Herschel PACS

DPU OBS
Software Specification Document

Document Ref.: PACS-CR-SR-013

Issue: 3.1

Prepared by:	Stefano Pezzuto Daniele Schito	Date:	5th May 2006
Approved by:	Renato Orfei		

Contents

Document Status Sheet

Document Title: DPU On Board Software Specification Document			
Issue	**Revision**	**Date**	**Reason for change**
Draft 1		21st January 2002	Formerly issued as PACS-CR-TN-005 and containing the logical model only
Draft 2		30th January 2002	Inserted comments on the logical model by HF. First version of the architecture design
1	0	5th December 2003	First Issue
1	1	22nd September 2004	Issue for delivery to CGS
2	0	9th November 2004	RID from CGS
2	1	25th November 2004	2nd set of RID from CGS
2	2	1st December 2004	Modification to take into account ECSS-E-40 standard
3	0	16th February 2006	Version for code review
3	1	5th May 2006	Comments received during code review

Document Change Records

Document Title: DPU On Board Software Specification Document	
Document Reference Number: PACS-CR-SR-013	
Document Issue/Revision Number: 3/0	
Section	**Reason For Change**
3.2	Clarified
3.3	Completed Figure 3-3
3.10	New section for IRQ_3
4.1.1	OBS-FSR-TC2 corrected
6	Upgraded • OBS-SEC-SR7 (see OBS-CUR-SM2) • OBS-SEC-SR8 (see OBS-CUR-SM2)
Appendix A	Upgraded

[...]
ogni merlo crede d'aver messo nel fischio
un significato fondamentale per lui, ma che solo lui intende;
l'altro gli ribatte qualcosa che non ha nessuna relazione
con quello che lui ha detto; è un dialogo tra sordi,
una conversazione senza capo né coda.
Ma i dialoghi umani sono forse qualcosa di diverso?
I. Calvino. Palomar

1 Introduction

This document presents the on board software of the DPU, the subsystem of PACS that interfaces the instrument with the Herschel spacecraft. The DPU runs two different software, the boot software that is started when the unit is switched on, and the application software, that is started by the boot software when a specific command is sent. This document covers the application software only.

For the SW development PACS adopts the standard given in AD1 tailored to be fully equivalent to RD1.

1.1 Definitions, acronyms and abbreviations

1.1.1 Definitions

Event As far as the telemetry is concern, an event is a kind of telemetry packet (Service 5 of RD2: Event Reporting) signaling that something non nominal occurred, or is occurring, in the instrument. In the VIRTUOSO terminology, an event is a kernel service (generically named KS_Event in the architecture design) used for intertask communications. The particular meaning of the word can be easily understood from the context in which it is used

Function Quoted from RD2: *The term* **Function** *refers to conceptual software processes, for which the ground normally has the capability to exercise direct control over these processes. Functions shall be used in support of external physical interfaces*. For the DPU OBS, subsystems act as external interfaces which are commanded by the ground to execute specific commands, so any subsystem (including the DPU itself) is seen by the OBS as a function. In this document, the term function is also used for the autonomy functions, software routines that the DPU executes in case of out-of-limit conditions. They are conceptually different from functions as defined in RD2 and for this reason in this document they are always named autonomy functions, and never functions alone

VIRTUOSO is the operating system (by WindRiver) in which the DPU OBS runs. The OBS is structured in tasks (see Sections 4 and 5) which are independent programs running in parallel. The operating system is in charge, among other things, to schedule/deschedule the tasks preserving their data in memory, to provide intercommunications services

1.1.2 Acronyms

APID	Application Identifier
CASE	Computer Aided Software Engineering
CDMS	Command and Data Management System
CNR	Consiglio Nazionale delle Ricerche
DEC	Detector Controller
DPU	Digital Processing Unit
EEPROM	Electrically Erasable Programmable Read Only Memory
ESA	European Space Agency
FIFO	First In First Out

HK	HouseKeeping
ICD	Interface Control Document
IFSI	Istituto di Fisica dello Spazio Interplanetario
INAF	Istituto Nazionale di Astrofisica
MEC	Mechanical Control Unit
OBCP	On Board Control Procedure
OBS	On Board Software
PACS	Photoconductor Array Camera and Spectrometer
PROM	Programmable Read Only Memory
RAM	Random Access Memory
ROM	Read Only Memory
SPU	Signal Processing Unit
SW	SoftWare
TC	TeleCommand
TM	TeleMetry
UR	User Requirement
URD	UR Document

1.2 References

1.2.1 Applicable documents

ID	Title	Number/version/date
AD1	Space engineering - Software - Part 1: Principles and requirements	ECSS-E-40 28 November 2003
AD2	DPU/ICU On Board Software User Requirements Document	PACS-CR-RD-001 Issue 3.0. 5 May 2006
AD3	DPU/ICU On Board Software Product Assurance Plan	IFSI/OBS/PL/2000-001 Issue 1.1. 2 April 2001
AD4	PACS PS-ICD Usage	PACS-CR-TN-015 Issue 2. 25 March 2003
AD5	Herschel Science Ground Segment to Instrument ICD	FIRST-FSC-DOC-0200 Issue 1.3. 13 March 2003
AD6	Interface Control Document DEC/MEC–DPU	PACS-CL-ID-003 Issue 3.5. 3 October 2003
AD7	SPU High Level Software to DPU Interface Description	PACS-TW-ID-001 Issue 6.0. 13 December 2005
AD8	PACS and LFI SPU_SUSW - DPU_ASW Protocol	PACS-IC-TN-001 Issue 01, Rev. A. 2 August 2001

1.2.2 References documents

ID	Title	Number/version/date
RD1	Guide to applying the ESA software engineering standards to small software projects	BSSC(96)2 Issue 1. May 1996
RD2	Herschel/PLANCK Packet Structure Interface Control Document	SCI-PT-ICD-07527 Issue 5.0. 20 July 2004
RD3	DPU OBS User Manual	PACS-CR-UM-024 Issue 2.0. 18 April 2006
RD4	DPU OBS Detailed Design Document	PACS-CR-DD-023 Issue 1.0. 24 February 2006
RD5	List of PACS Housekeeping and Telecommands	PACS-ME-LI-005 Issue 1.3. 6 April 2006
RD6	AxiomSys User's Manual	©1992-1999 by Structured Technology Group, Inc.

1.3 Overview of the document

In the next two sections the OBS architecture design is presented. On the base of this design the software requirements are derived and presented in Section 4. Some estimates of the required resources, mainly on memory size, are then given in Section 5 while the traceability of the software requirements is in Section 6. At the end of the document the Data Dictionary is given to fully explain all the Data and Control Flows used.

2 System design

In this section the architecture design is presented, showing the context diagram and the decomposition at the tasks level. In the next section each task is described and, for the most complex ones, resolved in its components.

2.1 Design method

The software has been designed in agreement with AD3: the CASE tool used is AxiomSys[1] which implements the Yourden-De Marco Structured Analysis method, and the Hatley-Pirbhai Architecture Modeling method. For more informations see RD6.

Process input and output data are defined as data dictionary items, or *data items* for short. They are stored in the Data Dictionary reported in the Appendix A of this document.

2.2 Decomposition description

The OBS has two external interfaces: the first one is towards the spacecraft through the MIL-STD-1553B serial interface, driven by the dedicated chip BU61582 (BU61580 in the non rad-hard configuration); the second one is with the subsystems, via three independent serial links, driven by the dedicated chip SMCS332, following the IEEE 1355 standard. The context diagram is shown in Figure 2-1.

Communications with the 1553 interface are activated by the reception of an interrupt (INT_1553). The OBS is informed (1553_Messages control flow) if the CDMS is sending a TC packet, is available for a TM packet transfert, or is providing other info (e.g. time informations). RX 1553 and TX 1553 are the data flows in input and output.

For the 1355 interface another interrupt line (INT_1355) is used which, as before, can have different meanings (reception/transmission of packets, errors on the link). The Crx 1355 controls the input data flow

[1]© 1992-1999 by Structured Technology Group, Inc.

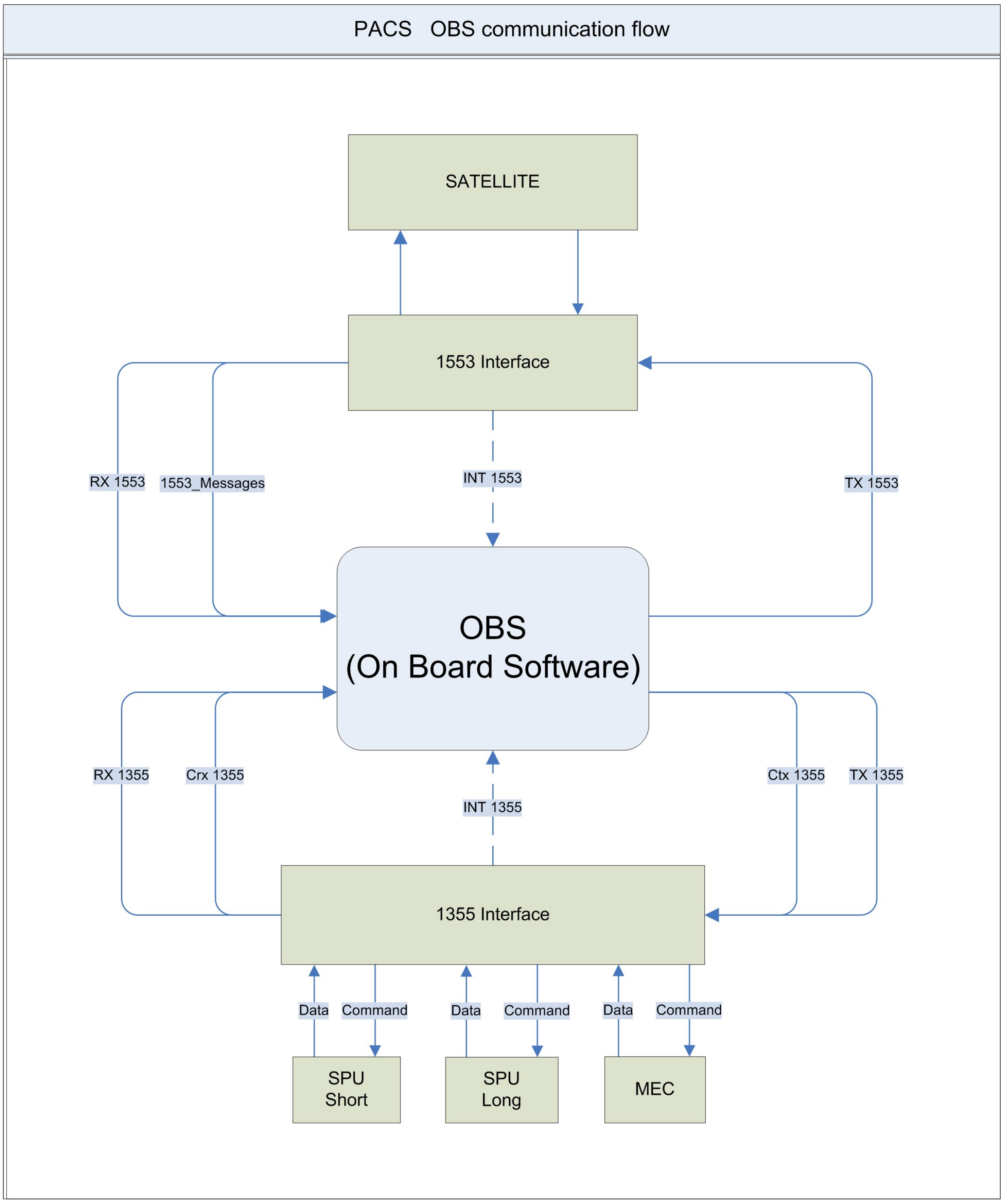

Figure 2-1: *The context diagram of the OBS architecture*

RX 1355 (HK, acknowledgment of commands, science data) from the subsystems while Ctx 1355 controls the output data flow TX 1355 (commands) from the OBS.

In Figure 2-2 the OBS decomposition at the task level is shown. The 1553 interface is served with one module (ISR IRQ2) and one task (T2TMTCIF); the 1355 interface requires another interrupt handler (ISR IRQ1) and one task (T3_IRQ1SV) that decodes the arrived data and decides which, if any, receiving task should be started (T6_MECRX, T7_SPSRX, T8_SPLRX). HK data (Dec_values, Sps_ values and Spl_values) are directly sent to the subsystem controller task (T5_HKMON), which monitors the instrument health by periodically checking that the collected HK parameters are in the expected range. The OBS controller (T4CNTRLR) decodes the telecommands from the satellite (TC_Packets), executes the DPU commands, sends the measurement instructions to the subsystems and drives the OBCP task (T9_OBCP). On top of them there is also a task at higher priority (T1_INIT, not shown in figure), which handles the unit when the application software is started, initialises all the components, and remains eventually idle.

UpdateTM is a library function that is called from all the tasks when a TM packet (TM_Struct) is to be sent to satellite. This function adds the time stamp to the packet and controls the status of three pools (Pool_EV_packets, Pool_HK_packets and Pool_SC_packets) where the packets are buffered before the final delivery to the satellite. 1355 TX is the library function that sends the commands to the subsystems, waits and processes the acknowledgment.

The following intertask communications services provided by VIRTUOSO are used: task T2TMTCIF is activated by the event ISR_1553_EVENT; if a telecommand has arrived the controller is informed through the FIFO TC_QUEUE; if the execution of an OBCP is requested the task T9_OBCP is woken up by event STARTPROC; the task T3_IRQ1SV is activated through the semaphore SEMA_1355_INT which calls the three 1355 receiving tasks through events INT_DEC, INT_SPS and INT_SPL.

3 Component description

The tasks introduced in the previous section are here discussed in more details showing, in particular, their components and the input and output data/control flows. The description at the single routines level can be found in RD4.

3.1 IRQ2 handling (1553 interrupt)

This module handles the 1553 interrupt. It consists of an assembly routine (ISR IRQ2) and an event handler routine (Handler_1553) (see Figure 3-1).

The 1553 interrupt line is connected to IRQ2, one of the external interrupt available on the 21020 chip. INT 1553 is the associated control flow and is signaled by hardware. Through 1553_Messages the bus controller informs the DPU about the kind of request (TC packets, TM packets, sync signal).

3.1.1 ISR IRQ2

This is the low level interrupt service routine written in assembler. It signals to VIRTUOSO, through the Call_Event_Handler control flow, to call Handler_1553 which signals back to VIRTUOSO if the 1553 associated event is to be activated.

3.1.2 Handler_1553

This function is called by VIRTUOSO on reception of a 1553 interrupt. On the base of the informations provided by the 1553 chip (1553_Messages), it decodes the kind of service requested by the Bus Controller. Low level actions like time reception or providing status words are directly done inside this function. On the other hand, if high level actions are necessary (TC/TM processing) the handler informs VIRTUOSO, again through Call_Event_Handler, that T2TMTCIF is to be scheduled.

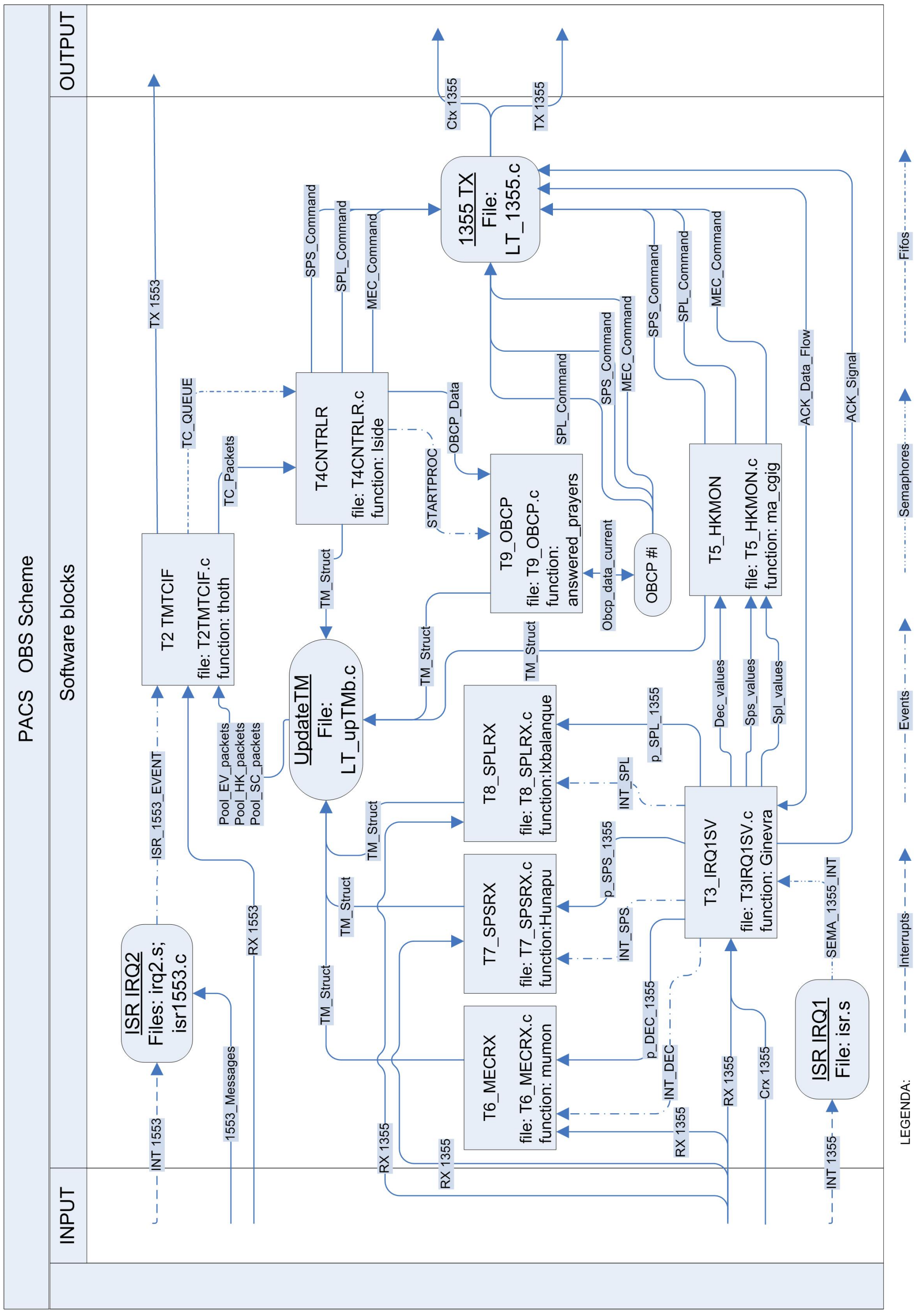

Figure 2-2: *The architecture design of the OBS. Rectangles are tasks: file: gives the name of the file containing the entry point shown after function:; ovals are library functions; OBCP #i is the i-th procedure called inside T9_OBCP. Note that in order to simplify the figure, the bidirectional data flow ACK_Data_Flow connects directly T3_IRQ1SV with 1355 TX while it is actually sent from T3_IRQ1SV to the three 1355 receiving tasks and then to 1355 TX*

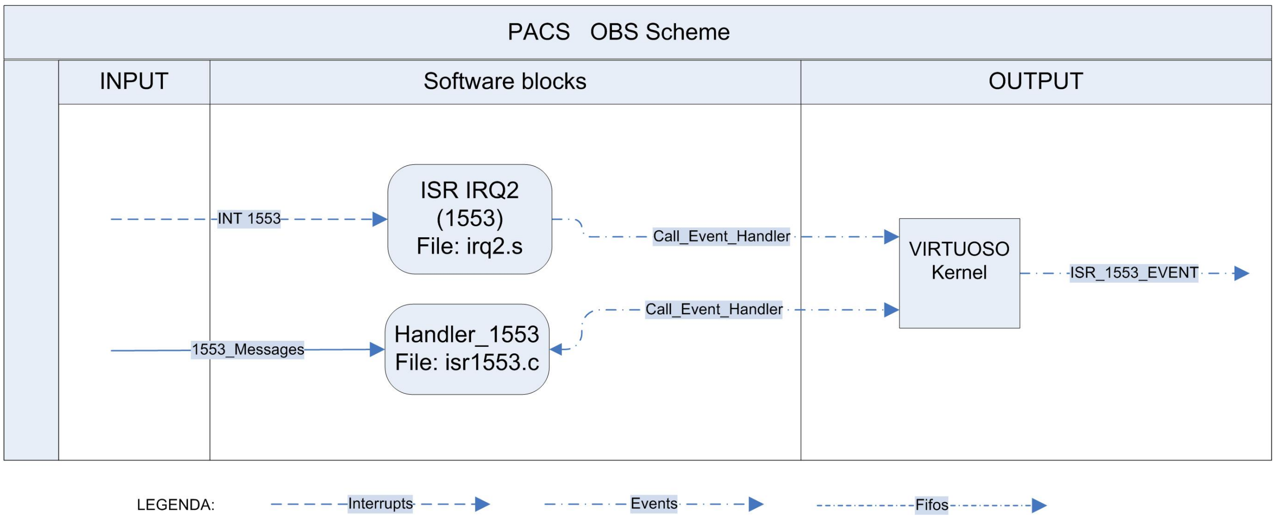

Figure 3-1: *Decomposition of the module which receives and decodes the 1553 interrupt. ISR IRQ2 is an assembly routine that, through the operating system VIRTUOSO, calls the event handler routine Handler_1553*

3.2 T2TMTCIF

In Figure 3-2 the task T2TMTCIF is shown. It receives in input the bit stream from the 1553 interface (RX 1553) and reconstructs the TC packet which is then copied (TC_Packets) by the controller: **no TC buffering is foreseen**. Informations on this packet are sent via the FIFO TC_QUEUE. Since the TC packet is handled autonomously by these two tasks, then it is possible for the DPU to receive a TC during the execution of the previous TC, but if more than one TC is received the newer are lost and a corresponding counter is incremented.

If the Bus Controller is available for a TM packet download, this task checks the status of the three pools Pool_EV_packets, Pool_HK_packets and Pool_SC_packets, in this order. If a packet is ready to be transmitted it is fetched from the pool, the source sequence counter is written according to its APID and finally the whole checksum is computed and written. The packet is then copied (TX 1553) in the Dual Port RAM of the 1553 interface.

Note that if more than one packet is found in the pool(s), the OBSW writes as many packets as possible according to the available space in the Dual Port RAM. It is then the 1553 chip that autonomously sends the packets, one by one, following the requests of the Bus Controller.

3.3 OBSW Controller

This task (see Figure 3-3) controls the TC execution. It is activated once the FIFO TC_QUEUE is filled in with the pointer to the telecommand buffer (TC Packets). The controller extracts the packet from the buffer and checks its content. The TC verification report is prepared and if the checks are passed the execution is started. Commands for the subsystems are sent using the library function 1355 TX, commands for the OBCP are sent to the OBCP task. The library function UpdateTM receives the telemetry packets, adds the time stamp and sends them to the TM pools.

3.3.1 Controller

This module contains the entry point of the task. It sleeps until a new telecommand is received. The telecommand is copied from TC Packets into a local buffer and then sent to TC_Acceptance to verify its consistency with respect to ESA standard. Acceptance_Result carries the result of the checks: if one of them is not passed the execution is not started, otherwise the command is interpreted according to the requested service. The following services are directly served inside the Controller: 9) Time Management; 14) Packet Transmission

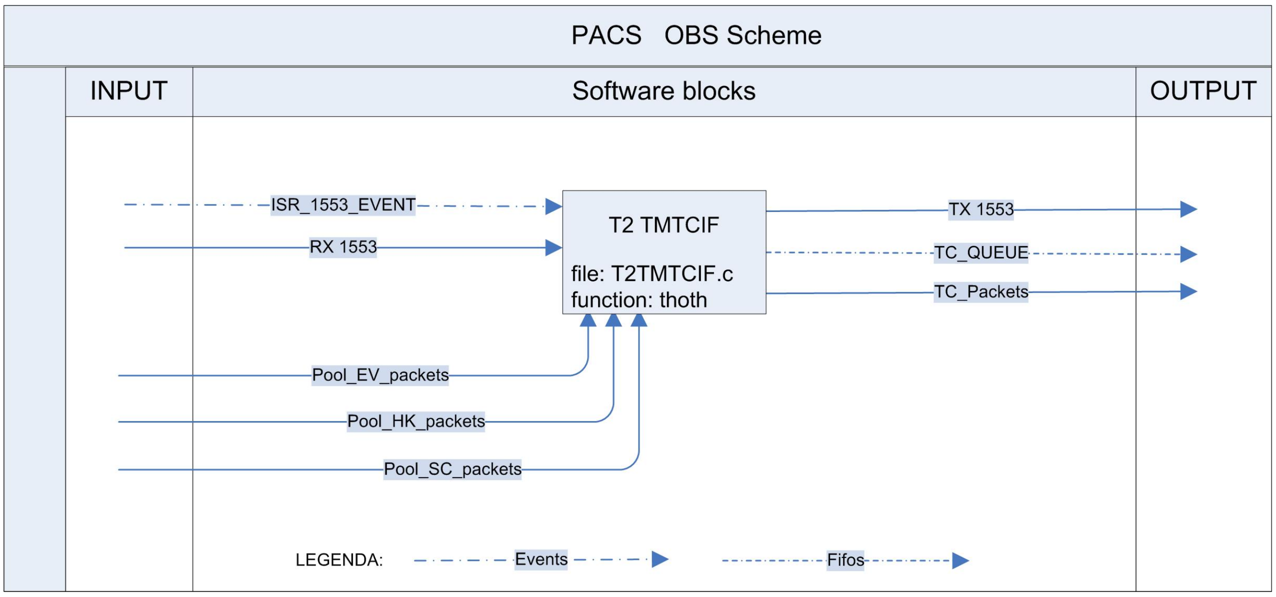

Figure 3-2: *Decomposition of the task T2TMTCIF*

Control; 17) Test Service. The memory management and the function management services are executed by specific modules. Another module takes care of service 18 (OBCP) which is, from an operational point of view, the most important service used by PACS.

Note that both TC Packets and the Controller local buffer have a depth of one single telecommand which means that at most one TC can be received while the previous one is under execution!

3.3.2 TC_Acceptance

This module implements the Telecommand Verification Service. The following fields, in this order, are verified: APID; packet length; checksum; type; subtype. If one check is not passed the TC is rejected without performing the next checks. There is another mandatory check to be performed, about the illegal or inconsistent application data field, that can not be done at this level since some TC are actually executed by the subsystems and their data field can not be checked by the DPU. This check is left to the following modules and/or to the subsystems.

The result of the checks is sent to the controller (Acceptance_Result) and written in a Verification Report: in case the checks are passed the report is a TM (1,1) which is sent (TM_Struct) to UpdateTM only if a specific bit in the telecommand packet is set to 1. If one of the checks fails the report is a TM (1,2) which is always sent to UpdateTM.

3.3.3 Memory_service

This module implements the Memory Management Service to load/dump/check the memory of the DPU as well as that of the subsystems.

In case of a DPU memory load, the sequence of the operations follows the prescription given in RD2: the data words to be loaded are verified by computing a checksum on them and comparing the result with the checksum written in the packet (this is a checksum different from that written at the end of the telecommand). If the two numbers are equal the data are copied in the final destination and the checksum is again computed and compared with that sent inside the packet. If all these steps are successfull a TM packet (1,7) is issued, otherwise a TM packet (1,8) is generated, specifying which was the failure. If the failure happens after writing the new data in memory, the old memory content is copied back.

The dump and check commands are easier to be executed even if the former can require more than one telemetry packet depending on how many words are required to be dumped.

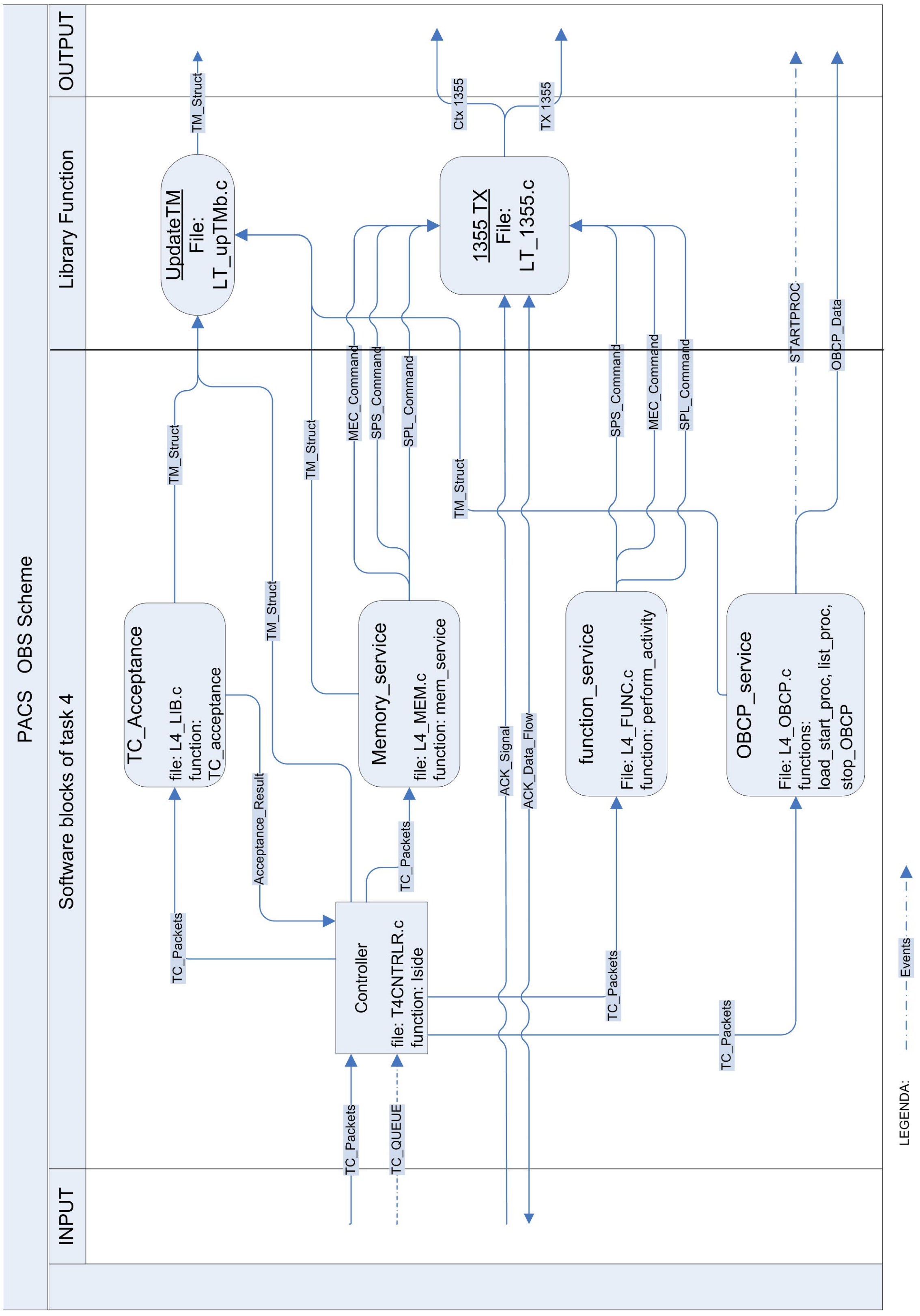

Figure 3-3: *Decomposition of the OBSW Controller*

In case the memory command is for one of the subsystems the module extracts the relevant data from the TC and prepares the packet for the subsystem according to the protocol defined in the ICDs. This packet is then sent to 1355 TX module which, in turn, transmits the packet to the subsystem. The dump command requires special attention because, as for the DPU, one single command can generate more than one TM packet.

3.3.4 function_service

This module is dedicated to Service 8 of RD2. PACS makes a limited use of functions, since the measurements are performed mainly via OBCP. The (8,4) request (perform activity of a function) is used to execute one single command for the DPU or for the subsystems. The commands implemented in the DPU OBS are reported in RD3. In case the command execution fails, a TM packet (1,8) is generated; in case of successful execution no TM packet is foreseen with the exception of the command to verify the integrity of PM for which a TM (1,7) is reported if the checksum is what expected (see RD3).

3.3.5 OBCP_service

Onboard control procedures (OBCP) are C functions compiled as part of the OBS or uploaded from ground. All the services requested in RD2 (starting, stopping, suspending and so on) are implemented using VIRTUOSO kernel services. This requires that the procedures are executed in a separate task (T9_OBCP in Figure 2-2).

The module OBCP_service starts, stops or suspends a running procedure, or stores in memory new uploaded parameters (OBCP_Data). Other services implemented are the provision of the list of all the procedures stored on-board, or of the list of all running procedures (for PACS at most one procedure is running). It is also possible to upload and to execute a brand new OBCP. The control flow STARTPROC, associated to a VIRTUOSO event, starts the execution of an OBCP.

3.4 T9_OBCP

This task sleeps until an event is set by the Controller when a Start OBCP command is received: the parameters contained in the TC are transported within the OBCP_Data flow. If the number of parameters sent with the telecommand is less than that used by the OBCP (a telecommand can be sent also with no parameters at all) the missing parameters are kept to the values of the last start command (or to zero the first time the OBCP is executed). Just before the beginning of the execution a TM packet (1,3) (TC execution started) is prepared and sent to UpdateTM. Once the procedure ends, a TM packet (1,7) or (1,8) is issued (TC execution completed or failed).

3.5 ISR_IRQ1

The 1355 interrupt line is connected to IRQ1, one of the external interrupt available on the 21020 chip. INT 1355 is the associated control flow and is signaled by hardware. The interrupt register of the 1355 chip is interpreted and if required the semaphore SEMA_1355_INT is incremented to start T3_IRQ1SV. The recognized interrupts are: end-of-packet sent, end-of-packet received, error (parity or disconnect) on the link.

3.6 T3_IRQ1SV

This task is executed when ISR_IRQ1, on reception of a 1355 interrupt which requires an action on DPU, increments the semaphore SEMA_1355_INT.

The content of the received packet is derived on the base of the header. Science and diagnostic HK packets are sent to the corresponding receiving 1355 task. Nominal HK packets are directly transmitted to T5_HKMON. Any other header is assumed to be an acknowledgment. In this case the reception of the ACK is first signaled to the module 1355 TX (ACK_Signal in Figure 2-2). Then the packet is sent to the corresponding 1355 task which cheks if the received acknowledgment is consistent with what DPU is waiting for (ACK_Data_Flow, but read the caption of Figure 2-2!). The receiving tasks are started by signaling the specific event: INT_DEC, INT_SPS and INT_SPL.

3.7 T6_MECRX

This task handles the packets received on the 1355 link connected to the DEC/MEC subsystem. It sleeps until the VIRTUOSO event INT_DEC is set by T3_IRQ1SV.

Diagnostic science data are packetised and sent to UpdateTM (see TM_Struct in Figure 2-2). Any other packet is considered a possible ACK. If the DPU was waiting an ACK from DEC, the sender task is informed (ACK_Data_Flow), otherwise an event is generated and the subsystem status is set to STOPPED. In order to simplify Figure 2-2, ACK_Data_Flow connects T3_IRQ1SV with 1355 TX. Actually this flow connects the three 1355 receiving tasks with the module function 1355 TX.

3.8 T7_SPSRX and T8_SPLRX

T7_SPSRX and T8_SPLRX are two independent tasks which perform the same activities, the only difference being that the former is linked to the short wavelength SPU, the latter to the long wavelength SPU.

Each task waits for an event set by T3_IRQ1SV (INT_SPS and INT_SPL respectively). Science data are put in a TM packet and sent to UpdateTM (see TM_Struct in Figure 2-2). A different APID is used for the two SPUs, while a different packet subtype is used to distinguish between photometry and spectroscopy data.

The ACK is treated exactly as in the case of T6_MECRX task.

3.9 T5_HKMON

The execution of this task starts every two seconds (some jitter may occur depending on the scheduling of the other tasks).

First, the hardware HK of the DPU are sampled and checked. Then the HK from each subsystem are monitored if the corresponding link status is not OFF and if a new HK packet has been received. If a new packet has not been received in the last two seconds, the HK status for that link is set to No New HK and the DPU writes in the HK packet the last received values.

Once all the checks have been performed, some status variables are set (see RD3) and then the TM HK packet is prepared and sent to UpdateTM (shown in Figure 2-2 as TM_Struct flow). In case of out-of-limit condition, an event is generated and, if defined, an autonomy function is started which may send commands to the subsystems.

3.10 ISR_IRQ3

This component is missing in Figure 2-2 because it is outside Virtuoso control. The FPGA external to the 21020 provides a programmable timer with a resolution of 1μs. Once the programmed time expires, the HW interrupt line connected to IRQ3 is asserted. The interrupt service routine starts its execution and checks the integrity of 6 DM memory cells. The sequence of operations is the following

1. Save the content of the cell
2. Compute the complement of the cell content (1's are turned into 0's and viceversa)
3. Write the new value
4. Read the new value and compares with what has been written
5. If the comparison is not OK write the address in a buffer; otherwise go on
6. Restore the original content of the memory cell

In this way the integrity of the Data Memory is verified. With the adopted choice of 6 memory cells, the whole DM is checked in about 1 days. With a memory load (see RD3) it is possible to change this value.

When the HK monitoring task wakes up, among the other checks it verifies both that this task is running (the next memory address to check must be different from the previous one) and that no new addresses have been written in the buffer (that can contain up to 32 entries) otherwise an event (5,4) is sent to the satellite.

3.11 Library functions

3.11.1 1355 TX

This module does not belong to a specific task but is available to the OBS as a library function. It transmits a command, with or without parameters, to a subsystem.

Each command is sent to the required link structured in packet as described in the ICDs. The OBS then waits for the acknowledgment from the subsystem within 200 msec: the reception of the ACK is communicated by T3_IRQ1SV using the flow ACK_Signal (see Figure 2-2). Then the ACK is processed by one of the three 1355 receiving tasks and finally it is sent to this module (ACK_Data_Flow). If the acknowledgment does not arrive or is negative the DPU raises an exception report (5,1) and the subsystem status is set to STOPPED.

This module ensures that until an ACK is not received it is not possible to send another command to the same or to another subsystem.

3.11.2 UpdateTM

This module is a library function available to all the tasks. It executes the following functions: 1) writes the time stamp of the packet; 2) checks if there is space in the specific buffer (one of the three pools); 3) copies the packet in the pool. In case of buffer overflow during step 2, HK or event packets are lost and an internal DPU counter is incremented (part of DPU SW HK). If the buffer overflow occurres for the generic TM buffer an event is generated and the science packets transmission is disabled, until the buffer is at least 25% free; then packet transmission is enabled again (the check of the available space is performed by the HK monitoring task, as part of the OBS autonomy functions).

4 Specific Requirements

In this section all the requirements are listed. Each requirement is uniquely identified by its ID code **OBS-xxx-yyii** where ii is a running number, and -xxx-yy are according to the following scheme:

-FSR- Functional requirements

TC	Telecommand acceptance
HK	Housekeeping reporting
EV	Event reporting
MM	Memory management
AF	autonomous functions requirements
FM	Function management
TM	Time management
PT	Packet transmission control
TS	Test service
CP	On-board control procedures
MC	Telemetry conventions

-PRF-SR Performance requirements

-IRF-SR Interface requirements

-OPR-SR Operational requirements

-SEC-SR Safety requirements

-QTY-SR Quality requirements

Unless otherwise specified, the source of requirements is AD2.

4.1 Functional Requirements

4.1.1 Telecommands acceptance

Req. ID	Description	Reference
OBS-FSR-TC1	The OBS shall receive all the instruments commands originating from the CDMS as Telecommands (TC) packets with the following fields: (see Figure 3.1-1 of RD2): `packet_id`, `packet_sequence_control`, `packet_length`, `data_field_header`, `application_data`, `packet_error_control`	OBS-CUR-TC3
OBS-FSR-TC2	The maximum packet length shall be 242 octets	OBS-CUR-TC3
OBS-FSR-TC3	The OBS shall interpret the two 16 bits words of `data_field_header` according to Section 3.1.2.1 of RD2	OBS-CUR-TC3
OBS-FSR-TC4	All data for the CDMS generated by the OBS shall be sent in Telemetry (TM) packets with the following fields (see Figure 4.1-1 of RD2): `packet_id`, `packet_sequence_control`, `packet_length`, `data_field_header`, `application_data`, `packet_error_control`	OBS-CUR-TM1, OBS-CUR-TM15
OBS-FSR-TC5	The maximum packet length shall be 1024 octets	OBS-CUR-TM1
OBS-FSR-TC6	The OBS shall fill in the five 16 bits words of `data_field_header` according to Section 4.1.2.1 of RD2	OBS-CUR-TM1
OBS-FSR-TC7	The OBS shall generate a Verification Report for each received TC in order to check the conformity of the packet to RD2	OBS-CUR-TC5, OBS-CUR-TC9
OBS-FSR-TC8	The OBS shall extract the APID of the TC from the first 11 bits of the `packet_id`. The APID of the Verification Report shall be identical to the received APID **Comment from IFSI**: in case the DPU receives a TC with wrong APID, the corresponding TM acceptance report shall contain the APID assigned to PACS, not the wrong one, which will be reported as part of the Application data field	OBS-CUR-TC3
OBS-FSR-TC9	The OBS shall check that the received TC have the APID field set according to the APID values contained in RD2 - Appendix 3. In case of invalid value the OBS shall generate a Telecommand Report Failure (1,2) with Failure-Code equal to 0 and Parameter field containing the `packet_id` of the TC. The TC shall not be executed	OBS-CUR-TC7, OBS-CUR-TC9

Telecommands acceptance (ctd.)

Req. ID	Description	Reference
OBS-FSR-TC10	If the previous check is passed, the OBS shall check that the value of `packet_length` is in accordance with the actual length of the received TC. If not so the OBS shall generate a Telecommand Report Failure (1,2) with Failure-Code equal to 1 and Parameter field containing the `packet_length` of the TC. The TC shall not be executed	OBS-CUR-TC7, OBS-CUR-TC9
OBS-FSR-TC11	If the previous checks are passed, the OBS shall compute the checksum of the whole TC packet and shall compare the result with the value of `packet_error_control`. The checksum algorithm is fixed for the complete mission and is defined in RD2, Appendix 4. If the two values do not match the OBS shall generate a Telecommand Report Failure (1,2) with Failure-Code equal to 2 and Parameter field containing the `packet_error_control` of the TC. The TC shall not be executed	OBS-CUR-TC7, OBS-CUR-TC9
OBS-FSR-TC12	If the previous checks are passed, the OBS shall check that the Packet Type value contained in the first word of `data_field_header` is one of the types used by PACS, reported in AD4 and forming a subset of the allowed values given in RD2. If not so the OBS shall generate a Telecommand Report Failure (1,2) with Failure-Code equal to 3 and Parameter field containing both words of `data_field_header` of the TC. The TC shall not be executed	OBS-CUR-TC7, OBS-CUR-TC9
OBS-FSR-TC13	If the previous checks are passed, the OBS shall check that the Packet Subtype value contained in the second word of `data_field_header` is one of the subtypes used by PACS, reported in AD4 and forming a subset of the allowed values given in RD2. If not so the OBS shall generate a Telecommand Report Failure (1,2) with Failure-Code equal to 4 and Parameter field containing both words of `data_field_header` of the TC. The TC shall not be executed **Comment from IFSI**: after this step RD2 requires to check the Application Data contained in the TC Data Field. Since PACS consists of many intelligent subsystems and the DPU is not requested to verify the application data of the commands for subsystem, this last check is not done. For DPU commands it is done as first step of the execution process; it is not done at all for subsystems commands	OBS-CUR-TC7, OBS-CUR-TC9
OBS-FSR-TC14	For all Telecommands which pass the acceptance checks, the DPU shall generate, only if required by a specific acknowledgment bit of the TC packet, a Telecommand Report Success (1,1) within 2 seconds. The Application data field shall follow Section 5.1.2.1 of RD2	OBS-CUR-TC5
OBS-FSR-TC15	The start of execution of a TC shall be reported, in case of an OBCP (see below), in a Telecommand Execution Report - Started (1,3), according to Section 5.1.2.2 of RD2	OBS-CUR-TC12

Telecommands acceptance (ctd.)

Req. ID	Description	Reference
OBS-FSR-TC16	At the end of successfull execution of particular telecommands, the OBS shall generate a Telecommand Execution Report - Success (1,7), according to Section 5.1.2.4 of RD2. The list of commands that require this report shall be available in RD3	OBS-CUR-TC12
OBS-FSR-TC17	For not started, unsuccessfully executed, or otherwise aborted TC executions, the OBS shall generate a TC Execution Report - Failure (1,8), according to Section 5.1.2.5 of RD2	OBS-CUR-TC13

4.1.2 Housekeeping reporting

Req. ID	Description	Reference
OBS-FSR-HK1	The OBS shall report HK data in TM packets (3,25). Three packets are presently defined: non prime (default at startup); spectroscopy; photometry. Each packet shall contain a pre-defined set of the instrument parameters (the full list shall be reported in RD3). The transmission of HK packets is started/changed automatically according to the instrument observing/operative mode	OBS-CUR-TM5
OBS-FSR-HK2	The OBS shall report a pre-defined set of HK parameters in an essential HK packet at a reduced data rate. This packet shall be distinguished from nominal ones by its APID, as specified in Appendix 3 of RD2	OBS-CUR-TM6, OBS-CUR-TM12
OBS-FSR-HK3	The structure of this essential HK packet shall be identical to that of non prime nominal HK packet. To be compatible with the TM data rate, this packet shall be generated every 10 seconds	OBS-CUR-TM12
OBS-FSR-HK4	The OBS shall report the collected HK parameters into the Reporting Housekeeping Data packet described in section 5.3.2.2 of RD2. The SID shall be: 1 for spectroscopy observing mode; 2 for photometry observing mode; 3 for non prime operative mode; 4 for essential HK packet	OBS-CUR-TM3
OBS-FSR-HK5	The OBS shall be able to select one of the three HK report with a specific TC	OBS-CUR-TM12
OBS-FSR-HK6	The OBS shall provide actual values of the HK parameters and not changes (or delta values) since the last readout	OBS-CUR-TM11

4.1.3 Event reporting

Req. ID	Description	Reference
OBS-FSR-EV1	The OBS shall support Event Reporting Service. The structure of the packet follows Section 5.5.2 of RD2. Parameters A shall be set to OBSID and BBID	OBS-CUR-TM16
OBS-FSR-EV2	The OBS shall support event report (5,1) for any asynchronous event or warning. The content of the fields Event ID, SID and Parameters B will be reported in RD3. Event Sequence Counter field shall have the two most significant bits set to 01	OBS-CUR-AF5

Field	Memory ID				Start Address
Bit	3	1	4	8	16
Name	Sub ID	Mem Type	Block	24 bit address	
Value	000 DPU	0 PRAM	0000 PROM		
	001 DEC	1 DRAM	0001 RAM		
	010 SPU blue		0010 Ext. RAM		
	011 SPU red		0011 EEPROM		
	100 SPU both		0100 SMCS DRAM		
			0101 1553 DRAM		
			0110 DRAM in PRAM		

Table 5: Memory ID and Start Address of Memory Management Service. The actual meaning of Block depends on the subunit, for the DPU it shall be given in RD3

Event reporting (ctd.)

Req. ID	Description	Reference
OBS-FSR-EV3	The OBS shall support exception report (5,2) in non-nominal cases for which an unscheduled onboard action is required. The content of the fields Event ID, SID and Parameters B will be reported in RD3. Event Sequence Counter field shall have the two most significant bits set to 10	OBS-CUR-AF5
OBS-FSR-EV4	The OBS shall support error/alarm report (5,4) for non-nominal events which require ground intervention. The content of the fields Event ID, SID and Parameters B will be reported in RD3. Event Sequence Counter field shall have the two most significant bits set to 11	OBS-CUR-AF5

4.1.4 Memory management

Req. ID	Description	Reference
OBS-FSR-MM1	The Smallest Addressable Unit (SAU) shall define the base for the memory management service. The OBS shall assume an address granularity of 32 bits for the 21020 Data Memory, and 48 bits for the 21020 Program Memory, for the DPU as well as for the other PACS subsystems memory	NTBT
OBS-FSR-MM2	The OBS shall support the memory load command using absolute addresses (6,2) adopting the procedure described in RD2, Section 5.6.1.2. At the end of successfull execution a TM Report - Success (1,7) shall be generated	OBS-CUR-TC2, OBS-CUR-TC12, OBS-CUR-TC15, OBS-CUR-SM6, OBS-CUR-SM7
OBS-FSR-MM3	The OBS shall decode the (6,2) packet content in accordance with RD2, Section 5.6.1.2. Memory ID and Start Address shall be decoded according to Table 5	OBS-CUR-SM6
OBS-FSR-MM4	The OBS shall support the memory dump command using absolute addresses (6,5) using the procedure described in RD2, Section 5.6.1.4	OBS-CUR-SM6
OBS-FSR-MM5	The OBS shall decode the (6,5) packet content in accordance with RD2, Section 5.6.1.4. Memory ID and Start Address shall be decoded according to Table 5	OBS-CUR-SM6

Memory management (ctd.)

Req. ID	Description	Reference
OBS-FSR-MM6	The OBS shall report the requested memory content in one or more telemetry packets according with Service (6,6) of RD2	OBS-CUR-SM6
OBS-FSR-MM7	The OBS shall support the memory check command using absolute addresses (6,9) using the procedure described in RD2, Section 5.6.1.5	OBS-CUR-SM3, OBS-CUR-SM6
OBS-FSR-MM8	The OBS shall decode the (6,9) packet content in accordance with RD2, Section 5.6.1.5. Memory ID and Start Address shall be decoded according to Table 5	OBS-CUR-SM6
OBS-FSR-MM9	The OBS shall report the resulting checksum in a telemetry packet according with Service (6,10) of RD2	OBS-CUR-SM6
OBS-FSR-MM10	The OBS shall be able to identify memory commands for the subsystems and shall send them according to the specific ICD	OBS-PUR-GE3
OBS-FSR-MM11	The OBS shall be able to receive memory dump packets from the subsystems, according to the specific ICD, and shall packet them according to service (6,6) of RD2	OBS-PUR-GE3
OBS-FSR-MM12	The OBS shall be able to receive memory check reports from the subsystems, according to the specific ICD, and shall packet them according to service (6,10) of RD2	OBS-PUR-GE3

4.1.5 Function management

The subsystems of PACS are intelligent units seen by DPU as external interfaces. For this reason Service 8 is used to command all the subsystems (in this sense the DPU itself is seen as a subsystem). The function ID identifies one particular subunit. The services to start and to stop functions are not actually used, since the units are switched on by the CDMS and the communications between the DPU and the subunits are started with a OBCP which involves the 1355 protocol. All the instances of Start/Stop function will be accepted with a (1,1) or (1,2) TM packet, but no action will be taken by the DPU. This also applies to TC (8,5): the DPU only reacts with a (1,1) or (1,2) without further action.

The OBS shall assign each function a status word. At startup this word shall be set to the (logical) value `OFF` until the corresponding 1355 link is activated, after that the word shall be set to the value `ON`. The communications with any subunit can be stopped via a DPU command: when it happens, the DPU shall set the status word to the value `STOPPED`. From now on the communications from DPU to subunit shall be stopped until restarted with another DPU command (see RD3). However, the communications from subunit to DPU (HK, science data) shall not be affected at all.

Req. ID	Description	Reference
OBS-FSR-FM1	The OBS shall implement Service 8 of RD2 to command the subsystems	OBS-CUR-TC1
OBS-FSR-FM2	The OBS shall accept TC (8,1) but beside reporting a TM (1,1) or TM (1,2) no further action will be taken	AD4
OBS-FSR-FM3	The OBS shall accept TC (8,2) but beside reporting a TM (1,1) or TM (1,2) no further action will be taken	AD4
OBS-FSR-FM4	The OBS shall send measurement instruction to subsystems using service (8,4) of RD2	OBS-CUR-TC11, OBS-CUR-TM9, OBS-CUR-TM13
OBS-FSR-FM5	The OBS shall decode the (8,4) packet content in accordance with RD2, Section 5.8.1.4. The meaning of the fields is reported in Table 8	OBS-CUR-TC3

Field	Function ID	Activity ID	SID	Parameter(s)
Bit	8	8	16	$n \times 32$[1]
Value	0x64 DPU		n	
	0x65 SPU blue			
	0x66 SPU red			
	0x67 DEC			

[1]also $n \times 16$ for DPU commands

Table 8: Function ID for Function Management Service. For Activity ID see the User Manual of each subunit (RD3 for the DPU). The DPU-subunits ICD distinguish between trigger and write commands: the latter have SID set to 4 and the Parameters field has a special meaning (see the ICD or RD3)

Function management (ctd.)

Req. ID	Description	Reference
OBS-FSR-FM6	If the function status is **active** the command shall be sent to the subunit	OBS-CUR-TC11
OBS-FSR-FM7	If the function status is **stopped** the command shall not be sent to the subunit and the DPU shall raise an event report (see RD3) as well as a TM (1,8)	OBS-CUR-TC11
OBS-FSR-FM8	The OBS shall accept TC (8,5) but beside reporting a TM (1,1) or TM (1,2) no further action will be taken	AD4

4.1.6 Time management

Req. ID	Description	Reference
OBS-FSR-TM1	The OBS shall synchronize its internal clock copying the Time Message received from the 1553 Bus Controller according to Appendix 9, Section 4.3 of RD2	OBS-CUR-SY1
OBS-FSR-TM2	The time-tagging of TM packets shall not be suspended at any time. The OBS shall adopt the time format described in Appendix 6.3.8 of RD2	OBS-CUR-TM15
OBS-FSR-TM3	The OBS shall verify its internal time with the CDMS master time on reception of telecommand (9,7) described in section 5.9.1.5 of RD2	OBS-CUR-SY1
OBS-FSR-TM4	On reception of a time verification telecommand, the OBS shall report its internal time in a telemetry packet (9,9) filled in according to section 5.9.2.2 of RD2	OBS-CUR-SY3

Note on this last requirement. Quoting from RD2: “*After reception of a TC(9,7) the application addressed shall generate a Time Verification Report TM(9,9) at the moment of reception of the next synchronisation signal.*”. As such, this requirement can not be implemented due to the fact that the telemetry packets can not be sent directly through the 1553 interface, but they are copied in the 1553 dual port RAM which acts as a buffer and delivered to the CDMS according to the low level protocol interely handled by the 1553 chip. If the dual port RAM is already occupied when the TC (9,7) is received, it is not possible to send the TM (9,9) at the moment of the reception of the next synchronisation signal. For this reason, the content of packet (9,9) is the time at the reception of the last synchronisation signal plus 1 second.

4.1.7 Packet transmission control

Req. ID	Description	Reference
OBS-FSR-PT1	The OBS shall support Service (14,1), Enable Generation of Telemetry Packets, as described in Section 5.14.1.1 of RD2	OBS-CUR-TC16
OBS-FSR-PT2	The OBS shall support Service (14,2), Disable Generation of Telemetry Packets, as described in Section 5.14.1.1 of RD2	OBS-CUR-TC16
OBS-FSR-PT3	The OBS shall ignore any telecommand to disable the generation of specific telemetry packets listed in RD3	OBS-CUR-TC16
OBS-FSR-PT4	The OBS shall support Service (14,3), Report Enabled Telemetry Packets, as described in Section 5.14.1.2 of RD2	AD4
OBS-FSR-PT5	On reception of a telecommand (14,3) the OBS shall report the list of enabled telemetry packets, as described in Section 5.14.2 of RD2	AD4

4.1.8 Test service

Req. ID	Description	Reference
OBS-FSR-TS1	The OBS shall support Service (17,1), Perform Connection Test, as described in Section 5.17.1 of RD2	OBS-CUR-SM8
OBS-FSR-TS2	On reception of a telecommand (17,1) the OBS shall generate a telemetry packet (17,2), as described in Section 5.17.2 of RD2	OBS-CUR-SM8

4.1.9 On-board control procedures

On-board Control Procedures (OBCP) are sequences of operations that the DPU executes one after the other, possibly with some logic inside. They are written in C and for the OBS they are callable functions. The entry points are stored in an array whose index is the procedure ID. Many of the operations the DPU executes are just measurement instructions for the other subunits, but a procedure can also contain commands for the DPU itself. The list of procedures with their ID and number of parameters shall be reported in RD3.

At any moment one only procedure can be under execution. It is not possible to start a new OBCP while another one is running, with the only exception of the `go_SAFE` procedure (see RD3): if the start command involves this OBCP while another OBCP is running, the latter is stopped and the former is started.

The execution of the telecommands for this service depends on the status of each procedure. For this reason, all the procedures shall have a status word which is set to one of the following values: `stopped`; `running`; `suspended`; `deleted`.

Deleting a procedure, according to RD2, implies to delete the code from the OBS. This is not implemented and deleting a procedure means that its status word is set to deleted and it can no longer be executed. To remove the code from OBS a patch can be applied (see RD3).

Req. ID	Description	Reference
OBS-FSR-CP1	The OBS shall support the upload of a new OBCP with Service (18,1), Load Procedure, according to Section 5.18.1.1 of RD2. At the end of a successfull load, a TM Report - Success (1,7) shall be generated	OBS-CUR-TC2
OBS-FSR-CP2	The OBS shall delete an OBCP on reception of a telecommand (18,2), Delete Procedure, according to Section 5.18.1.2 of RD2	OBS-CUR-TC2

On-board control procedures (ctd.)

Req. ID	Description	Reference
OBS-FSR-CP3	The OBS shall start an OBCP on reception of a telecommand (18,3), Start Procedure	OBS-CUR-TC1
OBS-FSR-CP4	The OBS shall start an OBCP according to Section 5.18.1.3 of RD2	OBS-CUR-AF9
OBS-FSR-CP5	On completion of an OBCP the OBS shall report either a telemetry packet (1,7), execution succesfully completed, or a packet (1,8), execution failure	OBS-CUR-TC12, OBS-CUR-AF9
OBS-FSR-CP6	The OBS shall stop an OBCP on reception of a telecommand (18,4), Stop Procedure, according to Section 5.18.1.4 of RD2	OBS-CUR-TC6, OBS-CUR-TC14
OBS-FSR-CP7	The OBS shall suspend an OBCP on reception of a telecommand (18,5), Suspend Procedure, according to Section 5.18.1.5 of RD2	AD4
OBS-FSR-CP8	The OBS shall resume the execution of an OBCP on reception of a telecommand (18,6), Resume Procedure, according to Section 5.18.1.6 of RD2	AD4
OBS-FSR-CP9	The OBS shall upgrade the parameters of an OBCP on reception of a telecommand (18,7), Communicate Parameters to a Procedure, according to Section 5.18.1.7 of RD2	OBS-CUR-TC15
OBS-FSR-CP10	The OBS shall report the list of OBCP on reception of a telecommand (18,8), Report List of OBCP, in a telemetry packet (18,9)	OBS-CUR-TM10
OBS-FSR-CP11	The OBS shall report the list of active OBCP on reception of a telecommand (18,10), Report List of Active OBCP, in a telemetry packet (18,11)	AD4
OBS-FSR-CP12	The OBS shall report the status and last set of parameters values of an OBCP on reception of a telecommand (18,12), Report OBCP Status, in a telemetry packet (18,13)	OBS-CUR-TM10

4.1.10 Telemetry conventions

Req. ID	Description	Reference
OBS-FSR-MC1	The OBS shall adopt the bit numbering conventions reported in Appendix A1.1 of RD2	OBS-CUR-TM1
OBS-FSR-MC2	The OBS shall adopt the field alignment conventions reported in Appendix A1.2 of RD2 **Comment from IFSI**: for HK packets this requirement can not be fullfilled, in order to insert as much parameters as possible. The HK values are written one after the other (packed)	OBS-CUR-TM1
OBS-FSR-MC3	The OBS shall adopt the parameters types and structures reported in Appendix A6 of RD2	OBS-CUR-TM1
OBS-FSR-MC4	The set of Telecommands packets handled by the OBS shall be in accordance with those reported in Table A-2.1 of RD2	OBS-CUR-TM1
OBS-FSR-MC5	The set of Telemetry packets handled by the OBS shall be in accordance with those reported in Table A-2.1 of RD2	AD4
OBS-FSR-MC6	The OBS shall adopt the APID assigned to PACS according to Table A-3.1 and Table A-3.2 of RD2	OBS-CUR-TM6

Telemetry conventions (ctd.)

Req. ID	Description	Reference
OBS-FSR-MC7	The OBS shall adopt the cyclic redundant check code according to the specification reported in Appendix A4.1 of RD2	OBS-CUR-TM1

4.2 Performance Requirements

Req. ID	Description	Reference
OBS-PRF-SR1	The OBS shall provide the conformity check report for each TC at the maximum command data rate (2 TC per second)	OBS-CUR-TC4
OBS-PRF-SR2	The OBS shall reject non-valid packets at the earliest possible stage in the on-board acceptance process	OBS-CUR-TC8
OBS-PRF-SR3	The OBS shall generate the TC verification report at the earliest possible stage in the on-board acceptance process	OBS-CUR-TC10
OBS-PRF-SR4	The OBS shall provide the nominal HK report at a rate of 0.5 Hz. For the subsystems no specific command is foreseen, they are expected to delivery their onw HK packet with a frequency not smaller than that of DPU	OBS-CUR-TM8
OBS-PRF-SR5	The OBS shall be able to support a total output telemetry rate of 100 Kbps averaged on 24 hours	OBS-CUR-TR1
OBS-PRF-SR6	The OBS shall be able to support a burst mode of 350 Kbps averaged on 30 minutes	OBS-CUR-TR2
OBS-PRF-SR7	Scientific data coming from the two SPU shall not be mixed during the packetisation process	OBS-PUR-GE5
OBS-PRF-SR8	The OBS shall not occupy more space in PM than supported by EEPROM size	OBS-CUR-TR3

4.3 Interface Requirements

4.3.1 1553 Interface Requirements

Unless differently specified, requirements, figures and tables of this section come from Appendix 9 of RD2.

Req. ID	Description	Reference
OBS-IRF-SR1	The OBS shall act as a Remote Terminal (RT) implementing the functionalities, allowed by the MIL1553B STD, needed to be compliant with the Transfer Layer protocol described in RD2	
OBS-IRF-SR2	The RT shall support Mode Commands as shown in Table 3.2.4-1	
OBS-IRF-SR3	The RT shall support the subaddress (SA) allocation shown in Table 3.2.3-1	
OBS-IRF-SR4	The RT shall support broadcast messages	
OBS-IRF-SR5	The OBS shall adopt word formats as shown in Figure 3.1.5-1	
OBS-IRF-SR6	The RT shall adopt message formats as shown in Figure 3.1.3.2-1. The messages are always initiated by a specific command of the BC	
OBS-IRF-SR7	The RT shall recognize three types of messages: Receive, Transmit and Broadcast	

1553 Interface Requirements (ctd.)

Req. ID	Description	Reference
OBS-IRF-SR8	The RT shall support command words in accordance with RD2	
OBS-IRF-SR9	The T/R bit field set to 0 indicates that the RT shall receive data from the BC, set to 1 indicates that the RT shall transmit data to the BC	
OBS-IRF-SR10	The RT shall adopt the SA utilization Table 3.2.3-1	
OBS-IRF-SR11	The RT shall use SA 0R for mode command	
OBS-IRF-SR12	The RT shall support the mode commands listed in Table 3.2.4-1	
OBS-IRF-SR13	The RT shall use SA 1T to transmit instrument status	
OBS-IRF-SR14	The RT shall use SA 8R to receive spacecraft time	
OBS-IRF-SR15	The RT shall use SA 10T to inform spacecraft that a new telemetry packet is ready	
OBS-IRF-SR16	The RT shall use SA 11–26T to transfer telemetry packets from instrument to spacecraft	
OBS-IRF-SR17	The RT shall use SA 11–14R to transfer telecommand packets from spacecraft	
OBS-IRF-SR18	The RT shall use SA 27R to prepare instrument for telecommand transfer	
OBS-IRF-SR19	The RT shall place in SA 27T, after reading the telecommand packet, the confirmation message	
OBS-IRF-SR20	The RT shall use SA 30T (Data Wrap read) for test purposes	
OBS-IRF-SR21	The RT shall use SA 30R (Data Wrap write) for test purposes	
OBS-IRF-SR22	The RT shall read/write in the data word count field the quantity of data words to be transferred from/to the BC	
OBS-IRF-SR23	The RT shall support Mode Commands as shown in Table 3.2.4-1	
OBS-IRF-SR24	After the reception of any Mil Bus message (except a Mil Bus broadcast message), the RT shall respond with a status word with the content shown in Table 3.3.2-1	
OBS-IRF-SR25	The RT shall generate TM-packets with a maximum size of 1024 octets	
OBS-IRF-SR26	The RT shall accept TC-packets with a maximum size of 248 octets	
OBS-IRF-SR27	The RT shall support the exchange of variable length packets	
OBS-IRF-SR28	The RT shall support a cyclic satellite Data Bus Protocol based on a 1 second period called Frame, divided into 64 subframes, each containing a number of Mil Std 1553B messages	
OBS-IRF-SR29	Any Instrument TM packet shall fit into the Subframe boundaries	
OBS-IRF-SR30	The first Subframe every second starts with the Mode Command “Sync without Data word”. This Subframe is reserved and no instrument TM data transfer shall be allowed	
OBS-IRF-SR31	The RT shall receive the time information in Subframe 33	

1553 Interface Requirements (ctd.)

Req. ID	Description	Reference
OBS-IRF-SR32	Each subframe has a timing structure of 24 message slots (see Table 4.1.3.1-1). TM/TC packets shal be transferred during message slots 5 to 20	
OBS-IRF-SR33	The RT shall accept ground commands which will start predefined operational modes	
OBS-IRF-SR34	Each packet transfer shall be controlled by the exchange of a Packet Transfer Request/Descriptor and a Packet Transfer Confirmation, providin the necessary (handshake) information about the transfer	
OBS-IRF-SR35	The RT shall check the status of the packet transfer that has taken place in the previous Subframe, within the receiving of the next Subframe Sync Message, at the latest	
OBS-IRF-SR36	If a packet transfer has been performed then the RT shall update the TM packet data buffer within 2 msec, and shall update the TM packet Packet Transfer Request Words within 2 msec	
OBS-IRF-SR37	Only one TM packet transfer from each RT at a time is allowed. If there is more than one packet to be send the RT shall queue the TM packets	
OBS-IRF-SR38	TM packets shall be transferred within one Subframe	
OBS-IRF-SR39	The RT shall support Packet Transfer Requests via SA 10T and Packet Transfer Descriptors via SA 27R	
OBS-IRF-SR40	In case of TM packets the RT shall provide the following parameters with these words: – The number of needed messages – The number of words in the last message (equal to the word count pattern)	
OBS-IRF-SR41	For TC packets the above parameters are provided by the BC. The RT shall utilize these parameters to re-assemble the TC packets	
OBS-IRF-SR42	The RT shall read the RT address in the Subframe User field (see Figure 4.2-1)	
OBS-IRF-SR43	The RT, if not in Burst Mode, shall not interpret the Subframe User field as a command or enable signal	
OBS-IRF-SR44	The RT shall support an internal Subframe Counter and shall provide the value for BC access	
OBS-IRF-SR45	When receiving the first Subframe each second the Subframe Counter shall be set to 0 and the RT shall increment this value by one with every received Sync with Data Word command	
OBS-IRF-SR46	The RT shall receive the time information as a broadcast message sent to SA 8R, as shown in Figure 4.3-1	
OBS-IRF-SR47	The RT shall consider the value of the time information as the time at the beginning of the next frame. Reference is the beginning of the Mode Command Synchronize of Subframe 1	

1553 Interface Requirements (ctd.)

Req. ID	Description	Reference
OBS-IRF-SR48	The RT shall copy the Time Message to SA 8T immediately after receiving the Mode Command Synchronize at the beginning of a frame. At initialisation, before receiving any valid Time Distribution Message, the RT shall set the buffer at SA 8T to zero	
OBS-IRF-SR49	The RT status information shall be available via SA 1T in two data words using the layout shown in Figure 4.4-1:	
OBS-IRF-SR50	In the first data word, the reserved bits shall be set to zero; Subframe count shall contain a copy of the Subframe count value provided by BC	
OBS-IRF-SR51	In the second data word, the RT shall provide: BIT information; dynamic status	
OBS-IRF-SR52	The RT shall be able to receive Low Level Commands at SA 1R	
OBS-IRF-SR53	The TC packet shall be received by RT in the TC Data receive SAs, beginning with SA 11R	
OBS-IRF-SR54	The Packet Transfer Descriptor shall be received by RT in the SA 27R, according to the layout shown in Table 4.5.1-1	
OBS-IRF-SR55	The RT shall evaluate the TC Packet Transfer Descriptor after the reception of the next Subframe Sync, within one Subframe	
OBS-IRF-SR56	The RT shall store the new TC packet immediately and copy the associated words of the Packet Transfer Descriptor to SA 27T, to become the TC Packet Confirmation, according to the layout shown in Table 4.5.1-2	
OBS-IRF-SR57	To receive a TC packet the RT shall adopt the procedure shown in Figure 4.5.1-1	
OBS-IRF-SR58	RT shall request a TM packet transfer (RT to BC) by setting its TM Packet Transfer Request control words (SA 10T)	
OBS-IRF-SR59	The layout of the Packet Transfer Request shall be in accordance with Table 4.6.1.1-1 where: a) reserved bits shall be set to zero;	
OBS-IRF-SR60	b) No. of messages for next packet shall indicate the number of messages needed for the packet the RT is intending to send in the next Subframe. The first message of a TM packet shall always stored at SA 11T;	
OBS-IRF-SR61	c) No. of Data Words shall indicate the number of data words transmitted in the last message. In case of 32 words this field shall be set to 00000B;	
OBS-IRF-SR62	d) since data packets have always a size of n$\times$16 bit, with n an even number, no filling area shall be foreseen;	
OBS-IRF-SR63	e) Event fields A and B shall be set to 0 (no Event message pending);	
OBS-IRF-SR64	f) the Burst Mode field shall be set to 0 (Nominal Mode) or 1 (Burst Mode);	
OBS-IRF-SR65	g) Flow Control field shall be set by RT according to the status of TM transfer immediately. Its value shall be: 00B (No transfer pending), 01B (Transfer is pending);	

1553 Interface Requirements (ctd.)

Req. ID	Description	Reference
OBS-IRF-SR66	h) Packet Count field shall be used to support a RT-generated counter. To avoid that after an RT initialisation or reset an identical packet number is used, there shall be one number foreseen for that case. This number shall never appear in the cyclical transmission	
OBS-IRF-SR67	The RT shall support a circular Packet Counter in the range 1 to 255 decimal	
OBS-IRF-SR68	After initialisation or restart the RT shall set the counter value to 0 for the first TM Packet Transfer	
OBS-IRF-SR69	The RT shall not use this counter for any other purpose than defined in Chapter 4.6	
OBS-IRF-SR70	After requesting a TM packet transfer, the RT shall determine if the packet transfer was performed via the handshake signal (TM Packet Confirmation) sent by BC	
OBS-IRF-SR71	The RT shall receive the Packet Confirmation on SA 10R according to the layout shown in Figure 4.6.1.2-1	
OBS-IRF-SR72	The RT shall request a TM packet transfer (RT to BC) by setting its TM Packet transfer control words (SA 10T)	
OBS-IRF-SR73	For the exchange of TM packets in normal data bus mode, the RT shall support the logic shown in Figure 4.6.1.3-1	
OBS-IRF-SR74	After a TM packet transfer, in case there is no new TM packet pending the RT shall set the first word of the TM Packet Transfer Request to 0000 0000B, and the Packet Count value of the second word shall stay unchanged. The Flow Control field bits shall be set to 00B	
OBS-IRF-SR75	The RT shall support the Burst Data Bus Mode as described in Section 5.22.2.1.1. In particular the science TM-packets shall have their maximum size of 1024 octets	
OBS-IRF-SR76	After a TM packet transfer, in case there is no new TM packet pending in Burst Data Bus Mode the RT shall indicate End of Transfer as in Normal Data Bus Mode	

4.3.2 1355 Interface Requirements

Req. ID	Description	Reference
OBS-IRF-SR77	The OBS shall operate the HOCI data port as a 32 bits port	AD6
OBS-IRF-SR78	The OBS shall set the Transmit base register to 160 Mbps	AD6
OBS-IRF-SR79	The OBS shall set the Transmit bitrate to 10 Mbps for the three links	AD6
OBS-IRF-SR80	The OBS shall set the COMI data port of each link to transmit and to receive as a 32 bits port	AD6
OBS-IRF-SR81	The OBS shall transmit the EOP1 token at the end of each packet	AD6
OBS-IRF-SR82	The OBS shall operate the SMCS332 chip in transparent mode	AD6
OBS-IRF-SR83	The OBS shall be able to start each link as master or slave	AD6

4.4 Operational Requirements

Req. ID	Description	Reference
OBS-OPR-SR1	The OBS shall support all the operative modes of PACS. The transition from one to the other mode may be automatic or commanded by satellite/ground	OBS-CUR-ON1, OBS-CUR-OF1, OBS-PUR-GE1
OBS-OPR-SR2	The OBS shall support all the instrument observing modes of PACS. The transition from one to the other mode will be commanded by satellite/ground	OBS-PUR-GE2
OBS-OPR-SR3	The OBS shall be able to update an OBCP. This service shall be implemented with Load OBCP service	OBS-CUR-TC2
OBS-OPR-SR4	The OBS shall avoid that the execution of TCs affects any other independent on board process. An exception is the Abort an OBCP telecommand	OBS-CUR-TC6
OBS-OPR-SR5	The OBS shall avoid that commands for a certain subsystem are sent to another subsystem	OBS-CUR-TC11
OBS-OPR-SR6	The OBS shall support the possibility of updating all the parameters tables stored on board. This service shall be implemented as a memory load	OBS-CUR-TC15
OBS-OPR-SR7	The OBS shall be able to acquire science data generated by the subsystems	OBS-CUR-TM2, OBS-CUR-TM14
OBS-OPR-SR8	The OBS shall collect the instrument HK from all the subsystems (DPU included) during all nominal modes of the instrument	OBS-CUR-TM3
OBS-OPR-SR9	The OBS shall be able to provide in its HK packet a status word indicating whether new HK values have been received from the subsystems during the last acquisition period (2 seconds). This capability shall be implemented by setting the function status word to the value `NO NEW HK` if new HK values have not been received	OBS-CUR-TM7
OBS-OPR-SR10	The OBS shall mantain a BBID (Building Block Identifier) and OBSID (Observation Identifier) fields as part of its HK. They are set by ground via two dedicated commands for DEC. DPU shall copy in its HK packet the last values received by DEC. The startup values shall be 0	OBS-CUR-TM16
OBS-OPR-SR11	The OBSID and BBID values received from DEC as part of its HK, shall be written in all HK packets starting from byte 18 (OBSID) and byte 22 (BBID) (start of TM packet is byte 0)	AD5
OBS-OPR-SR12	The OBSID and BBID values received from DEC as part of its HK, shall be written in all event packets starting from byte 20 (OBSID) and byte 24 (BBID) (start of TM packet is byte 0)	AD5

Operational Requirements (ctd.)

Req. ID	Description	Reference
OBS-OPR-SR13	Since the generation of TM packets inside the DPU and the retrieving of the packets by the CDMS are not synchronous, the OBS shall provide three buffers to store TM packets until they are transmitted to CDMS. To this aim, three buffers are foreseen: one for event reports, one for HK packets and one for all the other kind of packets. The sizes of the buffers are, respectively: 32, 64, 400 packets. In case overflow occurres for the generic packets buffer, the CDMS shall be informed with an event report	OBS-CUR-TM17
OBS-OPR-SR14	The OBS shall set the most significant bit of the time field in all TM packets to 1 until it synchronizes with CDMS. The rest of the field measures the time elapsed since DPU was switched on, in the format specified in RD2	OBS-CUR-SY2
OBS-OPR-SR15	When entering Test Mode, the OBS shall not disable autonomy functions	OBS-CUR-SM1
OBS-OPR-SR16	The OBS shall be able to perform regular HW self checks. The following HW parameters shall be sampled every two seconds and reported in the HK packets: +2.5V, +5V, +15V, –15V, Temperature	OBS-CUR-SM2
OBS-OPR-SR17	The OBS shall be stored in EEPROM, mapped in data memory. It is copied into program memory at startup by the DPU boot software	OBS-CUR-SM4
OBS-OPR-SR18	The OBS shall be able, with a dedicated OBCP, to copy the full image in EEPROM. The following program memory areas shall be copied: the interrupt table (segment `seg_rth`); the init segment `seg_init` and the program area `seg_pmco`	OBS-CUR-SM4
OBS-OPR-SR19	The generation of the OBS image depends on a file (architecture file) that organizes the 21020 memory in different areas. The OBS program code shall resides in program memory in a segment named `seg_pmco`; fixed data shall be stored in program memory in a segment named `seg_init` and copied at startup in data memory in a segment named `seg_dmda` (this copy involves a 21020 standard library function); variables and parameters shall resides in data memory in a segment named `heap1`	OBS-CUR-SM5
OBS-OPR-SR20	The OBS shall support a command to perform a warm reset of the DPU. The execution of this command shall fully (HW) reset the 1553 and 1355 boards and shall end with a jump to location `0x8` of program memory	OBS-CUR-AF11
OBS-OPR-SR21	The compression parameters specific to each observing modes shall be provided to DEC on reception of a specific command	OBS-PUR-GE4

4.5 Safety Requirements

Req. ID	Description	Reference
OBS-SEC-SR1	The OBS shall implement autonomy functions that will be called on detection of DPU or other instrument subsystems anomalies without ground intervention. The list of these functions will be reported in RD3	OBS-CUR-TM4, OBS-CUR-AF1
OBS-SEC-SR2	The OBS shall report in event packets failures and/or anomalies detected on-board. The list of the events as well as the level of criticality will be reported in RD3	OBS-CUR-AF2
OBS-SEC-SR3	In case of activation of an autonomy function the OBS shall report in event packet: 1) the ID of the started function; 2) the time of occurrence; 3) parameters passed to the function	OBS-CUR-AF3, OBS-CUR-AF4
OBS-SEC-SR4	The OBS shall wait a minimum period (specified in AD2 and currently set to 2 sec, the HK acquisition rate) before the next event packet reporting the same event can be issued	OBS-CUR-AF6
OBS-SEC-SR5	The OBS shall implement a command to enable/disable each individual autonomy function	OBS-CUR-AF7
OBS-SEC-SR6	The HW HK values for all the subsystems shall be available in all HK packets	OBS-CUR-AF10
OBS-SEC-SR7	The OBS shall implement a TC to verify the integrity of the code stored in PM. In case of failure an event (5,2) shall be generated	OBS-CUR-SM2
OBS-SEC-SR8	The OBS shall implement an autonomous and periodic check of HW integrity of DM. In case of failure an event (5,4) shall be generated	OBS-CUR-SM2

4.6 Quality Requirements

Req. ID	Description	Reference
OBS-QTY-SR1	For the OBS development the standard given in AD1 shall be adopted, tailored to be fully equivalent to RD1	AD3
OBS-QTY-SR2	The OBS shall be coded according to the rules given in AD3	AD3

5 Feasibility and Resources Estimates

This section addresses the problem of estimating the required resources to run the OBS.

For the DPU of PACS there are no special requirements on timing: the most demanding action to perform is the reaction to a 1553 interrupt. The shortest slot of a subframe lasts 150μsec (see RD2). We estimate that the OBS starts reading the first 1553 message in not more than 50μsec or 1/3 of the whole slot duration.

There are 512 Kwords of program memory. 256 are reserved for the interrupt table and then there is the space needed for the boot software. The area that the application software can use starts at 0x4000 so that there are 507904 usable words. Our present estimate is that the code will require less than 100000 words.

Regarding the EEPROM, its dimension is 256 Kwords of data memory, written in pages of 1024 words, for a total of 256 pages. Each page consists of an header followed by 678 program memory words. One page is required for the interrupt table and 7 pages for the initialization segment: there are 248 pages left which can store 168144 words of program memory.

Also data memory has 512 Kwords. 65536 words are reserved for VIRTUOSO idle task, so there are 448 Kwords available. We have reserved other 262144 (half DM) words for global and static variables (if this space is not enough, the compiler issues an error message and the executable is not produced), so there are 192 Kwords left. Each task works in itw own memory space (stack), the following sizes have been assigned

Task	Stack size
Init	5000
Controller	10000
Procedures Handler	10000
1553 Interface	10000
HK Monitoring	10000
DEC Interface	5000
SPS Interface	5000
SPL Interface	5000
1355 Interface	10000

for a total of 70000 words. There are still 126608 words. The space required by VIRTUOSO services is (see the User Manual): 4784 for the FIFO; 1000 for command packets; 120 for timers, for a total of 5904 words are used.

6 Software Requirements vs Components Traceability matrix

In the following matrix the list of User Requirements is reported. For each requirement the corresponding SW Requirement(s) is(are) indicated along with the testing procedure (T means that the requirement is to be tested with a specific procedure, D means that the requirement is tested with design review). The last column shows the architecture module where the SW requirement is mapped: *italic font* for tasks names, `this font` for single modules or files.

User Req. ID	SW Req. ID	Testing	Module
OBS-CUR-ON1	OBS-OPR-SR1	D	*T4CNTRLR*
OBS-CUR-OF1	OBS-OPR-SR1	D	*T4CNTRLR*
OBS-CUR-TC1	OBS-FSR-FM1	T	*T4CNTRLR*
	OBS-FSR-FM4	T	*T4CNTRLR*
	OBS-FSR-CP3	T	*T4CNTRLR*
OBS-CUR-TC2	OBS-FSR-MM2	T	*T4CNTRLR*
	OBS-FSR-CP2	T	*T4CNTRLR*
OBS-CUR-TC3	OBS-FSR-TC1	D	*T2TMTCIF, T4CNTRLR*
	OBS-FSR-TC2	D	*T4CNTRLR*
	OBS-FSR-TC3	T	*T4CNTRLR*
	OBS-FSR-TC8	T	*T4CNTRLR*
	OBS-FSR-FM5	T	*T4CNTRLR*
OBS-CUR-TC4	OBS-PRF-SR1	T	*T4CNTRLR*
OBS-CUR-TC5	OBS-FSR-TC7	T	*T4CNTRLR*
	OBS-FSR-TC14	T	*T4CNTRLR*
OBS-CUR-TC6	OBS-FSR-CP6	T	*T4CNTRLR*
	OBS-OPR-SR4	D	*T4CNTRLR*
OBS-CUR-TC7	OBS-FSR-TC9	T	*T4CNTRLR*
	OBS-FSR-TC10	T	*T4CNTRLR*
	OBS-FSR-TC11	T	*T4CNTRLR*
		(test to be performed in a engineering session with emulator)	
	OBS-FSR-TC12	T	*T4CNTRLR*
	OBS-FSR-TC13	T	*T4CNTRLR*

User Req. ID	SW Req. ID	Testing	Module
OBS-CUR-TC8	OBS-PRF-SR2	D	*T4CNTRLR*
OBS-CUR-TC9	OBS-FSR-TC7	T	*T4CNTRLR*
	OBS-FSR-TC9	T	*T4CNTRLR*
	OBS-FSR-TC10	T	*T4CNTRLR*
	OBS-FSR-TC11	T	*T4CNTRLR*
		(test to be performed in a engineering session with emulator)	
	OBS-FSR-TC12	T	*T4CNTRLR*
	OBS-FSR-TC13	T	*T4CNTRLR*
OBS-CUR-TC10	OBS-PRF-SR3	D	*T4CNTRLR*
OBS-CUR-TC11	OBS-FSR-FM4	T	*T4CNTRLR*
	OBS-FSR-FM6	T	*T4CNTRLR*
	OBS-FSR-FM7	T	*T4CNTRLR*
	OBS-OPR-SR5	T	*T4CNTRLR*
OBS-CUR-TC12	OBS-FSR-TC15	T	*T9_OBCP*
	OBS-FSR-TC16	T	*T4CNTRLR, T9_OBCP*
	OBS-FSR-CP5	T	*T9_OBCP*
OBS-CUR-TC13	OBS-FSR-TC17	T	*T4CNTRLR, T9_OBCP*
OBS-CUR-TC14	OBS-FSR-CP6	T	*T4CNTRLR*
OBS-CUR-TC15	OBS-FSR-MM2	T	*T4CNTRLR*
	OBS-FSR-CP9	T	*T4CNTRLR*
	OBS-OPR-SR6	T	*T4CNTRLR*
OBS-CUR-TC16	OBS-FSR-PT1	T	*T4CNTRLR*
	OBS-FSR-PT2	T	*T4CNTRLR*
	OBS-FSR-PT3	T	*T4CNTRLR*
OBS-CUR-TM1	OBS-FSR-TC4	T	`update_TM_buffer`
		(tested via inspection of the telemetry)	
	OBS-FSR-TC5	D	`update_TM_buffer`
	OBS-FSR-TC6	T	`update_TM_buffer`
	OBS-FSR-MC1	T	all the tasks
		(tested via inspection of raw telemetry)	
	OBS-FSR-MC2	D	all the tasks
	OBS-FSR-MC3	D	all the tasks
	OBS-FSR-MC4	D	*T4CNTRLR*
	OBS-FSR-MC7	T	*T2TMTCIF, T4CNTRLR, T9_OBCP*
OBS-CUR-TM2	OBS-OPR-SR7	T	*T3_IRQ1SV*
OBS-CUR-TM3	OBS-FSR-HK4	T	*T5_HKMON*
	OBS-OPR-SR8	T	*T3_IRQ1SV*
OBS-CUR-TM4	OBS-SEC-SR1	D	*T5_HKMON*
OBS-CUR-TM5	OBS-FSR-HK1	T	*T5_HKMON*
OBS-CUR-TM6	OBS-FSR-MC6	D	all the tasks
OBS-CUR-TM7	OBS-OPR-SR9	T	*T5_HKMON*
OBS-CUR-TM8	OBS-PRF-SR4	T	*T5_HKMON*
OBS-CUR-TM9	OBS-FSR-FM4	T	*T4CNTRLR, T6_MECRX*
OBS-CUR-TM10	OBS-FSR-CP10	T	*T4CNTRLR*
	OBS-FSR-CP12	T	*T4CNTRLR*
OBS-CUR-TM11	OBS-FSR-HK6	D	*T5_HKMON*
OBS-CUR-TM12	OBS-FSR-HK2	T	*T5_HKMON*
	OBS-FSR-HK3	T	*T5_HKMON*
	OBS-FSR-HK5	T	*T4CNTRLR*
OBS-CUR-TM13	OBS-FSR-FM4	T	*T4CNTRLR*

User Req. ID	SW Req. ID	Testing	Module
OBS-CUR-TM14	OBS-OPR-SR7	T	*T3_IRQ1SV*
OBS-CUR-TM15	OBS-FSR-TC4	T	`update_TM_buffer`
		(tested via inspection of the telemetry)	
	OBS-FSR-TM2	D	`update_TM_buffer`
OBS-CUR-TM16	OBS-FSR-EV1	T	`event_packet`
		(tested via inspection of raw telemetry)	
	OBS-OPR-SR10	T	*T5_HKMON*
OBS-CUR-TM17	OBS-OPR-SR13	T,D	`update_TM_buffer`
OBS-CUR-SY1	OBS-FSR-TM1	D	*T2TMTCIF*
	OBS-FSR-TM3	T	*T4CNTRLR*
OBS-CUR-SY2	OBS-OPR-SR14	T	*T1_INIT*
		(tested via inspection of raw telemetry)	
OBS-CUR-SY3	OBS-FSR-TM4	T	*T4CNTRLR*
OBS-CUR-SM1	OBS-OPR-SR15	D	*T4CNTRLR*
OBS-CUR-SM2	OBS-OPR-SR16	T	*T5_HKMON*
	OBS-SEC-SR7	T	*T4CNTRLR*
	OBS-SEC-SR8	T	ISR_IRQ3
OBS-CUR-SM3	OBS-FSR-MM7	T	*T4CNTRLR*
OBS-CUR-SM4	OBS-OPR-SR18	D	*T9_OBCP*
OBS-CUR-SM5	OBS-OPR-SR19	D	`Makefile, pacs.ach`
OBS-CUR-SM6	OBS-FSR-MM2	T	*T4CNTRLR*
	OBS-FSR-MM3	T	*T4CNTRLR*
	OBS-FSR-MM4	T	*T4CNTRLR*
	OBS-FSR-MM5	T	*T4CNTRLR*
	OBS-FSR-MM6	T	*T4CNTRLR*
	OBS-FSR-MM7	T	*T4CNTRLR*
	OBS-FSR-MM8	T	*T4CNTRLR*
	OBS-FSR-MM9	T	*T4CNTRLR*
OBS-CUR-SM7	OBS-FSR-MM2	T	*T4CNTRLR*
OBS-CUR-SM8	OBS-FSR-TS1	T	*T4CNTRLR*
	OBS-FSR-TS2	T	*T4CNTRLR*
OBS-CUR-AF1	OBS-SEC-SR1	D	*T5_HKMON*
OBS-CUR-AF2	OBS-SEC-SR2	T	`event_packet`
		(tested via inspection of raw telemetry)	
OBS-CUR-AF3	OBS-SEC-SR3	T	*T5_HKMON*
		(tested via inspection of raw telemetry)	
OBS-CUR-AF4	OBS-SEC-SR3	T	*T5_HKMON*
		(tested via inspection of raw telemetry)	
OBS-CUR-AF5	OBS-FSR-EV2	D	`event_packet`
	OBS-FSR-EV3	D	`event_packet`
	OBS-FSR-EV4	D	`event_packet`
OBS-CUR-AF6	OBS-SEC-SR4	D	*T5_HKMON*
OBS-CUR-AF7	OBS-SEC-SR5	T	*T4CNTRLR*
OBS-CUR-AF8	Not to be implemented for OBSW Version 1		
OBS-CUR-AF9	OBS-FSR-CP4	T	*T9_OBCP*
	OBS-FSR-CP5	T	*T9_OBCP*
OBS-CUR-AF10	OBS-SEC-SR6	D	*T5_HKMON*
OBS-CUR-AF11	OBS-OPR-SR20	T	*T4CNTRLR*
OBS-CUR-TR1	OBS-PRF-SR5	T	*T2TMTCIF*
OBS-CUR-TR2	OBS-PRF-SR6	T	*T2TMTCIF*

User Req. ID	SW Req. ID	Testing	Module
OBS-CUR-TR3	OBS-PRF-SR8	D	*T2TMTCIF*
OBS-PUR-GE1	OBS-OPR-SR1	D	*T4CNTRLR*
OBS-PUR-GE2	OBS-OPR-SR2	D	*T4CNTRLR*
OBS-PUR-GE3	OBS-FSR-MM10	T	*T4CNTRLR*
	OBS-FSR-MM11	T	*T6_MECRX*, *T7_SPSRX*, *T8_SPLRX*
	OBS-FSR-MM12	T	*T6_MECRX*, *T7_SPSRX*, *T8_SPLRX*
OBS-PUR-GE4	OBS-OPR-SR21	D	*T4CNTRLR*
OBS-PUR-GE5	OBS-PRF-SR7	D	`update_TM_buffer`

A Data dictionary

1553_Messages

Data flow from the CDMS. Messages here not only means TC and TM encoded words, but also control data.

ACK_Data_Flow

This data flow connects in both directions the 1355 receiving tasks with the 1355 TX module (note that in Figure 2-2 ACK_Data_Flow connects the T3_IRQ1SV with the 1355 TX module just to simplify the figure). The receivers use the flow to be informed if the DPU was waiting for an ACK; 1355 TX receives via this flow the ACK packet content to check if the command has been executed or not.

ACK_Signal

Used to inform the 1355 TX module that T3_IRQ1SV has received an ACK packet from a subsystem. The received data are not interpreted at all, this flow is only for synchronization.

Acceptance_Result

The result of the telecommand acceptance test is sent from the TC_Acceptance module to the Controller through this data flow. Only if the result is OK the TC is executed.

Call_Event_Handler

An event handler is a C function called by VIRTUOSO to decide if the task waiting on that event should be made runnable. In the OBS this feature is implemented for the 1553 interrupt handling.

Crx 1355

Data flow controlling the reception of packets from the subsystems.

Ctx 1355

Data flow controlling the transmission of packets to the subsystems.

Dec_values

HK received from DEC and sent from T3_IRQ1SV to T5_HKMON.

INT 1355

This control flow is activated when an interrupt is generated by one of the three serial links of the 1355 interface. The kind of interrupt is written in a specific register of the SMCS332 chip.

INT 1553

This control flow transports the interrupt generated by the 1553 interface when a request from the Bus Controller is received. This happens regularly at a rate of 64 interrupts per second. Since not all the interrupts require actions from PACS, the function Handler_1553 decides if T2TMTCIF should be made scheduled or not.

INT_DEC

Virtuoso control signal (EVENT) used to wake up T6_MECRX.

INT_SPL

Virtuoso control signal (EVENT) used to wake up T8_SPLRX.

INT_SPS

Virtuoso control signal (EVENT) used to wake up T7_SPSRX.

ISR_1553_EVENT

Virtuoso control signal (EVENT) used to wake up T2TMTCIF.

MEC_Command

Data flow used by the DPU to send a command to the DEC/MEC, through the 1355 interface link.

OBCP_Data

Parameters sent by ground to start the execution of an OBCP. If one or more parameters are not received, DPU uses previous values. At startup, all the parameters are set to 0.

OBCP_data_current

Parameters sent to the OBCP that is going to be started. These new values overwrite previous ones, otherwise old values (default at startup are zero) are kept.

p_DEC_1355

Pointer to a structure containing informations on the DEC link status.

p_SPL_1355

Pointer to a structure containing informations on the long wavelength SPU link status.

p_SPS_1355

Pointer to a structure containing informations on the short wavelength SPU link status.

Pool_EV_packets

Memory area (buffer) that contains event packets before the transmission to CDMS via 1553 interface. It contains up to 32 packets, in case of overflow the newer packets are lost and a specific counter (reported in the HK packets) is incremented.

Pool_HK_packets

Memory area (buffer) that contains HK packets before the transmission to CDMS via 1553 interface. It contains up to 64 packets, in case of overflow the newer packets are lost and a specific counter (reported in the HK packets) is incremented.

Pool_SC_packets

Memory area (buffer) that contains all packets but events and HK, before the transmission to CDMS via 1553 interface. It contains up to 400 packets, in case of overflow the newer packets are lost and a counter (reported in the HK packets) is incremented. An event is also generated.

RX 1355

Data written in the 1355 Dual Port RAM by the SMCS332 chip. T3_IRQ1SV reads the first word (header) to recognize the received packet: nominal or diagnostic HK data; science data; acknowledgment.

RX 1553

1553 raw data that T2TMTCIF uses in input to reconstruct TC packets.

SEMA_1355_INT

Virtuoso control signal (SEMAPHORE) used to start the execution of T3_IRQ1SV.

SPL_Command

Data flow used by the DPU to send a command to the long wavelength SPU, through the 1355 interface link.

SPS_Command

Data flow used by the DPU to send a command to the short wavelength SPU, through the 1355 interface link.

STARTPROC

Virtuoso control signal used (EVENT) to start the execution of T9_OBCP.

Spl_values

HK received from the long wavelength SPU and sent from T3_IRQ1SV to T5_HKMON.

Sps_values

HK received from the short wavelength SPU and sent from T3_IRQ1SV to T5_HKMON.

TC_Packets

The packet containing the TC received from the CDMS.

TC_QUEUE

Virtuoso control flow (FIFO) associated to the TC FIFO to start the execution of T4CNTRLR.

TM_Struct

This data flow represents the TM packets generated by the DPU OBS, ready to be sent to the CDMS.

TX 1355

Data (commands) written in the 1355 Dual Port RAM by the DPU before transimssion to the subsystem.

TX 1553

1553 raw data that T2TMTCIF uses in output to send TM packets to the CDMS.