HERSCHEL

SPIRE On-Board Software User Manual

Document Ref.: SPIRE-IFS-PRJ-001391

Release: 4.0.0
Issue: 4.0.0

Prepared by:
Scigè John, Liu
Sergio Molinari

Distribution List:

RAL	**K. King**
	B. Swinyard
	S. Sidher
	T. Grundy
IFSI	**R. Cerulli**
	R. Orfei

<BLANK PAGE>

➢ Index :

➢ INDEX : 3
➢ TABLES: 6
1 INTRODUCTION 9
1.1 PURPOSE OF THE DOCUMENT 9
1.2 ACRONYMS AND GLOSSARY 9
1.3 DOCUMENT LIST 10
1.3.1 Applicable Documents 10
1.3.2 Reference Documents 10
1.4 DOCUMENT CHANGE RECORD 11
2 OBS COMPILATION 13
2.1 EXTERNAL COMPONENTS 13
2.1.1 Compiler, assembler, linker and other tools used. 13
2.1.2 File System Environment and basic data manipulation 14
2.2 THE VIRTUOSO PROJECT FILE 14
2.3 THE ARCHITECTURE FILE 14
2.4 COMPILING THE OBS'S SOURCE CODES 15
2.5 THE COMPILATION PRODUCTS 15
2.6 PROGRAM MEMORY USAGE SUMMARIES 17
3 OBS OBJECTS 18
3.1 OBS TASKS 18
3.2 MEMORY POOLS 19
3.3 FIFOS 19
3.4 SEMAPHORES 20
4 MANAGING OBS ON THE DPU 21
4.1 RUNNING THE EEPROM-RESIDENT OBS 21
4.2 LOADING THE OBS VIA TELECOMMANDS 21
4.2.1 Generating the telecommands 21
4.2.2 Load the telecommands 22
4.2.3 Startup the uploaded image 22
4.3 VERIFICATION OF THE RUNNING OBS 22
4.4 MODIFICATION OF THE RUNNING OBS 23
4.5 STORE THE OBS INTO EEPROM 23
4.5.1 Damaged pages on EEPROM 23
4.5.2 Store the OBS into EEPROM [Example] 24
5 USAGE OF THE OBS SERVICES 25

6 TELECOMMAND VERIFICATION 25
7 HOUSEKEEPING DATA REPORTING 25
7.1 SITUATION AT START-UP 26
7.2 MODIFYING THE HK PACKET PROPERTIES 26
7.2.1 Sampling Interval 26
7.2.2 HK Parameters 26
7.2.3 Internal HK Parameters 27
7.2.3.1 Note 1: Internal Hk Parameter 31
7.2.3.2 Note 2: OBS Hk commands 31
7.2.3.3 Note 3: Monitoring and Autonomy Action flags 32
8 MEMORY MANAGEMENT 33
8.1 ABSOLUTE ADDRESSING 33
8.2 PROGRAM MEMORY VERIFICATION 33
8.3 DATA MEMORY HARDWARE VERIFICATION 33
9 TABLE MANAGEMENT 34
9.1 DEFAULT TABLES IN THE OBS 35
9.2 TABLE LOAD 37
9.3 TABLE UPDATE 37
9.4 TABLE DELETE 37
9.5 TABLE DEFRAGMENTATION 37
10 OBS PATCHING 38
11 WRITE TO EEPROM 39
12 VIRTUAL MACHINES 40
12.1 EMBEDDED ON BOARD FUNCTIONS FOR VIRTUAL MACHINES PROGRAMS 40
12.1.1 Setting IRQ sensing state 42
12.1.2 Subsystems Commanding Inhibition 42
12.1.3 Note 3: 42
12.1.4 Note 4: 42
12.1.5 Note 5: 42
12.2 FLOATING POINT NUMERICAL FORMAT 43
12.2.1 Floating Point operations 43
12.2.2 Floating Point comparators 43
12.2.3 Floating Point conversions 44
12.2.4 Floating Point operation status. 44
12.3 AUTONOMY ACTION FORMAT 45
13 MONITORING OF HOUSEKEEPING PARAMETERS 46
13.1 BASICS 46
13.2 HOW TO CONFIGURE THE MONITORING SYSTEM 47
13.3 ARGUMENT DEFINING A MONITORED PARAMETER 48
13.3.1 Parameter 48
13.3.2 Command_ID 48
13.3.3 Mon_Config 48
13.3.3.1 Note 1: Analogic parameters are handled as Signed Value 49
13.3.3.2 Note 2: Parameters' value masking and shifting 49
13.3.3.3 Note 3: Analogic Parameters with Unsigned value 49
13.3.4 XXX_YYY_Limits. 49

13.3.5 N_Fail_Values. 49
13.3.6 Fail_Val [1 to 16]. 49
13.3.7 Action_XY. 50
13.3.8 N_Dep. 50
13.3.9 Dep 1 ... N. 50
13.4 VM Tables and OBS Functions for Autonomy Services 51
13.4.1 Abort Peak-UP 51
13.4.2 Safe VM Environment 51
13.4.3 Goto SAFE Mode 52
13.5 Tele Command with different priority 53
13.6 How to Use the Monitoring System 54

14 SUB-SYSTEM I/F FDIR MANAGEMENT 55

14.1 Note on FDIR behaviour 56
14.2 Note on SDEX – Science Data Extraction 56
14.3 HS Transparent mode 56
14.4 LS Errors detection 57

15 OBS DIRECT FUNCTIONS 59

15.1 HS Management 59
15.1.1 Science Frame Data Selection 59
15.1.2 Science Data Buffer FIFO Flush / Reset 59
15.2 SubSystem Direct Commanding 59
15.3 LS Commanding Inhibition System 59
15.3.1 LS CIS – DCU, MCU and SCU 60
15.3.2 LS CIS Management of Single Commands 60
15.3.3 LS CIS Management of Range of Commands 60
15.4 Science Frame Data Extraction (SDEX) 61

16 THE PEAK-UP OPERATING MODE 61

17 THE SAFETY PROCEDURE GOTO_SAFE_MODE 61

18 EVENT REPORTING 62

18.1 Events, Reports and Warnings reports – TM(5,1) : 62
18.2 Anomalies and Exceptions – TM(5,2) : 72
18.3 Alarms and Errors reports – TM(5,4) : 73

19 TC VERIFICATION ERROR CODES 74

19.1 TC Verification Error Structure 74
19.2 TC Verification Error Codes 75

A. APPENDIX A – SPECIAL TELEMETRY PACKETS. 80

➢ Tables:

Table 1-1 Acronyms and Glossary 10
Table 1-2 Applicable Documents 10
Table 1-3 Reference Documents....10
Table 1-4 Document Change Record....12
Table 2-1 Tools used to compile 14
Table 2-2 – SPIRE.ACH – Board architecture – Memory mapping 15
Table 2-3 – SPIRE.MAP – Memory usage in Spire OBS 16
Table 2-4 Program Memory Usage 17
Table 3-1 – Entity Types and relative definitions....18
Table 3-2 OBS Task ID definition....18
Table 3-3 Memory Pool ID definition 19
Table 3-4 Virtuoso FIFO ID definition 19
Table 3-5 Virtuoso SEMA ID definition 20
Table 7-1 List of allowed HK packets 25
Table 7-2 Commands to get DPU HK parameters 30
Table 7-3 Subsystem Status Bits 31
Table 7-4 HK from data in Table....31
Table 7-5 HK from data in Table....32
Table 9-1 On Board Default Tables pre-allocated at boot-time. 36
Table 10-1 Patching example 38
Table 12-1 OBS Functions accessible from VM programs....41
Table 12-2 VM/OBS Functions I/O registries....42
Table 12-3 VM/OBS Functions Constant Values 43
Table 12-4 VM/OBS Comparison Functions Behaviour....43
Table 12-5 VM/OBS Converstion Functions Behaviour....44
Table 12-6 Floating Point operation status bit mapping....44
Table 12-7 Floating Point operation status bit semantics 44
Table 12-8 Single Autonomy Action format – 32 bit word....45
Table 13-1 Single Autonomy Action format – 32 bit word....47
Table 13-2 – Monitoring Item State and ID 48
Table 13-3 – Monitored HK parameter 48
Table 13-4 – Monitoring item configuration 48
Table 13-5 – Monitored HK Value Ranges....49
Table 13-6 – Autonomy Action....50
Table 13-7 – Autonomy OBS Functions 51
Table 13-8 Safe Enviroment VM. 52
Table 13-9 Safe Mode VM. 52
Table 14-1 Configuration table....56
Table 14-2 Two possible scenario: a single error on left, two contiguous error on right. 56
Table 15-1Science Data Selection example....59
Table 15-2 DCU, MCU & SCU CIS addressing 60
Table 18-1 Events, Reports and Warnings reports 71
Table 18-2 Anomalies and Exceptions reports....72
Table 18-3 Alarms and Errors reports 73
Table 19-1 TC Verification Error Structure 74
Table 19-1 TC Verification Error Codes....79
Table A-1 Special Telemetry Packets....80

<BLANK PAGE>

<BLANK PAGE>

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 up-linked 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 4.0.0.

1.2 Acronyms and Glossary

WORD	TERMS
AVM	Avionic Model
BC	Bus Controller
BP	BreakPoint
BSW	DPU Boot Software
CDMS	Command and Data Management System
CIS	LS Commanding Inhibition Status
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
LS	Low-speed Serial interface to sub-systems
NA	Not Applicable
OBS	On-Board Software
PM	Program Memory (DSP)
PROM	Programmable Read-Only Memory
RAM	Random Access Memory
S/C	Space-Craft
S/S	Sub-System
SUT	Software Under Test
TBC	To Be Confirmed
TBD	To Be Defined
TBW	To Be Written
TC	Tele-Command
TM	Telemetry
VME	Virtual Machine Executable Code

Table 1-1 Acronyms and Glossary

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 Issue:
AD2	SPIRE OBS Software Specifications Document	SPIRE-IFS-DOC-001352 Issue:
AD3	Packet Structure Interface Control Document (PSICD)	SCI-PT-ICD-7527 Issue:
AD4	Herschel/Planck Instrument Data Rates	H-P-1-ASPI-TN-0204 Issue:
AD5	SPIRE Autonomy requirements	SPIRE-RAL-PRJ-001855 Issue:
AD6	SPIRE Peak-up Mode Requirement	SPIRE-RAL-PRJ-001969 Issue:
AD7	SPIRE Failure Detection Isolation and Recovery	SPIRE-RAL-PRJ-1978 Issue
AD8		

Table 1-2 Applicable Documents

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	CNR. IFSI. 2003. TR01
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	VIG41R200
RD9	ADSP-21000 Family C Tools Manual	
RD10	SPIRE FDIR	SPIRE-RAL-PRJ-001978
RD11	SPIRE On-Board Software Configuration Report	IFSI/OBS/RP/2004-001 [SPIRE]
RD12	Memory Management Library Interface Version 1.1	
RD13	SPIRE On-Board Software Configuration Report - Delivery Notes	IFSI/OBS/RP/2004-001 [SPIRE] <DeliveryNoteAppendix>
RD14		

Table 1-3 Reference Documents

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	2	12/06/2006	Updates on: • Tasks List • Pool List • FIFO List • Semaphores List • Tables Definitions • Event packets List • Error codes List
2	2.E	20/11/2006	Updates on: • NCR 12 • Document naming aligned with Software Revision
2	2.G	12/01/2007	Updates on: • VM Functions for Floating Point Values
2	2.H	13/01/2008	Updates on: • Header CNR Logo to INAF Logo Changes for SxRs: • SCR-0165 – added new HK-Definition in § 7.2.3 :Get_FifoStat_TC_HP, Get_FifoStat_TC_LP, Get_FifoStat_TM_EV, Get_FifoStat_TM_HK, Get_FifoStat_TM_SD, Get_FifoStat_TM_RP, Get_FifoStat_LS_HP, Get_FifoStat_LS_LP, Get_FifoStat_AFX, Get_FifoStat_VM, Get_FifoStat_MEMD • SPR-0585 – Protection to Stack Over/Under Flows • SPR-0592 – Changing Addressing algebra • SPR-0602 – Removal of any TM(1,8), new Event set. • SCR-0608 – New HK Parameter Class : Moat table's Row MSW/LSW. • SPR-0615 – Changing Boolean Statement in TMTC • SCR-0622 – New structures for new procedure for SAFE mode • SCR-0634 – new behaviour for BBID implemented • SCR-0635 – New HK definition updated • SPR-0641 – Event ID behaviour aligned to ICD • SPR-0642 – Monitoring table ID now can be set freely • SPR-0643 – Now VM Stopped Event restored. • SCR-0646 – New HK parameter as SCR-0608 • SCR-0647 – Smec Selection table installed. • SPR-0650 – UM updated accordingly New Functionality: • OBS function for VM ○ SAFE_MODE ○ THROW_EVENT ○ Flush Fifo ○ Reset Fifo UM: • Par 4 mod introduction • 4.3 Embedding new paragraph on the OBSM
2	9.0	30/06/2008	• Added reference to Configuration Report and Delivery

Issue	Revision	Date	Reason for Change
			note for memory ranges and checksums. • Changing 0x41YYZZZZ to 0x50YYZZZZ for extraction for Table row MSW to HK.
2	9.1	20/08/2008	• Chapters tree one level lower for single functionality in OBS • Autonomy Action rewrote • Safe VM • Safe MODE
3	x	31/10/2008	Collected to Issue 4.0.0
4	0.0	23/10/2009	Aligning to changes applied in OBS v 4.0.0: • New HK parameters • New debug dump structure • New VM trace conditions • VM stop with ceiling for time to effect. • Monitoring task deletion of 3three time status, now with 2two time state. • Fixing Telecommand priority • Autonomy function “Safe mode” recoding • Autonomy function context now configurable by RAL • Separating Monitoring System from Action Sequencing • Monitoring Task now sending events through VM_SVC • Monitoring Task now sending any event even if not yet served • Action Sequencer recalculate the right action by the collected transitions • Monitoring Task now with different State Management • Telecommand Sequencer now with different insulation schema

Table 1-4 Document Change Record

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 on a Windows machine:

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

2.1.1 Compiler, assembler, linker and other tools used.

The following list of tools are the minimal requirement for building the SPIRE OBS:

- Spire OBS modules are compiled by
  “**gcc version rel3.3 21k/SHARC 3.3**”
  build by Analog Devices for the ADI-DSP 21K family that parses code written in “C” as known in 1991-1996.
- Spire OBS modules are linked together by
  “**Analog Devices ADSP-210x0 Linker Release 3.3, Version 2.21**”.
- Spire OBS uses library modules managed by
  “**Analog Devices ADSP-210x0 Librarian Release 3.3, Version 2.21**”.
- Spire OBS uses low level assembly function assembled by
  “**Analog Devices ADSP-210x0 Librarian Release 3.3, Version 2.21**”.
- Initialization structure for data memory is managed by
  “**Analog Devices ADSP-210x0 Initializer Release 3.3, Version 2.21**”.

A standard make is been used to rule the compilation procedure:

Tools used to compile

```
C:\ADI_DSP\BIN>g21k -version
gcc version rel3.3 21k/SHARC 3.3:
```

```
C:\ADI_DSP\BIN>ld21k -version
Analog Devices ADSP-210x0 Linker
Release 3.3, Version 2.21
Copyright (c) 1991-1996 Analog Devices, Inc.
… omissis …
```

```
C:\ADI_DSP\BIN> ld21k -version
Analog Devices ADSP-210x0 Initializer
Release 3.3, Version 2.21
Copyright (c) 1991-1996 Analog Devices, Inc.
```

```
C:\ADI_DSP\BIN>asm21k –version
Analog Devices ADSP-210x0 Assembler
Release 3.3, Version 2.21
Copyright (c) 1991-1997 Analog Devices, Inc.
```

```
C:\ADI_DSP\BIN>lib21k -version
Analog Devices ADSP-210x0 Librarian
Release 3.3, Version 2.21
Copyright (c) 1991-1996 Analog Devices, Inc.
```

Tools used to compile

```
C:\Documents and Settings\scige>make -version
Bypass to CYGWIN-make
GNU Make 3.81
Copyright (C) 2006  Free Software Foundation, Inc.
This is free software; see the source for copying conditions.
There is NO warranty; not even for MERCHANTABILITY or FITNESS FOR A
PARTICULAR PURPOSE.

This program built for i686-pc-cygwin
```

Table 2-1 Tools used to compile

2.1.2 File System Environment and basic data manipulation

Minimal file manipulation can be done via DOS-like command [i.e. copy, del] and via Unix-like command [i.e. cp, rm], we suggest to install a UNIX-Like environment like Cygwin or Mingw.

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. The **spire.vpf** file is part of the delivery pack of the OBS.

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 4.0.0 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.

```
!***** start of file *****!
! Spire Hardware Architecture Description
.system FirstDPU;
.processor = ADSP21020;
! == == == == == == == == == == == == == == == == == == == == == == == == == ==
!
!
! == == == == == == == == == == == == == == == == == == == == == == == == == ==
! program memory area:
!                          pm Interrupt table           256 Words 48 bits
!                          pm Initialization Segment   8192 Words 48 bits
!                          pm Program OBS    Segment 106496 Words 48 bits
!                          pm Program Patch  Segment 393216 Words 48 bits
!
.segment /pm /ram  /begin=0x000000 /end=0x0000FF   seg_rth;
! nino 18/01/2006 : segment expanded after MMLIB
.segment /pm /ram  /begin=0x004000 /end=0x005FFF   seg_init;
.segment /pm /ram  /begin=0x006000 /end=0x01FFFF   seg_pmco;
! nino 16/11/2005 : segment cutted-out by nino - for patching purpose
.segment /pm /ram  /begin=0x020000 /end=0x07FFFF   seg_pmhi;
! == == == == == == == == == == == == == == == == == == == == == == == == == ==
!
!
! == == == == == == == == == == == == == == == == == == == == == == == == == ==
! data memory area:
!                          dm data   383     KiWords 32 bits (385024 word)
!                          dm stack    2     KiWords 32 bits (  7168 word)
!                          dm heap   127     KiWords 32 bits (130048 word)
```

```
!
.segment /dm /ram  /begin=0x00000000 /end=0x0005FBFF          seg_dmda;
.segment /dm /ram  /begin=0x0005FC00 /end=0x000603FF /cstack  seg_stak;
.segment /dm /ram  /begin=0x00060400 /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=0x88FFFFFF Bus_IF_DCU;
.segment /dm /ram  /begin=0x89000000 /end=0x89FFFFFF Bus_IF_MCU;
.segment /dm /ram  /begin=0x8A000000 /end=0x8AFFFFFF Bus_IF_SCU;
.segment /dm /ram  /begin=0x8F000000 /end=0x8FFFFFFF Bus_IF_1553;
! == == == == == == == == == == == == == == == == == == == == == == == == == ==
!
!
! == == == == == == == == == == == == == == == == == == == == == == == == == ==
!Bank Description
.bank /pm0 /wtstates=0 /wtmode=internal /begin=0x000000;
!!the PM bank1 is mot mounted
!!.bank /pm1 /wtstates=0 /wtmode=internal /begin=0x080000;
! == == == ==                                                       == == == ==
!Bank Description
! 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 *****
```

Table 2-2 – SPIRE.ACH – Board architecture – Memory mapping

2.4 Compiling the OBS's Source Codes

The OBS distribution contains various **MAKEFILE** that manages the compilation and linking of the source codes. 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 installed.

2.5 The Compilation Products

The compilation will produce many intermediate files. The most important compilation product is obviously the **SPIRE_FM.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.
As example here is an extract from the **SPIRE_NM.MAP** file contained in the OBS distribution and valid for the SPIRE OBS version 4.0.0.

```
Analog Devices ADSP-210x0 Linker              spire_nm.map             Page 1
Release 3.3, Version 2.21                     Thu Oct 22 16:50:40 2009
Copyright (c) 1991-1996 Analog Devices, Inc.


Architecture Description: FirstDPU

Segment      Start      End        Length      Memory Type          Attribute    Width

seg_rth      000000     0000ff     256         Program Memory       RAM
seg_dmda     00000000   0005fbff   392192      Data Memory          RAM
seg_init     004000     005fff     8192        Program Memory       RAM
seg_pmco     006000     01ffff     106496      Program Memory       RAM
seg_pmhi     020000     07ffff     393216      Program Memory       RAM
seg_stak     0005fc00   000603ff   2048        Data Memory          RAM
heap1        00060400   0007ffff   130048      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_DCU   88000000   88ffffff   16777216    Data Memory          RAM
Bus_IF_MCU   89000000   89ffffff   16777216    Data Memory          RAM
Bus_IF_SCU   8a000000   8affffff   16777216    Data Memory          RAM
Bus_IF_1553  8f000000   8fffffff   16777216    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     006000     016057     65624       Program Memory       RAM
seg_pmhi     ******     ******     0           Program Memory       RAM
seg_dmda     00000000   0005d9be   383423      Data Memory          RAM
seg_stak     0005fc00   0005fc0a   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_D     ********   ********   0           Data Memory          RAM
Bus_IF_M     ********   ********   0           Data Memory          RAM
Bus_IF_S     ********   ********   0           Data Memory          RAM
Bus_IF_1     ********   ********   0           Data Memory          RAM


Memory Usage Summaries:

Memory Type     Attribute  Total

Program Memory  ROM        0
Program Memory  RAM        65895
Program Memory  PORT       0
Data Memory     ROM        0
Data Memory     RAM        383434
Data Memory     PORT       0


Cross Reference …………
---- Cross reference omitted ----
```

Table 2-3 – SPIRE.MAP – Memory usage in Spire OBS

The file **SPIRE_NM.MAP** is also the base reference for addressing information on DPU, for accessing the OBS Data Memory, Program Memory, Storage in EEPROM, and over.

Check RD11or RD13 for an extraction valid for the current revision of the OBS.

2.6 Program Memory Usage Summaries

The following table describe the used program memory space [extraction of the above memory map] :

Segment	Segment Scope	Segment Start	Segment End	Usage End	Usage [dec]	Usage [hex]
seg_rth	Interrupt Vector used store mainly to address	0x000000	0x0000FF	0x0000FF	256	0x100
seg_init	Initialization for Global/Static Variables	0x004000	0x005FFF	0x00400E	15	0x0E
seg_pmco	Actual Program ready to run	0x006000	0x01FFFF	0x0170DC	69853	0x110DC
seg_pmhi	Area for Patch	0x020000	0x07FFFF		393216	0x60000

Table 2-4 Program Memory Usage

Check RD11or RD13 for an extraction valid for the current revision of the OBS.

3 OBS Objects

The OBS is structured in a series of entities, some of them are defined as Virtuoso's Objects and other entities are defined by the Software Engineer in order to comply the task behaviour or integrate Virtuoso's Object actions in more abstract functionalities.
Those entities are:

Entiy Name/Type	Creator
Tasks	Virtuoso
Memory Pools	IFSI design
FIFOs	Virtuoso
Semaphores	Virtuoso

Table 3-1 – Entity Types and relative definitions

3.1 OBS Tasks

The OBS is structured in a series of Virtuoso Tasks defined in the VIRTUOSO Project file **spire.vpf**. Each task has an associated ID that can be returned as a parameter in special TM (5,1) telemetry packets generated in case of anomalies.
Task IDs are specified as follows:

TASK_ID	TASK NAME
0x0000	INIT
0x0001	TIME_TASK
0x0002	TMTC
0x0003	VM_1
0x0004	VM_2
0x0005	VM_3
0x0006	VM_AFX
0x0007	HS
0x0008	VM_SVC
0x0009	LS
0x000A	CMD_SEQ
0x000B	HK_ASK0
0x000C	HK_ASK1
0x000D	HK_ASK2
0x000E	HK_ASK3
0x000F	HK_MON
0x0010	AUTO_SEQ
0x0011	MEM_DUMP
0x0012	PEAK_UP
0x0013	IDLE

Table 3-2 OBS Task ID definition

3.2 Memory Pools

Due to incorrect Virtuoso's Memory Pools behavior, DPU Memory Pools are now managed by our internal handling procedures. The memory areas in which memory blocks are used to store packets are now statically placed in data memory. Pools IDs can be returned as parameters in special TM (5,1) telemetry packets generated in case of anomalies; values are specified as follows:

POOL_ID	Content Type
0	TC packets
1	Event TM packets
2	HK TM packets
3	Science Data TM packets
4	(RESERVED)
5	Execution Report TM Packets

Table 3-3 Memory Pool ID definition

3.3 FIFOs

Virtuoso FIFOs are message queues used to exchange information between different OBS Tasks. FIFO IDs can be returned as parameters in special TM (5,1) telemetry packets generated in case of anomalies; values are specified as follows:

FIFO ID	FIFO Names	Sender*	Receiver*
0x0	TC_HP_QUEUE	TMTC	CMD_SEQ
0x1	TC_LP_QUEUE	TMTC	CMD_SEQ
0x2	EV_TM_QUEUE	Any	TMTC
0x3	HK_TM_QUEUE	HK_ASK[0/1/2/3]	TMTC
0x4	SD_TM_QUEUE	HS CMD_SEQ VM_[1/2/3/AFX] HardwareVM	TMTC
0x5	RP_TM_QUEUE	CMD_SEQ	TMTC
0x6	LS_HP_QUEUE	VM_[1/2/3/AFX]	LS
0x7	LS_LP_QUEUE	HK_ASK[0/1/2/3]	LS
0x8	AUTO_HP_QUEUE	HK_MON	AUTO_SEQ
0x9	AUTO_LP_QUEUE	HK_MON	AUTO_SEQ
0xA	VM_TM_QUEUE	VM_[1/2/3/AFX] HardwareVM	VM_SVC
0xB	MEM_DUMP_QUEUE	CMD_SEQ	MEM_DUMP

Table 3-4 Virtuoso FIFO ID definition

* Indicative information, for deeper information see AD2.

3.4 Semaphores

Virtuoso SEMAPHORES are used to transfer control between OBS tasks. The following semaphores are implemented in the OBS:

ID	Semaphore Name
0x0	HK_0_SEMA
0x1	HK_1_SEMA
0x2	HK_2_SEMA
0x3	HK_3_SEMA
0x4	LS_SEMA
0x5	TC_READY
0x6	FRAG_SEMA
0x7	AUTO_SEMA
0x8	Reserved (Used for debug)
0x9	PKUP_SEMA

Table 3-5 Virtuoso SEMA ID definition

4 Managing OBS on the DPU

When the DPU is switched on, the Boot SW 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 (5,1) 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. After start-up the OBS is able to modify the running OBS, in the Boot SW there is no implementation of procedure managing OBS images. The OBS can manipulate a clone of the OBS itself, attempt of modification of the running OBS image are dangerous and discouraged.

4.1 Running the EEPROM-resident OBS

The OBS is resident in EEPROM. Two independent partitions are available on the EEPROM and both can store a copy of the OBS image. Once the (5,1) event (with no errors reported) is received by the CDMS simulator, the commands “Force boot Primary” or “Force boot Secondary” described in RD2 can be sent to the DPU; the BSW will copy the OBS from the requested EEPROM partition into 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.

4.2 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.

4.2.1 Generating the telecommands

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.txt*** 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.txt*** file contains the list of memory pages to be avoided (it can be empty).
SPIRE.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 ***path/suffix***TCnnnnn.dm where nnnnn is a count number.

4.2.2 Load the telecommands

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 4 TC/s can be uplinked, then the interval parameter should be set to 250.

4.2.3 Startup the uploaded image

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 uplink 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.

4.3 Verification of the running OBS

The Boot SW verify the integrity of the OBS on the EEPROM and then report it via a Event (5,1) indicating "NO ERROR", it check also the integrity of all TC loading a new OBS image. Once the OBS is running the can calculate a checksum word of the running OBS, and then compare it with the checksum calculated on ground.
Once the OBS in the EEPROM or in the just uploaded Image is launched it is possible generate reports on checksum calculated on-board on the OBS.
More details are demanded in §8.2.
Interest area for verification are reported in §2.6.

NOTE:
There is no way to generate a telemetry report of checksum on the EEPROM.
There is no way to generate a telemetry report of checksum on the uploaded OBS Image.

In the RD11[1] §2.3 are reported the right checksum calculated on the binary at delivery time.

[1] RD11 SPIRE On Board Software Configuration Report

4.4 Modification of the running OBS

Modification of the running image of the OBS are dangerous and strongly discouraged, the interested area is the seg_pmco segment in PM [§2.3 for memory architecture].
Once the OBS is running, the counseled way to actuate modification of OBS is to edit it via the PATCHING TC group (8, 4, CE).
More details are demanded in §10 OBS Patching.

4.5 Store the OBS into EEPROM

The running OBS can be stored in the onboard EEPROM and then reused at the next boot. In order to be bootable the image stored in the EEPROM must contain the OBS program code segment and the global initialization segment [seg_init and seg_pmco, see in §2.3]. The EEPROM handling library automatically embed the startup interrupt vector [seg_rth , see in §2.3].
The nominal way to store OBS in EEPROM is to use the WRITE2EEPROM TC[2] on a memory range that covers the whole seg_init segment and the used part of seg_pmco segment[3].
In the RD11[4] §2.2 is reported the minimal range to store into EEPROM.

4.5.1 Damaged pages on EEPROM

The EEPROM on which the OBS image is stored is split in page, each pages contains a chunk of the OBS image, the coordinates in the program code memory area identifying the data stored, the address of the next page and a checksum word calculated over the page itself.
An inconsistent CRC traps storage error in a page, and the page containing the damage must not be used in order to avoid propagation in program image and therefore the right execution.
The Boot Software loads the OBS reading this chain of linked EEPROM pages.
Once a corrupted page is identified it is possible to skip it when storing the OBS into EEPROM, to skip it just add it in the proper area in the "WRITE2EEPROM TC[5]", and the Boot Software will jump that page during startup.

[2] RD2 SPIRE DATA ICD §**3.2.8.3.27 Function 0xCA DPU, Activity 0x07: WRITE2EEPROM**
[3] See §**4.5.2 "Store the OBS into EEPROM [Example]"**
[4] **RD11** SPIRE On Board Software Configuration Report
[5] RD2 SPIRE DATA ICD §**3.2.8.3.27 Function 0xCA DPU, Activity 0x07: WRITE2EEPROM**

4.5.2 Store the OBS into EEPROM [Example]

An example: with seg_init start in 0x4000 and using seg_pmco ends in address 17100, the WRITE2EEPROM TC shall cover 0x4000-0x17100 address.

Segment Name	Purpose	Start Address	End Address	Length
seg_init	Initialization for Global/Static Variables	0x004000	0x00400e	15
seg_pmco	Program Code storage.	0x006000	0x016dfc	69117

To enforce the example, the primary partition is busy, an old version of the code is left for fault-back environment.

To enforce the example, consider that the page 168 [0xA8] is been identified as damaged due to a broken cell that flip to one after some weeks.

The resulting TC will be:

<table>
<tr><td>0</td><td>0</td><td>0</td><td>1</td><td>1</td><td colspan="11">APID: 0x0500</td></tr>
<tr><td>1</td><td>1</td><td colspan="3">Src</td><td colspan="11">Count</td></tr>
<tr><td colspan="16">Length</td></tr>
<tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td><td>0</td><td>0</td><td>0</td></tr>
<tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr>
<tr><td colspan="8">FUNCTIONID</td><td colspan="8">ACTIVITYID</td></tr>
<tr><td colspan="16">SID=0x001A</td></tr>
<tr><td colspan="16">Start_Address MSW</td></tr>
<tr><td colspan="16">Start_Address LSW</td></tr>
<tr><td colspan="16">End_Address MSW</td></tr>
<tr><td colspan="16">End_Address LSW</td></tr>
<tr><td colspan="16">PARTITION_FLAG</td></tr>
<tr><td colspan="16">JUMP_NPAGES</td></tr>
<tr><td colspan="16">Jump_PageIds</td></tr>
<tr><td colspan="16">Checksum</td></tr>
</table>

Parameters

Name	Value
FUNCTIONID	0xCA
ACTIVITYID	0x07
SID	0x001A
Start_Address	0x00004000
End_Address	0x00016DFC
PARTITION_FLAG	1 = Secondary Partition
JUMP_NPAGES	1

Jump_PageIds	An array of one element:
	{
	168 [0xA8]
	}

Check RD11or RD13 for valid ranges for the current revision of the OBS.

5 Usage of the OBS Services

In the following chapters all OBS services are described.

6 Telecommand Verification 25
7 Housekeeping Data Reporting 25
8 Memory Management 33
9 Table Management 34
10 OBS Patching 38
11 WRITE TO EEPROM 39
12 Virtual Machines 40
13 Monitoring of Housekeeping Parameters 46
14 Sub-System I/F FDIR Management 55
15 OBS Direct Functions 59
16 The Peak-up Operating mode 61
17 The safety procedure GOTO_SAFE_MODE 61

6 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.

7 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 7-1 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.

7.1 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 initialization and complies with requirements contained in RD2.

7.2 Modifying the HK Packet Properties

7.2.1 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.

Note 1:
Due to the strong relation with the Monitoring System, it is not possible to interrupt the HK-Sampling with immediate effect, at least a single repetition period shall be wait before to set a new HK definition.
Note 2:
To change the sampling rate, the repetition of the TC (8,4,0xCC-01) is sufficient and counseled.
A stop and restart is STRONGLY DISCOURAGED.

7.2.2 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 only be allowed if the monitoring task is not active. 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.

7.2.3 Internal HK Parameters

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	
0x40060000	Get Observing Mode	16	See note 2. Note: the Observing mode is stored in the table/ row 6/0.
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 sequence number of last received TC	16	
0x100E0000	Get number of received TC	16	
0x100F0000	Get sequence number of last executed TC	16	
0x10100000	Get number of executed TC	16	
0x04140000	Get S/S I/F monitoring flags	16	See Note 1
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	
0x041F0000	Get Monitoring Function Status	16	- reserved -
0x04210000	Get DPU 5V	16	
0x04220000	Get DPU 15V	16	
0x04230000	Get DPU –15V	16	
0x04240000	Get DPU temperature	16	
0x04270000	Get DPU 2.5V	16	
0x04280000	Get DPU processor workload	16	units per thousand value of 1000 → 100%
0x04290000	Get Autonomy Functions status	16	Flags useful for FDIR See Note 3
0x102A0000	Get DCU science frame counter	16	
0x102B0000	Get MCU science frame counter	16	
0x102C0000	Get SCU science frame counter	16	
0x102E0000	Get Tm Mode	16	Nominal/Burst-Mode 0=Nominal / 1= Burst
0x102F0000	Get LS channel duty time	16	Time in which the LS I/F was busy in the last second, in units of 16μs.

Command ID	Function	Bits	Content
0x04300000	Get Hard VM 0 status	16	The table ID of the currently running code on Virtual Machine. `0xFFFF` If not running.
0x04310000	Get Soft VM 1 status	16	Ibidem for VM1
0x04320000	Get Soft VM 2 status	16	Ibidem for VM2
0x04330000	Get Soft VM 3 status	16	Ibidem for VM3
0x04340000	Get Soft VM AFX status	16	Ibidem for VM AFX
0x10350000	Get FIFO DataFlow Flags	16	The 3 LSB are linked to the three S/S, in the SCU/MCU/DCU order. When expecting data from FIFO they goes to 0 until data are found.
Command ID	**Function**	**Bits**	**Content**
0x103C0000	GetSCPoolStat	16	Free slots in Science Pool
0x103D0000	GetHKPoolStat	16	Free slots in HK Pool
0x103E0000	GetEVPoolStat	16	Free slots in Event Pool
0x103F0000	GetRPPoolStat	16	Free slots in Report Pool
Command ID	**Function**	**Bits**	**Content**

0x04400000	Get_INIT_Task_State	16	INIT task status codified as follow:
			`RUNNING   0x0000`
			`STOPPED   0x0001`
			`ABORTED   0x0003`
			`SUSPEND   0x0004`
			`SLEEPING  0x0010`
			`EVENT_W   0x0080`
			`FIFO_W    0x0200`
			`SEMA_W    0x1000`
			`Unknown   0xFFFF`

0x04410000	Get_TIME_Task_State	16	Ibidem for TIME task
0x04420000	Get_TMTC_Task_State	16	Ibidem for TMTC task
0x04430000	Get_VM_1_Task_State	16	Ibidem for Soft VM1 task
0x04440000	Get_VM_2_Task_State	16	Ibidem for Soft VM2 task
0x04450000	Get_VM_3_Task_State	16	Ibidem for Soft VM3 task
0x04460000	Get_VM_AFX_Task_State	16	Ibidem for Soft VM AFX task
0x04470000	Get_HS_Task_State	16	Ibidem for HS task
0x04480000	Get_VM_SVC_Task_State	16	Ibidem for VM_SVC task
0x04490000	Get_LS_Task_State	16	Ibidem for LS task
0x044A0000	Get_CMD_SEQ_Task_State	16	Ibidem for CMD_SEQ task
0x044B0000	Get_HK_ASK0_Task_State	16	Ibidem for HK_ASK0 task
0x044C0000	Get_HK_ASK1_Task_State	16	Ibidem for HK_ASK1 task
0x044D0000	Get_HK_ASK2_Task_State	16	Ibidem for HK_ASK2 task
0x044E0000	Get_HK_ASK3_Task_State	16	Ibidem for HK_ASK3 task
0x044F0000	Get_HK_MON_Task_State	16	Ibidem for HK_MON task
0x04500000	Get_AUTO_SEQ_Task_State	16	Ibidem for AUTO_SEQ task
0x04510000	Get_MEM_DUMP_Task_State	16	Ibidem for MEM_DUMP task
0x04520000	Get_PEAKUP_SEQ_Task_State	16	Ibidem for PEAKUP_SEQ task
0x04530000	Get_IDLE_Task_State	16	Ibidem for IDLE task
Command ID	**Function**	**Bits**	**Content**
0x10580000	Get_Lost_block_TC	16	Get number of unallocated memory blocks for TeleCommand packets
0x10590000	Get_Lost_block_EV	16	ibidem for Event/Error/Alarm TM packets

Command ID	Function	Bits	Content
0x105A0000	Get_Lost_block_HK	16	ibidem for HouseKeeping TM packets
0x105B0000	Get_Lost_block_SD	16	ibidem for Science TM packets
0x105A0000	Get_Lost_block_NT *(RESERVED for IFSI-Internal purpose)*	16	ibidem for NON- TM temporary packet
0x105B0000	Get_Lost_block_RP	16	ibidem for Report TM packets
Command ID	**Function**	**Bits**	**Content**
0x10600000	Get_DPU_Stat	16	OBS Version
0x10610000	GetTmMode	16	Nominal/Burst Mode [0,1]
Command ID	**Function**	**Bits**	**Content**
0x10620000	Get_Missed_TCHPQ	16	Get number of failed enqueues on Virtuoso FIFO Queue for High Priority TC
0x10630000	Get_Missed_TCLPQ	16	ibidem for Low Priority TC
0x10640000	Get_Missed_EVTMQ	16	ibidem for Event TM packets
0x10650000	Get_Missed_HKTMQ	16	ibidem for HK TM packets
0x10660000	Get_Missed_SDTMQ	16	ibidem for Science TM packets
0x10670000	Get_Missed_RPTMQ	16	ibidem for Report TM packets
0x10680000	Get_Missed_LSHPQ	16	ibidem for High Priority LS Commands
0x10690000	Get_Missed_LSLPQ	16	ibidem for Low Priority LS Commands
0x106A0000	Get_Missed_AUTOQ	16	ibidem for High Priority calls to Autonomy Function
0x106B0000	Get_Missed_VMTMQ	16	ibidem for internal VM calls to OBS
0x106C0000	Get_Missed_MEDUQ	16	ibidem for calls to Memory Dump procedure
Command ID	**Function**	**Bits**	**Content**
0x04760000	Get_FifoStat_TC_HP	16	Items en-queued for a processing, number of: High-Priority TC
0x04770000	Get_FifoStat_TC_LP	16	Ibidem for Low-Priority TC
0x04760000	Get_FifoStat_TM_EV	16	Ibidem for Events/ Errors/ Alarm TM
0x04770000	Get_FifoStat_TM_HK	16	Ibidem for House Keeping TM
0x04760000	Get_FifoStat_TM_SD	16	Ibidem for Science Data TM
0x04770000	Get_FifoStat_TM_RP	16	Ibidem for Report TM
0x04760000	Get_FifoStat_LS_HP	16	Ibidem for High Priority S/S Command
0x04770000	Get_FifoStat_LS_LP	16	Ibidem for Low Priority S/S Command
0x04780000	Get_FifoStat_AFX	16	Ibidem for Autonomy Functions Request
0x04790000	Get_FifoStat_VM	16	Ibidem for VM Telemetry
0x047A0000	Get_FifoStat_MEMD	16	Ibidem for Memory Dump Request
Command ID	**Function**	**Bits**	**Content**
0x04800000	Get ID of selection Table for Frame ID 0x0	16	no_selection=0xFFFF
0x04810000	Get ID of selection Table for Frame ID 0x1	16	no_selection=0xFFFF

Command ID	Function	Bits	Content
0x04820000	Get ID of selection Table for Frame ID 0x2	16	no_selection=0xFFFF
0x04830000	Get ID of selection Table for Frame ID 0x3	16	no_selection=0xFFFF
0x04840000	Get ID of selection Table for Frame ID 0x4	16	no_selection=0xFFFF
0x04850000	Get ID of selection Table for Frame ID 0x5	16	no_selection=0xFFFF
0x04860000	Get ID of selection Table for Frame ID 0x6	16	no_selection=0xFFFF
0x04870000	Get ID of selection Table for Frame ID 0x7	16	no_selection=0xFFFF
0x04880000	Get ID of selection Table for Frame ID 0x8	16	no_selection=0xFFFF
0x04890000	Get ID of selection Table for Frame ID 0x9	16	no_selection=0xFFFF
0x048A0000	Get ID of selection Table for Frame ID 0xA	16	no_selection=0xFFFF
0x048B0000	Get ID of selection Table for Frame ID 0xB	16	no_selection=0xFFFF
0x048C0000	Get ID of selection Table for Frame ID 0xC	16	no_selection=0xFFFF
0x048D0000	Get ID of selection Table for Frame ID 0xD	16	no_selection=0xFFFF
0x048E0000	Get ID of selection Table for Frame ID 0xE	16	no_selection=0xFFFF
0x048F0000	Get ID of selection Table for Frame ID 0xF	16	no_selection=0xFFFF
Command ID	**Function**	**Bits**	**Content**
0x04900000	Get ID of selection Table for Frame ID 0x10	16	no_selection=0xFFFF
0x04910000	Get ID of selection Table for Frame ID 0x11	16	no_selection=0xFFFF
0x04920000	Get ID of selection Table for Frame ID 0x12	16	no_selection=0xFFFF
0x04930000	Get ID of selection Table for Frame ID 0x13	16	no_selection=0xFFFF
0x04940000	Get ID of selection Table for Frame ID 0x14	16	no_selection=0xFFFF
0x04950000	Get ID of selection Table for Frame ID 0x15	16	no_selection=0xFFFF
Command ID	**Function**	**Bits**	**Content**
0x04A00000	Get ID of selection Table for Frame ID 0x20	16	no_selection=0xFFFF
0x04A10000	Get ID of selection Table for Frame ID 0x21	16	no_selection=0xFFFF
Command ID	**Function**	**Bits**	**Content**
0x10FF0000	Dummy place holder	16	* Always 0 (zero)
0x40yyzzzz	Get the less significant 16 bits in a table row.	16	See note 2 Table ID: ***yy*** Offset: ***zzzz*** 0XFFFF on error
0x50yyzzzz	Get the most significant 16 bits in a table row.	16	See note 2 Table ID: ***yy*** Offset: ***zzzz*** 0XFFFF on error

Table 7-2 Commands to get DPU HK parameters

7.2.3.1 Note 1: Internal Hk Parameter

[0x04140000-Get S/S I/F monitoring flags]
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) :

Bits	Interface Type	Values
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** Low-Speed I/F status	• "00" → ALIVE • "01" → SICK • "10" → DEAD
7-8	**MCU** High-Speed I/F mode	• "00" → NOMINAL • "01" → TRANSPARENT
9-10	**SCU** Low-Speed I/F status	• "00" → ALIVE • "01" → SICK • "10" → DEAD
11-12	**SCU** High-Speed I/F mode	• "00" → NOMINAL • "01" → TRANSPARENT

Table 7-3 Subsystem Status Bits

7.2.3.2 Note 2: OBS Hk commands

1 [0x40YY*zzzz* Read a table row LSW]
2 [0x50YY*zzzz* Read a table row MSW]
This command is used to insert into HK data coming from other tables defined in the OBS session.

Item Schema	40/41	YY	ZZZZ	
Bits	31-24	23-16	15-13	12-00

Bits	Description	Values
31-24	Constant KEY value 0x40	40/50
23-16	YY Table ID 0x00←→0xFF	The **Id** of the table from which the data is read. If not defined will be reported in the housekeeping the value **0xFFFF**
15-13	ignored	
12-00	ZZZZ Table Offset 0x0000←→0x1FFF	The **Offset** of the data to read in the table. If outside the size of the table, will be reported in the housekeeping the value **0xFFFF**

Table 7-4 HK from data in Table

7.2.3.3 Note 3: Monitoring and Autonomy Action flags

This command is used to insert into HK information gathered from various task, involved in the FDIR flow.

NOTE:

NO ACTION CAN BE TRIGGERED BY HK_MON STATUS “EQUAL-TO” 0x00.

The value mapping is in the following table:

Bits	Description	Values	Interpretation
1-0	HKMON_STAT	0/1/3	0: Monitor System Stopped 1: Monitoring system Running 3: monitoring System Suspended
3-2	AFX_RUNNING	0/1/2/3	0: No 1: HW AFX_RUNNING 2: SW AFX_RUNNING 3: ANY AFX_RUNNING
4	TC_DROP	0/1	0: Nominal STATUS 1: OBS is dropping execution of TC
5	HW_VM_RUN	0/1	1: HW VM is Running 0: otherwise
6	HW_VM_LS_MTX	0/1	LS pre-emption mutex 1: HW VM is locking access 0: otherwise
7	PEAK_UP_RUN	0/1	1: Peak Up is Running
-	-	-	-
8	HW_VM CIS OVERRIDE	0/1	1: Command Inhibition System Switched Off 0: otherwise
9	VM_AFX CIS OVERRIDE	0/1	1: Command Inhibition System Switched Off 0: otherwise
F-A	DUMMY	0	

Table 7-5 HK from data in Table

8 Memory Management

8.1 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.

8.2 Program Memory verification

Diagnostic procedure for integrity of the hardware memory is not provided in OBS. Detailed analysis on the OBS image on board can be done via memory dump telecommands, that directly scopes inside the memory. The integrity of the SPIRE OBS can be verified comparing a checksum calculated on the image onboard with the one provided at delivery. The checksum can be retrieved by two ways: via the TC(6,9) `Mem_Check_Using_Abs_Addr` and via the TC(8,4,0xCA,0x15) `PM_MemoryCheck`. The counselled command is the TC(6,9) , it doesn't follow an instrument dependant structure. See RD11 for official delivery checksum.

8.3 Data Memory Hardware verification

SPIRE OBS can verify the integrity of the Data Memory hardware cells, sending a TC (8,4,0xCA,0x16) `DM_MemoryCheck` SPIRE OBS at low priority it iterates over the whole Data Memory checking cell by cell. Each check is done by reading the data in the cell, rewrite the negation of the data, verify if the data is been wrote correctly, then restore back the word read, each of this iteration is done in a freezed context with the interrupt inhibited, time safety for other functionality in OBS is granted by the enclosure of the previous algorithm in a low level assembly routine wrote directly in ADI DSP assembly.

9 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.

9.1 Default Tables in the OBS

This is the list of predefined tables available in the OBS at start-up. Those tables which are listed as not suitable for flight operations will have to be reloaded after the start-up.

Table ID [DEC]	TID [HEX]	Length (32-bit words)	Source	Contents/
0	0x00	23	RAL	Essential HouseKeeping TM packet definition
1	0x01	374	RAL	Nominal HouseKeeping TM packet definition
2	0x02	23	IFSI	Diagnostic HouseKeeping TM packet definition - predefined to report the status of the OBS Tasks
3	0x03	6	IFSI	Diagnostic HouseKeeping TM packet definition - predefined to report the status of the OBS Memory Pools
5	0x05	89	IFSI RAL	Monitoring Table definition – Not suitable for flight operations (Dummy)
6	0x06	36		OBS Configuration Table
7	0x07	32	RAL	[RAL] – VM HK storage area, v1.0
10	0x0A	12	RAL	[RAL] – Base SMEC Selection, v1.0
50	0x32	17	RAL	[RAL] – Jiggle Map 7, v1.4 (includes END word) – Not suitable for flight operations
51	0x33	129	RAL	[RAL] – Jiggle Map 64, v 1.1 (includes END word) – Not suitable for flight operations
52	0x34	2	RAL	[RAL] – S-Map 01, v1.0 (includes END word) – Not suitable for flight operations
53	0x35	17	RAL	[RAL] – S-Map 016, v1.0 (includes END word) – Not suitable for flight operations
54	0x36	2	RAL	[RAL] – S-Map 04, v1.0 – Not suitable for flight operations (Dummy)
64	0x40		IFSI	Auto Suspend VM 0 (size = 1)
65	0x41		IFSI	Auto Suspend VM 1 (size = 1)
66	0x42		IFSI	Auto Suspend VM 2 (size = 1)
67	0x43		IFSI	Auto Suspend VM 3 (size = 1)
68	0x44		IFSI	Auto Suspend VM AFX (size = 1)
70	0x46		RAL	[RAL] – Flash , v1.3 – Not suitable for flight operations
71	0x47		RAL	[RAL] – Chop, v1.4 – Not suitable for flight operations
72	0x48		RAL	[RAL] – JiggleMap, v1.4 – Not suitable for flight operations
73	0x49		RAL	[RAL] – BSM Move, v1.0 – Not suitable for flight operations
74	0x4A		RAL	[RAL] – Step and Chop, v1.2 – Not suitable for flight operations
80	0x50		RAL	[RAL] – SCAL, v2.1 – Not suitable for flight operations (contains an error)
81	0x51		RAL	[RAL] – PTC, 1.10 – Not suitable for flight operations
82	0x52	64	RAL	[RAL] – VMTM, v1.0
83	0x53		RAL	[RAL] – CREC, v1.7 – Not suitable for flight operations
100	0x64			[RAL] – Functions, v1.4 – Not suitable for flight operations
212	0xD4	9	IFSI	VM Procedure to go into SAFE MODE – Not suitable for flight operations (Dummy)
213	0xD5	14	IFSI	VM Procedure to Inhibit commanding to MCU – Not suitable for flight operations (Dummy)
248	0xF8	64	IFSI	DCU Command Inhibition Table (size = 64)
249	0xF9	64	IFSI	MCU Command Inhibition Table (size = 64)
250	0xFA	64	IFSI	SCU Command Inhibition Table (size = 64)
253	0xFD		-	Table to temporarily hold VM code uplinked via the *Execute_Command_List* TC
254	0xFE		IFSI	**MOAT SCHEMA REFLECTION Mirror** **[WARNING DON'T MODIFY]**
255	0xFF		IFSI	**MOAT SCHEMA REFLECTION Primary** **[WARNING DON'T MODIFY]**
256	0x100		IFSI	***Embedded On Board Function*** ***Unexistent Virtual Table. See §12.1***

Table 9-1 On Board Default Tables pre-allocated at boot-time.

9.2 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 (**8192** decimal).

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.

9.3 Table Update

To update an existing table it is sufficient to send a TC (8,4, 0x01-0x03) as specified in section “b” of 9.2.

9.4 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, VM code, Monitoring, etc …) a TM (1,8) will be issued.

9.5 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.

10 OBS Patching

For maintenance purpose a basic patching module is embedded in the system[6]. In the architecture of the DPU there is the definition of the `seg_pmhi` segment, this reserved area of the PM is used as desk-bench for editing the OBS images [see occupation at §2.6]. The patching procedure consists mainly in three phases:

- Clone of the OBS: from the current executable PM area to the editing workplace that also reside in PM area; this is done via the function `Execute_Patching` TC(8,4,CE,02,Direction=1);
- Edit of the Clone: this is done via a set functions (`Memory Load` TC(6,x)) manipulating the clone of the OBS image stored in the dedicated PM area `seg_pmhi`
- Rewrite the OBS: this is done via the function `Execute_Patching` TC(8,4,CE,02,Direction=2).

If the user try to apply patches prior to clone the actual OBS image a TM(1,8) is issued.

Address	.	0 – BASE OBS	1 – AFTER CLONE “Execute_Patching” TC (8, 4, CE, 02, Direction = 1 , address = 0x20000, nWord = 0x7003)	2 – AFTER EDITING IMAGE “Mem_Load” TC(6, 2, Address = 0x27001, nWord = 2, data = { 0xEEEE, 0xFFFF })	3 – AFTER Commit and reboot. “Execute_Patching” TC (8, 4, CE, 02, Direction = 2 , address = 0x20000, nWord = 0x7003)
0x000000		0xGGGG	0xGGGG	0xGGGG	0xGGGG
...		...	...	...	...
0x007000		0xAAAA	0xAAAA	0xAAAA	0xAAAA
0x007001		0xBBBB	0xBBBB	0xBBBB	0xEEEE
0x007002		0xCCCC	0xCCCC	0xCCCC	0xFFFF
0x007003		0xDDDD	0xDDDD	0xDDDD	0xDDDD
0x020000		n/a	0xGGGG	0xGGGG	0xGGGG
...		...	...	...	...
0x027000		n/a	0xAAAA	0xAAAA	0xAAAA
0x027001		n/a	0xBBBB	0xEEEE	0xEEEE
0x027002		n/a	0xCCCC	0xFFFF	0xFFFF
0x027003		n/a	0xDDDD	0xDDDD	0xDDDD

Table 10-1 Patching example

[6] See also RD12 Memory Management Library Interface Version 1.1

11 WRITE TO EEPROM

See at §4.5 Store the OBS into EEPROM

12 Virtual Machines

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 organized in TC (8,4, 0x01-0x03) ready to be sent to the DPU.

NOTES:

- ***i.*** ***H/W VM always resets the fifos before starting***
- ***ii.*** H/W and S/W VMs handle the insertion of commands into the LS command stream
  - ***a.*** H/W VM locking the port access for LS TASK through a mutex.
  - ***b.*** S/W VM pushing command into an High Priority Queue served before the one used by HK Collectors.

12.1 Embedded On Board Functions for Virtual Machines Programs

Some OBS functionality can be accessed by VM program. The Table 256 (0x100) contains a set of entry points these functions, VM program can call them via an indirect subroutine call (*IR-CALL*), passing arguments i registries starting from R0, the expected returning values are stored in registries starting from R0.

NOTE: As placeholder for unassigned ID a dummy function return the Caller Table ID in R0.

- See next page -

Entry #	Name	Arguments	Returns	Notes
0x00	No Action	N/A	N/A	No Action[7]
0x01	Disable irq0 sensing	N/A	N/A	See §12.1.1
0x02	Enable irq0 sensing	N/A	N/A	
0x03	Disable irq2 sensing	N/A	N/A	
0x04	Enable irq2 sensing	N/A	N/A	
0x05	Disable irq3 sensing	N/A	N/A	
0x06	Enable irq3 sensing	N/A	N/A	
0x07	Set Inhibition for Single Command	R[0] Command R[1] 0 – Allow 1 – Inhibit	N/A	See §12.1.1
0x08	Read Inhibition for Single Command	R[0] Command	R[0] 0 Allow 1 Inhibit	
0x09	Set Inhibition for Command Group [in a defined range]	R[0] First Command R[1] Last Command R[2] Selection Mask R[3] Selection Pattern R[4] 0 – Allow 1 – Inhibit	N/A	
0x0A	Flush All Fifo	R[0] Fifo Selection	N/A	See §15.1.2
0x0B	Reset All Fifo	R[0] Fifo Selection	N/A	See §15.1.2
0x0E	Execute Autonomy Action	R[0] Autonomy Action	N/A	See §12.3
0x0F	Packet Transfer Control Enable/Disable output of TM packets	R[0] Packet Type R[1] Packet Subtype R[2] SID/EventID R[3] Mode (0=Disable, 1=Enable)	R[0] Result 0x0E01 = Type not found 0x0E02 = Sub-type not found 0x0E03 = SID/EID not found	*Note*: *sub-Type is stored in the left octet. i.e. {1/2/4} shall be 0x{1/2/4}00*
0x10	Float ADD	R[0] Left Operand R[1] Right Operand	R[0] Result R[1] Status Flags	See §12.1.3
0x11	Float SUB			See §12.1.3
0x12	Float MUL			See §12.1.3
0x13	Float DIV			See §12.1.3 See Note 4
0x18	Float "is Greater"	R[0] Left Operand R[1] Right Operand	R[0] Result	See §12.2.2
0x19	Float "is Lower"			
0x1A	Float "is Equal"			
0x1B	Float "is NAN"			
0x1C	Float "to Integer"	R[0] Float	R[0] Integer	
0x1D	Float "from Integer"	R[0] Integer	R[0] Float	
0x1E	GOTO SAFE MODE	N/A	N/A	See §13.4.3
0x1F	THROW EVENT	R[0] TM Sub Type [1/2/4] R[1] Event ID R[2] Event SID R[3] Numer of Parameter R[4] 1st Parameter R[5] 2nd Parameter … R[4+n] nth Parameter	N/A	All registers are interpreted as 16 bits wide word aligned to the right 16 LSBits. *Note*: *sub-Type is stored in the left octet. i.e. {1/2/4} shall be 0x{1/2/4}00*

Table 12-1 OBS Functions accessible from VM programs

[7] *a dummy function that returns the Caller Table ID in R0*

12.1.1 Setting IRQ sensing state

Interrupt sensing enabling/disabling affect system behavior, use carefully.
Enabling/Disabling irq0 affects science data acquisition.
Enabling/Disabling irq2 affects communication with CDMS.
Enabling/Disabling irq2 affects Interrupt Driven Virtual Machine.

12.1.2 Subsystems Commanding Inhibition

See §15.3 for more explanation.

To set or unset the inhibition status of a single command:

Set the R0 with the command	R0 = 0x84320000;
Set the R1 with the inhibition status	R1 = 0; // Allow
Call the OBS Set Inhibition Status Command	IRCALL 0x100, 0x07;

To read the inhibition status of a single command:

Set the R0 with the command	R0 = 0x84320000;
Call the OBS Read Inhibition Status Command	IRCALL 0x100, 0x08;
Read from R0 the Inhibition Status	IRCALL 0x100, 0x08;

To set or unset the inhibition status of a range command:

Set the R0 with the first command in the range	`R0 = 0x84320000;  //`
Set the R1 with the last command in the range	`R1 = 0x84FF0000;  //`
Set the R2 with a selection mask	`R2 = 0xFF000000;  //`
Set the R3 with a selection pattern	`R3 = 0x84000000;  // Anything in range`
Set the R4 with the inhibition status	`R4 = 0;           // Allow`
Call the OBS Set Inhibition Command	`IRCALL 0x100, 0x09;`

12.1.3 Note 3:

Operator	Pseudo-Algebraic Equivalent.	
ADD	R0 = R0 + R1;	R1 = <status>
SUB	R0 = R0 - R1;	R1 = <status>
MUL	R0 = R0 * R1;	R1 = <status>
DIV	R0 = R0 / R1;	R1 = <status>

Table 12-2 VM/OBS Functions I/O registries

R[1] is used as operation status flags storage.
See §12.2 for more explanation.
See §12.2.1 for more explanation.

12.1.4 Note 4:

If the Right Operand is equal to 0 zero, the Division Zero flag is raised.

12.1.5 Note 5:

If the Right Operand is equal to 0 zero, the Division Zero flag is raised.

12.2 Floating Point Numerical format

Floating Point OBS Functions for VM Programs uses IEEE 754 Numerical Representation for Floating Point Variables:

	Sign Bit [31]	Exponent [30-23]	Mantissa [22-00]	Constant
Normal	S := 0/1 [Positive/ Negative]	E := -126←→127 in (x-127) Biased Format	M := 1.ffffff 22 Bit Fractions.	
ZERO	0	0	0	0x00000000
Negative ZERO	1	0	0	0x80000000
Positive Infinite	0	256 – 0xFF	0	0x7F800000
Negative Infinite	1	256 – 0xFF	0	0xFF800000
Not A Number	any	256 – 0xFF	any	0xFFFFFFFF

Table 12-3 VM/OBS Functions Constant Values

12.2.1 Floating Point operations

Basic algebra operation are provided, Addiction, Subtraction, Multiplication, Division.
The input must be already in floating point format, to obey to this see $12.2.3
After each call the R[0] contains the operations result, and R[1] contains operation status flags, see §12.2.4.

12.2.2 Floating Point comparators

Basic comparison operators are provider: '>' Greater Than, '<' Lower Than and '=' Equal To.
An additional comparator is provided: is 'isNaN' "is Not a Number" that checks if the given floating point number represent a valid real value.

Operator	Pseudo-Algebraic Equivalent.
`Is Greater`	$R0 = \begin{cases} 1 & \text{if } (R0 > R1) \\ 0 & \text{if } (R0 \le R1) \end{cases}$
`Is Lower`	$R0 = \begin{cases} 1 & \text{if } (R0 < R1) \\ 0 & \text{if } (R0 \ge R1) \end{cases}$
`Is Equal`	$R0 = \begin{cases} 1 & \text{if } (R0 = R1) \\ 0 & \text{if } (R0 \ne R1) \end{cases}$
`Is NAN`	$R0 = \begin{cases} 1 & \text{if } (R0 \notin \Re) \\ 0 & \text{if } (R0 \in \Re) \end{cases}$

Table 12-4 VM/OBS Comparison Functions Behaviour

12.2.3 Floating Point conversions

Basic conversion function are provided: Float From and To Integer.
The R[0] will contains the converted value, and R[1] contains operation status flags, see §12.2.4.
Register holds natively integer values, to interface floating point represented values with integer use the following converters:

Operator	Pseudo-Algebraic Equivalent.

To Integer	R0 = Floating Point representation
	R1 = <status>;

From Integer	R0 = Floating Point representation
	R1 = <status>;

Table 12-5 VM/OBS Converstion Functions Behaviour

12.2.4 Floating Point operation status.

Flags position and meaning in R[1] after call:

31-12	11	10	9	8	7	6	5	4	3	2	1	0
Ignored	0	0	CU	CV	0	0	ON	OV	0	IZ	IV	IN

Table 12-6 Floating Point operation status bit mapping

	Mean	Explanation
IN	Input Operand/s is not a number.	An operand is NAN
IV	Input Infinite.	An operand is an Infinite
IZ	Divisor is zero.	Divisor is ZERO
ON	Output is “NaN” Not a Number.	Result is NAN
OV	Output is Infinite.	Result is an Infinite
CV	Conversion Overflow	The value cannot be represented in 31Bit Signed Integer Format.
CU	Conversion Underflow	The value cannot be represented in 31Bit Signed Integer Format.

Table 12-7 Floating Point operation status bit semantics

12.3 Autonomy Action Format

The Autonomy Sequence Task AUTO_SEQ interprets autonomy action delegated from other tasks. An autonomy action is codified in a single 32bit word and each bits meaning is described in the following schema[8].
Note: The HK Monitoring Task HK_MON activates different autonomy actions in different state transition of a single monitored HK parameter, hence the user my set different configuration for the specified parameter transition:

31	30	29	28	27-26	25	24	23-16	15-0
Enabled	Throws Event	autonomy action	0 OBS FX 1 VM call	(0)	1 Hard 0 Soft	(0)	ID	Param
1	0	1	1/0	0	1/0	0	ID	PARAM

Table 12-8 Single Autonomy Action format – 32 bit word

Bit 31 → Enabled/Disabled (1/0) – Enable the current action
Bit 30 → *Enabled/Disabled 0 – No event will be thrown - (set to 0)*
Bit 29 → Enabled/Disabled (1/0) - The execution of an Autonomy Function
Bit 28 → Enabled/Disabled (1/0) - The execution of a VM program or an OBS Function.
Bit 27-26 → *Reserved- (set to 0)*
Bit 25 → Hard_VM/Soft_VM (1/0) - Action VM program will run on the Hard_VM or on the AFx dedicated Soft_VM.
Not applicable if [BIT-28] indicates an OBS Function.
Bit 24 → *Reserved (set to 0)*
Bit 23-16 → ID of the VM program or OBS Function to be executed
Bit 15-0 → Parameter to be passed to the VM program or OBS Function

[8] Schema extracted from paragraph §13.3.7

13 Monitoring of Housekeeping Parameters

13.1 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 monitoring of 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.

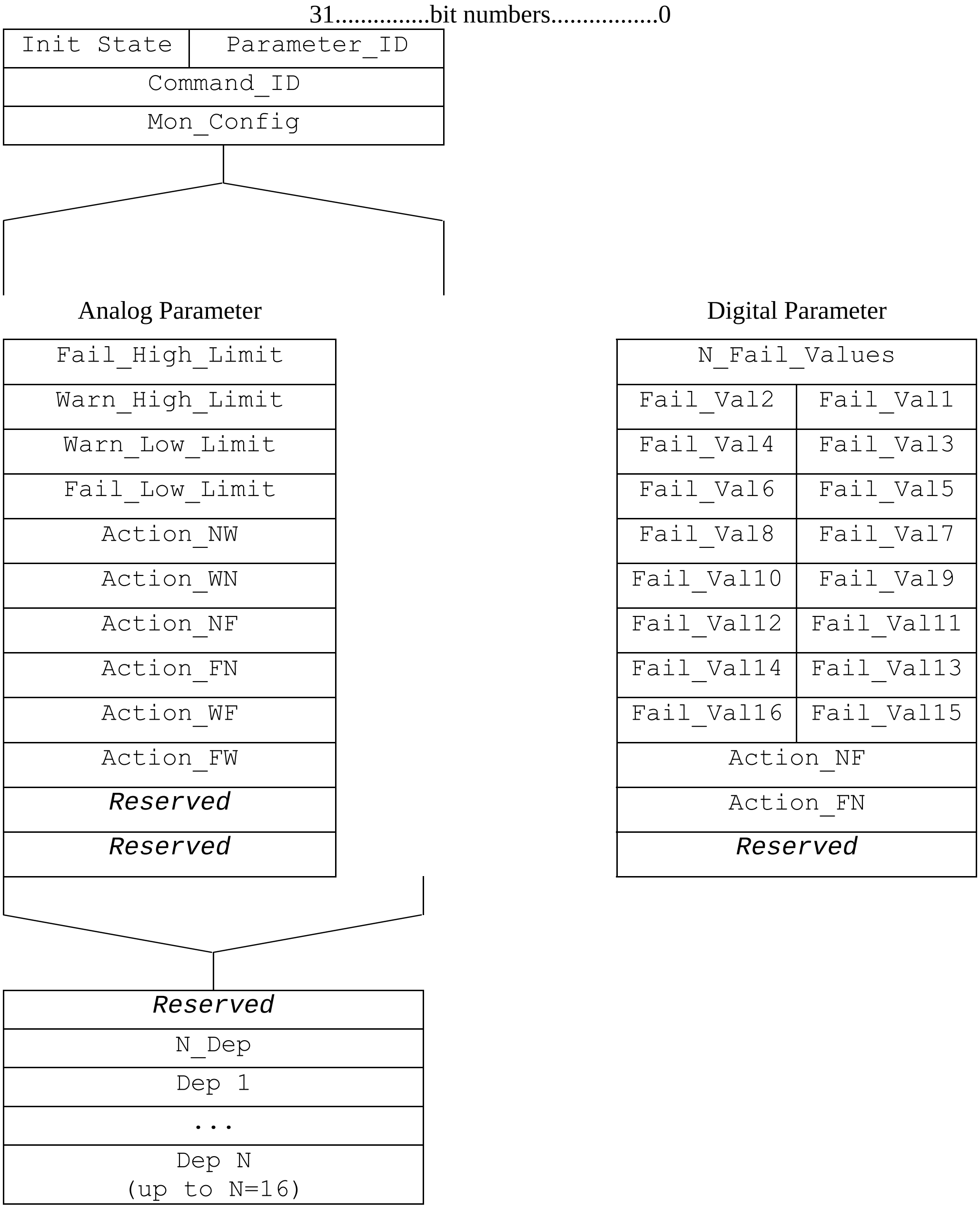
Two types of parameters are supported by the monitoring system: ANALOG parameters that can take a continuous range of values, and DIGITAL parameters that can only assume a discrete number of values.

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 realized 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 of these states (6 combinations in total). A separate behavior can be configured for each transition. As a default condition, the monitoring system IS NOT active at startup of the OBS, and it must be explicitly activated with the Telecommand Start_Monitoring.

13.2 How to Configure the Monitoring System

The configuration settings for the monitoring system must be stored in an 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.



Init State	Parameter_ID
Command_ID	
Mon_Config	

Analog Parameter

Fail_High_Limit
Warn_High_Limit
Warn_Low_Limit
Fail_Low_Limit
Action_NW
Action_WN
Action_NF
Action_FN
Action_WF
Action_FW
Reserved
Reserved

Digital Parameter

N_Fail_Values	
Fail_Val2	Fail_Val1
Fail_Val4	Fail_Val3
Fail_Val6	Fail_Val5
Fail_Val8	Fail_Val7
Fail_Val10	Fail_Val9
Fail_Val12	Fail_Val11
Fail_Val14	Fail_Val13
Fail_Val16	Fail_Val15
Action_NF	
Action_FN	
Reserved	

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

Table 13-1 Single Autonomy Action format – 32 bit word

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 §9.2).

13.3 Argument defining a monitored parameter

Here follows the detailed explanations of each field in the above structure:

13.3.1 Parameter

Bits:

31-28	27-24	23-20	19-16	15-12	11-8	7-4	3-0
Initial State		Monitoring Identifier					

Table 13-2 – Monitoring Item State and ID

Bit 31-24 → The Initial State of the parameter [0:Normal, 1: Warning, 2: Failure]
Bit 23-0 → It is a unique identifier for the monitoring item.

13.3.2 Command_ID

Bits:

31-28	27-24	23-20	19-16	15-12	11-8	7-4	3-0
32 Bit Wide HK COMMAND							

Table 13-3 – Monitored HK parameter

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 §7.2.2 should be used for DPU HK parameters.

13.3.3 Mon_Config

Bits:

31	30	29-22	21-20	19-16	15-12	11-8	7-4	3-0
Ena Dis	Ana Dig	Retry Limit	Word Offset	Bitwise Offset	Bitwise Data Mask			

Table 13-4 – Monitoring item configuration

Bit 31 → Enable/Disable (1/0) monitoring of this parameter [9] [10]
Bit 30 → Analog/Digital parameter (1/0)
Bits 29-22 → Retry Limit for transition sensing
Bits 21-20 → Number of words offset from where extract the parameter

- When using 16Bit wide perameters:
  this value must be set to "0" Zero.
- When using 32Bit wide parameters:
  the MSWord is referred by "0" Zero and
  the LSWord is referred by "1" One;
- When using 48Bit wide parameters:
  the MSWord is referred by "0" Zero ,
  the MidSWord is referred by "1" One and
  the LSWord is referred by "2" Two;

Bits 19-16 → Number of bits the parameter is bitwise right-shifted Note 2 and 3
Bits 16-00 → Extraction mask to be applied to the parameter Note 2 and 3

[9] See also at *"Note on dependency.§ 13.3.10"*
[10] See also at *"Note 4: enhancing VM code managing monitoring items. § 13.3.3.4"*

13.3.3.1 Note 1: Analogic parameters are handled as Signed Value

For Analog Parameters: when the 16th (sixteenth) Bit is asserted in the masked parameter the monitoring module will expand it as sign bit. i.e.:

- < Mask `0x7FFF` ; Parameter read `0xFFFF` > : → valued compared as `0x00007FFF - (32767)`
- < Mask `0xFFFF` ; Parameter read `0xFFFF` > : → valued compared as `0xFFFFFFFF - (-1)`

13.3.3.2 Note 2: Parameters' value masking and shifting

The masking/shifting action is done in the following order:

- The Parameter Value is BIT-WISE-MASKED with the given mask.
- The resulting value is then BIT-WISE-RIGHT-SHIFTED by the given count.

13.3.3.3 Note 3: Analogic Parameters with Unsigned value

If there is the needing to monitor an 16Bit wide unsigned parameter there is the possibility to avoid this by losing a bit in precision, removing the LSBit, and then monitoring it with a 15bit precision.
As example:

- <MASK 0xFFF7, Shift 0x01 (on right), parameter read 0x89AB> →
  → value compared will be 0x000044D5 - that will be compared with limits right-shifted by one.

13.3.3.4 Note 4: enhancing VM code managing monitoring items.

A different way to switch On/Off a single monitoring Item, and the referencing tree, is been used avoiding the control of the BIT 31.
Since Spire OBS Revision 4.0.0 this control is been avoided/removed, in order to promote the concept described in "*Note on dependency.§ 13.3.10*"

13.3.4XXX_YYY_Limits.

Soft and Hard limits for analog parameters
Fail_Low < Warn_Low < Normal < Warn_High < Fail_High

Fail_Low Limit	Warn_Low Limit	Normal Range	Warn_High Limit	Fail_High Limit
Failure	Warning	Normal	Warning	Failure

Table 13-5 – Monitored HK Value Ranges

Note:
The WARNING/FAILURE state is triggered when the value is equal to limit:

{Limit} <= {Value} <= {Limit}

The NORMAL state is triggered when the value is inside the boundary limits:

{Limit} < {Value} < {Limit}

13.3.5N_Fail_Values.

Number of discrete values for which a digital parameter should be considered in a FAIL state

13.3.6Fail_Val [1 to 16].

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.

13.3.7 Action_XY.

Configuration settings for the specified parameter transition:
Bits:

31	30	29	28	27	26	25	24	23-16	15-0
Enabled	Throws Event	Trigger action	0 OBS FX 1 VM call	(0)	(0)	1 Hard 0 Soft	(0)	ID	Param

Table 13-6 – Autonomy Action

Bit 31 → Enabled/Disabled (1/0) – Enable the current action
Bit 30 → Enabled/Disabled (1/0) – Enable the generation of a TM event (5,1)
Bit 29 → Enabled/Disabled (1/0) – Enable the execution of an Autonomy Function
Bit 28 → Enabled/Disabled (1/0) – Switch between a VM program or an OBS Function.
Bit 27-26 → Reserved (set to 0)
Bit 25 → Hard_VM/Soft_VM (1/0) - Action VM program will run on the Hard_VM or on the AFX dedicated Soft_VM.
Not applicable if [BIT-28] indicates an OBS Function.
Bit 24 → Reserved (set to 0)
Bit 23-16 → ID of the VM program or OBS Function to be executed
see also §13.4 VM Tables and OBS Functions for Autonomy Services
Bit 15-0 → Parameter to be passed to the VM program or OBS Function

13.3.8 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.

13.3.9 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.

13.3.10 Note on dependency.

The Enabling/Disabling of a single monitoring definition is been done using mainly the Boolean switch in “Mon_Config §13.3.3”, during optimization of VM programs used for Autonomy a different way is grown.
The new concept is to avoid the control of that Boolean Switch and add a dependency ID to well known Monitoring ID, then two additional monitoring items were defined to be EVER TRUE AND EVER FALSE: basically those was built by Digital HK-Monitoring definition using a bitmask of full zeroes 0x0000 and a fail value of ZERO or ONE, by the case to generate a ALWAYS TRUE or FALSE condition.
In this scenario the Enabling/Disabling of a single HK_Monitoring Item is done by writing a MonitoringID referring the ALWAYS-TRUE/FALSE HK_Monitoring Item.
This has some size impact over the monitoring definition table simplifying the VM code accessing that one [write the Monitoring ID in a fixed displacement], otherwise a 3-operations access should be done [1-Read the Configuration, 2-Bitwise AND/OR masking, 3- Store back the configuration.

13.4 VM Tables and OBS Functions for Autonomy Services

When `Bit 28` of the monitoring item configuration[11] is set (VM call) the `ID` field in the same configuration indicates the Table ID to be run, and the start address imposed as default is 0 zero, then the `param` field is passed to the VM in the register `0 R[0x00]`.
Otherwise if the `Bit 28` of the monitoring item configuration[12] is not set (OBS FX) the `ID` field in the same configuration indicates which OBS functionality, then the `param` field is passed to the OBS function as unique integer argument.

Here is the list of the OBS functionality accessible actually implemented:

ID	Name	Description
1	(reserved)	DON’T USE
2	(reserved)	DON’T USE
3	Abort Peak UP	13.4.1 - Abort Peak-UP
254	SAFE VM	13.4.2 - Safe VM Environment
255	SAFEMODE	13.4.3 - Goto SAFE Mode

Table 13-7 – Autonomy OBS Functions

.

13.4.1 Abort Peak-UP

Calling this function the OBS abort a running peak up procedure

13.4.2 Safe VM Environment

Calling this action the OBS schedule for a VM execution. The given field `param` is used to identify the right VM Table-ID to run, the entry point of the table is set to zero as default, and the remnant part of the field is passed at the VM as argument in Register 0.

```
Param Field Partitions:
0xYYZZ    16 Bits   Content
0xYY..    8 Bits    Table-ID to call
0x..ZZ    8 Bits    Argument copied in R[0x00]
```

The execution of the given VM is done in a “SAFE” context with other services switched down.

[11] Bit assignment at Error! Reference source not found. Error! Reference source not found. – and Table 13-4 – Monitoring item configuration

[12] Bit assignment at Error! Reference source not found. Error! Reference source not found. – and Table 13-4 – Monitoring item configuration

Here is an equivalent flow of the function:

```
1.   Set AFX flag
     - 1. Set TMTC to DROP Incoming Telecommand
          All incoming telecommand will be transferred in Datalink Layer but not
          passed to the OBS Command Sequencer CMD_SEQ.
          - 1.   Stop Currently Running Hardware VM
          - 2.   Abort any in running Memory Dump
          - 3.   Abort any in running Peak Up procedure
          - 4.   Clear Hardware VM Execution  and Clear Status for Function Reporting
                 - 1. Set R[0x00] with the ZZ argument
                 - 2. START the Hardware VM at table 0xYY
                 - 3. Wait INDEFINITELY for VM execution completion
     - 2. Set TMTC to ACCEPT Incoming Telecommand
2.   Unset AFX flag
```

Table 13-8 Safe Enviroment VM.

13.4.3 Goto SAFE Mode

Calling this action the OBS set the Observing mode to SAFE and switches off the OBS/MCU communications[13].

This function is similar to the one above, but there is no argument passed to the preset hardwired in code table *0xD4 GOTOSAFEMODE*. This function is called from the GO_TO_SAFEMODE TC.

Here is the flow:

```
1.   Set AFX flag – suspends other autonomy functions
     - 1. Set TMTC to DROP Incoming Telecommand
          All incoming telecommand will be transferred in Datalink Layer but not
          passed to the OBS Command Sequencer CMD_SEQ.
          - 1.   Kill currently running Hardware VM immediately
          - 2.   Stop all other running VMs – these will die on the next instruction or
                 within 32ms (whichever comes first)
          - 3.   Abort any running Memory Dump
          - 4.   Abort any running Peak Up procedure
          - 5.   Clear Hardware VM Execution  and Clear Status Report
                 - 1. START the Hardware VM at Table 0xD4 (212) GOTOSAFEMODE
                 - 2. Wait INDEFINITELY for VM execution completion
     - 2. Set TMTC to ACCEPT Incoming Telecommand
2.   Unset AFX flag
```

Table 13-9 Safe Mode VM.

Note 1: This is what is called from the GOTO_SAFE_MODE telecommand TM(8,4,CA 0A)
Note 2: It is not possible to stop the the single VM.

[13] - ACTUALLY THIS IS DONE VIA INHIBITION SYTEM -

13.5 Tele Command with different priority

Three different priority, with different code path, are been defined: Normal, High and Super priority.
In nominal case no TC shall be received if the previous one is still in execution.
But in some theoretical circumstance there is the needing to send some command even if no completion TM(1,7) is been received from the CDMU.
Actually the OBS recognize 3 conditions:

- Nominal Priority
  - Everything is executed serially.
  - If an autonomy action is running the command can be dropped.
- High Priority – STOP_VM and ABORT_MEM_DUMP
  - The Telecommands will be pushed through a different queue and then executed before other possible TC in the nominal queue.
  - If an autonomy action is running the command can be dropped.
- Super priority, - STOP Monitoring System
  - The Telecommands will be pushed through a different queue and then executed before other possible TC in the nominal queue.
  - If an autonomy action is running
    →the command WILL BE EXECUTED.
  - If a series of autonomy functions are queued, but
    →NO MORE OTHER ACTION WILL BE QUEUED.
    This is done mainly to avoid infinitive loop of autonomy actions.
  - →ALL THE QUEUED ACTION WILL BE EXECUTED

13.6 How to Use the Monitoring System

The monitoring system is started by sending the Start_Monitoring TC with the Table ID of the monitoring table as a parameter. In order for the monitoring system to start some conditions need to be met, or an execution failure TM(1,8) will be generated:

- All the monitored parameters must be present in the nominal HK TM packet definition
- All the dependencies must be defined as monitored parameters themselves, and positioned in a higher location of the monitoring table
- All the tables referred to as VM program autonomy actions in the proper fields of the Action_XY records, must be already existing
- the collection of the Nominal HK TM packets MUST be active

When starting the monitoring service, the OBS does a sort of compilation of the monitoring table to sort out all parameter dependencies; for this reason the monitoring table cannot be updated via TC if the monitoring system is running. In case a new monitoring parameter needs to be added to the list, or a parameter needs to be deleted, or the list of dependencies needs to be modified, it is necessary to: i) send Stop_Monitoring TC, then ii) update the table, and finally iii) re-send the Start_Monitoring TC.

If, however, only parameter-specific configuration needs to be changed (like monitring limits, monitoring flag, detailed behaviour upon state transition), the re-compilation of the monitoring table is not needed and it is sufficient to i) send the Suspend_Monitoring TC, ii) update the monitoring table as needed, and iii) send the Resume_Monitoring TC.

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.

The monitoring system suspends itself when an autonomy function is running.

Nested dependencies are possible as long as the correct order is followed: the deeper dependency should be listed first as a monitored parameter.

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 (e.g. a measurement).

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 minor anomaly.

14 Sub-System I/F FDIR Management

According to AD5, the OBS is able to handle anomalies arisirng in the communication between the DPU and the SPIRE subsystems. The possible anomalies are outlined in RD10.
The OBS implements a mechanism by which the anomaly must be detected for N number of times before the I/F status is changed and proper actions are taken according to specifications; the number N can be independently set by the user for each separate anomaly, and are stored in Table 6 (Configuration table). The behaviour of the Low-Speed I/F FDIR can then be modified at runtime (***and at your own risk!***) using the Update_Table TC and changing the desidered N value according to the following table, where the first column gives the record index to be accessed in Table 6:

#	**Configuration Parameters for High-Speed I/F FDIR behaviour**	**Value at switch-on**
0	Observing Mode	0x00000000
1	BSM_CHOP_OFFSET found at previous peak up procedure - in bsm unit -	0
2	BSM_JIGGLE_OFFSET – same as above	0
3	Reserved for debug purpose *IN FLY SHALL REMAIN 0 - ASK TO IFSI PERSONNEL* Use of this location may generate a lot of useless TM-packets.	0
4	Max number of attempts for a sync failure recovery on the DCU High-Speed Link	5
5	Max number of attempts for a sync failure recovery on the MCU High-Speed Link	5
6	Max number of attempts for a sync failure recovery on the SCU High-Speed Link	5
7	Reserved	
#	**Configuration Parameters for DCU Low-Speed I/F FDIR behaviour**	
8	Max number of retries for DCU Low-Speed I/F Timeout anomaly	3
9	Max number of retries for DCU Low-Speed I/F Command Echo Mismatch anomaly	3
10	Reserved	
11	Reserved	
12	Retry count if DCU command echo is OK	0xFFFFFFFF
13	Max number of retries for DCU Low-Speed I/F Unknwon S/S Command anomaly	0xFFFFFFFF
14	Max number of retries for DCU Low-Speed I/F Forbidden S/S Command anomaly	0xFFFFFFFF
15	Max number of retries for DCU Low-Speed I/F S/S Command Timeout anomaly	0xFFFFFFFF
#	**Configuration Parameters for MCU Low-Speed I/F FDIR behaviour**	
16	Max number of retries for MCU Low-Speed I/F Timeout anomaly	3
17	Max number of retries for MCU Low-Speed I/F Command Echo Mismatch anomaly	3
18	Reserved	
19	Reserved	
20	Retry count if MCU command echo is OK	0xFFFFFFFF
21	Max number of retries for MCU Low-Speed I/F Unknwon S/S Command anomaly	0xFFFFFFFF
22	Max number of retries for MCU Low-Speed I/F Forbidden S/S Command anomaly	0xFFFFFFFF
23	Max number of retries for MCU Low-Speed I/F S/S Command Timeout anomaly	0xFFFFFFFF
#	**Configuration Parameters for SCU Low-Speed I/F FDIR behaviour**	
24	Max number retries for SCU Low-Speed I/F Timeout anomaly	3
25	Max number of retries for SCU Low-Speed I/F Command Echo Mismatch anomaly	3
26	Reserved	
27	Reserved	
28	Retry count if SCU command echo is OK	0xFFFFFFFF
29	Max number of retries for SCU Low-Speed I/F Unknwon S/S Command anomaly	0xFFFFFFFF
30	Max number of retries for SCU Low-Speed I/F Forbidden S/S Command anomaly	0xFFFFFFFF
31	Max number of retries for SCU Low-Speed I/F S/S Command Timeout anomaly	0xFFFFFFFF
#	**Science Data Extraction.**	
32	SDEX0 (IFSI test value)	0x0123C1A0
33	SDEX1 (IFSI test value)	0x0124C1A0
34	SDEX2 (IFSI test value)	0x200EC1A0
35	SDEX3 (IFSI test value)	0x401CC1A0

Table 14-1 Configuration table

14.1 Note on FDIR behaviour

A parameter value of 0xFFFFFFFF means that the FDIR for that specific anomaly is switched-off

14.2 Note on SDEX – Science Data Extraction

It is possible to extract up to 4 data word from science data frames coming from subsystems. The data are copied into the table 6 at the related field. See §15.4 Science Frame Data Extraction (SDEX).

14.3 HS Transparent mode

Data frames coming from subsystems are packed by their content type, length and their content consistence is enforce by a vertical XOR-Checksum word at the end of each frame. When OBS encounter an error on one of these it increment an internal counter of error and rapidly download the content of the hardware fifo source of the error, by doing this the OBS is certain that the next frame's first word is aligned at the first word read from the fifo, this will re-synchronize the packet flow in the fifo. Once the OBS is synchronized the error counter is reset. When the internal counter reaches its own limit stored in the Configuration Table 6 the OBS won't try more attempt to re-synchronize the data in fifo, then the OBS will download any word coming from the fifo without any analysis nor packing.

On discontinuous errors	.	On Continous error [limit = 2]
Normal frame		Normal frame
Error		Error
Transparent Frame [transiently]		Transparent Frame [transiently]
Transparent Frame [transiently]		Transparent Frame [transiently]
Transparent Frame [transiently]		Transparent Frame [transiently]
Fifo Empty		Fifo Empty
Normal frame		Error
Normal frame		Transparent Frame [transiently]
Normal frame		Transparent Frame [transiently]
Normal frame		Transparent Frame [transiently]
Normal frame		Fifo Empty
Normal frame		Transparent Frame [permantly]
Normal frame		Transparent Frame [permantly]
Normal frame		Transparent Frame [permantly]
Normal frame		Transparent Frame [permantly]

**Table 14-2 Two possible scenario:
a single error on left, two contiguous error on right.**

14.4 LS Errors detection

For each command sent to subsystem the DPU goes in wait state for an acknowledgement answer. If DPU recognize an error in this, the command is repeated until the subsystem answers or the count of retry reaches the given limit linked to the queried subsystem [see table below].

When the system answer correctly [see below: "**Answer Acknowledgement bit OK**"] the retry count is reset, an event for exiting form SICK/DEAD condition is send and the system is marked to be **ALIVE**.

When retry commanding the subsystem is marked to be **SICK**, an event is raised; if the queried subsystem is the MCU special action are executed.

When retry commanding reaches the given limit, the subsystem is marked to be **DEAD**, an Event will be generated and the DPU follows the directive contained in RD10.

Here follow the error conditions trapped by OBS and relative actions:

- **Stepping into ALIVE STATE:**
    - **Answer Acknowledgement bit OK**
        - Condition: OK the Command is been correctly executed and replied.
        - Action for D/M/SCU: Clearance event for any other previously sent
- **Stepping into SICK STATE:**
    - Answer Timeout :
        - Condition : No answer received after the expected time period.
        - Action for DCU: Event: ERROR_NO_DCU_RES_SID
        - Action for MCU: Event: ERROR_NO_MCU_RES_SID
        - Action for SCU: Event: ERROR_NO_SCU_RES_SID
    - Answer Command ID Mismatch
        - Condition : CID in Answer and Commanding don't match.
        - Action for DCU: Event: ERROR_LS_DCU_RX_ID
        - Action for MCU: Event: ERROR_LS_MCU_RX_ID
        - Action for SCU: Event: ERROR_LS_SCU_RX_ID
    - Answer Acknowledgement bit UNKNOWN Command
        - Condition: the Command request has an UNKNOWN CID.
        - Action for D/M/SCU: event ERROR_LS_CID_UNKNOWN_ID
    - Answer Acknowledgement bit FORBIDDEN Command
        - Condition: the Command execution is currently FORBIDDEN.
        - Action for D/M/SCU: event ERROR_LS_CID_FORBIDDEN_ID
    - Answer Acknowledgement bit TIME OUT Command
        - Condition: the Command execution timeout please refer to RD3.
        - Action for D/M/SCU: event ERROR_SS_TIMEOUT_ID
- **Stepping into DEAD STATE**:

- Answer Timeout [IF OBS-MODE is “DPUON”]:
  - Condition: No answer received after the expected time period.
  - Action for D/M/SCU: no Action

- Answer Timeout [IF OBS-MODE is not “DPUON”]:
  - Condition : No answer received after the expected time period.
  - Action for DCU: Event: `ERROR_LS_DEAD_DCU_ID`
  - Action for MCU: Event: `ERROR_LS_DEAD_MCU_ID`
  - Action for SCU: Event: `ERROR_LS_DEAD_SCU_ID`
  - Action for D/M/SCU: RAPID SAFE MODE[14]

- Answer Command ID Mismatch
  - Condition : CID in Answer and Commanding don’t match..
  - Action for DCU: Event: `ERROR_LS_DEAD_DCU_ID`
  - Action for MCU: Event: `ERROR_LS_DEAD_MCU_ID`
  - Action for SCU: Event: `ERROR_LS_DEAD_SCU_ID`
  - Action for D/SCU: RAPID SAFE MODE[15]
  - Action for MCU: RAPID SWITCH OFF MCU[16]

- Answer Acknowledgement bit UNKNOWN Command
  - Condition: the Command request has an UNKNOWN CID.
  - Action for DCU: Event: `ERROR_LS_DEAD_DCU_ID`
  - Action for MCU: Event: `ERROR_LS_DEAD_MCU_ID`
  - Action for SCU: Event: `ERROR_LS_DEAD_SCU_ID`
  - Action for D/SCU: RAPID SAFE MODE[17]
  - Action for MCU: RAPID SWITCH OFF MCU[18]

- Answer Acknowledgement bit FORBIDDEN Command
  - Condition: the Command execution is currently FORBIDDEN.
  - Action for DCU: Event: `ERROR_LS_DEAD_DCU_ID`
  - Action for MCU: Event: `ERROR_LS_DEAD_MCU_ID`
  - Action for SCU: Event: `ERROR_LS_DEAD_SCU_ID`
  - Action for D/SCU: RAPID SAFE MODE[19]
  - Action for MCU: RAPID SWITCH OFF MCU[20]

- Answer Acknowledgement bit TIME OUT Command
  - Condition: the Command execution timeout TBD Please refer to RD3.
  - Action for DCU: Event: `ERROR_LS_DEAD_DCU_ID`
  - Action for MCU: Event: `ERROR_LS_DEAD_MCU_ID`
  - Action for SCU: Event: `ERROR_LS_DEAD_SCU_ID`
  - Action for D/SCU: RAPID SAFE MODE[21]
  - Action for MCU: RAPID SWITCH OFF MCU[22]

[14] “differently from 13.4.3-Goto SAFE Mode” this call the VM with table 0xD4 – “SAFE MODE”
[15] “differently from 13.4.3-Goto SAFE Mode” this call the VM with table 0xD4 – “SAFE MODE”
[16] this call the VM with table 0xD5 – “INHIBIT MCU”
[17] “differently from 13.4.3-Goto SAFE Mode” this call the VM with table 0xD4 – “SAFE MODE”
[18] this call the VM with table 0xD5 – “INHIBIT MCU”
[19] “differently from 13.4.3-Goto SAFE Mode” this call the VM with table 0xD4 – “SAFE MODE”
[20] this call the VM with table 0xD5 – “INHIBIT MCU”
[21] “differently from 13.4.3-Goto SAFE Mode” this call the VM with table 0xD4 – “SAFE MODE”
[22] this call the VM with table 0xD5 – “INHIBIT MCU”

15 OBS Direct Functions

15.1 HS Management

15.1.1 Science Frame Data Selection

The OBS can execute Frame Data Selection by storing in the TM packets a subset of science data frames instead of the entire frame. This is possible using the content of table as filter [RD2 §3.2.8.3.32], this table must have the same length of the related data frame and composed by an array of ONE and ZERO. Any zero in the table will drop a data word at the corresponding location in a science frame.

Example:

Selection Table	.	Data Flow	.	Data Acquired
1		0x0123		0x0123
1		0x4567		0x4567
0		~~0x89ab~~		0x1111
0		~~0xcdef~~		
1		0x1111		

Table 15-1Science Data Selection example

15.1.2 Science Data Buffer FIFO Flush / Reset

The “Flush FIFO” and “Reset FIFO” commands can be executed selectively on the three different FIFOs. Any FIFO the user is interested to flush or reset is marked in the FFLAGS Parameter present in the command itself. The FFLAGS parameter bits present in the first nibble map the DCU FIFO, MCU FIFO, SCU FIFO respectively.
[Check RD2 §3.2.8.3.22 - RD2 § 3.2.8.3.26].

0	SCU	MCU	DCU	0	0	0	0	0	0	0	0	0	0	0	0

Bits to Fifo Mapping for Fifo Flush/Reset Commands.

15.2 SubSystem Direct Commanding

The SEND_DRCU_COMMAND TC (see RD2) can be used to send direct commands to the subsystems; the first parameter to this command will be 32-bit command word to the S/S, while the second will be an OVERRIDE flag; if it is 1 then the command shall go through even if it is currently inhibited (see below).

15.3 LS Commanding Inhibition System

This service allows to inhibit the distribution of commands to the instrument subsystems. Prior to send any command to the subsystems the OBS checks if the outgoing command is inhibited, the inhibition status of each command is mapped into tables onboard. Inhibition information is stored in tables 248 (DCU commands), 249 (MCU commands) and 250 (SCU commands). As such, commands can be inhibited or released either via the simple use of the Update_Table TC and by VM programs. At boot the inhibition status of each command is reset allowing all command to be used. The OBS can bypass this control, the Virtual Machines can override the inhibition system with the apposite opcode OVRD ON/OFF, the Direct commanding also has a flag that, if set, allow overriding of the inhibition system.

15.3.1LS CIS – DCU, MCU and SCU

For the DCU, MCU and the SCU systems a bit of the related table represent an inhibition status, those bits are indexed by a column/row rule that fits on the 12bits of the CID field of commands to SubSystem.

Bit 31-28	Bit 27	Bit 26-21	Bit 20-16 Column	Bit 15-00
SSYS ID	ignored	Row in table	Bit in 32 bit word	ignored

Table 15-2 DCU, MCU & SCU CIS addressing

15.3.2LS CIS Management of Single Commands

The inhibition of a single command or a set of command is controlled by the TC (8,4,0xCA,0x13/0x14) `Enable_SubSys_Command`, `Disable_SubSys_Command`. This command scan the whole array of inhibition status and set/reset the inhibition status any time the system command in analysis matches with a bitwise mask, in this way is possible to inhibit/dis-inhibit a masked sequential set of command with just a telecommand.
The same function is accessible from VM and follows the same semantics.

15.3.3LS CIS Management of Range of Commands

To set or unset the CIS both the user and the virtual machines access to the same functionality. Via the VM the user can set/unset a single command's CIS status, by calling the VirtualTable 0x100 at row 0x07 [ref §12.1]. both via TC and VM the user can set/unset the CIS status of a group of command. That group is identified with a bitwise masking and matching, so many possible combination can set unset the same group of command.

For the TC way please refer to RD2 Spire data ICD [23]

For the VM The argument needed are the following.

	Example	
Set the R0 with the first command in the range	R0 = 0x84320000;	//
Set the R1 with the last command in the range	R1 = 0x84FF0000;	//
Set the R2 with a selection mask	R2 = 0xFF000000;	//
Set the R3 with a selection pattern	R3 = 0x84000000;	// Anything in range
Set the R4 with the inhibition status	R4 = 0;	// Allow
Call the OBS Set Inhibition Command	IRCALL 0x100, 0x09;	

The CIS proceed as following behavior:

- Iterate A_VARIABLE from R0 to R1
    - If A_VARIABLE is an DCU command
        - Then Iterate A_VARIABLE from R0 to R1 covering also parameters in data fields.
            - if A_VARIABLE masked by R2 matches R3
                - then set the inhibition status as R4

[23] RD2 Spire data ICD
§3.2.8.3.36 Function 0xCA DPU, Activity 0x13:ENABLE_SS_TC
§3.2.8.3.37 Function 0xCA DPU, Activity 0x14:DISABLE_SS_TC

- otherwise // is an SCU or MCU command.
    - if A_VARIABLE masked by R2 matches R3
        - then set the inhibition status as R4

15.4 Science Frame Data Extraction (SDEX)

A maximum of 4 words can be extracted from any science data frame being received from the Subsystems, and put into specific locations in Table 6. It is then possible from any VM to read and use those data.
The SDEX service is managed and used by configuring 4 registers, with indexes 32, 33 34 and 35 in Table 6. The 16 MSB of each register are used to select the word to be extracted from a science frames, while the 16 LSB will contain the desired value. The configuration of the 16 MSB of the register is according to the following syntax:

31	30-25	24-16	15-0
R	Frame ID	Record Index in science frame	Data

Bit 31 : this is a ready bit that can be used to realize if the SDEX value has been actually updated or not. Each time a SDEX value is needed from a certain location in science frame, the ready bit can be set to 0; before reading the SDEX value in the 16 LSB of the register, the ready bit can be checked and if it is 1 it means that the 16 LSB of the register have been updated.
Bit 30-25: this is the Frame ID which science frames are expected to income from S/S the extractions of data are from those frames
Bit 24-16: the offset inside the Frame from where extract the parameter word.
Bit 15-0: the parameter read at the previous cycle.

Note:
If couple of <Frame ID, Offset> are not consistent with any expected Frame-Id or expected length the data will be set to the fixed value 0xFFFF

16 The Peak-up Operating mode

Please refer to AD6 *SPIRE Peak-up Mode Requirement*.

17 The safety procedure GOTO_SAFE_MODE

Please refer to AD7 *SPIRE Failure Detection Isolation and Recovery*.

18 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.

18.1 Events, Reports and Warnings reports – TM(5,1) :

Event Code [hex]	**SID [hex]**	**Event Name**	**Explanation**	**Returned Parameters**
0501	5100	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 Observing Mode • Current Step Number
0504	5101	REPORT_PEAKUP	Final report from Peak-Up procedure	• "2" – Constant for Spire • Offset in CHOP direction • Offset in JIGGLE direction
~~0505~~	~~5102~~	~~DELETED~~	~~DELETED~~	~~DELETED~~
0506	5101	PEAKUP_NOT_FOUND_ID	Peak-Up procedure didn't found a Peak-Value in the requested range	• "2" – Constant for Spire • Offset in CHOP direction • Offset in JIGGLE direction
0507	5101	PEAKUP_ABORT_ID	Peak-Up procedure Aborted	
0509	5103	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
050A	5104	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
050B	5108	SS_TIMEOUT	In response to a "SET" command, the DRCU times out.	• 32-bits Command sent to the DRCU • 32-bits echo received
050C	5109	LS_DCU_RX	The command ID in the echo sent back by the DRCU in response to a command, is not identical to the one 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

Event Code [hex]	SID [hex]	Event Name	Explanation	Returned Parameters
050D	5109	LS_MCU_RX	Same as above	Same as above
050E	5109	LS_SCU_RX	Same as above	Same as above
050F	510C	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 microseconds in which the LS port was busy during the last second more than 990000 usec to go in alarm state, less than 980000 usec to return in normal state
0510	510D	UNKNOWN_TM_PCKT	A TM packet ready to be sent has an unknown combination of type, subtype and SID.	• Type, • Subtype, • Packet ID of the unknown TM packet
0511	5111	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
0512	5117	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 stamp format • The Internal High Resolution Timer at this sync • The Internal High Resolution Timer at the previous sync
0513	5115	Check_PM_Fail_ID	The CRC computed over the specified PM memory range is different from what it should be	• 16-bit true CRC expected • 16-bit computed CRC
0514	5116	Check_DM_Fail_ID	At least one hardware Data Memory Cell has a erroneous behaviour.	• 32-bit Sum of Error pattern found on cell tested

Event Code [hex]	SID [hex]	Event Name	Explanation	Returned Parameters
0523	5193	ERROR_VM_Stack_UnderFlow____ID	The vm identified with the ID in parameter is returning to an unexistent location	• 16-bit Vm ID • 16-bit Table ID • 16-bit Table Offset • 32-bit Current offending Instruction • 16-bit Mutex Status [LS preemption] • 32-bit Timing period High Resolution in micro-seconds • 32-bit Timing period Low Resolution in milli-seconds • 32-bit Register 255 standard Offset. • 32-bit last LS status word. • 16-bit Stack depth • 16-bit+16bit 16 time Each couple is a- Table ID b- Table Offset with {16-[Stack depth]} times 0x0000 0x0000 padding. • 32-bit 16 times: PARAMETERS = 16 x 0x0000 • 32-bit 16 times Registry 0 to 15 • 32-bit 16 times Registry 240 to 255
0524	5194	ERROR_VM_Stack_OverFlow_____ID	The vm identified with the ID in parameter is trying to call a subrouting over the call-nesting-limit	Same as above with: PARAMETERS = • 32-bit 16 times 0x0000
0515	5195	ERROR_VM_Stopped____________ID	The vm identified with the ID in parameter is been stopped.	Same as above with: PARAMETERS = • 32-bit 16 times 0x0000

Event Code [hex]	SID [hex]	Event Name	Explanation	Returned Parameters
0516	5196	ERROR_VM_CopyTableFault_____ID	The vm identified with the ID in parameter was copying data to/from table outside the running one, and a non valid addressing was used. Check the VM code on opcode like: ICPF ICPT IRCPF IRCPT	Same as above with: PARAMETERS = • 32-bit index of the register containing the extern tableID : [0xFFFFFFFF if immediate 32 bit value] • 32-bit The external tableId register • 32-bit the external table offset • 32-bit the external table size • 32-bit index of the ragister pointing data in the local table [0xFFFFFFFF if immediate 32 bit value] • 32-bit the position of data in the local table • 32-bit local table size • 32-bit number of word • 32-bit 8 times 0x0000
0517	5197	ERROR_VM_PageFault__________ID	The vm identified with the ID in parameter was parsing a non existing location of the current table. Check the VM binary code.	Same as above with: PARAMETERS = • 32-bit 16 times 0x0000
0518	5198	ERROR_VM_IllegalInstruction_ID	The vm identified with the ID in parameter was parsing a non existing opcode in the current table. Check the VM binary code.	Same as above with: PARAMETERS = • 32-bit the unknown instruction • 32-bit 15 times 0x0000

Event Code [hex]	SID [hex]	Event Name	Explanation	Returned Parameters
0519	5199	ERROR_VM_IllegalRegister____ID	The vm identified with the ID in parameter was copying data from a pointed register to another pointed register using invalid pointer to register. Check the VM code for opcode: XREQ	Same as above with: PARAMETERS = • 32-bit Index of the register pointing the source register • 32-bit The pointed source register index • 32-bit Index of the register pointing the destination register • 32-bit The pointed destination register index • 32-bit 12 times 0x0000
051A	519A	ERROR_VM_IllegalIdentity____ID	The vm identified with the ID in parameter was stopping another vm using an invalid VM identity number. Check the VM code for opcode: VMSTP	Same as above with: PARAMETERS = • 32-bit the non existent vm • 32-bit 15 times 0x0000
051B	519B	ERROR_VM_IllegalTiming______ID	The vm identified with the ID in parameter was setting a timing period too small for the architecure in use. The minimal timing is: 10 u sec for Hardware VM 1000 u sec for Software VM Check the VM code for opcode: TIM RTIM	Same as above with: PARAMETERS = • 32-bit Requested Timing • 32-bit Minimal Timing 10 µsec for Hard VM 1 msec for Soft VM • 32-bit 14 times 0x0000

Event Code [hex]	SID [hex]	Event Name	Explanation	Returned Parameters
051C	519C	ERROR_VM_DivisionByzero_____ID	The vm identified with the ID in parameter was dividing the value in a register with a zero value divisor. Check the VM code for opcode: RDIV RRDV	Same as above with: PARAMETERS = • 32-bit Dividend Register Index • 32-bit Dividend Value in Register • 32-bit Divisor Register Index [0xFFFFFFFF if immediate 32 bit value] • 32-bit Result/Destination Register Index • 32-bit 12 times 0x0000
051D	519D	ERROR_VM_CallTableFault_____ID	The vm identified with the ID in parameter was calling a subroutine that resides in an undefined table or outside the indicated table. Check the VM code for opcode: ICALL IRCALL	Same as above with: PARAMETERS = • 32-bit Destination Table Id • 32-bit Destination Table Offset • 32-bit 14 times 0x0000
051E	519E	ERROR_VM_TransmitTableFault_ID	The vm identified with the ID in parameter was transmitting content from a undefined table or outside the indicated table. Check the VM code for opcode: TXTBL	Same as above with: PARAMETERS = • 32-bit Id of Table to transmit • 32-bit Index of data in table • 32-bit Number of word to transmit • 32-bit 13 times 0x0000
051F	519F	ERROR_VM_Read_Twice_________ID	The vm identified with the ID in parameter was reading data from the LS (HW port / task), prior to send any command, or attempted to read data twice. Check the VM code for opcode: READ	Same as above with: PARAMETERS = • 32-bit 16 times 0x0000

Event Code [hex]	SID [hex]	Event Name	Explanation	Returned Parameters
0520	510E	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
0521	510F	NO_MCU_RES	As NO_DCU_RES error, but for MCU subsystem	ibidem
0522	5110	NO_SCU_RES	As NO_DCU_RES error, but for SCU subsystem	ibidem
7001	5112	ERROR_MONPAR_NOM_WARN	An HK parameter went from nominal to warning	• 32-bit get parameter command • 32-bit echo word • 32-bit parameter value • 32-bit action word
7002	5112	ERROR_MONPAR_NOM_FAIL	An HK parameter went from nominal to fail	ibidem
7012	5112	ERROR_MONPAR_WARN_FAIL	An HK parameter went from warning to fail	ibidem
7020	5112	ERROR_MONPAR_FAIL_NOM	An HK parameter went from fail to nominal	ibidem
7021	5112	ERROR_MONPAR_FAIL_WARN	An HK parameter went from fail to warning	ibidem
7010	5112	ERROR_MONPAR_WARN_NOM	An HK parameter went from warning to nominal	ibidem
vmArg	5113	EVENT_VM_EVENT[24]	An Event Report from VM. The Event Code is Dynamically defined by the Virtual Machine	• Variable length array of parameter defined by Virtual Machine.
0607	5114	DUMP_ABORTED_ID	The Memory Dump procedure has been aborted by TC	
0608	5114	DUMP_NO_MEMBLOCKS_IN_MEMDUMP_ID	The Memory Dump procedure has been aborted because of failed memory block allocation	
1500	510A	TC_POOL_FULL_ID	The DPU Memory Pool for Telecommand packets is more than 80% full; The warning status will be restored when the pool is less than 70% full.	• Pool ID • Pool occupation status • Pool occupation limit • TIME STAMP of the transition
1501	510A	EV_POOL_FULL_ID	Ibidem for Event TM packets	ibidem
1502	510A	HK_POOL_FULL_ID	Ibidem for HouseKeeping TM packets	ibidem
1503	510A	SD_POOL_FULL_ID	Ibidem for Science TM packets	ibidem
1505	510A	RP_POOL_FULL_ID	Ibidem for Report TM packets.	ibidem

[24] **EVENT_VM_EVENT** see at Appendix A – Special telemetry packets.

Event Code [hex]	SID [hex]	Event Name	Explanation	Returned Parameters
1510	510B	TC_HP_FIFO_FULL_ID	The VIRTUOSO FIFO Queue for high-priority TC packets is more than 70% full	• FIFO ID • FIFO occupation status • FIFO occupation limit • TIME STAMP of the transition
1511	510B	TC_LP_FIFO_FULL_ID	Ibidem for low-priority TC packets	ibidem
1512	510B	EV_TM_FIFO_FULL_ID	Ibidem for event TM packets	ibidem
1513	510B	HK_TM_FIFO_FULL_ID	Ibidem for HouseKeeping TM packets	ibidem
1514	510B	SD_TM_FIFO_FULL_ID	Ibidem for science TM packets	ibidem
1515	510B	RP_TM_FIFO_FULL_ID	Ibidem for science TM packets	ibidem
1516	510B	LS_HP_FIFO_FULL_ID	Ibidem for high-priority Sub-Systems commands	ibidem
1517	510B	LS_LP_FIFO_FULL_ID	Ibidem for low-priority Sub-Systems commands	ibidem
1518	510B	AUTO_FIFO_FULL_ID	Ibidem for Autonomy Function Execution Request	ibidem
1519	510B	VM_TM_FIFO_FULL_ID	Ibidem for TM packets generated by the VM	ibidem
151A	510B	MEM_DUMP_FIFO_FULL_ID	Ibidem for command to the Memory Dumper Task	ibidem
2540	5106	ERROR_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
2541	5106	ERROR_FIFO_DCU_FLEN_SPEC_FULL_ID	Wrong Frame Length for a Full Spectrometer DCU frame	ibidem
2542	5106	ERROR_FIFO_DCU_FLEN_PSW_ID	Wrong Frame Length for a PSW DCU frame	ibidem
2543	5106	ERROR_FIFO_DCU_FLEN_PMW_ID	Wrong Frame Length for a PMW DCU frame	ibidem
2544	5106	ERROR_FIFO_DCU_FLEN_PLW_ID	Wrong Frame Length for a PLW DCU frame	ibidem
2545	5106	ERROR_FIFO_DCU_FLEN_SSW_ID	Wrong Frame Length for a SSW DCU frame	ibidem
2546	5106	ERROR_FIFO_DCU_FLEN_SLW_ID	Wrong Frame Length for a SLW DCU frame	ibidem
2547	5106	ERROR_FIFO_DCU_FLEN_PHOT_OFF_ID	Wrong Frame Length for a Full Photometry Offset DCU frame	ibidem
2548	5106	ERROR_FIFO_DCU_FLEN_SPEC_OFF_ID	Wrong Frame Length for a Full Spectrometer Offset DCU frame	ibidem
2549	5106	ERROR_FIFO_DCU_FLEN_PHOT_FULL_TEST_ID	Wrong Frame Length for a Full Photometry Test DCU frame	ibidem
254A	5106	ERROR_FIFO_DCU_FLEN_PSW_TEST_ID	Wrong Frame Length for a PSW Test DCU frame	ibidem
254B	5106	ERROR_FIFO_DCU_FLEN_PMW_TEST_ID	Wrong Frame Length for a PMW Test DCU frame	ibidem
254C	5106	ERROR_FIFO_DCU_FLEN_PLW_TEST_ID	Wrong Frame Length for a PLW Test DCU frame	ibidem
254D	5106	ERROR_FIFO_DCU_FLEN_SPEC_FULL_TEST_ID	Wrong Frame Length for a Full Spectrometer Test DCU frame	ibidem
254E	5106	ERROR_FIFO_DCU_FLEN_SSW_TEST_ID	Wrong Frame Length for a SSW Test DCU frame	ibidem

Event Code [hex]	SID [hex]	Event Name	Explanation	Returned Parameters
254F	5106	ERROR_FIFO_DCU_FLEN_SLW_TEST_ID	Wrong Frame Length for a SLW Test DCU frame	ibidem
2550	5107	ERROR_FIFO_DCU_FCRC_PHOT_FULL_ID	Wrong checksum for a Full Photometry DCU frame	• Frame ID read • Computed checksum • Read checksum
2551	5107	ERROR_FIFO_DCU_FCRC_SPEC_FULL_ID	Wrong checksum for a Full Spectrometer DCU frame	ibidem
2552	5107	ERROR_FIFO_DCU_ FCRC _PSW_ID	Wrong checksum for a PSW DCU frame	ibidem
2553	5107	ERROR_FIFO_DCU_ FCRC _PMW_ID	Wrong checksum for a PMW DCU frame	ibidem
2554	5107	ERROR_FIFO_DCU_ FCRC _PLW_ID	Wrong checksum for a PLW DCU frame	ibidem
2555	5107	ERROR_FIFO_DCU_ FCRC _SSW_ID	Wrong checksum for a SSW DCU frame	ibidem
2556	5107	ERROR_FIFO_DCU_ FCRC _SLW_ID	Wrong checksum for a SLW DCU frame	ibidem
2557	5107	ERROR_FIFO_DCU_ FCRC _PHOT_OFF_ID	Wrong checksum for a Full Photometry Offset DCU frame	ibidem
2558	5107	ERROR_FIFO_DCU_ FCRC _SPEC_OFF_ID	Wrong checksum for a Full Spectrometer Offset DCU frame	ibidem
2559	5107	ERROR_FIFO_DCU_ FCRC _PHOT_FULL_TEST_ID	Wrong checksum for a Full Photometry Test DCU frame	ibidem
255A	5107	ERROR_FIFO_DCU_ FCRC _PSW_TEST_ID	Wrong checksum for a PSW Test DCU frame	ibidem
255B	5107	ERROR_FIFO_DCU_ FCRC _PMW_TEST_ID	Wrong checksum for a PMW Test DCU frame	ibidem
255C	5107	ERROR_FIFO_DCU_ FCRC _PLW_TEST_ID	Wrong checksum for a PLW Test DCU frame	ibidem
255D	5107	ERROR_FIFO_DCU_ FCRC _SPEC_FULL_TEST_ID	Wrong checksum for a Full Spectrometer Test DCU frame	ibidem
255E	5107	ERROR_FIFO_DCU_ FCRC _SSW_TEST_ID	Wrong checksum for a SSW Test DCU frame	ibidem
255F	5107	ERROR_FIFO_DCU_ FCRC _SLW_TEST_ID	Wrong checksum for a SLW Test DCU frame	ibidem
2560	5106	ERROR_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
2562	5106	ERROR_FIFO_MCU_FLEN_BSM_ID	Wrong Frame Length for a BSM MCU frame	ibidem
2564	5106	ERROR_FIFO_MCU_FLEN_ENGINEERING_ID	Wrong Frame Length for an Engineering MCU frame	ibidem
2565	5106	ERROR_FIFO_MCU_FLEN_TEST_ID	Wrong Frame Length for a Test MCU frame	ibidem
2568	5107	ERROR_FIFO_MCU_FCRC_SMEC_ID	Wrong checksum for a SMEC MCU frame	• Frame ID read • Computed checksum • Read checksum
256A	5107	ERROR_FIFO_MCU_FCRC_BSM_ID	Wrong checksum for a BSM MCU frame	ibidem
256C	5107	ERROR_FIFO_MCU_FCRC_ENGINEERING_ID	Wrong checksum for an Engineering MCU frame	ibidem
256D	5107	ERROR_FIFO_MCU_ FCRC _TEST_ID	Wrong checksum for a Test MCU frame	ibidem

Event Code [hex]	SID [hex]	Event Name	Explanation	Returned Parameters
2570	5106	ERROR_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
2571	5106	ERROR_FIFO_SCU_FLEN_TEST_ID	Wrong Frame Length for a Test SCU frame	ibidem
2574	5107	ERROR_FIFO_SCU_FCRC_HSK_ID	Wrong checksum for a nominal SCU frame	• Frame ID read • Computed checksum • Read checksum
2575	5107	ERROR_FIFO_SCU_FCRC_TEST_ID	Wrong checksum for a Test SCU frame	ibidem
2578	5105	ERROR_FIFO_DCU_FID_ID	Wrong Frame ID in science data received from DCU	• HW Fifo ID • Frame ID read
2579	5105	ERROR_FIFO_MCU_FID_ID	Wrong Frame ID in science data received from MCU	• HW Fifo ID • Frame ID read
257A	5105	ERROR_FIFO_SCU_FID_ID	Wrong Frame ID in science data received from SCU	• HW Fifo ID • Frame ID read
7FFF	51FF	DPU_DEBUG_EVENT_ID[25]	Flexible Debug Event Container Now used for Peak-UP	• Type : - 0x0001: Jiggle - 0x0002: Chop • 32bit x 10 Words Parameter Passed to VM in this Peak-UP phase

Table 18-1 Events, Reports and Warnings reports

[25] **DPU_DEBUG_EVENT_ID** see at Appendix A – Special telemetry packets.

18.2 Anomalies and Exceptions – TM(5,2) :

Event Code [hex]	SID [hex]	Event Name	Explanation	Returned Parameters
C000	5200	EXCP_DRCU_ANOMALY_SID	(NYI)	(NYI)
C010	5200	EXCP_DPU_ANOMALY_SID	(NYI)	(NYI)
C100	5200	EXCP_OBS_ANOMALY_SID	(NYI)	(NYI)
C110	5200	EXCP_OBS_CORRECT_SID	(NYI)	(NYI)
0832	0520	EXCP_FX_UNARMED_SID	Cannot execute the activity requested, the function wasn't activated.	None
vmArg	5201	EXCP_VM_EXCP_SID[26]	An Event Exception Report from VM The Event Code is Dynamically defined by the Virtual Machine	Variable length array of parameter defined by Virtual Machine.
5AFF	5202	EXCP_AFX_PUSH_ACT_ID	Autonomy action execution	• Autonomy action
5AFE	5202	EXCP_AFX_SAFE_MODE_ID	THE OBS IS GOING TO SAFE-MODE	Scan Mode Scan Iteration
afxArg	5202	EXCP_AFX_PUSH_ACT_SID	Flexible Structure – actually not used [OBS 4.0.0]	N/A
0503	5203	EXCP_PEAK_UP_ERROR_ID	Event Exception Report from Peak-Up Procedure	• Peak-Up Phase 0x00YZ: ○ Y = 1 if Jiggle ○ Y = 2 if Chop ○ Z = 0 – Entry ○ Z = 1 – VM Running implies Wait for Data ○ Z = 2 – HS Data Collection ○ Z = 3 – Exit Phase • Iteration Count
7FFF	52FF	DPU_DEBUG_EVENT_ID[27]	Flexible Debug Event Container Now not used for Peak-UP	• (NYI)

Table 18-2 Anomalies and Exceptions reports

[26] **EXCP_VM_EXCP_SID** see at Appendix A – Special telemetry packets.

[27] **DPU_DEBUG_EVENT_ID** see at Appendix A – Special telemetry packets.

18.3 Alarms and Errors reports – TM(5,4) :

Event Code [hex]	SID [hex]	Event Name	Explanation	Returned Parameters
550C	5420	ALARM_LSDCU_DEAD	The command ID in the echo sent back by the DRCU in response to a command, is not identical to the one 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
550D	5420	ALARM_LSMCU_DEAD	Same as above	Same as above
550E	5420	ALARM_LSSCU_DEAD	Same as above	Same as above

Table 18-3 Alarms and Errors reports

19 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.

19.1 TC Verification Error Structure

TM (1,8) packets are structured as follow:

Packet structure:

<table>
<tr><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td><td colspan="11">APID1</td></tr>
<tr><td>1</td><td>1</td><td colspan="14">Count</td></tr>
<tr><td colspan="16">Length</td></tr>
<tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td></tr>
<tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr>
<tr><td colspan="16">TIME</td></tr>
<tr><td colspan="16">TC_Packet_ID</td></tr>
<tr><td colspan="16">TC_Packet_Sequence_Control</td></tr>
<tr><td colspan="16"><i>Failure Code</i></td></tr>
<tr><td colspan="16">TC_Packet_Length</td></tr>
<tr><td colspan="16">TC_Packet_Type</td></tr>
<tr><td colspan="16">TC_Packet_SubType</td></tr>
<tr><td colspan="16">TC_Packet_CrcSum</td></tr>
<tr><td colspan="16"><i>Parameter</i></td></tr>
<tr><td colspan="16">Checksum</td></tr>
</table>

.

Field Description

Field Name	**Content**
TC_Packet_ID	Copied barely from the TC
TC_Packet_SSC	Copied barely from the TC
Failure Code	*Variable Error Code* *Described in the next paragraph*
TC_Packet_Length	Copied barely from the TC
TC_Packet_Type	Copied barely from the TC
TC_Packet_SubType	Copied barely from the TC
TC_Packet_CrcSum	Copied barely from the TC
Parameter	*Variable length of parameter dependently by Failure Code*

.

Table 19-1 TC Verification Error Structure

19.2 TC Verification Error Codes

TM (1,8) packets will contain an error code and a variable list of parameters according to the following table.

Code	Error Name	Description	Parameters
`0x0601`	`Illegal_Memory_ID`	The specified Memory ID is not in the valid range 0-3	• The requested Mem_ID
`0x0602`	`Illegal_Start_Address`	The Start Address is not in the valid range for the requested Memory ID (see below)	• 8 Bit Padding • 24 Bit Required Start Address
`0x0603`	`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
`0x0604`	`Bad_NSAU`	The number of SAUs does not match with the TC packet length contained in the TC packet header	• Uplinked number of SAUs
`0x0605`	`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
`0x0606`	`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 ***ATTENTION MAYBE A CORRUPTED MEMORY CELL***	• OBS computed CRC after load
`0x0805`	`Illegal_Table_ID`	The Table specified is not in the valid 0-255 range	• The required Table ID
`0x0806`	`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
`0x0808`	`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
`0x0809`	`Table_Space_Full`	Not enough memory to create the table of the required size (in the *Set Table* TC)	• Required table length
`0x080A`	`No_Command_List`	A Stop VM TC has been received, but the specified VM is not running	• The index of the VM required to stop (0 for HW VM)
`0x080C`	`VM_Running`	An *Execute* or *Start Comman List* TC was received, but the specified VM is already executing a command list	• The table in use
`0x080D`	`Bad_Ndata`	The munber of data words in a Report Table or Update Table TCs is inconsistent with the actual table length and start Index	• The uplinked number (in units of 32-bit words) • of data words

`0x080E`	`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 If a *Send DRCU Command* 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 DRCU command sent
`0x080F`	`ILLEGAL_FFLAGS`	The FIFO ID required to be reset or flush is not a valid FIFO ID	• The uplinked FIFO flag word
`0x0810`	`VM_UNDEFINED_TABLE_ID`	The Table ID specified as input for a VM was not previously defined	• The requested Table ID
`0x0811`	`Undefined_Table`	The table for which an update or report has been requested, was not previously defined	• The requested Table ID
`0x0812`	`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
`0x0813`	`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 • 32Bit User Lock Mask (for debug purpose only)
`0x0814`	`PeakUp_Running`	There is already another Peak-up procedures running	• The VM Table used during that Peak Up.
`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
`0x081A`	`Defrag_error`	An error during Garbage Collection *Hardware dependant* ***MAYBE A CORRUPTED MEMORY CELL.***	• none
`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	• The requested minimum and maximum S/S command IDs
`0x081E`	`CIS_WRONG_BROADCAST`	Inhibition has been requested for a broadcast command	• Requested minimum S/S command ID • Requested maximum S/S command ID
`0x081F`	`(DELETED)`	(DELETED)	(DELETED)
`0x0820`	`(DELETED)`	(DELETED)	(DELETED)

0x0821	Illegal_HK_Packet_ID	The HK Packet ID contained in a TC is not in the allowed range 0-3	• Uplinked HK 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.	• The 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 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	• MonItemID(32) • Command (32) • Bit select (16) • Bit offset (16)
0x082C	Undefined_Autonomy_Function	The autonomy function required to start following a HK monitoring anomaly, is not defined	As above
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
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	Illegal_Mon_Item	A requested monitoring item is extracted from an HK parameter, but with the wrong bit shift.	• MonItemID(32) • Command (32) • Bit select (16) • Bit offset (16)
0x0835	MON_MEM_FULL	Not enough DM available to allocate the data monitoring internal structures	• None
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 EE-PROM 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	(DELETED)	(DELETED)	(DELETED)
0x0841	(DELETED)	(DELETED)	(DELETED)
0x0842	(DELETED)	(DELETED)	(DELETED)
0x0843	LS_ENQUEUE_ERROR	Cannot Enqueue a LS request to the dedicated task LS	• The S/S Command
0x0844	MP_INTERNAL_ERROR	If “Direction” is “0x0001” : a low level error code : ▪ 0 – OK ▪ 1 – NUM OF WORDS WRONG ▪ 3 - ILLEGAL_DIRECTION	• Code [see on left] • 16 Bit Direction. • 32Bit Start Address in SEG_PMHI • 32Bit number of word to patch
0x0845	MP_NOT_MIRRORED_ERROR	Attempting patch the current OBS using an image of OBS that is never been cloned form the running one. This intention is forbidden. [since 2.2.H]	• Code 16 Bit :[0xCE00] • 16 Bit Direction. [0x0002] • 32Bit Start Address in SEG_PMHI • 32Bit number of word to patch
0x0846	(DELETED)	(DELETED)	(DELETED)
0x0847	(DELETED)	(DELETED)	(DELETED)
0x0848	(DELETED)	(DELETED)	(DELETED)
0x0849	(DELETED)	(DELETED)	(DELETED)
0x0830	Function_Active	The Function is already activated	• Function Enable Status
0x0831	Function_Stopped	The Function is already stopped	• Function Enable Status
0x0850	Function_Always_Active	The Function is always enabled Cannot start/stop the requested functionality because it is always active.	• The Function-Activity Id.
0x0851	Undefined_Mon_Dependency	A Monitoring Item refers to another that-s not present	• MonItemID(32) • Command (32) • Bit select (16) • Bit offset (16)
0x08FF	TM_1_8_Generic_DEBUG_error	Intentionally generated Error Acceptance Packet advising that a DEBUG/SELF TEST Function is enabled.	Dynamic Debug Content Usually: • TC Arguments cloned
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	• Sub-Type
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
0x0EEE	[28]AFX_TC_INHIBITHED	The Telecommand cannot be executed because an internal autonomous action is been triggered, and the nominal telecommand execution is been stop. NOTE: The only way to unlock this condition is to send a STOP_MONITORING COMMAND TC(8,2,CD). **WARNING:** **A stop monitoring command is sent, the Telecommand execution is recovered at the first useful monitoring moment, with a delay of a single autonomy action execution time.**	• NONE

Table 19-2 TC Verification Error Codes

[28] AFX_TC_INHIBITHED see at Appendix A – Special telemetry packets.

A. Appendix A – Special telemetry packets.

In order to trace the behaviours of the OBS and the VM-Operations, *in situ*, some debug information are collected and send back to ground.

Those packets are listed in the table here:

TYPE	SUBTYPE	SID	EVENT CODE	MORE INFO AT: NOTE/PAGE	RAPID INTERPRETATION
1	8	0x08FF	n/a	Note 28 Page 79	The OBS is executing an autonomy action. And it is in an inhibited state.
5	1	0x5113	any	Note 24 Page 68	An Event from Virtual Machine see the related Event Code
5	1	0x51FF	0x7FFF	Note 25 Page 71	Debug Data
5	2	0x5201	any	Note 26 Page 72	An Event from Virtual Machine see the related Event Code
5	2	0x52FF	0x7FFF	Note 27 Page 72	Debug Data
21	4	0x07FF	n/a		Debug Data

Table A-1 Special Telemetry Packets

-- end of document --