1	16		✓	
210	BF1R_VGL_R_1	16		✓	
211	BF1R_CKRLH_R_1	16		✓	
212	BF1R_CKRLL_R_1	16		✓	
213	BF1R_VDECXH_R_1	16		✓	
214	BF1R_VDECXL_R_1	16		✓	
215	BF1R_VSMSH_R_1	16		✓	
216	BF1R_VSMSL_R_1	16		✓	
217	BF1R_VDDPROT_CLR1	16		✓	
218	BF1R_GND_BU_R_1	16		✓	
219	BF1R_VDD_R_1	16		✓	
220	BF1R_VGG_R_1	16		✓	
221	BF1R_VSS_BU_R_1	16		✓	
222	BF1R_VDL_BU_R_1	16		✓	
223	BF1R_VGL_BU_R_1	16		✓	
224	BF1R_VDDPROT_BUR1	16		✓	
225	I_HEATER_R_1	16		✓	✓
226	I_VSS_R_1	16		✓	
227	I_VSS_BU_R_1	16		✓	
228	VH_BLIND_R_1	16		✓	
229	CKTRIL_REF_R_1	16		✓	
230	BC_PWR_ANA_P_5	16	✓	✓	
231	BC_PWR_ANA_N_5	16	✓	✓	
232	BC_PWR_DIG_5	16	✓	✓	
233	BF2R_VH_R_2	16		✓	
234	BF2R_VL_R_2	16		✓	
235	BF2R_VRL_R_2	16		✓	
236	BF2R_VINJ_R_2	16		✓	
237	BF2R_HEATER_R_2	16		✓	
238	BF2R_VDL_R_2	16		✓	
239	BF2R_VSS_R_2	16		✓	
240	BF2R_VGL_R_2	16		✓	
241	BF2R_CKRLH_R_2	16		✓	
242	BF2R_CKRLL_R_2	16		✓	
243	BF2R_VDECXH_R_2	16		✓	
244	BF2R_VDECXL_R_2	16		✓	
245	BF2R_VSMSH_R_2	16		✓	
246	BF2R_VSMSL_R_2	16		✓	
247	BF2R_VDDPROT_CLR2	16		✓	
248	BF2R_GND_BU_R_2	16		✓	
249	BF2R_VDD_R_2	16		✓	
250	BF2R_VGG_R_2	16		✓	
251	BF2R_VSS_BU_R_2	16		✓	
252	BF2R_VDL_BU_R_2	16		✓	
253	BF2R_VGL_BU_R_2	16		✓	
254	BF2R_VDDPROT_BUR2	16		✓	
255	I_HEATER_R_2	16		✓	✓
256	I_VSS_R_2	16		✓	
257	I_VSS_BU_R_2	16		✓	
258	VH_BLIND_R_2	16		✓	

259	CKTRIL_REF_R_2	16		✓	
260	BC_PWR_ANA_P_6	16	✓	✓	
261	BC_PWR_ANA_N_6	16	✓	✓	
262	BC_PWR_DIG_6	16	✓	✓	
263	BC_TEMP_BOLC_R_1	16	✓	✓	✓
264	BC_TEMP_BOLC_R_2	16	✓	✓	✓
265	BC_TEMP_BOLC_R_3	16	✓	✓	✓
266	BC_TEMP_BOLC_R_4	16	✓	✓	✓
DEC HK (Bolometers section 1st block— end)		Bol Total	352	2944	160
		Packet Total	1338	3930	810
DEC HK (DEC and SPU HW section — begin)					
267	DMC_SW_GLOBAL_ST	24	✓	✓	✓
268	DMC_SEQ_STATUS	24	✓	✓	✓
269	DMC_DPU_REC_STAT	24	✓	✓	✓
270	DMC_DPU_SEN_STAT	24	✓	✓	✓
271	DMC_DECB_REC_STA	24	✓	✓	✓
272	DMC_DECB_CTRL_ST	24	✓	✓	✓
273	DMC_BLUE_PAC_ENC	24	✓	✓	✓
274	DMC_DECR_REC_STA	24	✓	✓	✓
275	DMC_DECR_CTRL_ST	24	✓	✓	✓
276	DMC_RED_PAC_ENC	24	✓	✓	✓
277	DMC_BOL_REC_STAT	24	✓	✓	✓
278	DMC_BOL_CTRL_STA	24	✓	✓	✓
279	DMC_GRAT_CTRL_ST	32	✓	✓	
280	DMC_CHOP_CTRL_ST	32	✓	✓	
281	DMC_FW_SPEC_CTRL	32	✓	✓	
282	DMC_FW_PHOT_CTRL	32	✓	✓	
283	DMC_CHECKSUM	–	not transmitted to ground		
284	DMC_CS1_CTRL_STA	32	✓	✓	
285	DMC_CS2_CTRL_STA	32	✓	✓	
286	DMC_SEQ_OPTIONS	4	✓	✓	
287	DMC_SEQ_POINTER	8	✓	✓	
288	DMC_SEQ_LOOP_ID0	16	✓	✓	
289	DMC_SEQ_LOOP_ID1	16	✓	✓	
290	DMC_SEQ_LOOP_ID2	16	✓	✓	
291	DMC_SEQ_LOOP_ID3	16	✓	✓	
292	DMC_SEQ_LOOP_ID4	16	✓	✓	
293	DMC_SEQ_WAIT_IND	16	✓	✓	
294	DMC_SEQ_LABEL	8	✓	✓	
295	DMC_TIME_1	32	✓	✓	✓
296	DMC_TIME_2	16	✓	✓	✓
297	DMC_DECB_REC_PAC	16	✓	✓	✓
298	DMC_DECR_REC_PAC	16	✓	✓	✓
299	DMC_DECB_CTRL_PA	16	✓	✓	✓
300	DMC_DECR_CTRL_PA	16	✓	✓	✓
301	DMC_BLUE_ENC_PAC	16	✓	✓	✓
302	DMC_RED_ENC_PAC	16	✓	✓	✓
303	DMC_BOL_REC_PAC	16	✓	✓	✓
304	DMC_BOL_CTRL_PAC	16	✓	✓	✓
305	DMC_DPU_REC_PAC	16	✓	✓	✓

306	DMC_DPU_SEND_PAC	16	✓	✓	✓
307	DMC_B_SPEC_READ	32	✓	✓	✓
308	DMC_R_SPEC_READ	32	✓	✓	✓
309	DMC_BOL_READ_CNT	32	✓	✓	✓
310	DMC_CPU_LOAD	10	✓	✓	✓
311	DMC_IRS_CNT	32	✓	✓	✓
312	DMC_VID	32	✓	✓	✓
313	DMC_CHOP_CUR_POS	16	✓	✓	✓
314	DMC_CHOP_SETPOIN	16	✓	✓	
315	DMC_CHOP_TARGET	16	✓	✓	
316	DMC_CHOP_PID_ERR	16	✓	✓	
317	DMC_CHOP_PID_ACC	32	✓	✓	
318	DMC_CHOP_MAX_DIT	16	✓	✓	
319	DMC_GRAT_CUR_POS	24	✓	✓	✓
320	DMC_GRAT_SETPOIN	24	✓		
321	DMC_GRAT_TARGET	24	✓		
322	DMC_GRAT_PID_ERR	24	✓		
323	DMC_GRAT_PID_ACC	32	✓		
324	DMC_FWSP_CUR_POS	4	✓	✓	✓
325	DMC_FWGRT_HALLA	16	✓	✓	
326	DMC_FWGRT_HALLB	16	✓	✓	
327	DMC_CHOP_OUTPUT	32	✓	✓	
328	DMC_ISR_STAT	4	✓		
329	DMC_FWPH_CUR_POS	4	✓	✓	✓
330	DMC_PLL_RES_LO	32	✓	✓	
331	DMC_PLL_RES_HI	16	✓	✓	
332	DMC_DECB_VDDD_3	16	✓		
333	DMC_DECB_VSS_3	16	✓		
334	DMC_DECB_VGND_3	16	✓		
335	DMC_DECB_VCAN1_3	16	✓		
336	DMC_DECB_VCAN2_3	16	✓		
337	DMC_DECB_V0BIAS3	16	✓		
338	DMC_DECB_VBI_R_3	16	✓		
339	DMC_DECB_V0V_3	16	✓		
340	DMC_DECB_VSCP_3	16	✓		
341	DMC_DECB_VDDR_3	16	✓		
342	DMC_DECB_VDDA_3	16	✓		
343	DMC_DECB_VWELL_3	16	✓		
344	DMC_DECB_IDDA_3	16	✓		
345	DMC_DECB_IDDD_3	16	✓		
346	DMC_DECB_ISS_3	16	✓		
347	DMC_DECB_IGND_3	16	✓		
348	DMC_DECB_HEAT_C	16	✓		
349	DMC_DECB_HEAT_V	16	✓		
350	DMC_DECB_RED_0V3	16	✓		
351	DMC_DECB_DCDC_T3	16	✓		
352	DMC_DECB_DCDC_P5	16	✓		
353	DMC_DECB_AC_CUR	16	✓		
354	DMC_DECB_TS_ST_3	4	✓		
355	DMC_DECB_CL_RO_3	16	✓		
356	DMC_DECB_RO_RA_3	16	✓		

357	DMC_DECB_CR_ST_3	16	✓
358	DMC_DECB_BR_CM_3	12	✓
359	DMC_DECB_ZB_CM_3	12	✓
360	DMC_DECB_SR_RB_3	16	✓
361	DMC_DECB_TS_1_3	16	✓
362	DMC_DECB_TS_2_3	16	✓
363	DMC_DECB_RO_CO_3	16	✓
364	DMC_DECB_RA_CO_3	32	✓
365	DMC_DECB_VDDD_4	16	✓
366	DMC_DECB_VSS_4	16	✓
367	DMC_DECB_VGND_4	16	✓
368	DMC_DECB_VCAN1_4	16	✓
369	DMC_DECB_VCAN2_4	16	✓
370	DMC_DECB_V0BIAS4	16	✓
371	DMC_DECB_VBI_R_4	16	✓
372	DMC_DECB_V0V_4	16	✓
373	DMC_DECB_VSCP_4	16	✓
374	DMC_DECB_VDDR_4	16	✓
375	DMC_DECB_VDDA_4	16	✓
376	DMC_DECB_VWELL_4	16	✓
377	DMC_DECB_IDDA_4	16	✓
378	DMC_DECB_IDDD_4	16	✓
379	DMC_DECB_ISS_4	16	✓
380	DMC_DECB_IGND_4	16	✓
381	DMC_DECB_FLASH_C	16	✓
382	DMC_DECB_FLASH_V	16	✓
383	DMC_DECB_REF_0V4	16	✓
384	DMC_DECB_DCDC_T4	16	✓
385	DMC_DECB_DCDCP15	16	✓
386	DMC_DECB_DCDCN15	16	✓
387	DMC_DECB_TS_ST_4	4	✓
388	DMC_DECB_CL_RO_4	16	✓
389	DMC_DECB_RO_RA_4	16	✓
390	DMC_DECB_CR_ST_4	16	✓
391	DMC_DECB_BR_CM_4	12	✓
392	DMC_DECB_ZB_CM_4	12	✓
393	DMC_DECB_SR_RB_4	16	✓
394	DMC_DECB_TS_1_4	16	✓
395	DMC_DECB_TS_2_4	16	✓
396	DMC_DECB_RO_CO_4	16	✓
397	DMC_DECB_RA_CO_4	32	✓
398	DMC_DECR_VDDD_1	16	✓
399	DMC_DECR_VSS_1	16	✓
400	DMC_DECR_VGND_1	16	✓
401	DMC_DECR_VCAN1_1	16	✓
402	DMC_DECR_VCAN2_1	16	✓
403	DMC_DECR_V0BIAS1	16	✓
404	DMC_DECR_VBI_R_1	16	✓
405	DMC_DECR_V0V_1	16	✓
406	DMC_DECR_VSCP_1	16	✓
407	DMC_DECR_VDDR_1	16	✓

408	DMC_DECR_VDDA_1	16	✓
409	DMC_DECR_VWELL_1	16	✓
410	DMC_DECR_IDDA_1	16	✓
411	DMC_DECR_IDDD_1	16	✓
412	DMC_DECR_ISS_1	16	✓
413	DMC_DECR_IGND_1	16	✓
414	DMC_DECR_HEAT_C	16	✓
415	DMC_DECR_HEAT_V	16	✓
416	DMC_DECR_REF_0V_1	16	✓
417	DMC_DECR_DCDC_T1	16	✓
418	DMC_DECR_DCDCP5	16	✓
419	DMC_DECR_AC_CUR	16	✓
420	DMC_DECR_TS_ST_1	4	✓
421	DMC_DECR_CL_RO_1	16	✓
422	DMC_DECR_RO_RA_1	16	✓
423	DMC_DECR_CR_ST_1	16	✓
424	DMC_DECR_BR_CM_1	12	✓
425	DMC_DECR_ZB_CM_1	12	✓
426	DMC_DECR_SR_RB_1	16	✓
427	DMC_DECR_TS_1_1	16	✓
428	DMC_DECR_TS_2_1	16	✓
429	DMC_DECR_RO_CO_1	16	✓
430	DMC_DECR_RA_CO_1	32	✓
431	DMC_DECR_VDDD_2	16	✓
432	DMC_DECR_VSS_2	16	✓
433	DMC_DECR_VGND_2	16	✓
434	DMC_DECR_VCAN1_2	16	✓
435	DMC_DECR_VCAN2_2	16	✓
436	DMC_DECR_V0BIAS2	16	✓
437	DMC_DECR_VBI_R_2	16	✓
438	DMC_DECR_V0V_2	16	✓
439	DMC_DECR_VSCP_2	16	✓
440	DMC_DECR_VDDR_2	16	✓
441	DMC_DECR_VDDA_2	16	✓
442	DMC_DECR_VWELL_2	16	✓
443	DMC_DECR_IDDA_2	16	✓
444	DMC_DECR_IDDD_2	16	✓
445	DMC_DECR_ISS_2	16	✓
446	DMC_DECR_IGND_2	16	✓
447	DMC_DECR_FLASH_C	16	✓
448	DMC_DECR_FLASH_V	16	✓
449	DMC_DECR_REF_0V2	16	✓
450	DMC_DECR_DCDC_T2	16	✓
451	DMC_DECR_DCDCP15	16	✓
452	DMC_DECR_DCDCN15	16	✓
453	DMC_DECR_TS_ST_2	4	✓
454	DMC_DECR_CL_RO_2	16	✓
455	DMC_DECR_RO_RA_2	16	✓
456	DMC_DECR_CR_ST_2	16	✓
457	DMC_DECR_BR_CM_2	12	✓
458	DMC_DECR_ZB_CM_2	12	✓

459	DMC_DECR_SR_RB_2	16	✓		
460	DMC_DECR_TS_1_2	16	✓		
461	DMC_DECR_TS_2_2	16	✓		
462	DMC_DECR_RO_CO_2	16	✓		
463	DMC_DECR_RA_CO_2	32	✓		
464	DMC_FPU_T_SENS_ST	14	✓	✓	✓
465	DMC_FW_SPEC_TEMP	16	✓	✓	✓
466	DMC_FW_PHOT_TEMP	16	✓	✓	✓
467	DMC_CHOPPER_TEMP	16	✓	✓	✓
468	DMC_GRATING_TEMP	16	✓	✓	✓
469	DMC_PSC_V1	16	✓	✓	✓
470	DMC_PSC_V2	16	✓	✓	✓
471	DMC_PSC_V3	16	✓	✓	✓
472	DMC_PSC_V4	16	✓	✓	✓
473	DMC_DCDC_TEMP	16	✓	✓	✓
474	DMC_DSP_TEMP	16	✓	✓	✓
475	DMC_SPU_PSU_P15V	16	✓	✓	✓
476	DMC_SPU_SWL_TEMP	16	✓	✓	✓
477	DMC_SPU_LWL_TEMP	16	✓	✓	✓
478	DMC_SPU_PS_TEMP	16	✓	✓	✓
479	DMC_SPU_VCC_CUR	16	✓	✓	✓
480	DMC_SPU_VCC_VOL	16	✓	✓	✓
481	DMC_SPU_VP_CUR	16	✓	✓	✓
482	DMC_FPU_T1_T	16	✓	✓	✓
483	DMC_FPU_T2_T	16	✓	✓	✓
484	DMC_REF_VOLT_0V	16	✓	✓	✓
485	DMC_CAL_SRC_TEMP	16	✓	✓	✓
486	DMC_REF_VOLT_5V	16	✓	✓	✓
487	DMC_CUSTOM_ENT_1	32	✓	✓	✓
488	DMC_CUSTOM_ENT_2	32	✓	✓	✓
489	DMC_CUSTOM_ENT_3	32	✓	✓	✓
490	DMC_CUSTOM_ENT_4	32	✓	✓	✓
491	DMC_CUSTOM_ENT_5	32	✓	✓	✓
492	DMC_CUSTOM_ENT_6	32	✓	✓	✓
493	DMC_CUSTOM_ENT_7	32	✓	✓	✓
494	DMC_CUSTOM_ENT_8	32	✓	✓	✓
495	DMC_CUSTOM_ENT_9	32	✓	✓	✓
496	DMC_CUSTOM_ENT10	32	✓	✓	✓
497	DMC_DET_SIM_STAT	32	✓	✓	
498	DMC_DET_SIM_PER	32	✓	✓	
499	DMC_CS1_RES_VALUE	32	✓	✓	
500	DMC_CS1_OUTPUT	16	✓	✓	
501	DMC_CS2_RES_VALUE	32	✓	✓	
502	DMC_CS2_OUTPUT	16	✓	✓	
503	DMC_BOLC_STATUS	16	✓	✓	✓
504	DMC_BSPU_TR_MODE	32	✓	✓	
505	DMC_RSPU_TR_MODE	32	✓	✓	
506	DMC_GRAT_OUTPUT	32	✓		
507	DMC_OBT_COUNT	32	✓	✓	
508	DMC_MIM_ST	32	✓	✓	
509	DMC_DM_SF_IND	8	✓	✓	✓

510	DMC_PM_SF_IND	8	✓	✓	✓
511	DMC_DM_DF_IND	8	✓	✓	✓
512	DMC_PM_DF_IND	8	✓	✓	✓
513	DMC_CS1_TARGET	32	✓	✓	
514	DMC_CS2_TARGET	32	✓	✓	
515	DMC_HK_CTRL_STAT	24	✓	✓	✓
516	DMC_HK_DIAG_STAT	24	✓	✓	✓
517	DMC_HK_DIAG_PERI	32	✓	✓	✓
518	DMC_LAST_ERR_ID	4	✓	✓	✓
519	DMC_LAST_ER_BF1	16	✓	✓	✓
520	DMC_LAST_ER_BF2	16	✓	✓	✓
521	DMC_LAST_ER_BF3	16	✓	✓	✓
522	DMC_LAST_ER_BF4	16	✓	✓	✓
523	DMC_LAST_ER_BF5	16	✓	✓	✓
524	DMC_LAST_ER_BF6	16	✓	✓	✓
525	DMC_LAST_ER_BF7	16	✓	✓	✓
526	DMC_LAST_ER_BF8	16	✓	✓	✓
527	DMC_LAST_ER_BF9	16	✓	✓	✓
528	DMC_LAST_ER_BF10	16	✓	✓	✓
529	DMC_LAST_ER_BF11	16	✓	✓	✓
530	DMC_LAST_ER_BF12	16	✓	✓	✓
531	DMC_LAST_ER_BF13	16	✓	✓	✓
532	DMC_LAST_ER_BF14	16	✓	✓	✓
533	DMC_LAST_ER_BF15	16	✓	✓	✓
534	DMC_LAST_ER_BF16	16	✓	✓	✓
DEC HK (DEC and SPU HW section — end)		DEC Total	4892	2656	1788
		Packet Total	6230	6586	2598
DEC HK (Bolometers section 2nd block— begin)					
535	BC_TEMP_BOLC_R_5	16	✓	✓	✓
536	BC_TEMP_BOLC_B_1	16	✓	✓	✓
537	BC_TEMP_BOLC_B_2	16	✓	✓	✓
538	BC_TEMP_BOLC_B_3	16	✓	✓	✓
539	BC_TEMP_BOLC_DAQ	16	✓	✓	✓
540	BC_TEMP_PSU_1	16	✓	✓	✓
541	BC_TEMP_PSU_2	16	✓	✓	✓
542	BCLR_TEMP_SP	16	✓	✓	✓
543	BCLR_TEMP_SP_SWT	16	✓	✓	✓
544	BCLR_TEMP_TS	16	✓	✓	✓
545	BCLR_TEMP_EV_SWT	16	✓	✓	✓
546	BFBR_TEMP_FPU_ST	16	✓	✓	✓
547	BCLR_TEMP_EV	16	✓	✓	✓
548	BFBR_TEMP_FPU1	16	✓	✓	✓
549	BFBR_TEMP_FPU2	16	✓	✓	✓
550	BCLR_HEATER_SP	16		✓	✓
551	BCLR_HEAT_SP_SWT	16		✓	✓
552	BCLR_HEAT_EV_SWT	16		✓	✓
553	BFBR_HEATER_FPU	16		✓	✓
554	BC_PWR_ANA_P_7	16	✓	✓	✓
555	BC_PWR_ANA_N_7	16	✓	✓	✓
556	BC_PWR_DIG_7	16	✓	✓	✓

DEC HK (Bolometers section 2nd block— end)Bol Total	288	352	352
Packet Total	6518	6938	2950

D go_SAFE OBCPs

In this appendix the commands sent by DPU to the other subsystems during the execution of the go_SAFE OBCPs are given. Note that commands for BOLC subsystem are sent via MEC unit.

D.1 go_SAFE ID=24

This is the OBCP more frequently used.

```
Begin
DPU_Set_Function(Function ID = 5; Set function OFF)
DPU_Set_Function(Function ID = 12; Set function OFF)
DPU_Set_Function(Function ID = 14; Set function OFF)
DPU_Set_Function(Function ID = 15; Set function OFF)
DPU_Set_Function(Function ID = 17; Set function OFF)
DPU_Set_Function(Function ID = 18; Set function OFF)
DPU_Set_Function(Function ID = 20; Set function OFF)
DPU_Set_Function(Function ID = 101; Set function ON)
DPU_Set_Function(Function ID = 102; Set function ON)
DPU_Set_Function(Function ID = 103; Set function ON)
DMC_ABORT_SEQUENCE()
    DPU_Set_Function(Function ID = 103; Set function ON)
DMC_STOP_DIAG_HK()
    DPU_Set_Function(Function ID = 103; Set function ON)
DMC_SYNCHRONIZE_ON_DETECTOR(0x800)
    DPU_Set_Function(Function ID = 103; Set function ON)
DMC_DISABLE_BB_1_CONT()
    DPU_Set_Function(Function ID = 103; Set function ON)
DMC_SWOF_BB_1_CONT()
    DPU_Set_Function(Function ID = 103; Set function ON)
DMC_DISABLE_BB_2_CONT()
    DPU_Set_Function(Function ID = 103; Set function ON)
DMC_SWOF_BB_2_CONT()
    DPU_Set_Function(Function ID = 103; Set function ON)
DMC_DISABLE_GRAT_CONT()
    DPU_Set_Function(Function ID = 103; Set function ON)
DMC_SWOF_GRAT_CONT()
    DPU_Set_Function(Function ID = 103; Set function ON)
DMC_DISABLE_CHOP_CONT()
    DPU_Set_Function(Function ID = 103; Set function ON)
DMC_SWOF_CHOP_CONT()
    DPU_Set_Function(Function ID = 103; Set function ON)
STOP_REDUCTION_COMPRESSION_BLUE()
    DPU_Set_Function(Function ID = 101; Set function ON)
STOP_REDUCTION_COMPRESSION_RED()
    DPU_Set_Function(Function ID = 102; Set function ON)
SPU_WRITE_DET_SEL_TABLE_1_BLUE(129, 24, 0xFA1, 0x81, 0x200,
0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF,
```

```
0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF,
0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0, 0, 0, 0, 0,  crc)
    DPU_Set_Function(Function ID = 101; Set function ON)
SPU_WRITE_DET_SEL_TABLE_1_RED(129, 24, 0xFA1, 0x81, 0x200,
0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF,
0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF,
0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0, 0, 0, 0, 0,  crc)
    DPU_Set_Function(Function ID = 102; Set function ON)
SPU_WRITE_DET_SEL_TABLE_2_BLUE(130, 24, 0xFA1, 0x82, 0x200,
0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF,
0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF,
0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0, 0, 0, 0, 0,  crc)
    DPU_Set_Function(Function ID = 101; Set function ON)
SPU_WRITE_DET_SEL_TABLE_2_RED(130, 24, 0xFA1, 0x82, 0x200,
0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF,
0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF,
0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0, 0, 0, 0, 0,  crc)
    DPU_Set_Function(Function ID = 102; Set function ON)
SPU_WRITE_DET_SEL_TABLE_3_BLUE(131, 24, 0xFA1, 0x83, 0x200,
0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF,
0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF,
0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0, 0, 0, 0, 0,  crc)
    DPU_Set_Function(Function ID = 101; Set function ON)
SPU_WRITE_DET_SEL_TABLE_3_RED(131, 24, 0xFA1, 0x83, 0x200,
0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF,
0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF,
0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0, 0, 0, 0, 0,  crc)
    DPU_Set_Function(Function ID = 102; Set function ON)
SPU_WRITE_DET_SEL_TABLE_4_BLUE(132, 24, 0xFA1, 0x84, 0x200,
0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF,
0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF,
0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0, 0, 0, 0, 0,  crc)
    DPU_Set_Function(Function ID = 101; Set function ON)
SPU_WRITE_DET_SEL_TABLE_4_RED(132, 24, 0xFA1, 0x84, 0x200,
0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF,
0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF,
0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0, 0, 0, 0, 0,  crc)
    DPU_Set_Function(Function ID = 102; Set function ON)
SPU_WRITE_DET_SEL_TABLE_5_BLUE(133, 24, 0xFA1, 0x85, 0x200,
0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF,
0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF,
0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0, 0, 0, 0, 0,  crc)
    DPU_Set_Function(Function ID = 101; Set function ON)
SPU_WRITE_DET_SEL_TABLE_5_RED(133, 24, 0xFA1, 0x85, 0x200,
0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF,
0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF,
0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0, 0, 0, 0, 0,  crc)
    DPU_Set_Function(Function ID = 102; Set function ON)
SPU_WRITE_DET_SEL_TABLE_6_BLUE(134, 24, 0xFA1, 0x86, 0x1C2,
0xFFFFFFBF, 0xFFFFEFFF, 0xFFFBFFFF, 0xFEFFFFFF, 0xBFFFFFEF, 0xFFFFFBFF,
0xFFFEFFFF, 0xFFBFFFFF, 0xEFFFFFFB, 0xFFFFFEFF, 0xFFFFBFFF, 0xFFEFFFFF,
```

```
0xFBFFFFFE, 0xFFFFFFBF, 0xFFFFE000, 0, 0, 0, 0, 0, 0,  crc)
    DPU_Set_Function(Function ID = 101; Set function ON)
SPU_WRITE_DET_SEL_TABLE_6_RED(134, 24, 0xFA1, 0x86, 0x1C2,
0xFFFFFFBF, 0xFFFFEFFF, 0xFFFBFFFF, 0xFEFFFFFF, 0xBFFFFFEF, 0xFFFFFBFF,
0xFFFEFFFF, 0xFFBFFFFF, 0xEFFFFFFB, 0xFFFFFEFF, 0xFFFFBFFF, 0xFFEFFFFF,
0xFBFFFFFE, 0xFFFFFFBF, 0xFFFFE000, 0, 0, 0, 0, 0, 0,  crc)
    DPU_Set_Function(Function ID = 102; Set function ON)
SPU_WRITE_DET_SEL_TABLE_7_BLUE(135, 24, 0xFA1, 0x87, 0x1C2,
0xFFFFFFBF, 0xFFFFEFFF, 0xFFFBFFFF, 0xFEFFFFFF, 0xBFFFFFEF, 0xFFFFFBFF,
0xFFFEFFFF, 0xFFBFFFFF, 0xEFFFFFFB, 0xFFFFFEFF, 0xFFFFBFFF, 0xFFEFFFFF,
0xFBFFFFFE, 0xFFFFFFBF, 0xFFFFE000, 0, 0, 0, 0, 0, 0,  crc)
    DPU_Set_Function(Function ID = 101; Set function ON)
SPU_WRITE_DET_SEL_TABLE_7_RED(135, 24, 0xFA1, 0x87, 0x1C2,
0xFFFFFFBF, 0xFFFFEFFF, 0xFFFBFFFF, 0xFEFFFFFF, 0xBFFFFFEF, 0xFFFFFBFF,
0xFFFEFFFF, 0xFFBFFFFF, 0xEFFFFFFB, 0xFFFFFEFF, 0xFFFFBFFF, 0xFFEFFFFF,
0xFBFFFFFE, 0xFFFFFFBF, 0xFFFFE000, 0, 0, 0, 0, 0, 0,  crc)
    DPU_Set_Function(Function ID = 102; Set function ON)
DPU_Set_BusList(Function ID = (Exiting Burst Mode)
DMC_WRT_B_DEC_REC_OPT(23, 1, 4,  crc)
    DPU_Set_Function(Function ID = 103; Set function ON)
DMC_WRT_R_DEC_REC_OPT(24, 1, 4,  crc)
    DPU_Set_Function(Function ID = 103; Set function ON)
DMC_WRT_BOL_REC_OPT(22, 1, 4,  crc)
    DPU_Set_Function(Function ID = 103; Set function ON)
DMC_SWOF_R_SPEC()
    DPU_Set_Function(Function ID = 103; Set function ON)
DMC_SWOF_B_SPEC()
    DPU_Set_Function(Function ID = 103; Set function ON)
DMC_SWOF_R_DEC()
    DPU_Set_Function(Function ID = 103; Set function ON)
DMC_SWOF_B_DEC()
    DPU_Set_Function(Function ID = 103; Set function ON)
BOL_Set_data_mode(Mode = 0)
    DPU_Set_Function(Function ID = 103; Set function ON)
BOL_Set_seq_mode(Mode = 00)
    DPU_Set_Function(Function ID = 103; Set function ON)
BOL_Reset_Bias(Matrix group = 0)
    DPU_Set_Function(Function ID = 103; Set function ON)
BOL_Set_HSP_heater_current(Current value = 3010)
    DPU_Set_Function(Function ID = 103; Set function ON)
BOL_Set_HSE_heater_current(Current value = 0)
    DPU_Set_Function(Function ID = 103; Set function ON)
BOL_Set_SP_heater_current(Current value = 0)
    DPU_Set_Function(Function ID = 103; Set function ON)
BOL_Set_TS_heater_current(Current value = 0)
    DPU_Set_Function(Function ID = 103; Set function ON)
BOL_Set_on/off_group(On/off word = 0)
    DPU_Set_Function(Function ID = 103; Set function ON)
DPU_Set_Function(Function ID = 21; Set function ON)
DPU_Wait_time(3000)
DPU_Set_HK_list(HK selection = NPRI; Array selected = BOTH)
```

D.2 go_SAFE ID=17

This is the alternative OBCP to put PACS in SAFE mode. It is similar to the previous one but without switching off the controller of calibrations sources.

```
Begin
DPU_Set_Function(Function ID = 5; Set function OFF)
DPU_Set_Function(Function ID = 12; Set function OFF)
DPU_Set_Function(Function ID = 14; Set function OFF)
DPU_Set_Function(Function ID = 15; Set function OFF)
DPU_Set_Function(Function ID = 17; Set function OFF)
DPU_Set_Function(Function ID = 18; Set function OFF)
DPU_Set_Function(Function ID = 20; Set function OFF)
DPU_Set_Function(Function ID = 101; Set function ON)
DPU_Set_Function(Function ID = 102; Set function ON)
DPU_Set_Function(Function ID = 103; Set function ON)
DMC_ABORT_SEQUENCE()
    DPU_Set_Function(Function ID = 103; Set function ON)
DMC_STOP_DIAG_HK()
    DPU_Set_Function(Function ID = 103; Set function ON)
DMC_SYNCHRONIZE_ON_DETECTOR(0x800)
    DPU_Set_Function(Function ID = 103; Set function ON)
DMC_DISABLE_GRAT_CONT()
    DPU_Set_Function(Function ID = 103; Set function ON)
DMC_SWOF_GRAT_CONT()
    DPU_Set_Function(Function ID = 103; Set function ON)
DMC_DISABLE_CHOP_CONT()
    DPU_Set_Function(Function ID = 103; Set function ON)
DMC_SWOF_CHOP_CONT()
    DPU_Set_Function(Function ID = 103; Set function ON)
STOP_REDUCTION_COMPRESSION_BLUE()
    DPU_Set_Function(Function ID = 101; Set function ON)
STOP_REDUCTION_COMPRESSION_RED()
    DPU_Set_Function(Function ID = 102; Set function ON)
SPU_WRITE_DET_SEL_TABLE_1_BLUE(129, 24, 0xFA1, 0x81, 0x200,
0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF,
0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF,
0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0, 0, 0, 0, 0,  crc)
    DPU_Set_Function(Function ID = 101; Set function ON)
SPU_WRITE_DET_SEL_TABLE_1_RED(129, 24, 0xFA1, 0x81, 0x200,
0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF,
0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF,
0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0, 0, 0, 0, 0,  crc)
    DPU_Set_Function(Function ID = 102; Set function ON)
SPU_WRITE_DET_SEL_TABLE_2_BLUE(130, 24, 0xFA1, 0x82, 0x200,
0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF,
0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF,
0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0, 0, 0, 0, 0,  crc)
    DPU_Set_Function(Function ID = 101; Set function ON)
```

```
SPU_WRITE_DET_SEL_TABLE_2_RED(130, 24, 0xFA1, 0x82, 0x200,
0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF,
0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF,
0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0, 0, 0, 0, 0,  crc)
    DPU_Set_Function(Function ID = 102; Set function ON)
SPU_WRITE_DET_SEL_TABLE_3_BLUE(131, 24, 0xFA1, 0x83, 0x200,
0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF,
0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF,
0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0, 0, 0, 0, 0,  crc)
    DPU_Set_Function(Function ID = 101; Set function ON)
SPU_WRITE_DET_SEL_TABLE_3_RED(131, 24, 0xFA1, 0x83, 0x200,
0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF,
0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF,
0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0, 0, 0, 0, 0,  crc)
    DPU_Set_Function(Function ID = 102; Set function ON)
SPU_WRITE_DET_SEL_TABLE_4_BLUE(132, 24, 0xFA1, 0x84, 0x200,
0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF,
0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF,
0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0, 0, 0, 0, 0,  crc)
    DPU_Set_Function(Function ID = 101; Set function ON)
SPU_WRITE_DET_SEL_TABLE_4_RED(132, 24, 0xFA1, 0x84, 0x200,
0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF,
0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF,
0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0, 0, 0, 0, 0,  crc)
    DPU_Set_Function(Function ID = 102; Set function ON)
SPU_WRITE_DET_SEL_TABLE_5_BLUE(133, 24, 0xFA1, 0x85, 0x200,
0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF,
0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF,
0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0, 0, 0, 0, 0,  crc)
    DPU_Set_Function(Function ID = 101; Set function ON)
SPU_WRITE_DET_SEL_TABLE_5_RED(133, 24, 0xFA1, 0x85, 0x200,
0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF,
0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF,
0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0, 0, 0, 0, 0,  crc)
    DPU_Set_Function(Function ID = 102; Set function ON)
SPU_WRITE_DET_SEL_TABLE_6_BLUE(134, 24, 0xFA1, 0x86, 0x1C2,
0xFFFFFFBF, 0xFFFFEFFF, 0xFFFBFFFF, 0xFEFFFFFF, 0xBFFFFFEF, 0xFFFFFBFF,
0xFFFEFFFF, 0xFFBFFFFF, 0xEFFFFFFB, 0xFFFFFEFF, 0xFFFFBFFF, 0xFFEFFFFF,
0xFBFFFFFE, 0xFFFFFFBF, 0xFFFFE000, 0, 0, 0, 0, 0, 0,  crc)
    DPU_Set_Function(Function ID = 101; Set function ON)
SPU_WRITE_DET_SEL_TABLE_6_RED(134, 24, 0xFA1, 0x86, 0x1C2,
0xFFFFFFBF, 0xFFFFEFFF, 0xFFFBFFFF, 0xFEFFFFFF, 0xBFFFFFEF, 0xFFFFFBFF,
0xFFFEFFFF, 0xFFBFFFFF, 0xEFFFFFFB, 0xFFFFFEFF, 0xFFFFBFFF, 0xFFEFFFFF,
0xFBFFFFFE, 0xFFFFFFBF, 0xFFFFE000, 0, 0, 0, 0, 0, 0,  crc)
    DPU_Set_Function(Function ID = 102; Set function ON)
SPU_WRITE_DET_SEL_TABLE_7_BLUE(135, 24, 0xFA1, 0x87, 0x1C2,
0xFFFFFFBF, 0xFFFFEFFF, 0xFFFBFFFF, 0xFEFFFFFF, 0xBFFFFFEF, 0xFFFFFBFF,
0xFFFEFFFF, 0xFFBFFFFF, 0xEFFFFFFB, 0xFFFFFEFF, 0xFFFFBFFF, 0xFFEFFFFF,
0xFBFFFFFE, 0xFFFFFFBF, 0xFFFFE000, 0, 0, 0, 0, 0, 0,  crc)
    DPU_Set_Function(Function ID = 101; Set function ON)
SPU_WRITE_DET_SEL_TABLE_7_RED(135, 24, 0xFA1, 0x87, 0x1C2,
```

```
0xFFFFFFBF, 0xFFFFEFFF, 0xFFFBFFFF, 0xFEFFFFFF, 0xBFFFFFEF, 0xFFFFFBFF,
0xFFFEFFFF, 0xFFBFFFFF, 0xEFFFFFFB, 0xFFFFFEFF, 0xFFFFBFFF, 0xFFEFFFFF,
0xFBFFFFFE, 0xFFFFFFBF, 0xFFFFE000, 0, 0, 0, 0, 0, 0,  crc)
    DPU_Set_Function(Function ID = 102; Set function ON)
DPU_Set_BusList(Function ID = (Exiting Burst Mode)
DMC_WRT_B_DEC_REC_OPT(23, 1, 4,  crc)
    DPU_Set_Function(Function ID = 103; Set function ON)
DMC_WRT_R_DEC_REC_OPT(24, 1, 4,  crc)
    DPU_Set_Function(Function ID = 103; Set function ON)
DMC_WRT_BOL_REC_OPT(22, 1, 4,  crc)
    DPU_Set_Function(Function ID = 103; Set function ON)
DMC_SWOF_R_SPEC()
    DPU_Set_Function(Function ID = 103; Set function ON)
DMC_SWOF_B_SPEC()
    DPU_Set_Function(Function ID = 103; Set function ON)
DMC_SWOF_R_DEC()
    DPU_Set_Function(Function ID = 103; Set function ON)
DMC_SWOF_B_DEC()
    DPU_Set_Function(Function ID = 103; Set function ON)
BOL_Set_data_mode(Mode = 0)
    DPU_Set_Function(Function ID = 103; Set function ON)
BOL_Set_seq_mode(Mode = 00)
    DPU_Set_Function(Function ID = 103; Set function ON)
BOL_Reset_Bias(Matrix group = 0)
    DPU_Set_Function(Function ID = 103; Set function ON)
BOL_Set_HSP_heater_current(Current value = 3010)
    DPU_Set_Function(Function ID = 103; Set function ON)
BOL_Set_HSE_heater_current(Current value = 0)
    DPU_Set_Function(Function ID = 103; Set function ON)
BOL_Set_SP_heater_current(Current value = 0)
    DPU_Set_Function(Function ID = 103; Set function ON)
BOL_Set_TS_heater_current(Current value = 0)
    DPU_Set_Function(Function ID = 103; Set function ON)
BOL_Set_on/off_group(On/off word = 0)
    DPU_Set_Function(Function ID = 103; Set function ON)
DPU_Set_Function(Function ID = 21; Set function ON)
DPU_Wait_time(3000)
DPU_Set_HK_list(HK selection = NPRI; Array selected = BOTH)
```

E Glossary

AOT Astronomical Observation Template
APID Application Identifier
AVM Avionic Model
BBID Building Block Identifier — This parameter, along with the OBSID, identifies the measurement to which each science data packet belongs
BOL BOLometers
BSW Boot SoftWare — The software the DPU executes when switched on, as opposite to the application program (OBS)
CDMS Command and Data Management System

CNR Consiglio Nazionale delle Ricerche

CRC Cyclic Redundancy Code — In this document this acronym is used as synonym of checksum. The mechanism implemented in the OBS complies with the specification of Appendix 4 of AD–2. The checksum is computed on the whole packet content and written in the last two bytes. For the memory load command (6,2) and the write command of service (8,4), a checksum is also computed on the data to load/write: this additional checksum is indicated with the acronym crc

crc — See CRC

CSL Centre Spatial de Liège

DEC Detector Controller — One of the PACS subsystems. Actually the DPU does not interface with it, but with the MEC. However, in this document DEC and MEC are used as synonyms

DPU Digital Processing Unit

DM Data Memory — The 21020 stores the code in program memory, and data in data memory. The former is based on words of 48 bits; the latter on words of 32 bits

DRAM Data RAM — Part of the 21020 memory used to store data (variables). Each word is 4 bytes wide

EEPROM Electrically Erasable Programmable ROM

EOP End Of Packet

EPROM Erasable Programmable ROM

ESA European Space Agency

HIFI Heterodyne Instrument for FIRST

HK HouseKeeping

HLSW High Level SoftWare — It is for the SPU what is called application software for the DPU

HW HardWare

ICC Instrument Control Centre

ICD Interface Control Document

ICU Instrument Control Unit — This is the DPU name for the HIFI instrument

ID Identifier

IFSI Istituto di Fisica dello Spazio Interplanetario

INAF Istituto Nazionale di Astrofisica

LLSW Low Level SoftWare — It is for the SPU what the BSW is for the DPU

LSB Least Significant Byte

LSb Least Significant bit

MEC Mechanical Control Unit — One of the PACS subsystems. In this document there is no difference between this subsystem and the DEC, even if the DPU only interfaces with MEC

MIB Mission Implementation Base — The data base which contains all the data used by SCOS2000 to prepare the TC and to handle the TM packets

MPE Max-Planck-Institut für Extraterrestrische Physik

MSB Most Significant Byte

MSb Most Significant bit

NACK Negative Acknowledgment — When the DPU sends a command to a subsystem an acknowledgment is expected within 200 msec, packetized according to the specific ICD

NoP Number of Parameters — It is used as shortcut in Section 4.18 and 5.18, when discussing the OBCP

OBCP On Board Control Procedure

OBS On Board Software

OBSID Observation Identifier — This parameter, along with the BBID, identifies the measurement to which each science data packet belongs

PACK Positive Acknowledgment — When the DPU sends a command to a subsystem an acknowledgment is expected within 200 msec, packetized according to the specific ICD

PACS Photoconductor Array Camera and Spectrometer

PM Program Memory — The 21020 stores the code in program memory, and data in data memory. The former is based on words of 48 bits; the latter on words of 32 bits

PRAM Program RAM — Part of the 21020 memory used to store the program code. Each word is 6 bytes wide

PROM Programmable ROM — It contains the boot software. For AVM an EPROM is actually used
RAM Random Access Memory
ROM Read Only Memory
SA Subaddress — Acronym specific to the 1553 protocol implementation (see Appendix 9 of AD–2)
SAU Smallest Addressable Unit — Size of one memory word: 4 bytes for DRAM and 6 bytes for PRAM
SCOS2000 Space COntrol System — Roughly speaking, the system used to send commands to the spacecraft and to receive and to store telemetry packets
SID Structure Identifier — In some packets it is used to define in an implicite way the number and type of the parameters sent
SMCS Scalable Multichannel Communication Subsystem — The chip (actually SMCS332) used to implement the 1355 interface
SPL Long Wavelength SPU
SPR Software Problem Report
SPS Short Wavelength SPU
SPU Signal Processing Unit
SSMM Solid State Mass Memory — The spacecraft memory used to store all the telemetry packets, downloaded during ground contact
SW SoftWare
TC Telecommand
TM Telemetry