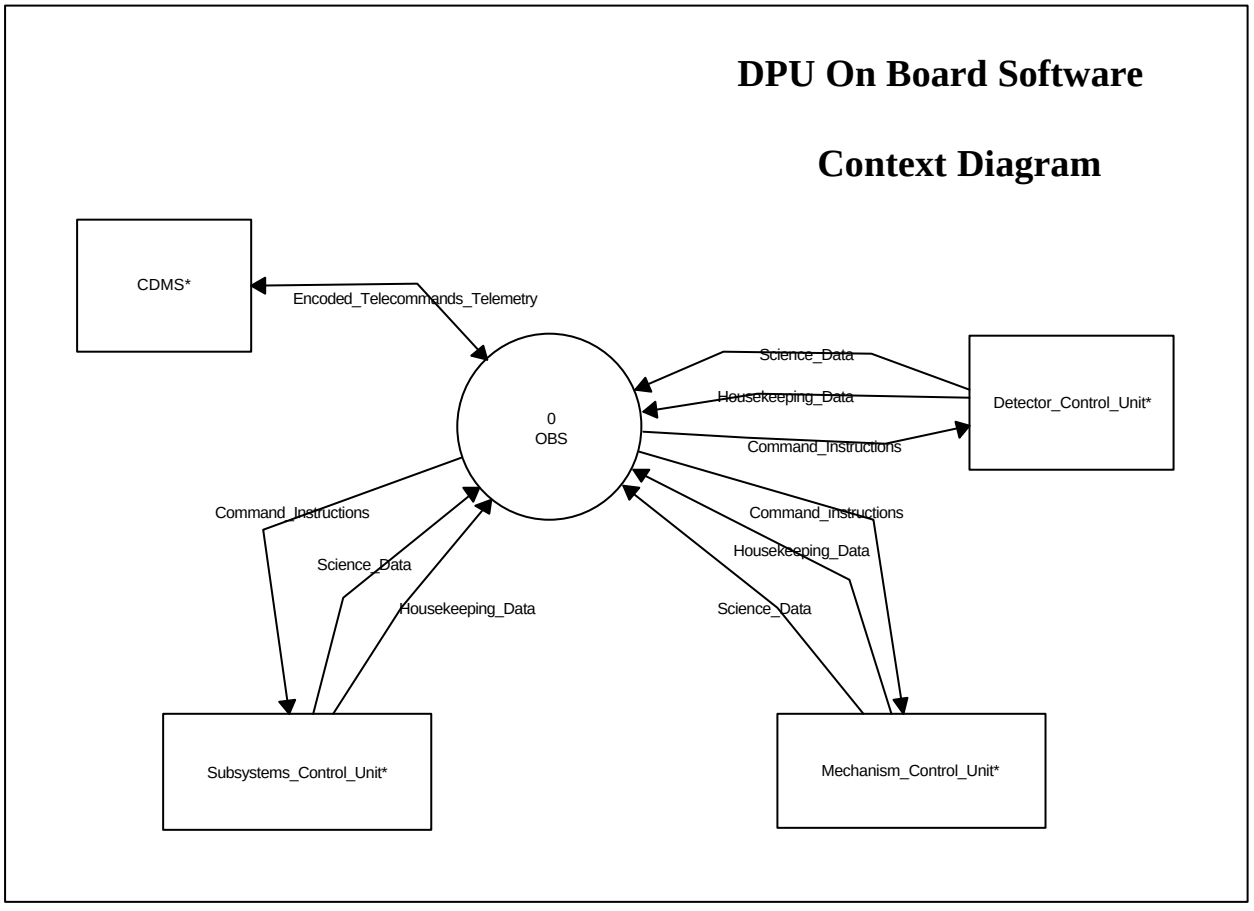
This note is intended to present the FIRST DPU/ICU On Board Software logical decomposition obtained by the IFSI software development team on the basis of the FIRST DPU/ICU OBS User Requirement document, draft 0, circulated the past November.
The obtained logical model is an implementation independent representation of what is needed by the user. Not necessarily there will be a one to one mapping of the highlighted main logical components on the final software architecture tasks.
This model will be included in a section of the OBS Software Specification Document, together with the Architectural design presentation.
Only one logical model has been provided for all the three instruments, mapping the common subset of user requirements. A tentative generalisation of the subsystems interfaces has been suggested.
No detailed decomposition of the subsystems communication component has been provided, these will be discussed with the three teams separately.
All comments/inputs are welcome.

1. FIRST DPU/ICU OBS Context Diagram



The interfaces of the DPU/ICU On Board Software are reported in the context diagram.
The interface with the spacecraft OBDH is represented by the Encoded_Telecommands_Telemetry data flow. The Telecommands packets and Telemetry packets will be in fact encoded according to the interface bus control protocol that will be adopted by ESA.

The interfaces with the subsystems can be generically divided into two types of interfaces: the first one linking the DPU/ICU to the subsystems in charge of controlling the mechanical/optical devices onboard the instruments and the second one linking the subsystems containing the measurement devices (detectors etc.).
In the first case there is only one input data flow, for the housekeeping data, while for the detectors subsystems a Science_data input data flow is foreseen in addition to the housekeeping data flow.
In both cases there is one output data flow only, containing the Command_Instructions for the subsystems.

2. Logical component OBS

The DPU/ICU is the only data interface of the FIRST instruments with the spacecraft OBDH, so the main capabilities of the OBS will be related with TCMs and TM handling. In particular, TCMs packets will be received (OBDH_Messages_Interface component), interpreted according to their priority, translated into instrument instructions (Command_Execution and IPU_Commands_Handler components for subsystems commands and DPU/ICU commands handling respectively) and sent to the specific subsystem (Subsyste_Communication component, for the different subsystems interfaces handling). The DPU/ICU will then receive (or will request in the case of HIFI) the science data and all the HK data, which are used for monitoring the instrument behaviour (Subsystem_Controller component).
A check will be performed on some of the parameters and, if the corresponding critical
values will be reached, the OBS will start the pre-defined autonomous functions in order to prevent any damage to the instrument. Depending on the severity of the detected
anomaly, the measurement could even be reset and/or DPU/ICU could ask OBDH to switch off some subsystems.
Furthermore, the OBS has to manage the uploading and downloading of (part of) the processor memory: this will allow to upgrade the OBS as well as all the subsystems parameters tables (IPU_Commands_Handler component).
Some subsystems may run their own programs which will likely need upgrades: OBS will be in charge of receiving these patches and passing them to the appropriate subsystem, following the same item necessary for the other subsystems instructions.

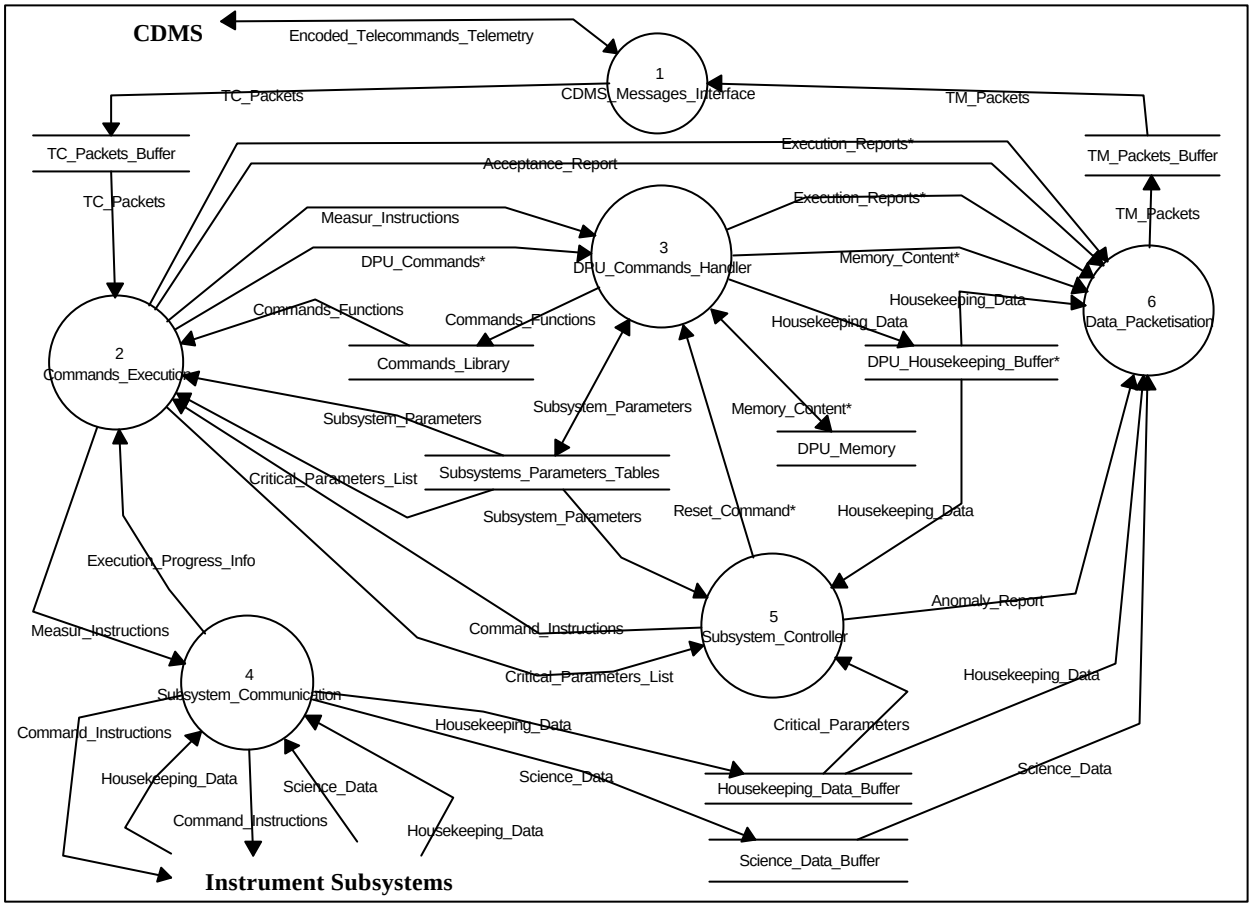


Figure 2 Flow Diagram OBS

2.1. Logical component CDMS_Messages_Interface

This logical component includes all the functionalities necessary to interface the DPU/ICU OBS with the OBDH. The messages are exchanged according to the protocol supported by the electrical interface with the spacecraft (MIL-STD-1553B): the telecommand packets and the telemetry packets will be encoded in 20 bit words and the overall data flow on the interface bus will be controlled by the Bus Controller (OBDH). Therefore, two buffers are necessary to maintain the OBS actions independent from the external requests: an input buffer for the decoded telecommand packets (TCM_Packets_Buffer) and an output buffer (TM_Packets_Buffer) for all the telemetry packets provided by the instrument, to be delivered on request to the OBDH.

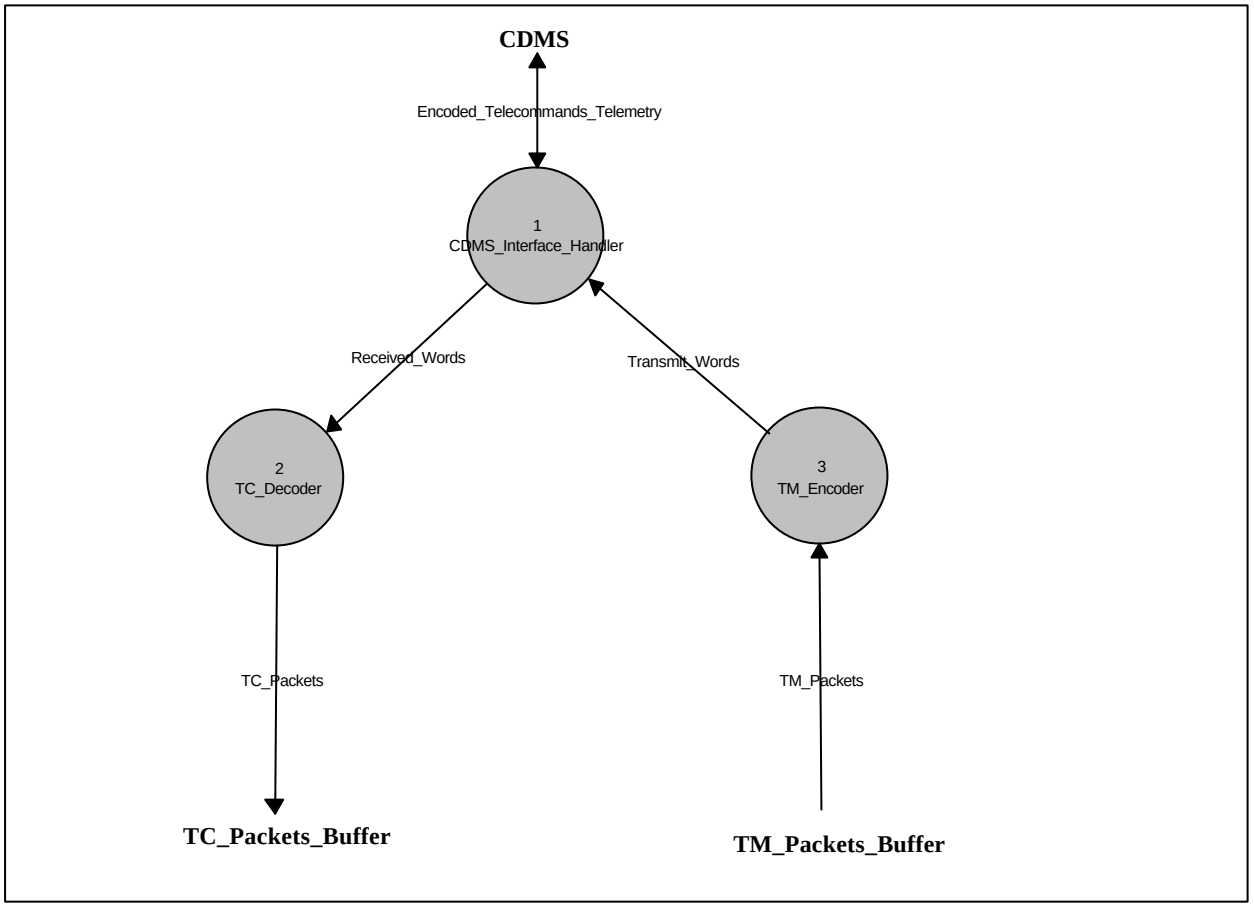


Figure 3 Flow Diagram CDMS_Messages_Interface

2.1.1. Logical component CDMS_Interface_Handler

This component is necessary to couple the DPU OBS with the CDMS interface bus protocol. Presently, the specifications of what will be the Transfer Layer Protocol (the way the exchange of large blocks of data, i.e. TC/TM packets, will be implemented) adopted by ESA for this interface are not yet defined and therefore the only very general functionality that can be assigned to this logical component is the identification of the nature of the CDMS requested packet (transmit or receive) and the decoding of the remote terminal address bits. Moreover, this component will be the generator of all the status messages foreseen by the protocol.

This process is missing its child diagram.

2.1.2. Logical component TC_Decoder

The messages received from the CDMS are encoded into 20 bit words (according to the MIL-STD-1553B standard). The TC_Decoder component extracts the relevant bits from all the received words and reconstructs the telecommands (TC), packetised according to the ESA Packet Telemetry Standard. All the received TC packets are stored sequentially into the OBS input data buffer, the TC_Packets_Buffer.

This process is missing its child diagram.

2.1.3. Logical component TM_Encoder

All the telemetry packets provided by the system shall be encoded into 20 bit words (according to the MIL-STD-1553B standard). This component prepares the encoded messages to be transmitted on request to the OBDH.

This process is missing its child diagram.

2.2. Logical component Commands_Execution

The Commands_Execution logical component is the main component of the OBS. It is in charge of the telecommands handling and execution. It includes the following functionalities:
- the TCM_Packets are read from the TCM_Packets_Buffer and all the validity checks are performed. The results of these checks are contained in the Acceptance Report data flow and will be packtised (by the Data_Packetisation component) in an output report telemetry packet. The non valid packets are rejected.
- all the telecommands are executed in the order they are received: there can be exceptions in the case of "immediate" commands, as , for example, the Abort_Command. In the case it is received, it is put in an internal immediate commands buffer in order to be read and executed at the end of the instruction presently under execution;
- the packet telemetry services requested are identified in order to execute the telecommand;
- the function commands are executed according to the corresponding on board stored routines. The measurement instructions generated by the function execution are transferred to the other instrument subsystems. When necessary, before sending an instruction, a feedback of the completion of the previous instruction is waited for;
- the feedback from the subsystems can be used to verify the level of execution of the telecommand (execution progress monitoring, TBC). At the completion of the command execution, a dedicated report is prepared, in order to be packetised and sent to OBDH together with the results of the measurement;
- the request of the On Board Operations Procedures service is handled in this component by stepping through the procedure code (either uploaded via telecommand or already stored in the on board libraries), interpreting the simple logic possibly contained in it and executing the functions/instructions called by it;
- the commands invoking the DPU/ICU on board memory handling are routed to the IPU_Commands_Handler component in order to be executed. In addition to the load and dump of the software executables, these commands will include the upgrades of all the tables stored on board;
- the time synchronisation of the subsystems is performed, being the time signal considered as part of the Measur_Instructions data flow.

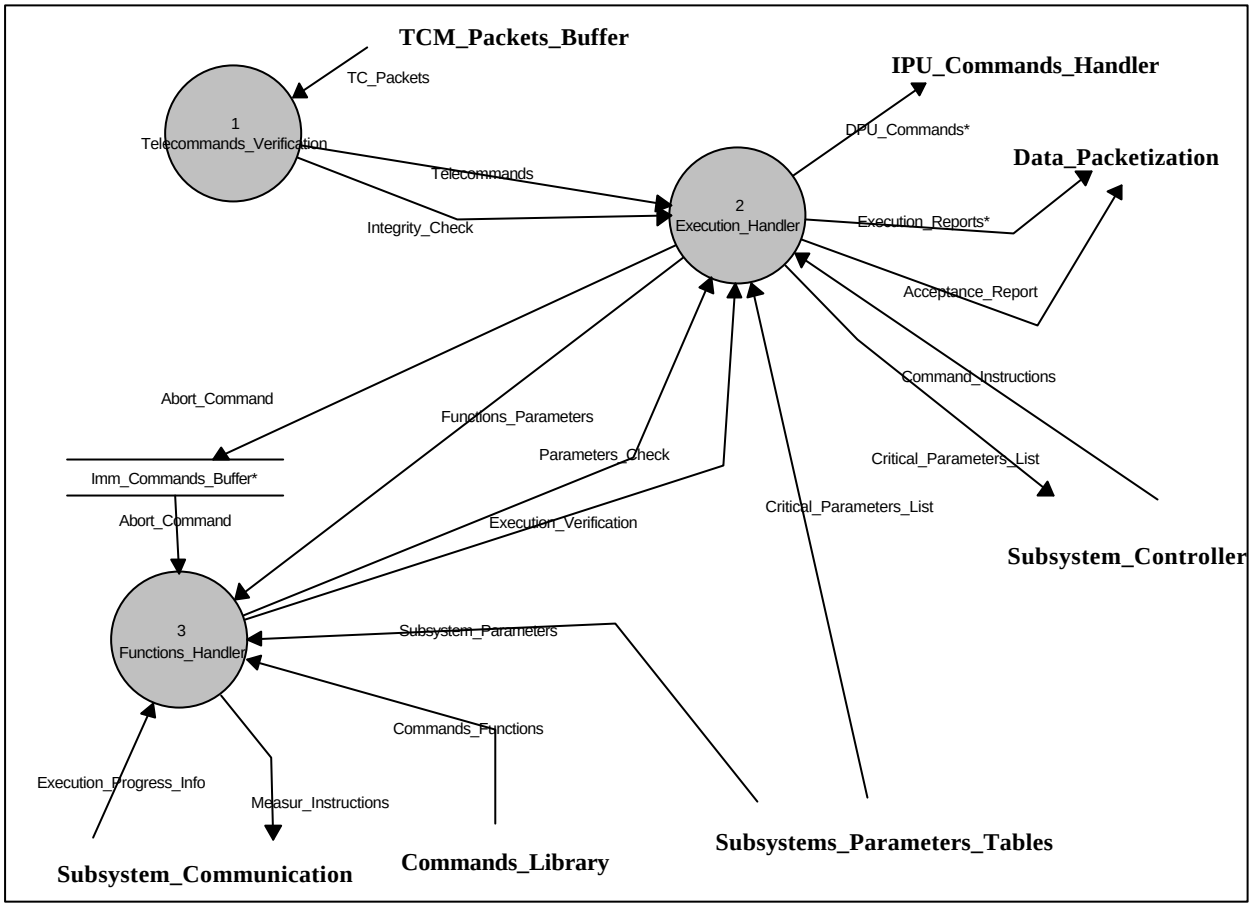


Figure 4 Flow Diagram Commands_Execution

2.2.1. Logical component Telecommands_Verification

The Telecommands_Verification component performs all the activities necessary to verify that the received packets have not been corrupted. In particular, it reads the telecommands packets stored in the TCM_Packets_Buffer, checks their validity (APID, length of packet, checksum, packet type and packet subtype) and provides TCM_Decoder with the Integrity_Check results. The valid TCM_Packets are unpacked and the Telecommands are extracted in order to be processed by the Execution_Handler. The non valid packets are rejected.

This process is missing its child diagram.

2.2.2. Logical component Execution_Handler

The Execution_Handler component is the one logically dedicated to the high level interpretation of the telecommands and to their routing to the other components involved in the command execution. Therefore its main functions are:
- identification of the immediate commands, e.g. abort command;
- identification of the requested services;
- extraction of the relevant info contained in the telecommands and preparation of the data for the command execution dedicated components;
- preparation of the acceptance report, putting together the integrity checks results and the executability checks results (Parameters_Check data flow);

- provision of the necessary info to the Functions_Handler component for the subsystems commands execution and the generation of the related measurement instructions;
- preparation of the Execution_Report, based on the command execution verification info provided by the Functions_Handler;
-management of the execution of the On Board Operation Procedures.
In addition, depending on the instruction performed, the corresponding Critical_Parameters_List of housekeeping parameters to be monitored is passed to the Subsystems_Controller.

This process is missing its child diagram.

2.2.3. Logical component Functions_Handler

The Functions_Handler verifies that the parameters associated with the received telecommand are in the nominal range and provides the Execution_Handler with the Parameters_Check results.
The software routines associated with the requested functions (Commands_Library) are activated and the Measur_Instructions (internal instrument actions) are generated and passed to the Subsytem_Communication component, in order to be transmitted to the other instrument subsystems. The instructions for the DPU/ICU subsystem are passed to the IPU_Commands_Handler. When necessary, the feedback from the subsystems (Execution_Progress_Info) is checked before sending the following instructions. This feedback is used to prepare an Execution_Verification message to be passed to the Execution_Handler component.
The presence of an immediate command (i.e. abort) in the Imm_Command_Buffer is checked at the completion of the execution of each measurement instruction (TBC), and the corresponding command is immediately executed.

This process is missing its child diagram.

2.3. Logical component DPU_Commands_Handler

The IPU_Commands_Handler logical component is dedicated to execute all the commands for the DPU/ICU instrument subsystem: the IPU_Commands (all the commands related to the memory upload/download activities, such as upgrading of tables on board, software patching, memory dumping etc.) and the Measur_Instructions are executed by reading, if necessary, the Subsystems_Parameters_Tables. The relative Execution_Reports are generated on request.
In case of a dump command the IPU_Commands_Handler reads from the IPU_Memory the Memory_Content and sends it to Data_Packetization.
The Housekeeping_Data generated from the DPU/ICU are stored in the IPU_Housekeeping_Buffer, in order to be monitored by the Subsystem_Controller.
This component is in charge of handling also the possible (warm) Reset _Command, which can be generated either by the Subsystem_Controller component, as a result of the instrument monitoring activities, or from the OBDH (in this case it can be considered as part of the IPU_Commands data flow, mainly for testing activities).

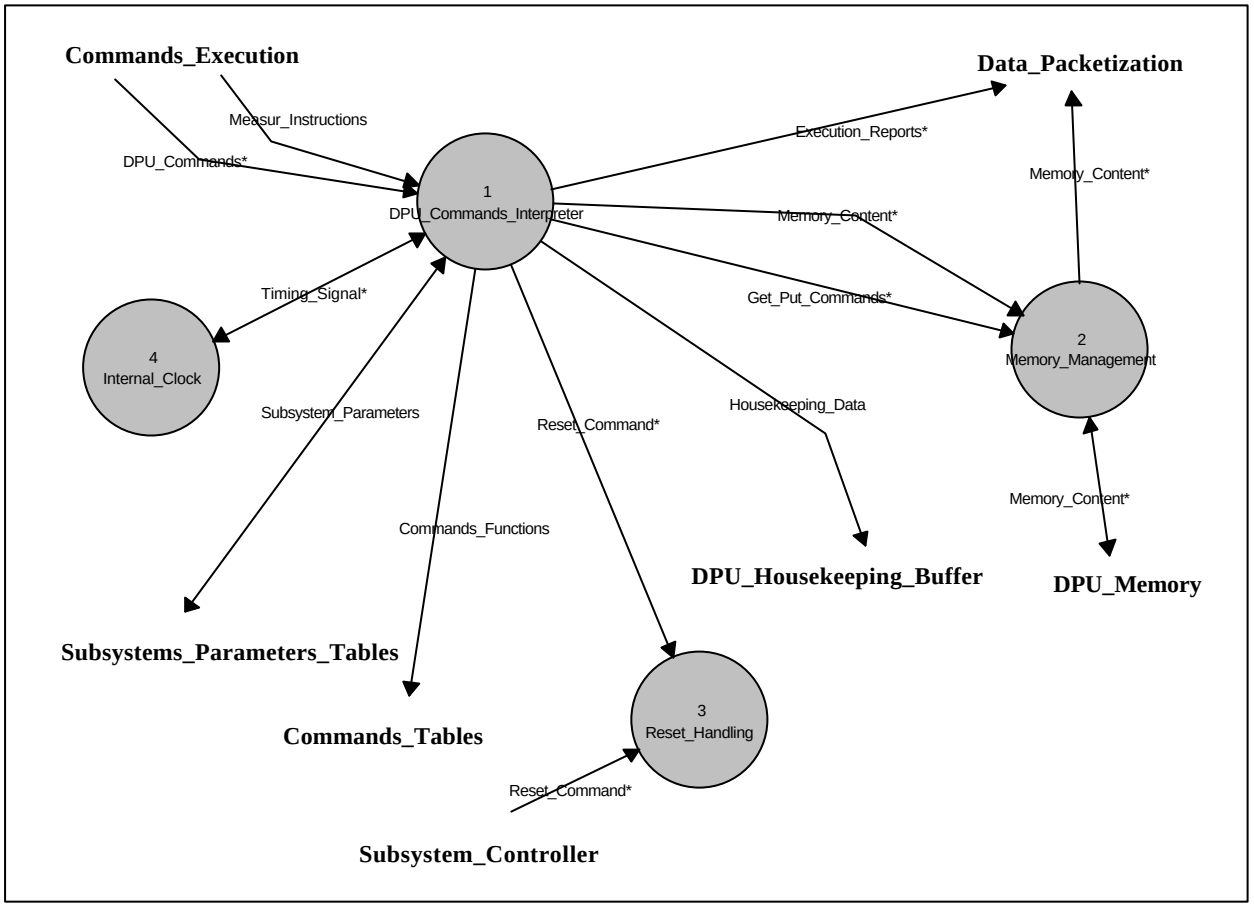


Figure 5 Flow Diagram DPU_Commands_Handler

2.3.1. Logical component DPU_Commands_Interpreter

The IPU_Commands_Interpreter is the main component for handling the IPU_Commands and the DPU/ICU Measur_Instructions routed by the Commands_Execution component, after their verification/acceptance.
In particular, it ensures the implementation of the various services related to these commands: in case of the memory management commands, it generates Get_Put_Commands and sends them to the Memory_Management component for their execution.
The IPU_Commands for updating of the tables stored on board (housekeeping/diagnostic parameters reports definition tables, subsystems measurements parameters tables, commands tables etc.) are handled in this component as well.
The Reset_Command, when it is received as part of the IPU_Commands input data flow, is routed to the Reset_Handling module.
The IPU_Commands_Interpreter synchronises the Internal_Clock with the OBDH master clock whenever the timing synchronisation command is received.
The Housekeeping_Data generated from the DPU/ICU are acquired and stored in the IPU_Housekeeping_Buffer.
The Execution_Reports related to the progress/completion
of the execution of the above described commands are provided.

This process is missing its child diagram.

2.3.2. Logical component Memory_Management

The Memory_Management component handles the Get_Put_Commands generated by the IPU_Commands_Interpreter (mainly containing the addresses of the ICU_Memory to be loaded, dumped or checked). In case of a memory dump command, the memory image (Memory_Content) is prepared in order to be packetised by the Data_Packetisation component. In case of a memory load command, the Memory_Content is extracted from the IPU_Commands and put in memory starting from the requested absolute address.

This process is missing its child diagram.

2.3.3. Logical component Reset_Handling

The Reset_Handling executes the reset procedure after the reception of a Reset_Command either from the Subsystem_Controller or the IPU_Commands_Interpreter.

This process is missing its child diagram.

2.3.4. Logical component Internal_Clock

The Internal_Clock component contains all the functionsnecessary to handle the DPU internal clock. It shall be seen as a sort of functions library which can be used by all the components whenever a time information is needed.
For example, it is used in the DPU_Commands_Handler component for the synchronisation of the internal clock with the CDMS master clock.
In the Data_Packetisation component, it is necessary for adding the generation time (expressed either in internal clock units or in UT) to the output telemetry packets.

This process is missing its child diagram.

2.4. Logical component Subsystem_Communication

The Subsystem_Communication logical component contains all the functionalities for handling the communication between the DPU/ICU and the other Instrument Subsystems.The measurement instructions provided by the Commands_Execution are encoded accordingly to the interface protocols adopted for the various interfaces with the different instrument subsystems and the related Command_Instructions are prepared and sent. The Science_Data and the Housekeepig_Data collected from the Instrument Subsystems are decoded and stored into the Science_Data_Buffer and the Housekeeping_Data_Buffer respectively.

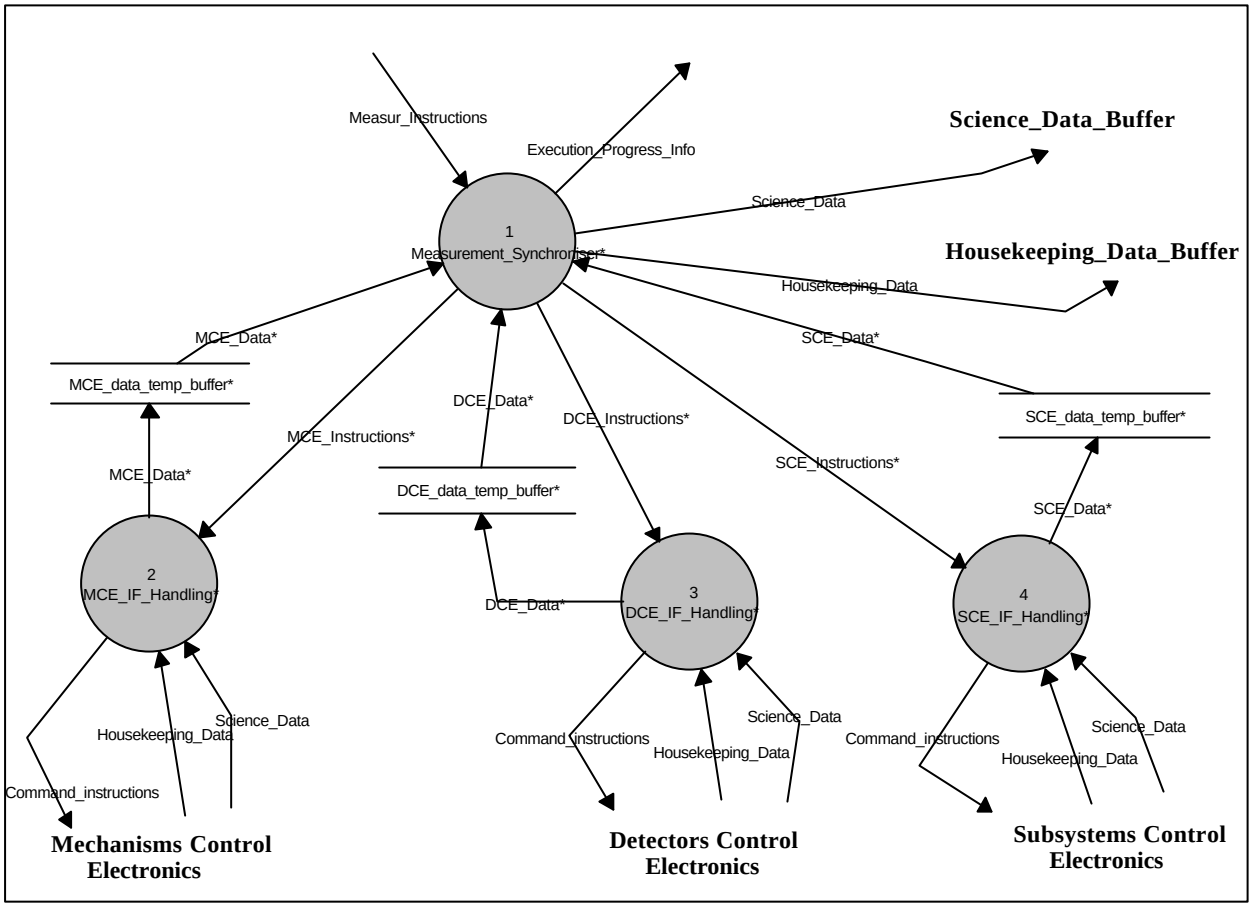


Figure 6 Flow Diagram Subsystem_Communication

2.4.1. Logical component Measurement_Synchroniser

This process is missing its child diagram.

2.4.2. Logical component MCE_IF_Handling

This process is missing its child diagram.

2.4.3. Logical component DCE_IF_Handling

This process is missing its child diagram.

2.4.4. Logical component SCE_IF_Handling

This process is missing its child diagram.

2.5. Logical component Subsystem_Controller

The Subsystem_Controller logical component contains all the functionalities for monitoring the instrument subsystems and for implementing all the autonomy functions necessary to guarantee the correct instrument operation.
The management of anomalies will be handled in a hierarchical manner such that the solution of the problems shall be sought at the lowest level possible. This will imply that, in the case of SPIRE, the autonomy functions implemented by the OBS

will be limited to the ones related to the monitoring of the DPU and DRCU housekeeping data, while for the other instruments the implemented procedures will have more extended possibilities.
In particular, the Subsystem_Controller extracts from the housekeeping data buffers (Housekeeping_Data_Buffer and IPU_Housekeeping_Buffer) the parameters to be monitored, compares their values with the reference values/ranges read from the related subsystem parameters table and reacts on the basis of the comparison result: in case of normal operations, an Event_Report is prepared to be packetised by the Data_Packetisation component; in case of anomaly the corresponding anomaly handling function is activated and, possibly, some corrective Command_Instructions are generated and sent to the Commands_Execution component in order to be executed as immediate commands. If the instrument warm reset is the only way to recover the situation, the Reset_Command is provided to the IPU_Commands_Handler.
All the inputs actually used during the execution of the autonomy functions are collected into a dedicated Event_Report, to be packetised by the Data_Packetisation component.

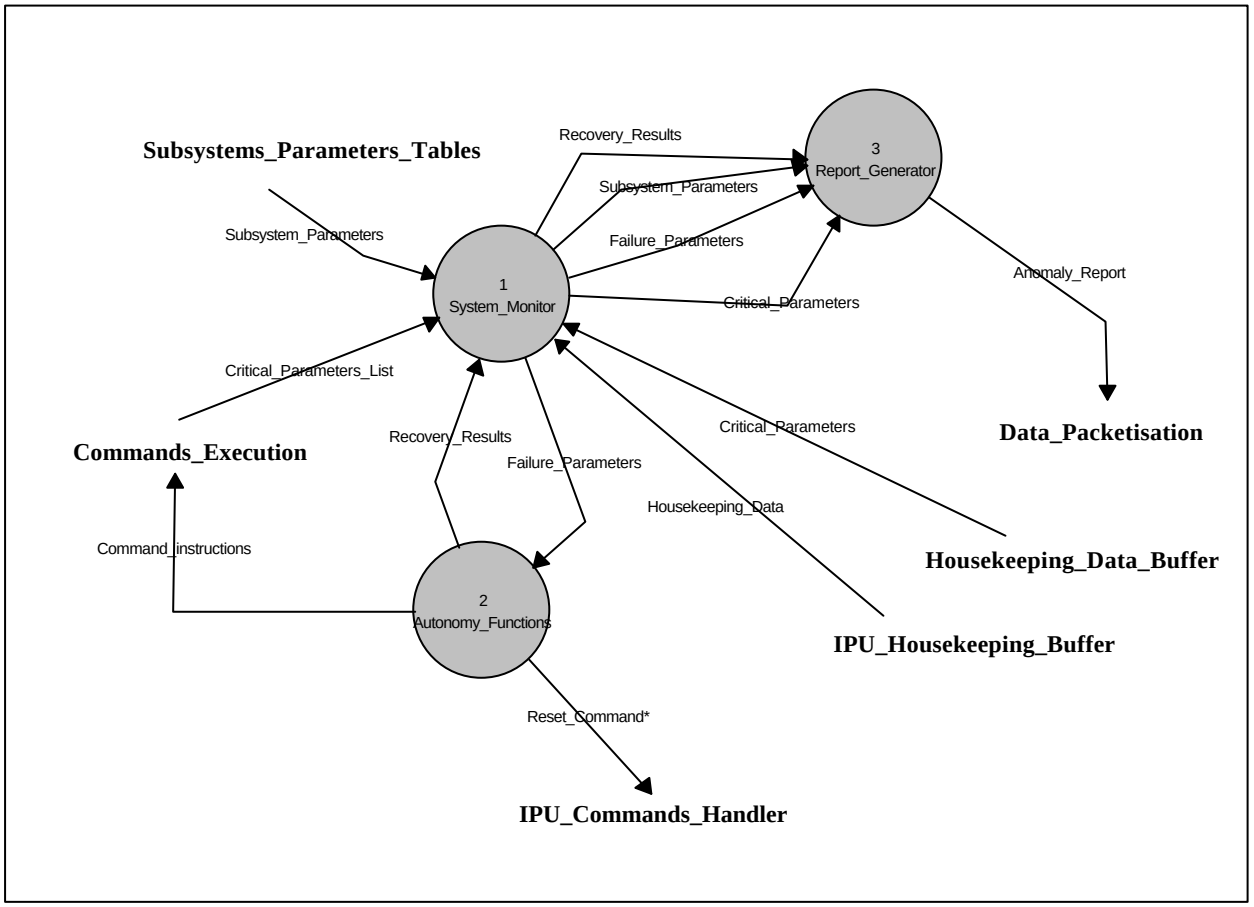


Figure 7 Flow Diagram Subsystem_Controller

2.5.1. Logical component System_Monitor

The System_Monitor component monitors the values of the critical housekeeping parameters and, in case of normal operations, sends the results of the comparison to the Report_Generator for the compilation of a normal progress report. In case the

parameters values are out of the reference ranges, the System_Monitor reacts by activating the corresponding recovery function/procedure (Autonomy_Functions component).

This process is missing its child diagram.

2.5.2. Logical component Autonomy_Functions

The Autonomy_Functions component includes all the functions/procedures to be activated in case of a system anomaly detection. The actions performed by these functions are reflected in the output Command_Instructions data flow and/or in the Reset_Command to the IPU_Commands_Handler.

This process is missing its child diagram.

2.5.3. Logical component Report_Generator

The Report_Generator component gathers all the inputs actually used during the execution of the monitoring activity and of the autonomy functions (Subsystem_Parameters, Critical_Parameters, Failure_Parameters and Recovery_Results) and prepares a dedicated Event_Report, to be packetised by the Data_Packetisation component.

This process is missing its child diagram.

2.6. Logical component Data_Packetisation

The Data_Packetisation logical component includes all the functionalities related to the implementation of the ESA Packetization standard.
The Science_Data are sequentially read from the related buffer and the corresponding source packets are generated, eventually including some housekeeping information.
The Housekeeping (IPU and Subsystems) Data and the diagnostic data are read from the related buffers and the corresponding packets generated, containing either all the default parameters or a subset of parameters configured by telecommand.
A generation time information is included in the packets and the component is in charge of synchronising the reading of the housekeeping and science data obtained for the same measurement.
Another important functionality of this component is the packetisation of all the other reports requestes on ground, for example: Telecommands Acceptance and Execution Reports (collecting information both from the Command_Executions and the IPU_Commands_Handler components), Error/Anomaly Reports (based on the Subsystem_Controller provided information), etc..
The Memory_Content corresponding to a request of memory dump is packetised as well. All the generated telemetry packets are stored in the OBS output buffer TM_Packets_Buffer.

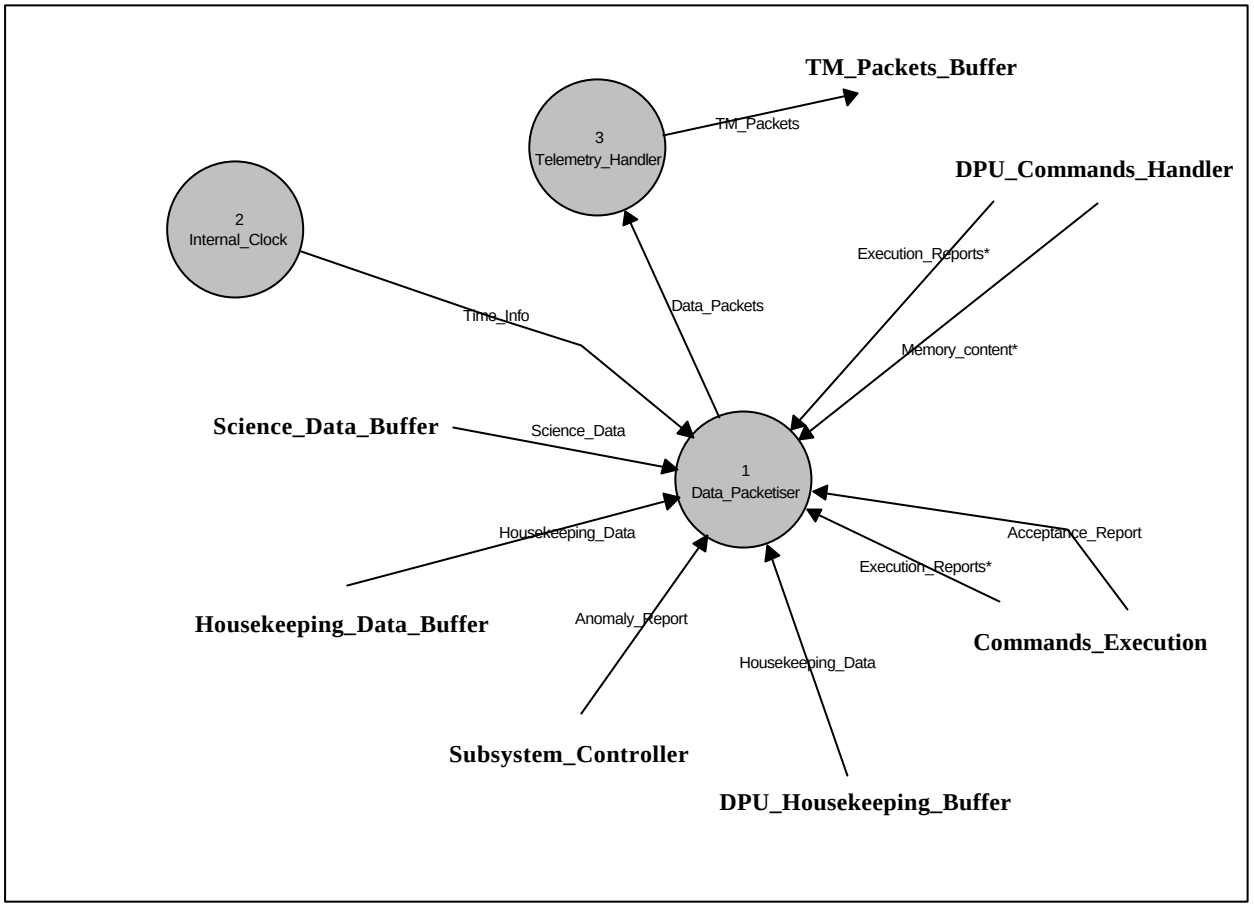


Figure 8 Flow Diagram Data_Packetisation

2.6.1. Logical component Data_Packetiser

The Data_Packetiser implements all the data formatting activities described in the Data_Packetisation logical component. The output of these activities will be the Data_Packets data flow, containing the complete packets data field.
Since the data field header will include the generation time of the packets, this component will get the necessary time info from the DPU/ICU internal clock.

This process is missing its child diagram.

2.6.2. Logical component Internal_Clock

The Internal_Clock component contains all the functionsnecessary to handle the DPU internal clock. It shall be seen as a sort of functions library which can be used by all the components whenever a time information is needed.
For example, it is used in the DPU_Commands_Handler component for the synchronisation of the internal clock with the CDMS master clock.
In the Data_Packetisation component, it is necessary for adding the generation time (expressed either in internal clock units or in UT) to the output telemetry packets.

This process is missing its child diagram.

2.6.3. Logical component Telemetry_Handler

The Telemetry_Handler component adds to the packet data field (Data_Packets), prepared by the Data_Packetiser component, the packets primary header and stores the obtained final telemetry packet into the TM_Packets_Buffer.

This process is missing its child diagram.