HERSCHEL

SPIRE On-Board Software User Manual

Document Ref.: SPIRE-IFS-PRJ-001391

Issue: 2.1

Prepared by: Sergio Molinari
Scigè John, Liu

Distribution List:

RAL	**K. King**
	B. Swinyard
	S. "Duke" Sidher
IFSI	**R. Cerulli**
	R. Orfei

Istituto di **F**isica dello **S**pazio **I**nterplanetario – CNR Area di Ricerca Tor Vergata
Via del Fosso del Cavaliere 00133 Roma
Tel.: (39) 06 4993 4490 – Fax: (39) 06 4993 4399

1 Introduction

1.1 Purpose of the document

This document describes in detail the procedures to start-up and run the SPIRE OBS, the contents of the TC packets to be uplinked in order to perform the required function, and the contents of the TM packets that the OBS generates. This document does not duplicate the information provided in RD2, but rather represents its complement for all that is not therein specified.
The present version of this document applies to SPIRE OBS Version 2.0.C

1.2 Acronyms and Glossary

AVM	Avionic Model
BC	Bus Controller
BP	BreakPoint
BSW	DPU Boot Software
CDMS	Command and Data Management System
DM	Data Memory (DSP)
DPU	Digital Processing Unit
DSP	Digital Signal Processor
DTST	Dedicated Test Software Tools
EGSE	Electrical Ground Support Equipment
EEPROM	Electrically Erasable Programmable Read-Only Memory
ESA	European Space Agency
HERSCHEL	Herschel Space Observatory
HK	Housekeeping
HW	Hardware
ICE	DSP In-Circuit Emulator
I/F	Interface
IFSI	Istituto di Fisica dello Spazio Interplanetario
NA	Not Applicable
OBS	On-Board Software
PM	Program Memory (DSP)
PROM	Programmable Read-Only Memory
RAM	Random Access Memory
S/C	Spacecraft
S/S	Subsystem
SUT	Software Under Test
TBC	To Be Confirmed
TBD	To Be Defined
TBW	To Be Written
TC	Telecommand
TM	Telemetry
VME	Virtual Machine Executable Code

1.3 Document List

1.3.1 Applicable Documents

Document Reference	Name	Number/version/date
AD1	SPIRE OBS User Requirements Document	SPIRE-IFS-PRJ-000444
AD2	SPIRE OBS Software Specifications Document	SPIRE-IFS-DOC-001352
AD3	Packet Structure Interface Control Document (PSICD)	SCI-PT-ICD-7527 Issue 2.0
AD4	Herschel/Planck Instrument Data Rates	H-P-1-ASPI-TN-0204 Issue: 1

1.3.2 Reference Documents

Document Reference	Name	Number/version
RD1	DPU/ICU Spacecraft Interface Test Plan	
RD2	SPIRE Data ICD	SPIRE-RAL-PRJ-001078
RD3	DRCU/DPU ICD	SPIRE-SAP-PRJ-001324
RD4	Virtual Machine Compiler and Simulator	
RD5	MCU Command List	
RD6	DPU-BSW Software Requirement Document	DPU-SQ-CGS-001
RD7	Switch-on Procedure TM Packets User Manual	DPU-MA-CGS-004
RD8	VIRTUOSO User Guide	
RD9	ADSP-21000 Family C Tools Manual	

1.4 Document Change Record

Issue	Revision	Date	Reason for Change
1	1	26/11/2004	Updated list of event packets
2	0	19/04/2005	Updated to go with OBS 2.0.C. • changed ID of VMSTAT HK commands • updated HK Packet IDs

2 OBS Compilation

This section describes the basic components that must be available to compile the OBS and the procedure to do it.

2.1 External Components

In order to be able to recompile the OBS two components must be installed ona Windows machine:

- ADSP- C Compiler and Tools (see RD9)
- VIRTUOSO Real-Time Software Development Tool (see RD8).

2.2 The VIRTUOSO Project File

The **spire.vpf** file contains the settings of the VIRTUOSO services that are used in the OBS. It can either be edited under VIRTUOSO, or with any text editor. This is where objects like Tasks, Semaphores, FIFO services, Events, Timers are defined. Refer to RD8 for a detailed description of the various services used. Here is the current content of the project file for version 1.2.J of the SPIRE OBS, that is part of the OBS distribution.

```
% Virtuoso Project File C:\Virtuoso\ADI21020\Rev33\Sigma\MyProj\spire.N1x\SPIRE.vpf
% Generated by Sysgen Backend version 4.1 R2.03 on Wed Nov 13 11:10:39 2002
% GLOBALPAR NAME        PARVALUE
% ===================================
  GLOBALPAR TICKFREQ    1000
  GLOBALPAR DATALEN     16384
  GLOBALPAR CEILING_PRIO 4
  GLOBALPAR KERNEL_PRIO  0
  GLOBALPAR DRIVER_PRIO  0

% NLIFILE NLIFILEPATH
% ===================================
  NLIFILE 'spire.nli'

% NODE NAME       NLINAME     NDPACKS NCPACKS NTIMERS KSTACK MONITSIZE MONITMASK
% ============================================================================
  NODE NODE1      NODE1         16        100      20     256      512         0

% DRIVERTYPE     NODE       CALL
% ============================================================
  TIMERDRIVER    NODE1      'timer_driver (TMZHI)       '
  USERDRIVER     NODE1      'StackOverFlowChecker(NULL)     '

% NETLINK NODE1       CALL1                    DIR NODE2      CALL2
% ============================================================

% TASKGROUP NAME       STARTUP
% ==============================
  TASKGROUP EXE           1
  TASKGROUP SYS           1
```

```
  TASKGROUP FPU            0
  TASKGROUP EXE_NOBOOT      0
  TASKGROUP VM_GROUP        1
  TASKGROUP HK_GROUP        1

% TASK NAME       NODE          PRIO      ENTRY        STACK   GROUPS
% ============================================================
  TASK INIT            NODE1       4       init           2048   [EXE]
  TASK TIME_TASK       NODE1       4       time_tsk       1024   [EXE]
  TASK TMTC            NODE1       5       tmtc           2048   [EXE]
  TASK VM_0            NODE1       5       vm_0           1024   [EXE_NOBOOT]
  TASK VM_1            NODE1       5       vm_1           1024   [VM_GROUP]
  TASK VM_2            NODE1       5       vm_2           1024   [VM_GROUP]
  TASK VM_3            NODE1       5       vm_3           1024   [VM_GROUP]
  TASK VM_AFX          NODE1       5       vm_AFX         1024   [EXE_NOBOOT]
  TASK HS              NODE1       6       hs             8192   [EXE]
  TASK DBG_SEQ         NODE1       7       dbg_seq        2048   [EXE_NOBOOT]
  TASK VM_SVC          NODE1       7       vm_svc         4096   [EXE]
  TASK LS              NODE1       7       ls             8192   [EXE]
  TASK CMD_SEQ         NODE1       8       cmd_seq        4096   [EXE]
  TASK HK_ASK0         NODE1       9       hk_ask0        2048   [HK_GROUP]
  TASK HK_ASK1         NODE1       9       hk_ask1        2048   [HK_GROUP]
  TASK HK_ASK2         NODE1       9       hk_ask2        2048   [HK_GROUP]
  TASK HK_ASK3         NODE1       9       hk_ask3        2048   [HK_GROUP]
  TASK HK_MON          NODE1       9       hk_mon         2048   [EXE_NOBOOT]
  TASK AUTO_SEQ        NODE1      10        auto_seq      2048   [EXE]


% FIFO NAME                 NODE       DEPTH WIDTH
% ========================================
  FIFO TC_HP_QUEUE          NODE1        8       40
  FIFO TC_LP_QUEUE          NODE1        8       40
  FIFO EV_TM_QUEUE          NODE1       80       40
  FIFO HK_TM_QUEUE          NODE1       32       40
  FIFO SD_TM_QUEUE          NODE1      128       40
  FIFO LS_HP_QUEUE          NODE1       64       16
  FIFO LS_LP_QUEUE          NODE1     1024       16
  FIFO AUTO_HP_QUEUE        NODE1      512       40
  FIFO AUTO_LP_QUEUE        NODE1      512       40
  FIFO VM_TM_QUEUE          NODE1       64       12

% EVENT NAME                NODE       CALL
% ============================================
  EVENT ISR_1553_EVENT      NODE1      'NULL        '
  EVENT ISR_FIFO_EVENT      NODE1      'NULL        '
  EVENT ISR_TIMER_EVENT     NODE1      'NULL        '
  EVENT TS_EVENT            NODE1      'NULL        '
  EVENT HK_0_EVENT          NODE1      'NULL        '
  EVENT HK_1_EVENT          NODE1      'NULL        '
  EVENT HK_2_EVENT          NODE1      'NULL        '
  EVENT HK_3_EVENT          NODE1      'NULL        '
  EVENT LS_TC_EVENT         NODE1      'NULL        '
  EVENT LS_0_EVENT          NODE1      'NULL        '
  EVENT LS_1_EVENT          NODE1      'NULL        '
  EVENT LS_2_EVENT          NODE1      'NULL        '
  EVENT LS_3_EVENT          NODE1      'NULL        '
```

```
% MAP NAME        NODE        BLOCKS BLOCKSIZE
% ============================================

% SEMA NAME           NODE
% ============================
  SEMA HK_0_SEMA      NODE1
  SEMA HK_1_SEMA      NODE1
  SEMA HK_2_SEMA      NODE1
  SEMA HK_3_SEMA      NODE1
  SEMA LS_SEMA        NODE1
  SEMA TC_READY       NODE1
  SEMA FRAG_SEMA      NODE1
  SEMA AUTO_SEMA      NODE1
  SEMA DBG_SEMA       NODE1


% MAILBOX NAME        NODE
% ================================

% RESOURCE NAME              NODE
% ================================
  RESOURCE TIMER_RESOURCE NODE1

% POOL NAME        NODE        SIZE_SMALL SIZE_LARGE BLOCK_NUMBER
% ========================================================
```

2.3 The Architecture File

.

The **spire.ach** file contains the definition of the various segments of the DPU PM and DM. Here is the current content of the architecture file for the version 1.2.J of the SPIRE OBS, that is part of the OBS distribution. Refer to RD9 for a detailed description of the various segments and directives used in creating this file.

```
!============================================================
.system FirstDPU;
.processor = ADSP21020;
! Program Memory
!==== Interrupt table
.segment /pm /ram  /begin=0x000000 /end=0x0000FF   seg_rth;
!==== Code
.segment /pm /ram  /begin=0x004000 /end=0x004FFF   seg_init;
.segment /pm /ram  /begin=0x005000 /end=0x07FFFF   seg_pmco;
! Data Memory
.segment /dm /ram  /begin=0x00000000 /end=0x0004FFFF         seg_dmda;
.segment /dm /ram  /begin=0x00050000 /end=0x000503FF /cstack  seg_stak;
.segment /dm /ram  /begin=0x00050400 /end=0x0007FFFF /cheap   heap1;
!
.segment /dm /ram  /begin=0x40000000 /end=0x400FFFFF       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 mot mounted
.bank /pm0 /wtstates=0 /wtmode=internal /begin=0x000000;
!
! 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=0 /wtmode=internal /begin=0x00000000;
.bank /dm1 /wtstates=1 /wtmode=both /begin=0x20000000;
.bank /dm2 /wtstates=0 /wtmode=internal /begin=0x40000000;
.bank /dm3 /wtstates=1 /wtmode=both /begin=0x80000000;
!============================================================
.endsys;
!***** end of file *****
```

2.4 Compiling the OBS

The OBS distribution contains a **makefile** that manages the compilation and linking of the source code. Typing make on the command line will compile all source files that have been updated with respect to previous compilation, or that depend on include files that have been modified; make rebuild will recompile all C and Assembler source code files.
Any compilation subsequent to a modification of the VIRTUOSO Project File (e.g. after adding another semaphore) will need a valid VIRTUOSO license.

2.5 The Compilation Products

The compilation will produce many intermediate files. The .o object files whose name starts with the suffix MIL should never be deleted since they contain the compiled MIL-1553B-STD drivers whose source codes are not included in the OBS delivery.
The most important compilation product is obviously the **SPIRE.EXE** that will contain the executable code.
Another useful ouput file is the memory map file that documents the actual DPU memory usage by the OBS. Here is an extract from the **SPIRE.MAP** file contained in the OBS distribution and valid for the version 1.2.J of the SPIRE OBS.

```
Analog Devices ADSP-210x0 Linker          spire.map          Page 1
Release 3.3, Version 2.21                 Thu Aug 19 10:26:32 2004
Copyright (c) 1991-1996 Analog Devices, Inc.

Architecture Description: FirstDPU

Segment     Start       End         Length    Memory Type       Attribute

seg_rth     000000      0000ff      256       Program Memory    RAM
seg_dmda    00000000    0004ffff    327680    Data Memory       RAM
seg_init    004000      004fff      4096      Program Memory    RAM
seg_pmco    005000      07ffff      503808    Program Memory    RAM
seg_stak    00050000    000503ff    1024      Data Memory       RAM
```

heap1	00050400	0007ffff	195584	Data Memory	RAM
1355_IF	40000000	400fffff	1048576	Data Memory	RAM
EEPROM	80000000	8003ffff	262144	Data Memory	RAM
Timer	81000000	81ffffff	16777216	Data Memory	PORT
watchdog	82000000	82ffffff	16777216	Data Memory	PORT
Int_mng	83000000	83ffffff	16777216	Data Memory	PORT
SMCS_reg	84000000	84ffffff	16777216	Data Memory	RAM
Bus_IF	88000000	8fffffff	134217728	Data Memory	RAM

Memory Usage (Actual):

Segment	Start	End	Length	Memory Type	Attribute
seg_rth	000000	0000ff	256	Program Memory	RAM
seg_init	004000	00400e	15	Program Memory	RAM
seg_pmco	005000	01151e	50463	Program Memory	RAM
seg_dmda	00000000	0004abb4	306101	Data Memory	RAM
seg_stak	00050000	0005000a	11	Data Memory	RAM
heap1	********	********	0	Data Memory	RAM
1355_IF	********	********	0	Data Memory	RAM
EEPROM	********	********	0	Data Memory	RAM
Timer	********	********	0	Data Memory	PORT
watchdog	********	********	0	Data Memory	PORT
Int_mng	********	********	0	Data Memory	PORT
SMCS_reg	********	********	0	Data Memory	RAM
Bus_IF	********	********	0	Data Memory	RAM

Memory Usage Summaries:

Memory Type	Attribute	Total
Program Memory	ROM	0
Program Memory	RAM	50734
Program Memory	PORT	0
Data Memory	ROM	0
Data Memory	RAM	306112
Data Memory	PORT	0

3 OBS Objects

3.1 Memory Pools

Due to incorrect Virtuoso's Memory Pools behaviour, DPU Memory Pools is now managed by our internal handling procedures. The memory areas in which memory blocks used to store packets are now statically placed in data memory. Pools IDs are specified as follows:

Pool_ID	Data type
1	TC packets
2	Event TM packets
3	TC verification report TM packets
4	HK TM packets

5	Science Data TM packets
6	Execution Report TM Packets

Table 3-1 Memory Pool ID definition

3.2 OBS Tasks

The OBS is structured in a series of Virtuoso Tasks. Task IDs are specified as follows:

Task_ID	Task Name
0x0	(Reserved for debugging)
0x1	INIT
0x2	TIME_TASK
0x3	TMTC
0x4	VM_0
0x5	VM_1
0x6	VM_2
0x7	VM_3
0x8	VM_AFX
0x9	HS
0xA	(Reserved for debugging)
0xB	VM_SVC
0xC	LS
0xD	CMD_SEQ
0xE	HK_ASK0
0xF	HK_ASK1
0x10	HK_ASK2
0x11	HK_ASK3
0x12	HK_MON
0x13	ERRAUTO_SEQ
0x14	TABLER
0x15	(Reserved for debugging)

Table 3-2 OBS Task ID definition

3.3 FIFOs

Virtuoso FIFOs are message queues used to exchange information between different OBS Tasks. FIFO IDs are specified as follows:

FIFO_ID	FIFO Name
0x0	TC_HP_QUEUE
0x1	TC_LP_QUEUE
0x2	EV_TM_QUEUE
0x3	HK_TM_QUEUE
0x4	SD_TM_QUEUE
0x5	LS_HP_QUEUE
0x6	LS_LP_QUEUE
0x7	ERR_HP_QUEUE
0x8	ERR_LP_QUEUE
0x9	VM_TM_QUEUE

Table 3-3 Virtuoso FIFO ID definition

4 DPU Memory Map

The OBS organizes the DPU memory as specified in the architecture file **spire.ach**, summarized in the table below:

Memory Type	Start Address	End Address	Segment Description
Program Memory	0x000000	0x0000FF	Used to store the Interrupt Vector Table
	0x000100	0x003FFF	BootUp OBS Loader Program
	0x004000	0x004FFF	Static Variable Inititalizer Storing Area
	0x005000	0x07FFFF	Used to store the OBS executable at runtime. It is currently sized to the whole PM.
Data Memory	0x00000000	0x0004FFFF	Used for static data allocation, e.g. all variables declared in the OBS routines.
	0x00050000	0x000503FF	Used as stack space for the idle task and interrupt service routines.
	0x00050400	0x0007FFFF	This is the heap. The VIRTUOSO task and fifoes are resident here.

Table 4-1 DPU Memory Map

5 OBS Loading on the DPU

When the DPU is switched on, the BSW is copied from PROM to PM and run. The details of the boot procedure can be found elsewhere (see RD6); here we simply note that after all the tests are carried out, a (5,2) event is generated and the boot enters an infinite loop waiting for a TC. The contents of the generated event are described in RD7; the last word in the packet contains the number of errors found in the memory checks, and should be 0. At this point there are two modes of loading and executing the OBS: using the image resident on the EEPROM on-board, or loading a new image via standard TCs.

5.1 Running the EEPROM-resident OBS

The OBS is resident in EEPROM. Once the (5,2) event (with no errors reported) is received by the CDMS simulator, the command "Force boot" described in RD2 can be sent to the DPU; the BSW will copy the OBS from EEPROM to PM, jump at the start location of the OBS in the PM, and the OBS will start running. If the DPU is connected to the CDMS simulator or SCOS2000, HK packets will be received (SIDs 0x300 and 0x301). This can be considered as the confirmation that the startup procedure has been successfully completed.

5.2 Loading the OBS via Telecommands

Once the BSW puts the DPU in a wait state, it is possible to uplink from SCOS2000 a new image of the OBS using standard TCs. The C program **TCGen** provided by Gavazzi is available under Windows to translate the OBS image SPIRE.EXE into a list of TC (6,2) ready to be sent to the DPU. The ADI21020 C Compiler must also be installed, since TCGen uses some C-tools (like cdump). The command to invoke the procedure is:

>tcgen –i ***segfile.txt*** –p ***pagefile.txt*** –f ***SPIRE.EXE*** –a 0x500 –o ***path/suffix*** –m 0

the segfile file contains the list of memory segments (one per line) defined in the program memory of the DPU and reported in the architecture file (spire.ach); typically the segments are seg_rth, seg_init and seg_pmco.
The pagefile file contains the list of memory pages to be avoided (it can be empty).
OBS.EXE is the executable file as produced by the compilation of the OBS code.
Path is the directory where the output TCs will be stored and suffix is a string that will be attached to the TC file names: the ouput files will be named *suffix*TCnnnnn.dm where nnnnn is a count number.

Once the set of TCs containing the image of the OBS have been produced, they can be uplinked using the "**ObswLoader**" script. The script loads TCs from a local directory and sends them to the CDMS that, in turn, sends them to the DPU. The following syntax should be used to invoke the script.

> ObswLoader –dpu –apid 1280 –interval ***XXX path***/*

where ***path*** is the directory that hosts the telecommands prepared with the TCGen program, and XXX is the interval in milliseconds for the dispatch of subsequent TCs to the CDMS. Clearly, the dispatching interval should match the capabilities of the buslist currently running on the CDMS. For fast uploads a dedicated buslist has been prepared that allows the CDMS to send to the DPU a maximum of 20 TC/s; using this buslist allows to invoke the ObswLoader script with an interval parameter of 50 (milliseconds). If one uses the nominal buslist where only 2 TC/s can be uplinked, then the interval parameter should be set to 500.

Once all TCs have been sent, it will be necessary to send the "Load TC and boot" TC (see RD2) from SCOS2000 to command the BSW to copy the full image from DM to PM and start the application program. If large areas of DM are damaged so that there is not enough space to store the image before copying it in PM, it is possible to upload a subset of TC. After any subset has been uploaded, the command to send is "Load TC and wait": when the BSW receives this command, this part of the image is copied in PM but the application program is not started. The BSW waits for the next subset and so on. When the last subset of memory packets is received, by sending the command "Load TC and boot" the DPU copies this last piece of code and then starts the execution of the application software. This command has not been tested so far and it should not be used.
It is also possible to restart the Boot Software, and thus reload the EEPROM-stored OBS or up-link the OBS via TCs, without switching off and on again the DPU: this can be done while the OBS is running by sending the "Call Boot" telecommand from SCOS2000.

6 Runtime Instructions

6.1 Telecommand Verification

The generation of the telecommand verification packets TM (1,3) (Execution Start), TM (1,5) (Execution Progress) and TM (1,7) (Execution End) are controlled by the Ack bits in the TC sent to the DPU, as specified in AD3 (§3.1). The TM (1,1) (successful TC verification), TM (1,2) (unsuccessful TC verification) and TM (1,8) (Execution Failure) are issued by the OBS irrespectively of the TC Ack bits. The actual dispatching of these TM packets to the CDMS can be inhibited using service 14 (Packet Transmission Control) as specified in AD3.
Error codes contained in TM (1,2) are listed in AD3, while error codes in TM (1,8) are reported in RD2.

6.2 Housekeeping Data Reporting

The OBS only generates HK packets TM (3,25). No Diagnostic packets are generated. The HK packet definition is stored in tables in the OBS. Four independent HK packets can be generated simultaneously, each with its own sampling interval.

HK Packet ID	Packet
300	Essential HK Packet
301	Nominal HK Packet. A subset of the parameters contained in this packet will be used for monitoring.
302	Free
303	Free

Table 6-1 List of allowed HK packets

The OBS does not perform any check on the DPU workload implied by the HK parameters collection. In particular, it should be remembered that the minimum time to issue a HK parameter request to the DRCU and receive the correspondent parameter is 2 milliseconds. This means that nominally the cumulative number of DRCU parameters requested for the various HK packets should not exceed 500/sec to avoid losing data. In reality the number should be kept below that limit because the OBS will likely be performing other tasks requiring communication to the subsystems at the same time.

6.2.1 Situation at Start-Up

At OBS start-up two types of HK packets are generated by default: the critical HK packet and the nominal HK packet. Both packets are TM (3,25) and the header only differs for the APID (0x0500 and 0x0502 respectively) and for the SID (0x0300 and 0x0301 respectively). The two packets are issued every 2s and every 1s respectively. The definition of the two packets is loaded on OBS initialisation and complies with requirements contained in RD2.

6.2.2 Modifying the HK Packet Properties

6.2.2.1 Sampling Interval

The sampling interval of an HK packet can be modified via a TC (8,4,0xCC-01), inserting the required interval in milliseconds in the proper TC field as specified in RD2. the new sampling interval is applied at the start of HK sampling cycle immediately following TC (8,4, 0xCC-01) reception. This means that if the current sampling interval of an HK packet is 10 seconds and a TC (8,4, 0xCC-01) with a 1 second sampling interval is received 2 seconds after the last HK packet has been issued, the 1-second HK packets will start to be sent after about 8 seconds from TC reception.

6.2.2.2 HK Parameters

The contents of the HK packets are defined by on-board tables that contain the list of DRCU 32-bits command words needed to get those parameters. The order in which the commands are stored in the HK definition tables defines the order in which the HK parameters are stored in the HK packets. To modify the contents of an HK packet, the first thing to do is then to uplink a new table (with its own ID number) containing the list of 32-bits commands needed to get the required HK parameters. The sequence of actions is then the following:

a) Load a new HK definition Table. The mechanism to do this will be explained when dealing with Tables management.
b) Stop HK acquisition using a TC (8,4, 0xCC-02) with the required HK Packet ID as specified in RD2.
c) Restart HK acquisition using a TC (8,4, 0xCC-01) with the required HK Packet ID, the Table ID of the table uplinked in a) and the required sampling interval in milliseconds.

Warning: since the Nominal HK packet (ID 0x301) will be used for monitoring purposes, stopping HK Packet ID 0x301 will also stop the monitoring task. Besides, when redefining the table to be used for Nominal HK packet, particular care must applied in making sure that no parameter used by the monitoring task is removed from the HK packet definition.

Note: A TC (8,4, 0xCC-01) with a Table ID different from the one currently in use for that HK Packet ID must be preceeded by a TC (8,40xCC-02), or a TM (1,8) with code 0x0827 (RD2) will be issued.

The list of commands for the DRCU is reported in RD3 and RD5. The commands to get DPU internal HK parameters are built according to the same structure (see AD2) so that the HK acquisition task can handle both types of HK requests. The list of available commands to get DPU HK parameters is the following:

Command ID	Function	Bits	Content
0x20010000	Get Observation ID	32	
0x20020000	Get Building Block ID	32	
0x10030000	Get Observing Mode	16	
0x10040000	Get Observation Step	16	
0x30050000	Get time of last DRCU timer reset	48	
0x30060000	Get Last Time Stamp	48	
0x30070000	Get absolute time drift	48	Time difference between last CDMS Time Stamp and OBS internal clock
0x30080000	Get Time of Start HK0 parameters collection	48	
0x30090000	Get Time of Start HK1 parameters collection	48	
0x300A0000	Get Time of Start HK2 parameters collection	48	
0x300B0000	Get Time of Start HK3 parameters collection	48	
0x100D0000	Get number of received TC	16	
0x100E0000	Get sequence number of last received TC	16	
0x100F0000	Get number of executed TC	16	
0x10100000	Get sequence number of last executed TC	16	
0x20110000	Get observation length in seconds	32	
0x20120000	Get data rate of current observation in byte/s	32	
0x30130000	Get Memory check info (NYI)	48	*
0x04140000	Get S/S I/F monitoring flags	16	See Note 1
0x10150000	Get DCU flags (NYI)	16	*
0x10160000	Get SCU flags (NYI)	16	*
0x10170000	Get MCU flags (NYI)	16	*
0x04180000	Get number of packet sent under Apid1	16	
0x04190000	Get number of packet sent under Apid2	16	
0x041A0000	Get number of packet sent under Apid3	16	
0x041B0000	Get number of packet sent under Apid4	16	
0x041C0000	Get number of packet sent under Apid5	16	
0x10200000	(RESERVED)		
0x04210000	Get DPU 5V	16	
0x04220000	Get DPU 15V	16	
0x04230000	Get DPU –15V	16	
0x04240000	Get DPU temperature	16	
0x10250000	(RESERVED)		
0x10260000	(RESERVED)		
0x04270000	Get DPU 2.5V	16	
0x04280000	Get DPU processor workload	16	units per thousand

0x08290000	Get LS internal queue Workload	32	Single LS command
0x102A0000	Get DCU fifo frame counter	16	
0x102B0000	Get MCU fifo frame counter	16	
0x102C0000	Get SCU fifo frame counter	16	
0x102D0000	Get Event status (NYI)	16	*
0x102E0000	Get Tm Mode	16	0=Nominal / 1= Burst
0x102F0000	Get LS channel duty time	32	In microsecond
0x04300000	Get Hard VM 0 status	16	VM code Table ID
0x04310000	Get Soft VM 1 status	16	VM code Table ID
0x04320000	Get Soft VM 2 status	16	VM code Table ID
0x04330000	Get Soft VM 3 status	16	VM code Table ID
0x04340000	Get Soft VM AFX status	16	VM code Table ID
0x10350000	Get FIFO DataFlow Flags	16	Only the 3 LSB are relevant (1 per FIFO); they are 0 if data are not found on FIFO when expected
0x103C0000	GetSCPoolStat	16	N. of slots available
0x103D0000	GetHKPoolStat	16	Ibidem
0x103E0000	GetEVPoolStat	16	Ibidem
0x103F0000	GetRPPoolStat	16	Ibidem
0x04400000	Get_HUP_Task_State	16	RUNNING 0x0000 STOPPED 0x0001 ABORTED 0x0003 SUSPEND 0x0004 SLEEPING 0x0010 EVENT_W 0x0080 FIFO_W 0x0200 SEMA_W 0x1000 Unknown 0xFFFF
0x04410000	Get_INIT_Task_State	16	Ibidem
0x04420000	Get_TIME_Task_State	16	Ibidem
0x04430000	Get_TMTC_Task_State	16	Ibidem
0x04440000	Get_VM_0_Task_State	16	Ibidem
0x04450000	Get_VM_1_Task_State	16	Ibidem
0x04460000	Get_VM_2_Task_State	16	Ibidem
0x04470000	Get_VM_3_Task_State	16	Ibidem
0x04480000	Get_VM_AFX_Task_State	16	Ibidem
0x04490000	Get_HS_Task_State	16	Ibidem
0x044A0000	Get_DBG_SEQ_Task_State	16	Ibidem
0x044B0000	Get_VM_SVC_Task_State	16	Ibidem
0x044C0000	Get_LS_Task_State	16	Ibidem
0x044D0000	Get_CMD_SEQ_Task_State	16	Ibidem
0x044E0000	Get_HK_ASK0_Task_State	16	Ibidem
0x044F0000	Get_HK_ASK1_Task_State	16	Ibidem
0x04500000	Get_HK_ASK2_Task_State	16	Ibidem
0x04510000	Get_HK_ASK3_Task_State	16	Ibidem
0x04520000	Get_HK_MON_Task_State	16	Ibidem
0x04530000	Get_AUTO_SEQ_Task_State	16	Ibidem
0x04540000	Get_TABLER_Task_State	16	Ibidem
0x04550000	Get_TUP_Task_State	16	Ibidem
0x10580000	Get number of unallocated memory blocks for TeleCommand packets	16	
0x10590000	Get number of unallocated memory blocks for Event TM packets	16	
0x105A0000	Get number of unallocated memory blocks for HouseKeeping TM packets	16	

0x105B0000	Get number of unallocated memory blocks for Science TM packets	16	
0x105C0000	Reserved	16	
0x10600000	Get_DPU_Stat	16	OBS_Version
0x10610000	GetTmMode	16	Nominal/BurstMode
0x10620000	Get number of failed enqueues on Virtuoso FIFO Queue for High Priority TC	16	
0x10630000	Get number of failed enqueues on Virtuoso FIFO Queue for Low Priority TC	16	
0x10640000	Get number of failed enqueues on Virtuoso FIFO Queue for Event TM packets	16	
0x10650000	Get number of failed enqueues on Virtuoso FIFO Queue for HK TM packets	16	
0x10660000	Get number of failed enqueues on Virtuoso FIFO Queue for Science TM packets	16	
0x10670000	Get number of failed enqueues on Virtuoso FIFO Queue for Report TM packets	16	
0x10680000	Get number of failed enqueues on Virtuoso FIFO Queue for High Priority LS Commands	16	
0x10690000	Get number of failed enqueues on Virtuoso FIFO Queue for Low Priority LS Commands	16	
0x106A0000	Get number of failed enqueues on Virtuoso FIFO Queue for High Priority calls to Autonomy Function	16	
0x106B0000	Get number of failed enqueues on Virtuoso FIFO Queue for Low Priority calls to Autonomy Function	16	
0x106C0000	Get number of failed enqueues on Virtuoso FIFO Queue used for internal VM calls to OBS	16	
0x106D0000	Get number of failed enqueues on Virtuoso FIFO Queue for calls to Memory Dump procedure	16	
0x04700000	Get ID of the selection Table for Frame ID 0x0	16	no_selection=0xFFFF
0x04710000	Get ID of the selection Table for Frame ID 0x1	16	no_selection=0xFFFF
0x04720000	Get ID of the selection Table for Frame ID 0x2	16	no_selection=0xFFFF
0x04730000	Get ID of the selection Table for Frame ID 0x3	16	no_selection=0xFFFF
0x04740000	Get ID of the selection Table for Frame ID 0x4	16	no_selection=0xFFFF
0x04750000	Get ID of the selection Table for Frame ID 0x5	16	no_selection=0xFFFF
0x04760000	Get ID of the selection Table for Frame ID 0x6	16	no_selection=0xFFFF
0x04770000	Get ID of the selection Table for Frame ID 0x7	16	no_selection=0xFFFF
0x04780000	Get ID of the selection Table for Frame ID 0x8	16	no_selection=0xFFFF
0x04790000	Get ID of the selection Table for Frame ID 0x9	16	no_selection=0xFFFF
0x047A0000	Get ID of the selection Table for Frame ID 0xA	16	no_selection=0xFFFF
0x047B0000	Get ID of the selection Table for Frame ID 0xB	16	no_selection=0xFFFF
0x047C0000	Get ID of the selection Table for Frame ID 0xC	16	no_selection=0xFFFF
0x047D0000	Get ID of the selection Table for Frame ID 0xD	16	no_selection=0xFFFF
0x047E0000	Get ID of the selection Table for Frame ID 0xE	16	no_selection=0xFFFF
0x047F0000	Get ID of the selection Table for Frame ID 0xF	16	no_selection=0xFFFF
0x04800000	Get ID of the selection Table for Frame ID 0x10	16	no_selection=0xFFFF
0x04810000	Get ID of the selection Table for Frame ID 0x11	16	no_selection=0xFFFF
0x04820000	Get ID of the selection Table for Frame ID 0x12	16	no_selection=0xFFFF
0x04830000	Get ID of the selection Table for Frame ID 0x13	16	no_selection=0xFFFF
0x04840000	Get ID of the selection Table for Frame ID 0x14	16	no_selection=0xFFFF
0x04850000	Get ID of the selection Table for Frame ID 0x15	16	no_selection=0xFFFF
0x04900000	Get ID of the selection Table for Frame ID 0x20	16	no_selection=0xFFFF
0x04910000	Get ID of the selection Table for Frame ID 0x21	16	no_selection=0xFFFF
0x10FF0000	Dummy place holder	16	* Always 0 (zero)

Table 6-2 Commands to get DPU HK parameters

Notes:

1 This parameter reports the status of SubSystem Interfaces condensed in one 16-bit word. The bit coding is as follows (first bit is number 1 starting from LSB) :
1-2: DCU Low-Speed I/F status: "00"→ ALIVE, "01" → SICK, "10" → DEAD
3-4: DCU High-Speed I/F mode: "00" → NOMINAL, "01" → TRANSPARENT
5-6: MCU I/F status: "00"→ ALIVE, "01" → SICK, "10" → DEAD
7-8: MCU High-Speed I/F mode: "00" → NOMINAL, "01" → TRANSPARENT
9-10: SCU I/F status: "00"→ ALIVE, "01" → SICK, "10" → DEAD
11-12: SCU High-Speed I/F mode: "00" → NOMINAL, "01" → TRANSPARENT

6.3 Memory Management

6.3.1 Absolute Addressing

Loading and dumping of memory areas using absolute addresses can be performed using the dedicated TCs of Service 6 as described in AD3 and RD2. The Start_Address parameter in the TC (6,2) is a relative address for each allowed memory area identified by the Memory_ID parameter. The allowed Memory IDs are listed in RD2.

6.3.2 Table Management

All HK packet definitions and VM codes needed to perform the SPIRE observations (see AD2 for a description of the concept) are stored on-board as Tables. Each table is characterised by an ID and a length in 32-bits words. The absolute memory addresses of all on-board tables are managed by the OBS and are not available to the user. The TCs to load and delete on-board tables are described in RD2. Here we describe how to use those TCs.

Warning: No action on tables is allowed during execution of command lists on HW VM, to avoid memory changes during VM execution. This because a VM code can call subroutines residing in other tables and it is not possible to predict which tables will be in use during execution of a complex VM code.

6.3.2.1 Table Load

The sequence to load a new table is the following:

a. Send a TC (8,4, 0x01-0x01) specifying the Table ID and the length in 32-bits words of the Table. **Warnings**:
   a.1 if the specified Table ID exists, the table is deleted. The only exception is if the table is in use (by an HK-collection task or VM), in which case a TM (1,8) is issued with a *Busy_table* error code.
   a.2 A table ***cannot*** be longer than 0x2000 words.
b. Send a TC (8,4, 0x01-0x03) containing the list of 32-bits words. Since the TC holds 16-bits words, each 32-bits word will have to be split in two, with the MSBs preceeding the

LSBs. The number of the 32-bits words contained in the TC must not exceed the length specified in afor that Table ID, or a *Bad_NData* TM (1,8) will be generated.

6.3.2.2 Table Update

To update an existing table it is sufficient to send a TC (8,4, 0x01-0x03) as specified in bof 6.3.2.1.

6.3.2.3 Table Delete

To delete an existing table it is sufficient to send a TC (8,4, 0x01-0x01) specifying the Table ID and setting the length to 0.

Note: if the table is in use (HK packet or VM code) a TM (1,8) will be issued.

6.3.3 Table Defragmentation

Tables are stored in a dedicated DM area. After a while the continuous creation, update and deletion of tables may lead to an excessive memory fragmentation within that area. This may result in the inability to create new tables even when enough space is available but it is not contiguous. The OBS defragments the DM either via a dedicated TC (8,4, 0x01-0x04), or upon reception of a *Set_Table* TC (8,4, 0x01-0x01) when it realizes that the required memory space is available only if DM is defragmented.

6.3.4 Default Tables in the OBS

This is the list of predefined tables available in the OBS at start-up.

Table ID	Contents
0	Critical HouseKeeping TM packet definition
1	Nominal HouseKeeping TM packet definition
5	Monitoring Table definition
6	Observing Mode value
80	Table to temporarily hold VM code uplinked via the *Execute_Command_List* TC
212	VM Procedure to go into DPU SAFE MODE
248	DCU Command Inhibition Table
249	MCU Command Inhibition Table
250	SCU Command Inhibition Table
255	MOAT

6.4 Virtual Machine Programs

VM programs are stored in tables in a dedicated DM area. RD4 describes how to write and compile a VM program using a GUI available under windows. The GUI is able to produce the executable VM code already organised in TC (8,4, 0x01-0x03) ready to be sent to the DPU.

6.5 Monitoring of HouseKeeping Parameters

6.5.1 Basics

The OBS can monitor S/S and DPU housekeeping parameters against soft and hard limits. The monitored item MUST be contained in the nominal HK TM packet, and it can also be extracted (using a bit mask & shift) from any HK parameter. The monitored item will be checked at the same frequency of the nominal HK parameter collection. The value checking for and HK parameter can be configured to be dependent on the NOMINAL value of up to 16 other HK parameters (i.e. if one of independent HK parameters is out of limits then the limit check on the dependent HK parameter is not done). On turn, each of the independent parameters can be dependent on another set of parameters.

Two types of parameters are supported by the monitoring system: ANALOG parameters that can take a continuous range of values, and DIGITAL parameters that can only assume a discrete number of values.

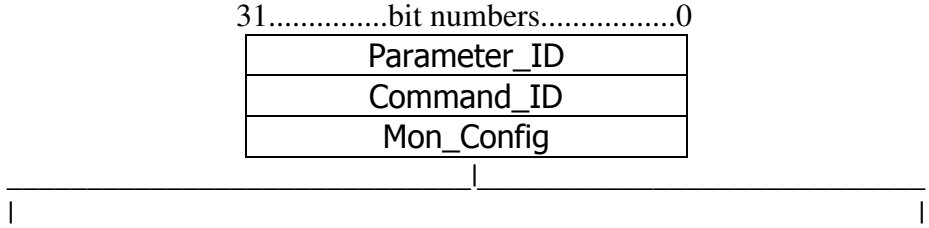
The OBS monitoring system supports three possible states for each ANALOG parameter: NOMINAL (parameter within limits), WARNING (parameter out of soft limits), FAILED (parameter out of hard limits). In case of a DIGITAL parameter, there is no WARNING state.

The state transitions (excluding those back to NOMINAL) are triggered after the offending conditions is realised for N consecutive times. The value of RETRY_LIMT can be independently set for each transition of each monitored parameter. The system can be configured to react to a transition between any two of these states (6 combinations in total). A separate behaviour can be configured for each transition.

As a default condition, the monitoring system IS NOT active at startup of the OBS. It must be explicitely activated by Telecommand (this can easily be changed and requires to recompile the code).

6.5.2 How to Configure the Monitoring System

The configuration settings for the monitoring system must be stored in any On-Board Table (see later to learn how to tell the OBS to use this table for monitoring). The definition of each monitored parameter with its configuration settings, has the precise structure defined below where each record is a 32-bits word.



Analog Parameter		Digital Parameter	
Fail_High_Limit		N_Fail_Values	
Warn_High_Limit		Fail_Val1	Fail_Val2
Warn_Low_Limit		Fail_Val3	Fail_Val4
Fail_Low_Limit		Fail_Val5	Fail_Val6
Action_NW		Fail_Val7	Fail_Val8
Action_WN		Fail_Val9	Fail_Val10
Action_NF		Fail_Val11	Fail_Val12
Action_FN		Fail_Val13	Fail_Val14
Action_WF		Fail_Val15	Fail_Val16
Action_FW		Action_NF	
Reserved		Action_FN	
Reserved		Reserved	

Reserved
N_Dep
Dep 1
...
Dep N (up to N=16)

The above structure must be replicated for each monitoring item. Finally we will end up with long column of 32-bits words that can be regularly loaded in any of the SPIRE On-Board Tables using an Update_Table TC (see §6.3.2.1). Here follows the detailed explanations of each field in the above structure

- **Parameter_ID**. It is a unique identifier for the monitoring item.
- **Command_ID**. This is the complete 32-bit command word used to obtain the desired HK parameter to be monitored. The defintions given in RD3 and RD5 should be used for S/S parameters, while the definitions given in §6.2.2.2 should be used for DPU HK parameters.
- **Mon_Config**.
  - Bit 31 → Enable/Disable (1/0) monitoring of this parameter
  - Bit 30 → Analog/Digital parameter (1/0)
  - Bits 22-29 → Retry Limit for transition sensing
  - Bits 16-21 → Number of bits the parameter has to be bitwise right-shifted
  - Bits 0-15 → Extraction mask to be applied to the parameter
- **Xxx_yyy_Limits**. Soft and Hard limits for analog parameters
- **N_Fail_Values**. Number of discrete values for which a digital parameter should be considered in a FAIL state

- **Fail_Valx**. Series of 16-bit values (up to 16) for which a digital parameter should be considered in a FAIL state. Only the first N_Fail_Values shall be considered.
- **Action_XY**. Configuration settings for the specified parameter transition:
  - Bit 31 → The specified action is Enabled/Disabled (1/0)
  - Bit 30 → The generation of a TM (5,1) event in the occurrence of a state transition is Enabled/Disabled (1/0)
  - Bit 28-29 → The execution of an Autonomy Function (i.e. a VM program) for this state transition is Enabled/Disabled (11/00)
  - Bit 26-27 → Reserved (set to 0)
  - Bit 25 → Autonomy Action VM program is to be run on the Hard_VM (1) or on the AFx dedicated Soft_VM (0)
  - Bit 24 → Reserved (set to 0)
  - Bits 16-23 → Table ID of the VM program to be executed
  - Bits 0-15 → Parameter to be passed to the VM program
- **N_Dep**. Number of dependencies; it is the number of other HK parameters that all must have NOMINAL values for the present monitoring item to be checked.
- **Dep_1 ... N**. Parameter IDs (see above) of the dependencies. Important warning: a parameter listed in the dependency list of another parameter, MUST itself be present as a monitoring item in the monitoring table BEFORE the location of the dependent parameter.

6.5.3 How to Use the Monitoring System

a. The monitoring system is started by sending the Start_Monitoring TC with the Table ID of the monitoring table as a parameter. If a TM(1,8) error is received in response, it means the monitoring system has not started.
b. All the monitored parameters must be present in the nominal HK TM packet definition, or a TM(1,8) is generated
c. All the dependencies must be defined as monitored parameters themselves, and positioned in a higher location of the monitoring table, or a TM(1,8) is generated
d. All the tables referred to as VM progam autonomy actions in the proper fields of the Action_XY records, must be already existing or a TM(1,8) is generated
e. In order for the monitoring system to start, the nominal HK collection MUST be active, or a TM(1,8) is generated
f. Given the complicated set of dependencies among monitored parameters, the monitoring table cannot be updated if the monitoring system is running. To update the monitoring table first send a Stop_Monitoring TC, then update the table, and finally re-send the Start_Monitoring TC.
g. To tell the monitoring system that another monitoring table should be used, the system must be first stopped with the Stop_Monitoring TC and then restarted with the Start_Monitoring TC and the new Table ID as a parameter.
h. The monitoring system suspends itself when an autonomy function is running.
i. You can have nested dependencies as long as the correct order is followed: the deeper dependency should be listed first as a monitored parameter.
j. If an autonomy function VM program is started on the Hard_VM (setting bit 25 =1 in the Action_XY field), it suspends (without any possibility of resuming) any other program that may be currently running on the Hard_VM (i.e. a measurement).
k. An autonomy function VM program started on the AFx Soft_VM (setting bit 25 =0 in the Action_XY field), does not interfere with another program that may be currently running

on the Hard_VM. This mode may be suited when the currently on-going measurement can be carried out also in presence of a mildly severe failure.

7 Event Reporting

Warning event TM (5,x) packets are issued by the OBS in several occasions. Here follows a table of warning/exception conditions so far identified and that result in the generation of a TM (5,x) packet.

TM Packet	Event Code	SID	Event Name	Explanation	Returned Parameters
(5,1)	0x0501	0x5100	REPORT_STEP	Indicates a new step in the current operation Mode. This event is issued every time the MODE or STEP Number is changed	• Current Mode • Current Step Number
(5,1)	0x0504	0x5101	REPORT_PEAKUP	Final report from Peak-Up procedure	• Offset in CHOP and JIGGLE directions
(5,1)	0x0509	0x5103	ERROR_LS_CID_UNKNOWN	In response to a "SET" command, the DRCU notifies that the command ID is not known.	• 32-bits Command sent to the DRCU • 32-bits echo received
(5,1)	0x050A	0x5104	ERROR_LS_CID_FORBIDDEN	In response to a "SET" command, the DRCU notifies that the command ID is forbidden.	• 32-bits Command sent to the DRCU • 32-bits echo received
(5,1)	0x050B	0x5108	ERROR_SS_TIMEOUT	In response to a "SET" command, the DRCU times out.	• 32-bits Command sent to the DRCU • 32-bits echo received
(5,1)	0x050C	0x5109	ERROR_LS_DCU_RX	The 2 MSB of the echo sent back by the DRCU in response to a "GET" command, are not identical to the pattern sent by the DPU. In this case the parameter cannot be trusted and is discarded.	• 32-bits Command sent to the DRCU • 32-bits echo received
(5,1)	0x050D	0x5109	ERROR_LS_MCU_RX	Same as above	Same as above
(5,1)	0x050E	0x5109	ERROR_LS_SCU_RX	Same as above	Same as above
(5,1)	0x050F	0x510C	ERROR_LS_OVERFLOW	The number of commands sent to the DRCU (HK collection + all VMs) exceeds the maximum allowed rate.	• 32-bits word with number of micro-seconds in which the LS port was busy during the last second (should be more than 10^6)

(5,1)	0x0510	0x510D	UNKNOWN_TM_PCKT	A TM packet ready to be sent has an unknown combination of type, subtype and SID.	• Type, subtype and SID of the unknown TM packet
(5,1)	0x0511	0x5111	TC_SEQ_ERROR	A gap in TC Packet counter of the TC PTD has been detected.	• Previous TC PTD Counter. • Currently read TC PTD counter
(5,1)	0x0512	0x5117	ERROR_NO_TIMESYNC_ID	No updated Time information has been received from CDMS at the Start-of-Frame sync	• Currently valid TIME-at-next-frame-sync in usual 48-bit time stap format
(5,1)	0x0513	0x5115	Check_PM_Fail_ID	The CRC computed over the specified PM memory range is different from what it should be	• 16-bit true CRC (uplinked) • 16-bit computed CRC
(5,1)	0x0520	0x510E	ERROR_NO_DCU_RES	The DCU DRCU does not respond to a command. This event is raised when the status bit 2 in the LS port status register is not asserted within 2 milliseconds from the command dispatch to the DRCU, or when the response read doesn't match with sent command, this might as well imply that the LS Hardware interface is not working correctly.	• Command sent to the DRCU
(5,1)	0x0521	0x510F	ERROR_NO_MCU_RES	As NO_DCU_RES error, but for MCU subsystem	• Command sent to the DRCU
(5,1)	0x0522	0x5110	ERROR_NO_SCU_RES	As NO_DCU_RES error, but for SCU subsystem	• Command sent to the DRCU
(5,1)	0x0607	0x5114	DUMP_ABORTED_ID	The Memory Dump procedure has been aborted by TC	
(5,1)	0x0608	0x5114	DUMP_NO_MEMBLOCKS_IN_MEMDUMP_ID_ID	The Memory Dump procedure has been aborted because of failed memory block allocation	
(5,1)	0x1500	0x510A	ERROR_TC_POOL_FULL_ID	The DPU Memory Pool for Telecommand packets is more than 80% full	• Pool ID • Pool occupation status • Pool occupation limit
(5,1)	0x1501	0x510A	ERROR_EV_POOL_FULL_ID	The DPU Memory Pool for Event TM packets is more than 80% full	• Pool ID • Pool occupation status • Pool occupation limit

(5,1)	0x1502	0x510A	ERROR_HK_POOL_FULL_ID	The DPU Memory Pool for HouseKeeping TM packets is more than 80% full	• Pool ID • Pool occupation status • Pool occupation limit
(5,1)	0x1503	0x510A	ERROR_SD_POOL_FULL_ID	The DPU Memory Pool for Science TM packets is more than 80% full	• Pool ID • Pool occupation status • Pool occupation limit
(5,1)	0x1510	0x510B	ERROR_TC_HP_FIFO_FULL_ID	The VIRTUOSO FIFO Queue for high-priority TC packets is more than 80% full	• FIFO ID • FIFO occupation status • FIFO occupation limit
(5,1)	0x1511	0x510B	ERROR_TC_LP_FIFO_FULL_ID	The VIRTUOSO FIFO Queue for low-priority TC packets is more than 80% full	• FIFO ID • FIFO occupation status • FIFO occupation limit
(5,1)	0x1512	0x510B	ERROR_EV_TM_FIFO_FULL_ID	The VIRTUOSO FIFO Queue for event TM packets is more than 80% full	• FIFO ID • FIFO occupation status • FIFO occupation limit
(5,1)	0x1513	0x510B	ERROR_HK_TM_FIFO_FULL_ID	The VIRTUOSO FIFO Queue for HouseKeeping TM packets is more than 80% full	• FIFO ID • FIFO occupation status • FIFO occupation limit
(5,1)	0x1514	0x510B	ERROR_SD_TM_FIFO_FULL_ID	The VIRTUOSO FIFO Queue for science TM packets is more than 80% full	• FIFO ID • FIFO occupation status • FIFO occupation limit
(5,1)	0x1515	0x510B	ERROR_LS_HP_FIFO_FULL_ID	The VIRTUOSO FIFO Queue for high-priority Sub-Systems commands is more than 80% full	• FIFO ID • FIFO occupation status • FIFO occupation limit
(5,1)	0x1516	0x510B	ERROR_LS_LP_FIFO_FULL_ID	The VIRTUOSO FIFO Queue for low-priority Sub-Systems commands is more than 80% full	• FIFO ID • FIFO occupation status • FIFO occupation limit
(5,1)	0x1519	0x510B	ERROR_VM_TM_FIFO_FULL_ID	The VIRTUOSO FIFO Queue for TM packets generated by the VM is more than 80% full	• FIFO ID • FIFO occupation status • FIFO occupation limit
(5,1)	0x2578	0x5105	ERROR_FIFO_DCU_FID_ID	Wrong Frame ID in science data received from DCU	• HW Fifo ID • Frame ID read
(5,1)	0x2579	0x5105	ERROR_FIFO_MCU_FID_ID	Wrong Frame ID in science data received from MCU	• HW Fifo ID • Frame ID read

(5,1)	0x257A	0x5105	ERROR_FIFO_SCU_FID_ID	Wrong Frame ID in science data received from SCU	• HW Fifo ID • Frame ID read
(5,1)	0x2540	0x5106	ER-ROR_FIFO_DCU_FLEN_PHOT_FULL_ID	Wrong Frame Length for a Full Photometry DCU frame	• Frame ID read • Frame length read • Expected Frame length for that Frame ID
(5,1)	0x2541	0x5106	ER-ROR_FIFO_DCU_FLEN_SPEC_FULL_ID	Wrong Frame Length for a Full Spectrometer DCU frame	• Frame ID read • Frame length read • Expected Frame length for that Frame ID
(5,1)	0x2542	0x5106	ER-ROR_FIFO_DCU_FLEN_PSW_ID	Wrong Frame Length for a PSW DCU frame	• Frame ID read • Frame length read • Expected Frame length for that Frame ID
(5,1)	0x2543	0x5106	ER-ROR_FIFO_DCU_FLEN_PMW_ID	Wrong Frame Length for a PMW DCU frame	• Frame ID read • Frame length read • Expected Frame length for that Frame ID
(5,1)	0x2544	0x5106	ER-ROR_FIFO_DCU_FLEN_PLW_ID	Wrong Frame Length for a PLW DCU frame	• Frame ID read • Frame length read • Expected Frame length for that Frame ID
(5,1)	0x2545	0x5106	ER-ROR_FIFO_DCU_FLEN_SSW_ID	Wrong Frame Length for a SSW DCU frame	• Frame ID read • Frame length read • Expected Frame length for that Frame ID
(5,1)	0x2546	0x5106	ER-ROR_FIFO_DCU_FLEN_SLW_ID	Wrong Frame Length for a SLW DCU frame	• Frame ID read • Frame length read • Expected Frame length for that Frame ID
(5,1)	0x2547	0x5106	ER-ROR_FIFO_DCU_FLEN_PHOT_OFF_ID	Wrong Frame Length for a Full Photometry Offset DCU frame	• Frame ID read • Frame length read • Expected Frame length for that Frame ID
(5,1)	0x2548	0x5106	ER-ROR_FIFO_DCU_FLEN_SPEC_OFF_ID	Wrong Frame Length for a Full Spectrometer Offset DCU frame	• Frame ID read • Frame length read • Expected Frame length for that Frame ID
(5,1)	0x2549	0x5106	ER-ROR_FIFO_DCU_FLEN_PHOT_FULL_TEST_ID	Wrong Frame Length for a Full Photometry Test DCU frame	• Frame ID read • Frame length read • Expected Frame length for that Frame ID

(5,1)	0x254A	0x5106	ER-ROR_FIFO_DCU_FLEN_PSW_TEST_ID	Wrong Frame Length for a PSW Test DCU frame	• Frame ID read • Frame length read • Expected Frame length for that Frame ID
(5,1)	0x254B	0x5106	ER-ROR_FIFO_DCU_FLEN_PMW_TEST_ID	Wrong Frame Length for a PMW Test DCU frame	• Frame ID read • Frame length read • Expected Frame length for that Frame ID
(5,1)	0x254C	0x5106	ER-ROR_FIFO_DCU_FLEN_PLW_TEST_ID	Wrong Frame Length for a PLW Test DCU frame	• Frame ID read • Frame length read • Expected Frame length for that Frame ID
(5,1)	0x254D	0x5106	ER-ROR_FIFO_DCU_FLEN_SPEC_FULL_TEST_ID	Wrong Frame Length for a Full Spectrometer Test DCU frame	• Frame ID read • Frame length read • Expected Frame length for that Frame ID
(5,1)	0x254E	0x5106	ER-ROR_FIFO_DCU_FLEN_SSW_TEST_ID	Wrong Frame Length for a SSW Test DCU frame	• Frame ID read • Frame length read • Expected Frame length for that Frame ID
(5,1)	0x254F	0x5106	ER-ROR_FIFO_DCU_FLEN_SLW_TEST_ID	Wrong Frame Length for a SLW Test DCU frame	• Frame ID read • Frame length read • Expected Frame length for that Frame ID
(5,1)	0x2550	0x5107	ER-ROR_FIFO_DCU_FCRC_PHOT_FULL_ID	Wrong checksum for a Full Photometry DCU frame	• Frame ID read • Computed check-sum • Read checksum
(5,1)	0x2551	0x5107	ERROR_FIFO_DCU_ FCRC _SPEC_FULL_ID	Wrong checksum for a Full Spectrometer DCU frame	• Frame ID read • Computed check-sum • Read checksum
(5,1)	0x2552	0x5107	ERROR_FIFO_DCU_ FCRC _PSW_ID	Wrong checksum for a PSW DCU frame	• Frame ID read • Computed check-sum • Read checksum
(5,1)	0x2553	0x5107	ERROR_FIFO_DCU_ FCRC _PMW_ID	Wrong checksum for a PMW DCU frame	• Frame ID read • Computed check-sum • Read checksum
(5,1)	0x2554	0x5107	ERROR_FIFO_DCU_ FCRC _PLW_ID	Wrong checksum for a PLW DCU frame	• Frame ID read • Computed check-sum • Read checksum
(5,1)	0x2555	0x5107	ERROR_FIFO_DCU_ FCRC _SSW_ID	Wrong checksum for a SSW DCU frame	• Frame ID read • Computed check-sum • Read checksum

(5,1)	0x2556	0x5107	ERROR_FIFO_DCU_ FCRC _SLW_ID	Wrong checksum for a SLW DCU frame	• Frame ID read • Computed check-sum • Read checksum
(5,1)	0x2557	0x5107	ERROR_FIFO_DCU_ FCRC _PHOT_OFF_ID	Wrong checksum for a Full Photometry Offset DCU frame	• Frame ID read • Computed check-sum • Read checksum
(5,1)	0x2558	0x5107	ERROR_FIFO_DCU_ FCRC _SPEC_OFF_ID	Wrong checksum for a Full Spectrometer Off-set DCU frame	• Frame ID read • Computed check-sum • Read checksum
(5,1)	0x2559	0x5107	ERROR_FIFO_DCU_ FCRC _PHOT_FULL_TEST_ID	Wrong checksum for a Full Photometry Test DCU frame	• Frame ID read • Computed check-sum • Read checksum
(5,1)	0x255A	0x5107	ERROR_FIFO_DCU_ FCRC _PSW_TEST_ID	Wrong checksum for a PSW Test DCU frame	• Frame ID read • Computed check-sum • Read checksum
(5,1)	0x255B	0x5107	ERROR_FIFO_DCU_ FCRC _PMW_TEST_ID	Wrong checksum for a PMW Test DCU frame	• Frame ID read • Computed check-sum • Read checksum
(5,1)	0x255C	0x5107	ERROR_FIFO_DCU_ FCRC _PLW_TEST_ID	Wrong checksum for a PLW Test DCU frame	• Frame ID read • Computed check-sum • Read checksum
(5,1)	0x255D	0x5107	ERROR_FIFO_DCU_ FCRC _SPEC_FULL_TEST_ID	Wrong checksum for a Full Spectrometer Test DCU frame	• Frame ID read • Computed check-sum • Read checksum
(5,1)	0x255E	0x5107	ERROR_FIFO_DCU_ FCRC _SSW_TEST_ID	Wrong checksum for a SSW Test DCU frame	• Frame ID read • Computed check-sum • Read checksum
(5,1)	0x255F	0x5107	ERROR_FIFO_DCU_ FCRC _SLW_TEST_ID	Wrong checksum for a SLW Test DCU frame	• Frame ID read • Computed check-sum • Read checksum
(5,1)	0x2560	0x5106	ER-ROR_FIFO_MCU_FLEN_SMEC_ ID	Wrong Frame Length for a SMEC MCU frame	• Frame ID read • Frame length read • Expected Frame length for that Frame ID
(5,1)	0x2561	0x5106	ER-ROR_FIFO_MCU_FLEN_BSM_I D	Wrong Frame Length for a BSM MCU frame	• Frame ID read • Frame length read • Expected Frame length for that Frame ID
(5,1)	0x2563	0x5106	ER-ROR_FIFO_MCU_FLEN_ENGIN EERING_ID	Wrong Frame Length for an Engineering MCU frame	• Frame ID read • Frame length read • Expected Frame length for that Frame ID

(5,1)	0x2565	0x5106	ER-ROR_FIFO_MCU_FLEN_TEST_ID	Wrong Frame Length for a Test MCU frame	• Frame ID read • Frame length read • Expected Frame length for that Frame ID
(5,1)	0x2568	0x5107	ER-ROR_FIFO_MCU_FCRC_SMEC_ID	Wrong checksum for a SMEC MCU frame	• Frame ID read • Computed check-sum • Read checksum
(5,1)	0x2569	0x5107	ERROR_FIFO_MCU_ FCRC _BSM_ID	Wrong checksum for a BSM MCU frame	• Frame ID read • Computed check-sum • Read checksum
(5,1)	0x256B	0x5107	ERROR_FIFO_MCU_ FCRC _ENGINEERING_ID	Wrong checksum for an Engineering MCU frame	• Frame ID read • Computed check-sum • Read checksum
(5,1)	0x256D	0x5107	ERROR_FIFO_MCU_ FCRC _TEST_ID	Wrong checksum for a Test MCU frame	• Frame ID read • Computed check-sum • Read checksum
(5,1)	0x2570	0x5106	ER-ROR_FIFO_SCU_FLEN_HSK_ID	Wrong Frame Length for a nominal SCU frame	• Frame ID read • Frame length read • Expected Frame length for that Frame ID
(5,1)	0x2571	0x5106	ER-ROR_FIFO_SCU_FLEN_TEST_ID	Wrong Frame Length for a Test SCU frame	• Frame ID read • Frame length read • Expected Frame length for that Frame ID
(5,1)	0x2574	0x5107	ER-ROR_FIFO_SCU_FCRC_HSK_ID	Wrong checksum for a nominal SCU frame	• Frame ID read • Computed check-sum • Read checksum
(5,1)	0x2575	0x5107	ER-ROR_FIFO_SCU_FCRC_TEST_ID	Wrong checksum for a Test SCU frame	• Frame ID read • Computed check-sum • Read checksum
(5,1)	0x7001	0x5112	ERROR_MONPAR_NOM_WARN	An HK parameter went from nominal to warn-ing	• 32-bit get parame-ter command • 32-bit echo word • Read parameter-value
(5,1)	0x7002	0x5112	ERROR_MONPAR_NOM_FAIL	An HK parameter went from nominal to fail	• 32-bit get parame-ter command • 32-bit echo word • Read parameter-value
(5,1)	0x7012	0x5112	ERROR_MONPAR_WARN_FAIL	An HK parameter went from warning to fail	• 32-bit get parame-ter command • 32-bit echo word • Read parameter-value

(5,1)	0x7020	0x5112	ERROR_MONPAR_FAIL_NOM	An HK parameter went from fail to nominal	• 32-bit get parameter command • 32-bit echo word • Read parameter-value
(5,1)	0x7021	0x5112	ERROR_MONPAR_FAIL_WARN	An HK parameter went from fail to warning	• 32-bit get parameter command • 32-bit echo word • Read parameter-value
(5,1)	0x7010	0x5112	ERROR_MONPAR_WARN_NOM	An HK parameter went from warning to nominal	• 32-bit get parameter command • 32-bit echo word • Read parameter-value
(5,1)	vmArg	0x5113	EVENT_VM_EVENT	An Event Report from VM	• VM Arg.
(5,2)	0xC000	0x0520	EXCP_DRCU_ANOMALY_SID	(NYI)	• (NYI)
(5,2)	0xC010	0x0520	EXCP_DPU_ANOMALY_SID	(NYI)	• (NYI)
(5,2)	0xC100	0x0520	EXCP_OBS_ANOMALY_SID	(NYI)	• (NYI)
(5,2)	0xC110	0x0520	EXCP_OBS_CORRECT_SID	(NYI)	• (NYI)
(5,2)	0x0832	0x0520	EXCP_FX_UNARMED_SID	Cannot execute the activity requested, the function wasn't activated.	None
(5,2)	vmArg	0x5201	EXCP_VM_EXCP	An Event Exception Report from VM	• VM Arg.
(5,4)	0x550C	0x5420	ALARM_LSDCU_DEAD	Same as code 0x050C	Same as code 0x050C
(5,4)	0x550D	0x5420	ALARM_LSMCU_DEAD	Same as code 0x050D	Same as code 0x050D
(5,4)	0x550E	0x5420	ALARM_LSSCU_DEAD	Same as code 0x050E	Same as code 0x050E

Table 7-1 Event Warning/Error Codes

8 TC Verification Error Codes

In case of errors in the application data of the received telecommands, the DPU, in accordance with AD3, issues TM (1,8) packets. These packets will contain an error code and a variable list of parameters according to the following table.

Error Code	Error Name	Description	Parameters
Memory Management			
0x601	Illegal_Memory_ID	The specified Memory ID is not in the valid range 0-3	The requested Mem_ID
0x602	Illegal_Start_Address	The Start Address is not in the valid range for the requested Memory ID (see below)	Required Start Address
0x603	Illegal_NSAU	The uplinked number of SAUs will place the memory patch outside the valid range for the requested Memory ID and Start Address	Uplinked number of SAUs
0x604	Bad_NSAU	The number of SAUs does not match with the TC packet length contained in the TC packet header	Uplinked number of SAUs
0x605	Bad_CRC	The CRC computed by the OBS on the uplinked memory patch is not equal to the one sent with the TC	CRC value uplinked with the memory patch
0x606	Bad_Load	The CRC computed by the OBS after the memory patch has been written into the DPU memory is not equal to the CRC value in the TC which contained the memory patch	OBS computed CRC
Function Management			
0x805	Illegal_Table_ID	The Table specified is not in the valid 0-127 range	The required Table ID
0x806	Illegal_Table_Index	The INDEX in the *Update Table* TC is larger that the table length specified in the *Set Table* TC	The uplinked INDEX
0x808	Bad_Data	The number of data words contained in the TC is not consistent with the Length field in the TC packet header	The uplinked number N of 32-bit data words
0x809	Table_Space_Full	Not enough memory to create the table of the required size (in the *Set Table* TC)	Required table length
0x80A	No_Command_List	A Stop VM TC has been received, but the specified VM is not running	The index of the VM (0 for HW VM) required to stop
0x80C	VM_Running	An *Execute* or *Start Comman List* TC was received, but the specified VM is already executing a command list	The index of the VM (0 for HW VM) required to stop
0x80D	Bad_Ndata	The munber of data words in a	The uplinked

		Report Table or Update Table TCs is inconsistent with the actual table length and start Index	number (in units of 32-bit words) of data words
0x80E	LS_RECEPTION_ERROR	If a *Reset DRCU Counter* TC was received but the command could not be successfully dispatched to the DRCU because of an LS transmission error, this error code notifies that no DRCU sync was done	The sync DRCU command sent
0x80F	ILLEGAL_FFLAGS	The FIFO ID required to be reset is not a valid FIFO ID	The uplinked FIFO flag word
0x810	VM_UNDEFINED_TABLE_ID	The Table ID specified as input for a VM was not previously defined	The requested Table ID
0x811	Undefined_Table	The table for which an update or report has been requested, was not previously defined	The requested Table ID
0x812	EEPROM_Failed	The procedure to write the image of the OBS currently running in PM into the EEPROM, failed	The number of errors occured during the procedure
0x813	Busy_table	The table for which a creation or update has been requested, is currently in use (by either a VM, or HK sampling or monitoring)	The requested Table ID
0x0815	Illegal_Frame_ID	The Frame_Id number contained in a TC is outside the allowed 00-0F,10-15 or 20-21 ranges	The requested Frame_Id
0x0816	Illegal_Sel_Table_ID	The Table ID number contained in a TC is outside the allowed 0-127 range	The requested Table_ID
0x0817	Undefined_Sel_Table	Table n. Table_ID not defined	Ibidem
0x0818	Invalid_len_Sel_Table	The length of table n. Table_ID doesn't match with selected Frame_ID's lentgh	The tables's length
0x0819	Invalid_content_Sel_Table	The content of table n. Table_ID doesn't contain a valid boolean {0,1} value array for selection	The requested Table_ID
0x081B	Illegal_Table_Len	The length of the table exceeds 8192 words, and this is not allowed	Required table length
0x081C	LS_INHIBITED_CMD_ERROR	The S/S command sent is inhibited	Sent S/S Command
0x081D	CIS_WRONG_CMD_RANGE	An incorret range of command IDs has been required for S/S command inhibition	Requested minimum and maximum S/S command IDs
0x081E	CIS_WRONG_BROADCAST	Inhibition has been requested for a broadcast command	
0x081F	VM_UNDEFINED_REF_TABLE_ID	A call to an undefined table is found within a VM code	Table ID that is undefined
0x0820	Peak_Up_error	(NYI)	(NYI)
0x0821	Illegal_HK_Packet_ID	The HK Packet ID contained in	Uplinked HK

		a TC is not in the allowed range 0-3	Packet ID
0x0822	Illegal_HK_SID	The MSB of the HK SID in a (3,x) TC is not 0x03	Uplinked HK SID
0x0823	Illegal_HK_Table_ID	The Table ID number contained in a TC is outside the allowed 0-127 range	Uplinked Table ID
0x0824	Illegal_HK_Sampling_Interval	The sampling interval contained in a TC is below the minimum allowed threshold (10ms) specified in RD2.	Required sampling interval (in ms)
0x0825	Undefined_HK_Table	A TC was received, linking an HK Packet ID to a Table ID which was not previously defined	The required undefined Table ID
0x0826	Undefined_Monitoring_Table	The specified Monitoring Table is not defined	The requested Table ID
0x0827	Err_HK_Sampling_Running	A new TC HK report definition was received while the sampling is still running (i.e, before a TC was sent. The only exception is the case where the only modification requested is the sampling interval.	The HK Packet ID contained in the received TC, and which is still running
0x0828	Illegal_Monitoring_Table_ID	The specified Monitoring Table ID is out of range	The requested Table ID
0x0829	Undefined_HK_ID	The HK packet ID requested in a TC does not correspond to a currently running sampling either defined one	The HK Packet ID contained in the TC, and which is not running
0x082A	HK_Nominal_not_Running	The nominal HK packet collection task is not running; in this condition monitoring cannot start	Status of HK collection process
0x082B	Undefined_HK_item	The HK parameter for which monitoring is requested is not defined in the nominal HK packet	HK parameter
0x082C	Undefined_Autonomy_Function	The autonomy function required to start following a HK monitoring anomaly, is not defined	HK parameter
0x082D	Monitoring_Suspended	It is required to suspend the monitoring checks when the task is already suspended	None
0x082E	Monitoring_Resumed	It is required to resume the monitoring checks when the task is already resumed	None
0x082F	Monitoring_Active	It is required to switch-off nominal HK collection while the monitoring task is active	None
0x0830	Function_Active	The request Function is already activated	Function Enable Status
0x0831	Function_Stopped	The request Function is already stopped	Function Enable Status
0x0832	PM_CHECK_Illegal_Start_Address	An incorrect start address was requested to perform a PM Checksum check	the requested start address

0x0833	PM_CHECK_Illegal_LENGTH	The length requested for a PM CRC check is incorrect (out of memory)	The requested lenght of memory area
0x0834	Mon_Active	An attempt was made to clear the nominal HK packet definition before stopping the monitoring task	
0x0835	MON_MEM_FULL	Not enough DM available to allocate the data monitoring internal structures	
0x0836	WRITE_EE_Illegal_Partition	A PM write onto the EEPROM was requested with the wrong memory bank (0 and 1 only allowed)	ID of the requested EEPROM memory bank
0x0837	WRITE_EE_Illegal_Addressing	Start Address and End Address are specified in a reversed order when a EEPROM write is requested	The requested start and end addresses
0x0838	WRITE_EE_Illegal_pageAvoidanceNum	The requested number of EEPROM pages to avoid in a requested EEPROM write exceeds the maximum number (114) that can fit in a TC	The requested number of EEPROM pages to avoid
0x0840	VMCall_Table_non_existent	An operation dealing with an undefined table has been detected in a VM program	The undefined table ID
0x0841	VMRead_Twice	In a VM program the READ register of the Low-Spped port is being read before an S/S command is sent	
0x0842	VMPageFault	In a VM program a table is accessed out of its memory	The affected table ID
0x0E01	Illegal_Type	The Packet Type to be Enabled/Disable not compliant to OBS's ICD	Type
0x0E02	Illegal_SubType	The Packet SubType to be Enabled/Disable not compliant to OBS's ICD	SybType
0x0E03	Illegal_SID	The Packet SID to be Enabled/Disable not compliant to OBS's ICD	SID
0x0E04	Bad_NPCKTS	The number of Packets to be Enabled/Disable don't match with packet length	NPCKTS

Table 8-1 TC Verification Error Codes