DEC/MEC SSD

Doc. PACS-CL-SR-001, issue 1.1
11 April, 2006

Prepared by : A. Mazy

Verified by :

Authorised by : E. Renotte

Approved by :

filename : PACS-CL-SR-001v1.1 SSD.doc

Distribution List

Recipients	Affiliation	Copies
CSL PACS Team c/o E. Renotte	CSL - Liege	\\Titan_pacs\documents (sources)\e) Software\PACS-CL-SR-001v1.1.doc
Central File @ PACS Project Office c/o N. Gradmann	MPE - Garching	1 PDF copy
O. Bauer, H. Feuchtgruber	MPE - Garching	1 PDF copy

Document Change Record

Issue	Date	Comments
draft 0.1	14/02/2001	initial issue
draft 0.2	25/01/2002	Removed useless details and added cooler controller + CS controller + MEC ISR servo
Issue 1.0	03/10/2003	90% rewritten, all pages changed to take into account IBDR comments.
Issue 1.1	11/04/2006	Updated to take into account the changes introduced during EM and QM OBS development. The SSD is now representative of the QM and FM flight software.

last saved by amazy on 11-Apr-06

List of Abbreviations

ASW	Application Software
AVM	Avionic Verification Model
BOLA	Bolometer Amplifier
BOLC	Bolometer Controller
CSL	Centre Spatial de Liège
DAC	Digital-to-Analog Converter
DEC/MEC	Detector & Mechanism Controller
DPU	Digital Processing Unit
EEPROM	Electrically Erasable PROM
EGSE	Electrical Ground Support Equipment
EM	Engineering/Electrical Model
ESA	European Space Agency
FM	Flight Model
FPGA	Field Programmable Gate Array
FPU	Focal Plane Unit
HK	HouseKeeping
HW	Hardware
ICD	Interface Control Document
NA	Not Applicable
OBS	On-Board Software
PACS	Photodetector Array Camera and Spectrometer
PROM	Programmable ROM
QM	Qualification Model
RAM	Random Access Memory
ROM	Read-Only Memory
S/C	SpaceCraft
SPU	Signal Processing Unit
SSD	Software Specification Document
SW	Software
SUSW	Start-Up Software
TBC	To Be Confirmed
TBD	To Be Defined
URD	User Requirement Document

Table of Contents

1 Scope

1.1 Introduction

The Photodetector Array Camera and Spectrometer (PACS) is an imaging spectrometer-photometer which forms part of the science payload of the Herschel Space Observatory (formerly called FIRST), an ESA cornerstone mission (CS4) to be launched in 2007 on Ariane 5.

A presentation of the Herschel mission and status is available at URL: http://sci.esa.int/home/herschel/. Useful information on PACS instrument, mission and Consortium can be found at http://pacs.mpe-garching.mpg.de and http://pacs.ster.kuleuven.ac.be/.

1.2 Purpose

This document presents the requirements and the architecture of the software embedded in the DEC/MEC.

This document is addressed to :

- The programmer who will code the DEC/MEC application.
- The programmer who will code the DPU application (since they need to know how data are organized inside the application).
- The system engineers who need to know how to use the DEC/MEC subsystem.

The software embedded in the DEC/MEC is split in two independent module:

- The Sart-up Software (SUSW) is used only at start-up to check the memory, perform patching and start the Application software. The SUSW is not described in this document.
- The Application Software (ASW) is the main application that is described in this document.

1.3 Organisational Responsibilities

The PACS project activities including project management and system engineering will be done at MPE-Garching under the direction of A. Poglitsch (PI). Design, fabrication, testing, and integration of the flight units will be done at Co-I and commercial facilities as appropriate.

In this programme the CSL is responsible for the design, production and unit-level verification of:

- the focal plane Grating Assembly;
- the Detector & Mechanism Controller (DEC/MEC);
- the Warm Interconnecting Harness.

2 Documents

2.1 Applicable Documents

[AD1]	ESA PT-IID-A-04624	FIRST/PLANCK Instrument Interface Document - Part A
[AD2]	ESA PT-RQ-04410	PA Requirements for FIRST/PLANCK Scientific Instruments
[AD3]	PACS-ME-RS-004	PACS Science Requirements Document
[AD4]	PACS-ME-RS-005	PACS Instrument Requirements Document
[AD5]	PACS-ME-PL-007	PACS Project Product Assurance Plan
[AD6]	PACS-CL-SP-001	DEC/MEC Software User Requirements Document
[AD7]	PACS-CL-ID-003	Interface Control Document DEC/MEC – DPU
[AD8]	PACS-CL-ID-004	Interface Control Document DEC/MEC - SPU
[AD9]	PACS-CL-ID-006	DEC to MEC and DEC to EGSE interface control document
[AD10]	HPL-IC-1248-01-CRS	HW-SW Interface Control Document
[AD11]	HPL-ICD-1248-01-CRS	Low Level SW Drivers SW Interface Control Document
[AD12]	PACS-CL-SR-002	DEC/MEC User Manual

2.2 Reference Documents

[RD1]	ESA PT-PACS-02126	Instrument Interface Document - Part B - Instrument "PACS"
[RD2]	PACS-ME-PL-002	PACS Design, Development and Verification Plan
[RD3]	-	CSL naming conventions for software development
[RD4]	1355-1995 (ISO/IEC 14575)	IEEE Standard for Heterogeneous InterConnect (HIC)
[RD5]	-	AxiomSys User's Manual
[RD6]	-	Virtuoso user guide for Version 4.1

3 General Description

3.1 Function and purpose

This document presents the requirements and the architecture of the Application Software embedded in the DEC/MEC.

This document is formally the result of merging the SRD and ADD of the ESA software engineering standards (BSSC(96)) strategies for document simplification for small software projects are applied). It is the development team response to the User Requirements Document [AD6] while taking into account the hardware design of the DEC/MEC

The software resides on the DSP board which is a part of the DEC/MEC subsystem.

This software will :

- Control and monitor the FPU : Chopper, Grating, Filter Wheels, Temperature sensors.
- Control the detectors arrays (setting timing parameters, bias voltages, …)
- Forward command to BOLC
- Receive the (raw) science data from the detector arrays and BOLC
- Format the science data and some HK data for transmission to the SPU.
- Receive its commands from the DPU
- Perform HK acquisition and format the data for transmission to the DPU.

A complete description of the user requirements related to the DEC/MEC on-board software can be found in [AD6].

3.2 Environment

3.2.1 Operating system

The DEC/MEC software shall be implemented as a set of tasks under a real-time operating system (VIRTUOSO[tm]) running on a 21020 DSP processor. [1]

3.2.2 Hardware

The DEC/MEC hardware (as viewed by the software) will be made of :

- a processor module with DSP CPU, RAM, ROM, EEPROM, Clock, start-up and power save logic, and 6 IEEE1355 « spacewire » serial bi-directional communication links.
    - In the EM and further models, this module will consist of the CRISA DSP board, identical to the SPU processor board, with an additional "mezzanine" board for 3 links.
- Two « DEC » modules, made of 5 boards each, in charge of the detector arrays timing and data acquisition, each one using a 1355 link for communication with the processor board. The DEC modules themselves will have no CPU.
- One « mechanism interface » module, made of 3 boards, containing the input and output circuits for the instrument mechanisms and housekeeping sensors, interfaced through the back-plane and appearing as a set of registers to the software.

3.2.3 Interfaces

The external interfaces will be implemented by 1355 links, possibly at reduced speed or unidirectional, according to the actual needs :

[1] VIRTUOSO is a real time operating system from the company Windriver

- a 10 Mbps bi-directional interface with the DPU will be used to receive commands and operation sequences, and transmit status and housekeeping data
- two 10 Mbps unidirectional interfaces with the SPU, each one being dedicated to the data from one of the detector arrays (called hereafter « blue » or "short wavelength" and « red » or "long wavelength")
- two 10 Mbps bi-directional interface with the DEC modules.
- one 10 Mbps bi-directional interface with the controller of the bolometers array and cooler

Additionnally, the hardware units exchange readout synchronisation signals (DEC and BOLC to MEC)

3.2.4 Microprocessor (DSP) constraints

1. Program bus: 48 bits

2. Data bus: 32 bits

3. Memory dimensions 512K*40bits of DRAM + 512K*56bits of PRAM + 256K*64bits of EEPROM + 32K*8bits of start-up PROM

4. Clock speed (18 MHz)

3.2.5 Development environment

The software will be developed using the following environment:

- Virtuoso 4.2R3 as operating system (Virtuoso is a trademark of Windriver)
- AxyomSys 3 as structured analysis based CASE tool (AxyomSys is a trademark of Structured Technology Group, Inc.)
- SIGMA 4 33MHz ADSP 21020 development board for SW testing (the Sigma board is manufactured by BittWare research systems)
- MOSAIC020 DSP board (the board is manufactured by DASA)
- One PC with a PCI-SpaceWire board running Virtuoso Development Tools.
- One PC with 2 PCI-SpaceWire boards running Sub-systems simulators (coded in C++ with MS Visual Studio 6 SP3)

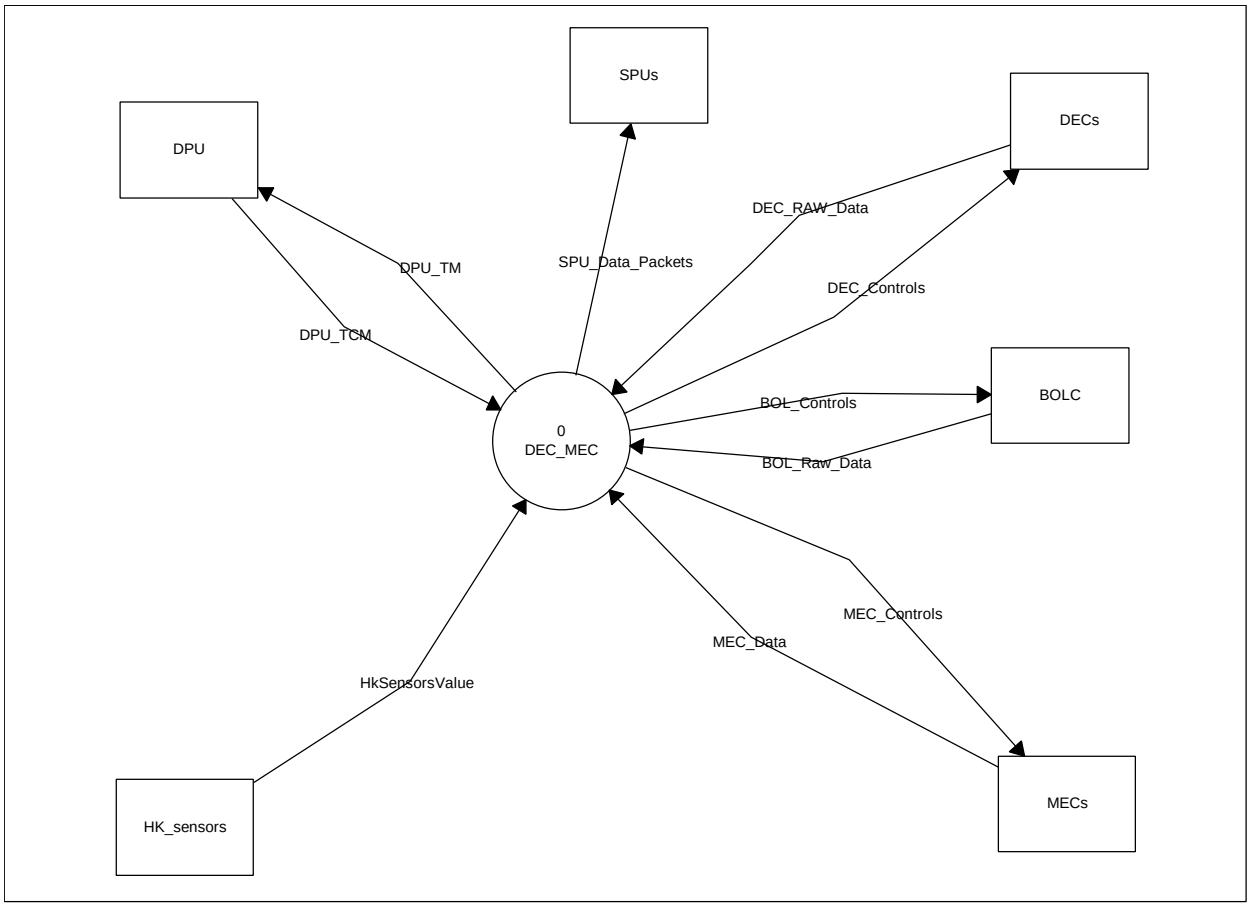
3.3 Relation to other systems

DEC/MEC on-board software must interface with the following other sub-systems, internal or external to the DEC/MEC :

- The DPU sends the tele-commands and receives the telemetry. Furthermore, DPU may access the DSP module memory in order to check it or to upgrade its content.

  The DPU is connected to the DEC/MEC with a 1355 link.
- Two DSP units in the SPU process the science data coming from the detectors.
  Each unit is connected to the DEC/MEC with a 1355 link.

- Two DEC modules control the two arrays of Photoconducting detectors.
  Each DEC module is connected to the DSP module with a 1355 link.
- One BOLC controls the two arrays of Bolometer detectors.
  DEC/MEC and BOLC are connected with a 1355 link.
- Focal plane mechanisms (Chopper, Grating, filter wheels. Heaters are also included in this group of components).
  Mechanisms are commanded by DEC/MEC through a bus extension of the DSP board and a specific Mechanisms Interface Module described in [AD9]
- Housekeeping sensors are acquired in several locations within the DEC/MEC and other subsystems. The DEC/MEC software sends the housekeeping measurements to the DPU.



3.4 General Constraints

3.4.1 Development tools and operating system standardisation

The standardisation of the software development tools and operating system is considered a user requirement.

The following software tools and operating system have been standardised at project level :

- Virtuoso 4.2R3 as operating system (Virtuoso is a trademark of Windriver)
- AxyomSys 3 as structured analysis based CASE tool (AxyomSys is a trademark of Structured Technology Group, Inc.)

3.4.2 Design philosophy

The software will be implemented as a set of tasks. Each one being as much as possible independent from the others. The software has been designed such as the failure or the dysfunction of a component will not affect any process not related to this component.

The software implementation will only use Virtuoso operating system functions and mathematics library functions that have been declared “safe” by ESA. No condition or combination of conditions will result in a “deadlock” or unknown state.

The software implementation will use only static allocation of variables and a fixed number of tasks all created at start-up.

4 Software Requirements

DECMEC-SR-1 Each software requirements shall be uniquely identified by its ID code according to the following template: DECMEC-SR-nnnn;

4.1 Functional Requirements

4.1.1 Switch-on requirements

DECMEC-SR-2 The DEC/MEC software shall put all mechanisms and both DECs in a defined electrical state at switch-on (detector arrays OFF, all mechanisms powered-off, all thermal regulations inactive), by writing initial values to all command outputs.

DECMEC-SR-3 After switch-on, the DEC/MEC shall initialize the 1355 link with DPU.

DECMEC-SR-4 After the 1355 link with DPU is initialized, the DEC/MEC shall start its housekeeping task and wait for a command from the DPU.

DECMEC-SR-5 The DEC/MEC software start-up operations shall not last more than 5 seconds (before the first HK packet is sent and it is able to accept a DPU command).

4.1.2 Switch-off requirements

DECMEC-SR-6 The DEC/MEC software shall handle all possible cases arising from the switching off of the hardware at any time, even while a mechanism is operating.

DECMEC-SR-7 In the event of an accidental switch-off of the DEC/MEC, it shall be possible to recover full functionality without any specific recovery actions. Once switched-on again, the DEC/MEC should be in the same initial state as after a ‘clean’ switch-off – switch-on.

4.1.3 Housekeeping requirements

DECMEC-SR-8 The DEC/MEC software shall autonomously sample a set of measurement variables (HW and SW related) at a fixed interval of 2 seconds.

DECMEC-SR-9 The set of sampled variables should be sent to DPU without DPU request.

DECMEC-SR-10 The hk measurement set shall contain all the variables defined in [AD12].

4.1.4 Diagnostic Housekeeping requirements

DECMEC-SR-11 The DEC/MEC software shall, on DPU request, sample a limited set of measurement variables (HW and SW related) at custom interval.

DECMEC-SR-12 The sampling rate and list of measurement variables shall be configured by the DPU.

DECMEC-SR-13 The maximum sampling rate shall be 256 samples per second. There shall be 3 special modes where the sampling rate is given by either the red DEC, the blue DEC or the BOLC.

4.1.5 DPU Interface requirements

DECMEC-SR-14 The DEC/MEC software shall receive the commands from the DPU through the 1355 link and interpret them.

DECMEC-SR-15 The DEC/MEC software shall accept the following command types from DPU (as defined in [AD7]) :

- Trigger commands
- Memory commands (read/write/dump/check)

DECMEC-SR-16 The format of all the commands (trigger and memory) shall be documented. Each command shall be identified by a unique ID.

DECMEC-SR-17 The read-only memory commands (read/dump/check) shall uniquely identify the target memory area by specifying the starting address and range.

DECMEC-SR-18 The write memory commands shall uniquely identify the target memory area by specifying a unique ‘memory ID’ that shall be mapped to a starting address in the DEC/MEC software.

DECMEC-SR-19 The DEC/MEC software shall accept all the write commands defined in [AD12].

DECMEC-SR-20 The DEC/MEC software shall accept all the trigger commands defined in [AD12].

DECMEC-SR-21 The DEC/MEC shall be able to receive from the DPU a list of instructions called “sequence”.

DECMEC-SR-22 Specific trigger commands shall be available for starting, stopping, aborting the execution of the operation sequence under DPU command.

DECMEC-SR-23 The DEC/MEC software shall send acknowledge (positive or negative) to DPU commands.

4.1.6 Instrument control requirements

DECMEC-SR-24 The DEC/MEC software shall handle the execution of the trigger commands and sequences of commands.

DECMEC-SR-25 The mechanisms shall not be moved unless commanded by a DPU command.

4.1.6.1 Sequences

DECMEC-SR-26 The DEC/MEC shall execute the instructions stored in a sequence sequentially or as directed by control instruction (loop, wait)

DECMEC-SR-27 The DEC/MEC shall be able to derive the timing of execution from different sources:

- The last readout of an integration ramp (from blue or red DEC)
- Readout from BOLC

DECMEC-SR-28 There shall be a trigger command that DPU can use to select the synchronisation source

DECMEC-SR-29 There shall be a trigger command that DPU can use to start the execution of a sequence.

DECMEC-SR-30 There shall be a trigger command that DPU can use to abort the execution of a sequence at any time.

DECMEC-SR-31 During the execution of a sequence, only the abort trigger command shall be accepted. Any other trigger command shall be rejected.

DECMEC-SR-32 During the execution of a sequence, no memory management command shall be accepted.

4.1.6.2 Mechanisms Control

DECMEC-SR-33 The DEC/MEC software shall be capable of executing the position servo closed-loop algorithm for the grating, chopper and filter wheels with an update rate of at least 8192Hz – other (lower) rates shall be possible (the rate is given by the FPGA timing)

DECMEC-SR-34 The DEC/MEC software shall be able to switch-on/off each of the servo closed-loop algorithm for each mechanism independently.

DECMEC-SR-35 For each of the mechanisms, the DEC/MEC software shall generate a trajectory for implementing the optimal move between successive set-points values generated by the move commands.

DECMEC-SR-36 For each of the mechanisms, the DEC/MEC software shall perform plausibility range check, report status and housekeeping data to the DPU in the housekeeping data set. The DEC/MEC software shall work in encoder unit only.

DECMEC-SR-37 The movement of the chopper and grating shall be synchronized with the readout reception. The interval between the readout and the beginning of the movement shall be configurable by suitable parameters sent to the hardware (timing FPGA).

4.1.6.3 Grating Control

DECMEC-SR-38 The DEC/MEC software shall implement the conversion of the torque command into the currents for the 0° and 90° coils taking into account the angle readout.

DECMEC-SR-39 The DEC/MEC software shall contain a specific procedure for unlocking and relocking the launch lock under DPU command.

DECMEC-SR-40 The DEC/MEC software shall find the origin of the grating encoder by moving slowly the grating until the limit switches and resetting the 8 MSB of the position when the limit position is reached.

DECMEC-SR-41 The DEC/MEC software shall be able to move the grating without using the encoder. Sine and cosine waveform at low frequency shall be sent to the motor drive to move the grating at reduced speed.

DECMEC-SR-42 The DEC/MEC software shall have a lock procedure for the grating. It shall be able to short-circuit the coils and/or active the launch lock.

DECMEC-SR-43 When switching-off the grating controller, the DEC/MEC software shall make sure all related components (drive + inductosyn) located in the cold focal plane unit shall dissipate no power, by setting the appropriate control bits in the hardware interface

4.1.6.4 Chopper Control

DECMEC-SR-44 The DEC/MEC software shall implement a degraded mode where broken sections of the coil are bypassed by hardware switches.

DECMEC-SR-45 The DEC/MEC software shall implement the conversion of the torque command into the coil currents taking into account which sections(s) of the coil are in use.

DECMEC-SR-46 When switching-off the chopper controller, the DEC/MEC software shall make sure all related components located in the cold focal plane unit shall dissipate no power by setting the appropriate control bits in the hardware interface

4.1.6.5 Black bodies Control

DECMEC-SR-47 The DEC/MEC software shall implement the temperature control closed-loop algorithm for the black bodies with an update rate of 1Hz (TBC).

DECMEC-SR-48 The DEC/MEC software shall generate a trajectory for implementing the optimal move between successive set-points values generated by the move commands (TBC).

4.1.6.6 Filter Wheels Control

DECMEC-SR-49 The DEC/MEC software shall handle 2 filter wheels with the same drive. Only one wheel shall be operated at a time.

DECMEC-SR-50 The DEC/MEC software shall move the filter-wheels in open loop until they reach limit switches signalling the target position.

DECMEC-SR-51 The DEC/MEC software shall also be able to move the filter-wheels in open loop for a given number of steps without checking the limit switches.

DECMEC-SR-52 When switching-off the filter-wheels controller, the DEC/MEC software shall make sure all related components located in the cold focal plane unit shall dissipate no power by setting the appropriate control bits in the hardware.

4.1.7 Detector control and data handling requirements

4.1.7.1 Photoconducting detector array parameters

DECMEC-SR-53 The DEC/MEC software shall be able to send commands to both DECs.

DECMEC-SR-54 There shall be a command to change the timing parameters of DECs. No check of the parameters shall be done in DEC/MEC software.

DECMEC-SR-55 There shall be a command to synchronize both DECs such that their integration ramp start at the same time.

4.1.7.2 Photoconducting arrays image acquisition and SPU interface

DECMEC-SR-56 The DEC/MEC software shall be able to to receive the science data from the DECs at 256 readouts/seconds/array.

DECMEC-SR-57 The DEC/MEC software shall be able to synchronize the execution of a sequence on the destructive readouts of any of the DECs.

DECMEC-SR-58 The DEC/MEC software shall extract the HK data from the packets and make it available to other tasks and DPU.

DECMEC-SR-59 The DEC/MEC software shall extract the raw detector array image from the DEC packets and send them to SPU. Packets shall also contain HK info as described in [AD8].

DECMEC-SR-60 It shall be possible to configure which DEC data are sent to which SPU.

DECMEC-SR-61 It shall be possible to replace DEC detector data by simulated data.

DECMEC-SR-62 The DEC/MEC software shall check the interval between DEC readouts and report the time-out in the HK.

4.1.7.3 Bolometers array parameters and commands

DECMEC-SR-63 The DEC/MEC software shall be able to send commands to BOLC.

DECMEC-SR-64 The DEC/MEC software shall simply forward commands from DPU to BOLC.

4.1.7.4 Bolometers arrays image acquisition and SPU interface

DECMEC-SR-65 The DEC/MEC software shall be able to receive the science data from BOLC at 40 readouts/sec (each readout is divided in 5 science packets and 1 HK packet).

DECMEC-SR-66 The DEC/MEC software shall extract the raw detector array image from the BOLC packets and send them to SPU. Packets shall also contain HK info as described in [AD8].

DECMEC-SR-67 The DEC/MEC software shall be able to synchronize the execution of a sequence on the readouts of BOLC.

DECMEC-SR-68 It shall be possible to replace BOLC detector data by simulated data.

DECMEC-SR-69 The DEC/MEC software shall check the interval between BOLC readouts and report the time-out in the HK.

4.2 Performance Requirements

None

4.3 Interface Requirements

DECMEC-SR-70 The DEC/MEC software shall handle 6 1355 links running at 10Mbps :

- 1 bi-directional interface with DPU
- 2 uni-directional interface with SPUs
- 2 bi-directional interface with DECs
- 1 bi-directional interface with BOLC

DECMEC-SR-71 The DEC/MEC software shall detect communication loss and signal it in the HK.

DECMEC-SR-72 In the case of a communication loss on DPU link, the DEC/MEC software shall restart the link autonomously until it reconnects to DPU.

4.4 Operational Requirements

None

4.5 Resource Requirements

DECMEC-SR-73 The DEC/MEC software shall be implemented as a set of tasks under a real-time operating-system (VIRTUOSO™).

DECMEC-SR-74 The DEC/MEC software shall run on the following hardware : a processor board with a 21020 DSP (18 Mhz), 512 kwords of data RAM, 512 kwords of program RAM, 256 kwords of EEPROM, 32 kwords of ROM (TBC))

DECMEC-SR-75 The DEC/MEC software shall never use more than 70% of CPU load.

4.6 Verification Requirements

DECMEC-SR-76 The software shall undergo acceptance verification based on the Software Validation and Verification Plan at the following levels:

- Subsystem level
- System level

4.7 Acceptance Requirements

None

4.8 Documentation Requirements

DECMEC-SR-77 The list of commands, their format, usage and availability in different modes (trigger, in sequence) shall be documented.

DECMEC-SR-78 The list of HK measures, their format, meaning and availability in diagnostic mode shall be documented.

4.9 Security Requirements

None

4.10 Portability Requirements

None

4.11 Quality Requirements

DECMEC-SR-79 The DEC/MEC software design shall use static allocation of variables and a fixed number of tasks, and generally refrain from using techniques that can result in an unknown or random amount of resources being used. Any dynamic allocation of resource shall be clearly identified.

4.12 Reliability Requirements

DECMEC-SR-80 The DEC/MEC software shall be modular. Independent functions shall be implemented in separate tasks. The DEC/MEC software shall continue to operate properly even if a specific hardware component or mechanism and its associated software task is not present, does not answer or fails prior or during any sequence.

DECMEC-SR-81 The DEC/MEC software shall only use such Virtuoso operating system services that have been declared “safe” by ESA. No condition or combination of conditions shall result in a “deadlock” or unknown state.

4.13 Maintainability Requirements

DECMEC-SR-82 The DEC/MEC software shall be designed to allow “patching” i.e. TBD parts of the software shall be replaceable in flight by uploading code through the DPU without requiring modification to the remaining part of the software.

4.14 Safety Requirements

None

4.15 User requirements – Software requirement traceability matrix

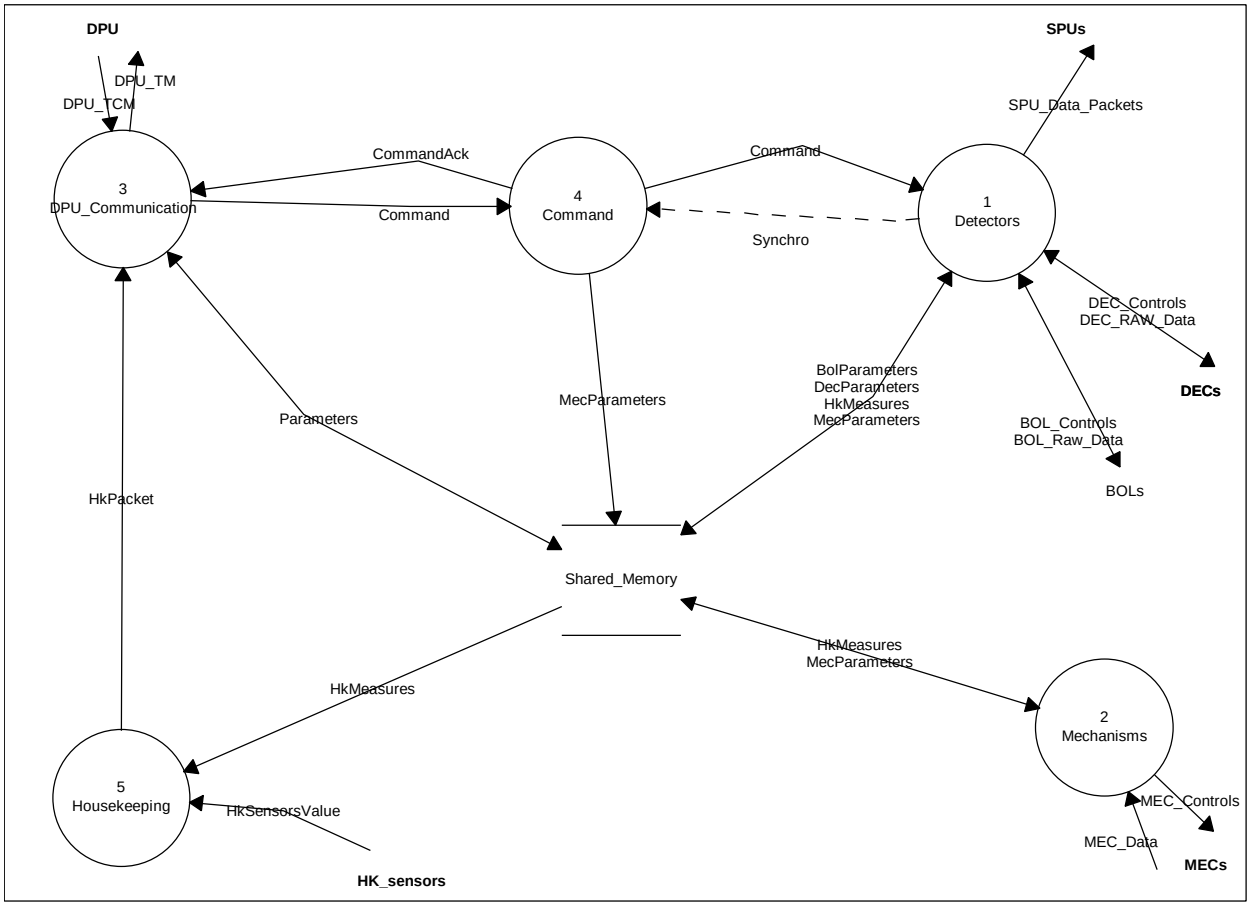
User Requirements	Software Requirements
DECMEC-UR-00	DECMEC-SR-1
DECMEC-UR-01	DECMEC-SR-8 + DECMEC-SR-19
DECMEC-UR-02	DECMEC-SR-80
DECMEC-UR-03	DECMEC-SR-81
DECMEC-UR-04	DECMEC-SR-79
DECMEC-UR-05	DECMEC-SR-77 + DECMEC-SR-78
DECMEC-UR-06	DECMEC-SR-82
DECMEC-UR-07	Requirement on the SUSW
DECMEC-UR-08	DECMEC-SR-2
DECMEC-UR-09	DECMEC-SR-25
DECMEC-UR-10	DECMEC-SR-4
DECMEC-UR-11	DECMEC-SR-72
DECMEC-UR-12	DECMEC-SR-5
DECMEC-UR-13	Deleted
DECMEC-UR-14	DECMEC-SR-6
DECMEC-UR-15	Deleted
DECMEC-UR-16	DECMEC-SR-7
DECMEC-UR-17	DECMEC-SR-8
DECMEC-UR-18	DECMEC-SR-8
DECMEC-UR-19	Deleted
DECMEC-UR-20	Deleted
DECMEC-UR-21	DECMEC-SR-8
DECMEC-UR-22	DECMEC-SR-11 + DECMEC-SR-12 + DECMEC-SR-13
DECMEC-UR-23	DECMEC-SR-59 + DECMEC-SR-66
DECMEC-UR-24	DECMEC-SR-10
DECMEC-UR-25	DECMEC-SR-10
DECMEC-UR-26	DECMEC-SR-10
DECMEC-UR-27	DECMEC-SR-10
DECMEC-UR-28	DECMEC-SR-10
DECMEC-UR-29	DECMEC-SR-10
DECMEC-UR-30	DECMEC-SR-10
DECMEC-UR-31	DECMEC-SR-10
DECMEC-UR-32	DECMEC-SR-10
DECMEC-UR-33	DECMEC-SR-15
DECMEC-UR-34	DECMEC-SR-61 + DECMEC-SR-68
DECMEC-UR-35	DECMEC-SR-82
DECMEC-UR-36	DECMEC-SR-82
DECMEC-UR-37	DECMEC-SR-15
DECMEC-UR-38	DECMEC-SR-10
DECMEC-UR-39	DECMEC-SR-10

DECMEC-UR-40	Deleted
DECMEC-UR-41	DECMEC-SR-19 + DECMEC-SR-20
DECMEC-UR-42	DECMEC-SR-15
DECMEC-UR-43	Deleted
DECMEC-UR-44	Deleted
DECMEC-UR-45	DECMEC-SR-15
DECMEC-UR-46	DECMEC-SR-17 + DECMEC-SR-18
DECMEC-UR-47	DECMEC-SR-17 + DECMEC-SR-18
DECMEC-UR-48	DECMEC-SR-18
DECMEC-UR-49	DECMEC-SR-9
DECMEC-UR-50	DECMEC-SR-20
DECMEC-UR-51	DECMEC-SR-10
DECMEC-UR-52	DECMEC-SR-19
DECMEC-UR-53	DECMEC-SR-21
DECMEC-UR-54	DECMEC-SR-22
DECMEC-UR-55	Deleted
DECMEC-UR-56	Deleted
DECMEC-UR-57	DECMEC-SR-21
DECMEC-UR-58	DECMEC-SR-26
DECMEC-UR-59	DECMEC-SR-27 + DECMEC-SR-28
DECMEC-UR-60	DECMEC-SR-55
DECMEC-UR-61	DECMEC-SR-27 + DECMEC-SR-28
DECMEC-UR-62	DECMEC-SR-20
DECMEC-UR-63	DECMEC-SR-57 + DECMEC-SR-67
DECMEC-UR-64	DECMEC-SR-20
DECMEC-UR-65	DECMEC-SR-10
DECMEC-UR-66	DECMEC-SR-30
DECMEC-UR-67	DECMEC-SR-31
DECMEC-UR-68	DECMEC-SR-20
DECMEC-UR-69	DECMEC-SR-20
DECMEC-UR-70	DECMEC-SR-20
DECMEC-UR-71	DECMEC-SR-20
DECMEC-UR-72	DECMEC-SR-20
DECMEC-UR-73	DECMEC-SR-33
DECMEC-UR-74	DECMEC-SR-38
DECMEC-UR-75	DECMEC-SR-35
DECMEC-UR-76	DECMEC-SR-36
DECMEC-UR-77	DECMEC-SR-40
DECMEC-UR-78	DECMEC-SR-41
DECMEC-UR-79	DECMEC-SR-39
DECMEC-UR-80	DECMEC-SR-43
DECMEC-UR-81	DECMEC-SR-42

DECMEC-UR-82	DECMEC-SR-19
DECMEC-UR-83	DECMEC-SR-33
DECMEC-UR-84	DECMEC-SR-44
DECMEC-UR-85	DECMEC-SR-45
DECMEC-UR-86	DECMEC-SR-35
DECMEC-UR-87	DECMEC-SR-36
DECMEC-UR-88	Can be done by moving the chopper to the end of range
DECMEC-UR-89	DECMEC-SR-46
DECMEC-UR-90	DECMEC-SR-19
DECMEC-UR-91	DECMEC-SR-47
DECMEC-UR-92	DECMEC-SR-48
DECMEC-UR-93	DECMEC-SR-20
DECMEC-UR-94	DECMEC-SR-10
DECMEC-UR-95	DECMEC-SR-19
DECMEC-UR-96	DECMEC-SR-49
DECMEC-UR-97	DECMEC-SR-49
DECMEC-UR-98	DECMEC-SR-50
DECMEC-UR-99	DECMEC-SR-35
DECMEC-UR-100	DECMEC-SR-50
DECMEC-UR-101	DECMEC-SR-52
DECMEC-UR-102	Can be done by enabling the FW controller
DECMEC-UR-103	Can be done with a nominal move command
DECMEC-UR-104	DECMEC-SR-20
DECMEC-UR-105	DECMEC-SR-51
DECMEC-UR-106	DECMEC-SR-19
DECMEC-UR-107	DECMEC-SR-20
DECMEC-UR-108	DECMEC-SR-20 + DECMEC-SR-19
DECMEC-UR-109	DECMEC-SR-19
DECMEC-UR-110	DECMEC-SR-55
DECMEC-UR-111	DECMEC-SR-54
DECMEC-UR-112	Deleted
DECMEC-UR-113	Deleted
DECMEC-UR-114	DECMEC-SR-56
DECMEC-UR-115	DECMEC-SR-56
DECMEC-UR-116	DECMEC-SR-55
DECMEC-UR-117	DECMEC-SR-57
DECMEC-UR-118	DECMEC-SR-62
DECMEC-UR-119	DECMEC-SR-58
DECMEC-UR-120	DECMEC-SR-59
DECMEC-UR-121	DECMEC-SR-60
DECMEC-UR-122	DPU/Ground responsibility
DECMEC-UR-123	DPU/Ground responsibility

DECMEC-UR-124	DPU/Ground responsibility
DECMEC-UR-125	DECMEC-SR-64
DECMEC-UR-126	DECMEC-SR-64
DECMEC-UR-127	DECMEC-SR-20
DECMEC-UR-128	DECMEC-SR-65
DECMEC-UR-129	DECMEC-SR-65
DECMEC-UR-130	DECMEC-SR-37
DECMEC-UR-131	DECMEC-SR-69
DECMEC-UR-132	DECMEC-SR-66
DECMEC-UR-133	DECMEC-SR-60
DECMEC-UR-134	DPU/Ground responsibility
DECMEC-UR-135	DPU/Ground responsibility
DECMEC-UR-136	DPU/Ground responsibility

5 Model Description



The application may be split into 5 logical units :

- ***DPU Communication Handler*** shall handle all the communication with the DPU (reception and emission).
- The ***Command Handler*** shall handle the interpretation and execution of commands. Commands may be direct commands or included in sequences of commands.
- ***Detector Handler*** shall control the photoconducting and bolometer arrays. This unit shall also receive the data from the detectors and send them to the SPU processors.
- ***Mechanisms Handler*** shall control all the mechanisms and the temperature sensors.
- ***HK Handler*** shall handle the housekeeping data acquisition and the generation of HK packets.

Furthermore, all the units shall be organized around a shared memory area used by all the units to exchange information (configuration parameters and hk measures). This memory area shall also be accessible to the DPU through read/write commands.

From now on, we will detail more and more each of the logical unit to reach a description of all individual tasks of the software. On the diagrams, logical units will be represented by white bubbles while tasks will be represented by grey bubbles. In the text, identifier for logical units will be written in *italic* while identifier for tasks will be written in **bold.**

5.1.1 DPU Communication Handler

This logical unit shall shall handle the communication with the DPU. This unit is the interface between the *Command Handler* and the *HK Handler* on one side and the DPU on the other side. Some commands from the DPU (read/write) will also be executed directly in this task.

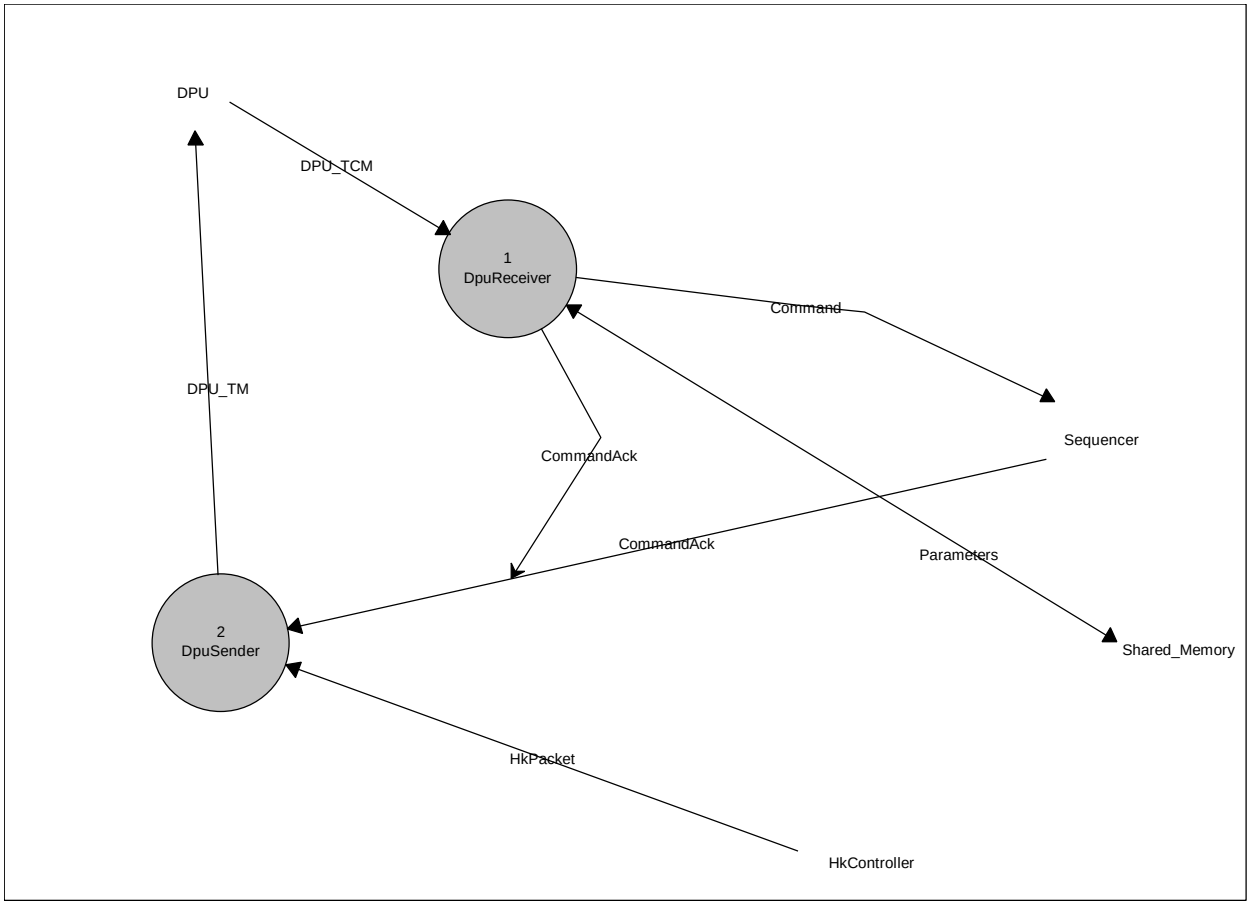
The *DPU Communication Handler* shall :

- Execute read/write commands
- Forward trigger commands and sequences of commands to the *Command Handler*.
- Forward commands acknowledgement to the DPU.
- Send HK Packets prepared by the *HK Handler*.

Since the communication protocol between DEC/MEC and DPU is asynchronous, two tasks are neeeded to handle the communication. Each task is responsible for one way of communication.

The **DpuSender** is presented in section 6.2.

The **DpuReceiver** is presented in section 6.3.



5.1.2 Detector Handler

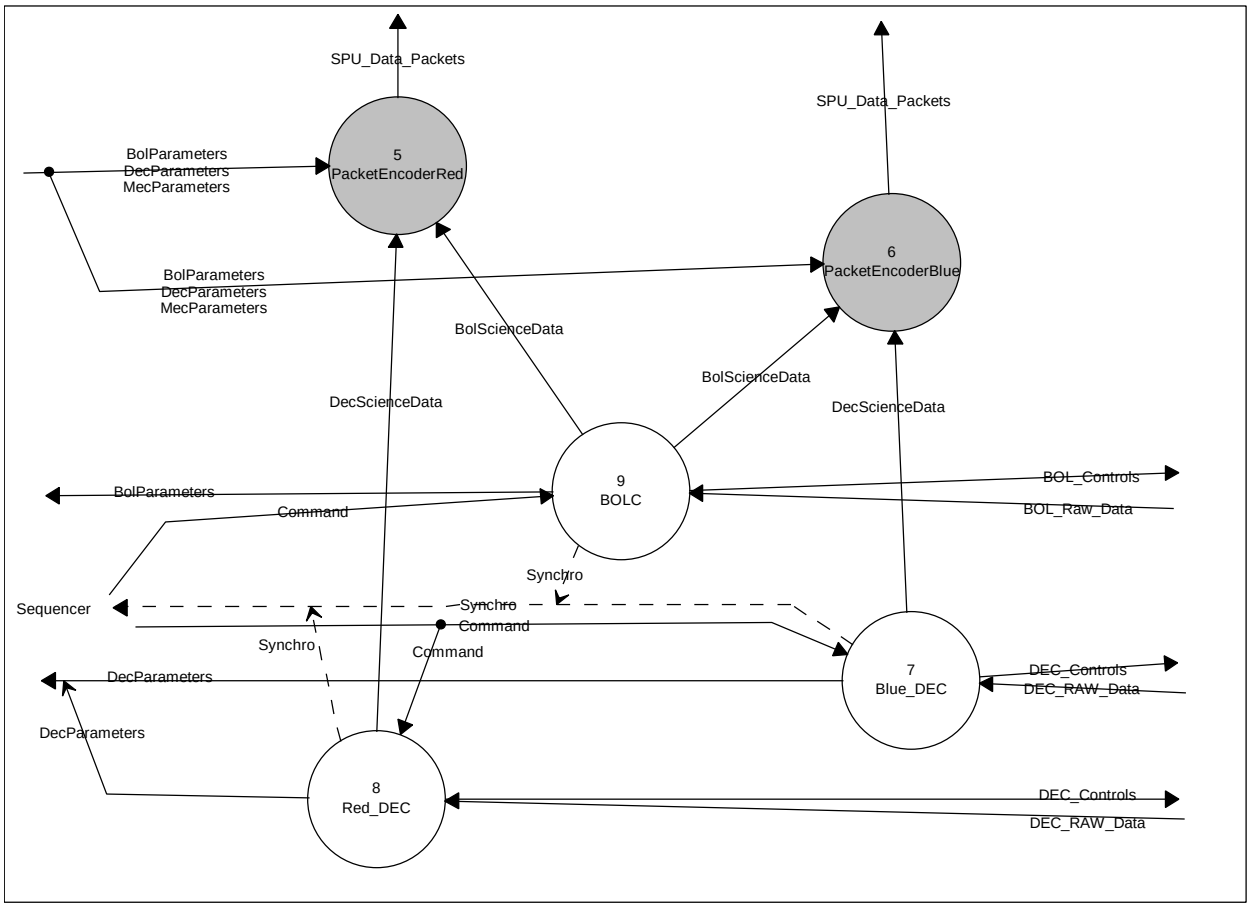
The *Detector Handler* unit shall be in charge of:

- sending commands to the electronic modules controlling the detector arrays
- receiving the science data and the housekeeping data.
- Formatting the data to send to the SPUs.
- Generating synchronisation events to be used by the *Command Handler*.

Two **PacketEncoder** tasks are responsible for the formation of packets and for sending them to the SPU. Each task is attached to one SPU. See section 6.13 for a deeper presentation of **PacketEncoder**.

Two units *DEC Handler* are responsible for the communication with the DECs. Each unit is attached to one DEC. They will be presented in more detail in section 5.1.3.

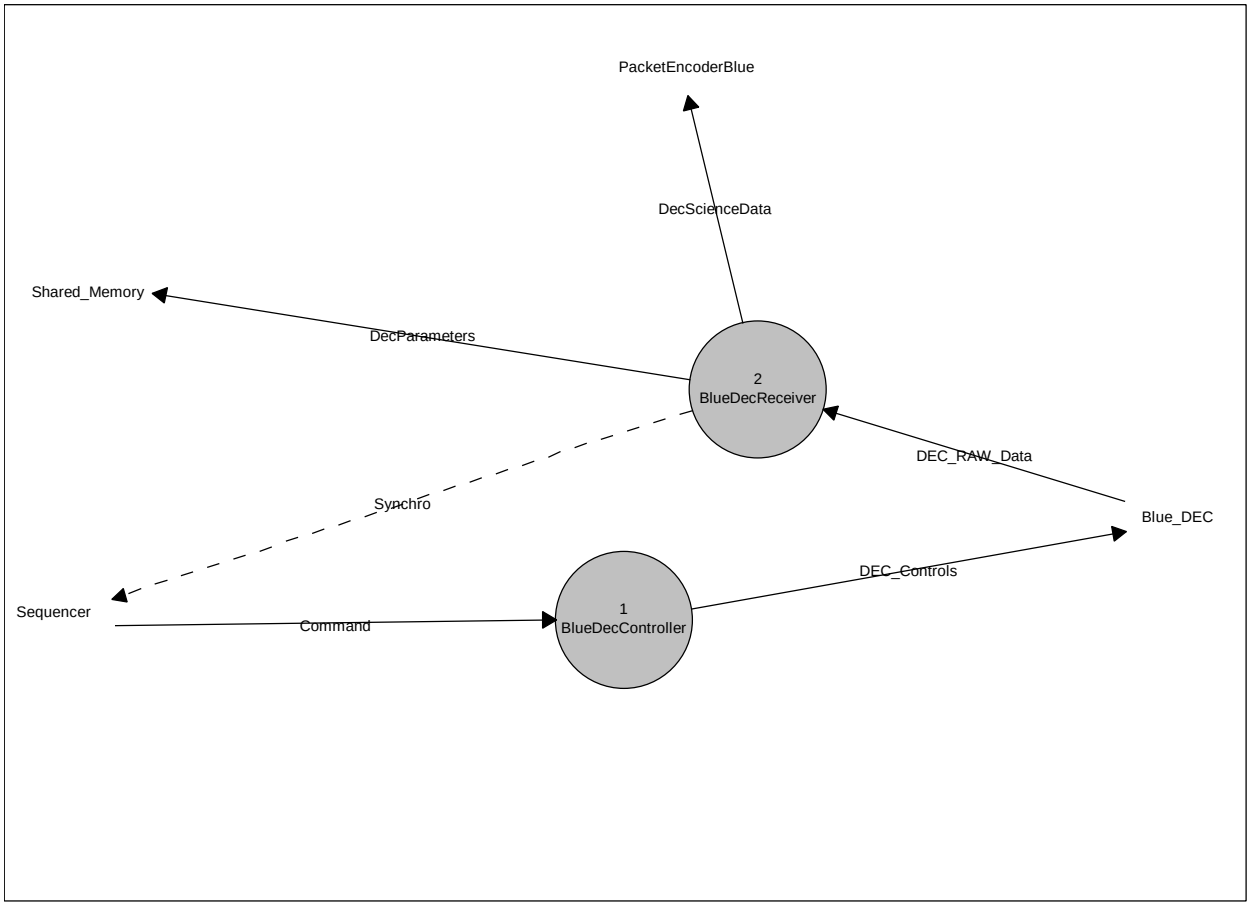
Since there is only one BOLC, there is only one unit *BOLC Handler*. This unit shall handle the communication with the BOLC. It will be presented in section 5.1.4.


SPU_Data_Packets
SPU_Data_Packets
BolParameters
DecParameters
MecParameters
5
PacketEncoderRed
6
PacketEncoderBlue
BolParameters
DecParameters
MecParameters
BolScienceData
BolScienceData
DecScienceData
DecScienceData
9
BOLC
BolParameters
Command
BOL_Controls
BOL_Raw_Data
Synchro
Sequencer
Synchro
Command
Synchro
Command
7
Blue_DEC
DEC_Controls
DEC_RAW_Data
DecParameters
DecParameters
8
Red_DEC
DEC_Controls
DEC_RAW_Data

5.1.3 DEC Handler

Each *DEC Handler* is split in two tasks (here, we consider only the *Blue DEC Handler*) :

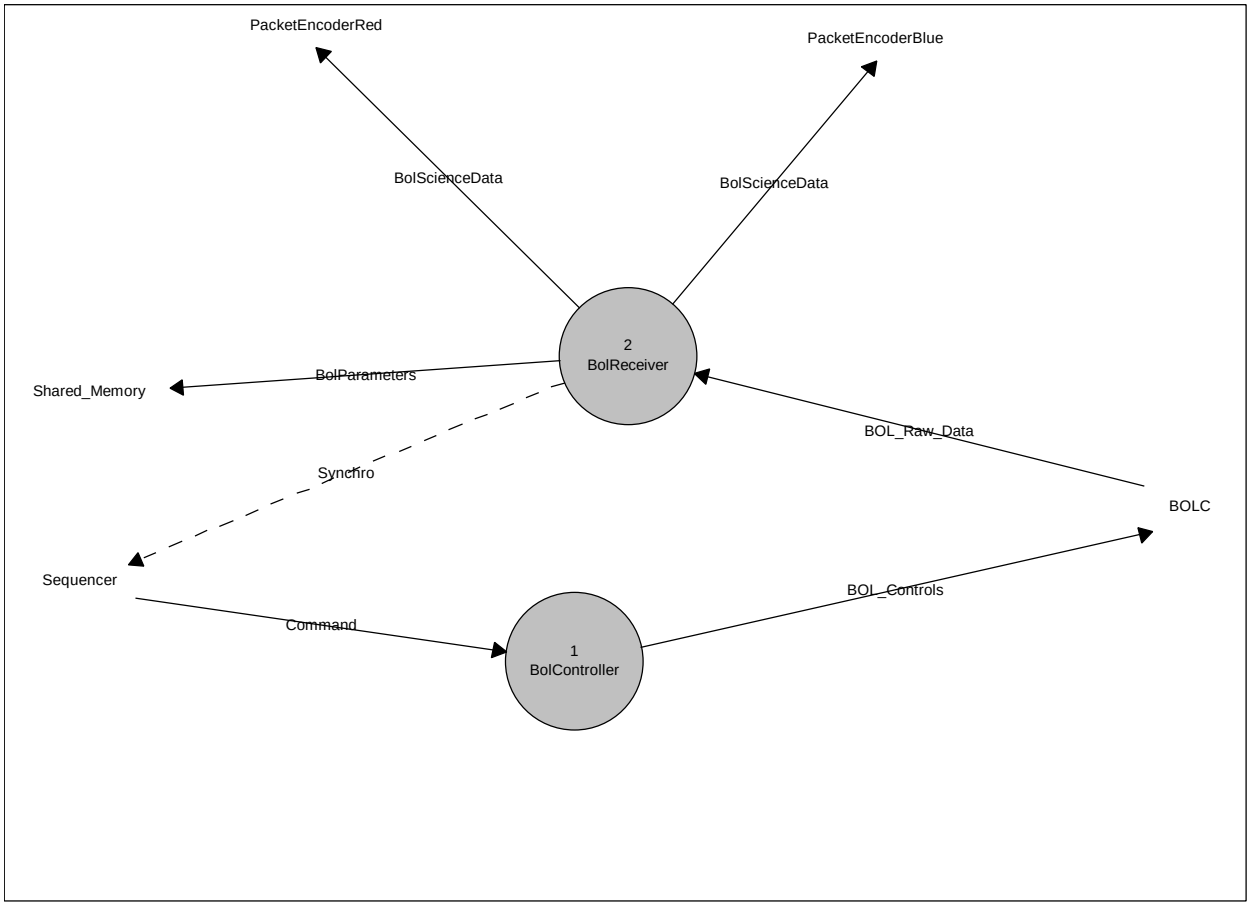
- **DecBlueReceiver** is receiving the raw data from the electronic module. The scientific data are forwarded to the **PacketEncode** while the housekeeping data are used to update a part of shared memory. **DecBlueReceiver** also generates the synchronisation event used by the *Command Handler*. See section 6.10 for a deeper presentation
- **DecBlueController** is forwarding the commands from the *Command Handler* to the electronic module. See section 6.9 for a deeper presentation.



5.1.4 BOLC Handler

There is only one *BOLC Handler*. It shall be implemented by a group of two tasks :

- **BolReceiver** shall receive the raw data from the BOLC. The scientific data shall be forwarded to the **PacketEncoders** while the housekeeping data shall be used to update the shared HK memory. **BolReceiver** shall also generate the synchronisation events used by the *Command Handler* to execute sequences. See section 6.12 for a deeper presentation
- **BolController** is forwarding the commands from the *Command Handler* to the electronic module. See section 6.11 for a deeper presentation.

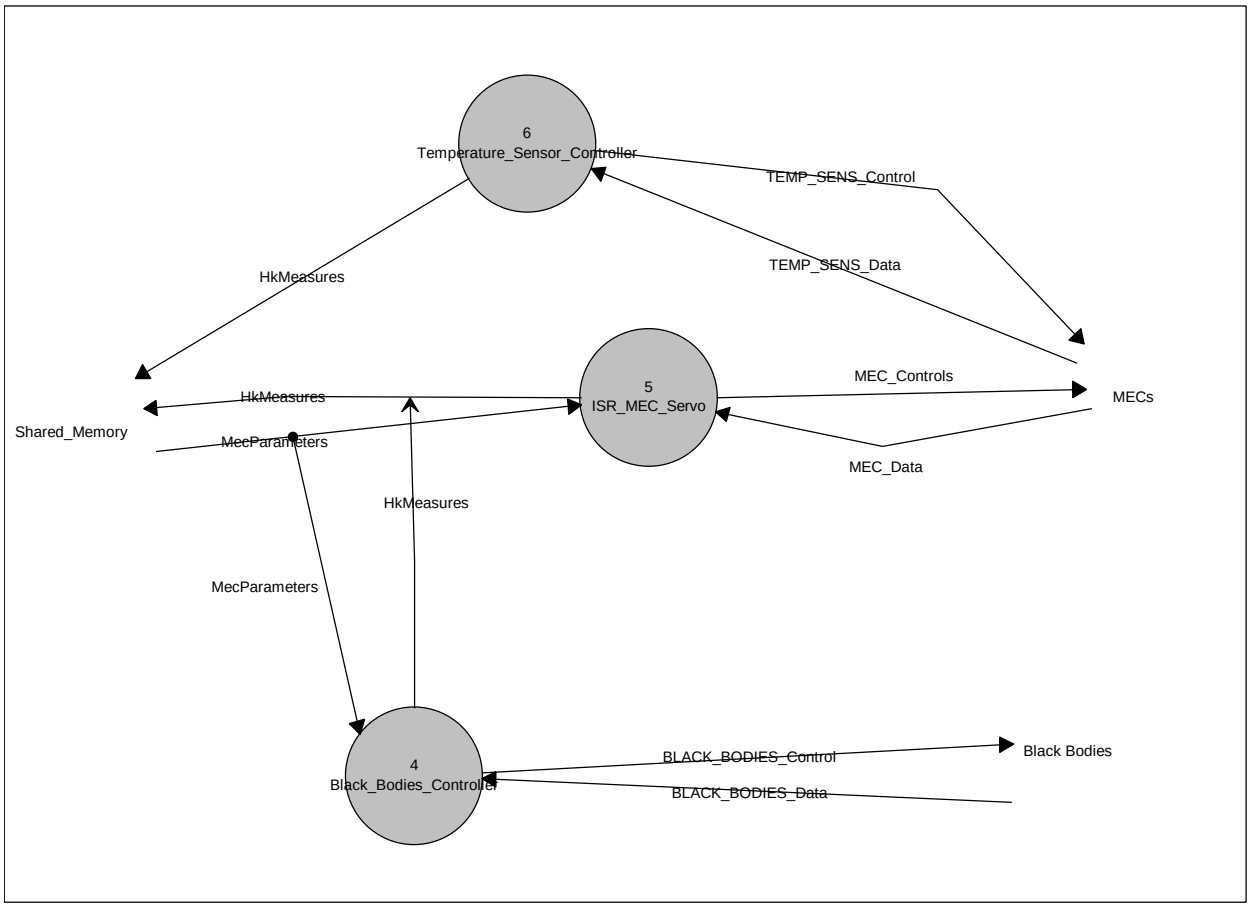


5.1.5 Mechanisms Handler

The *Mechanisms Handler* shall be implemented by a group of tasks controlling all the mechanisms (Grating, chopper, filter wheels) as well as controlling the black bodies and the temperature sensors.

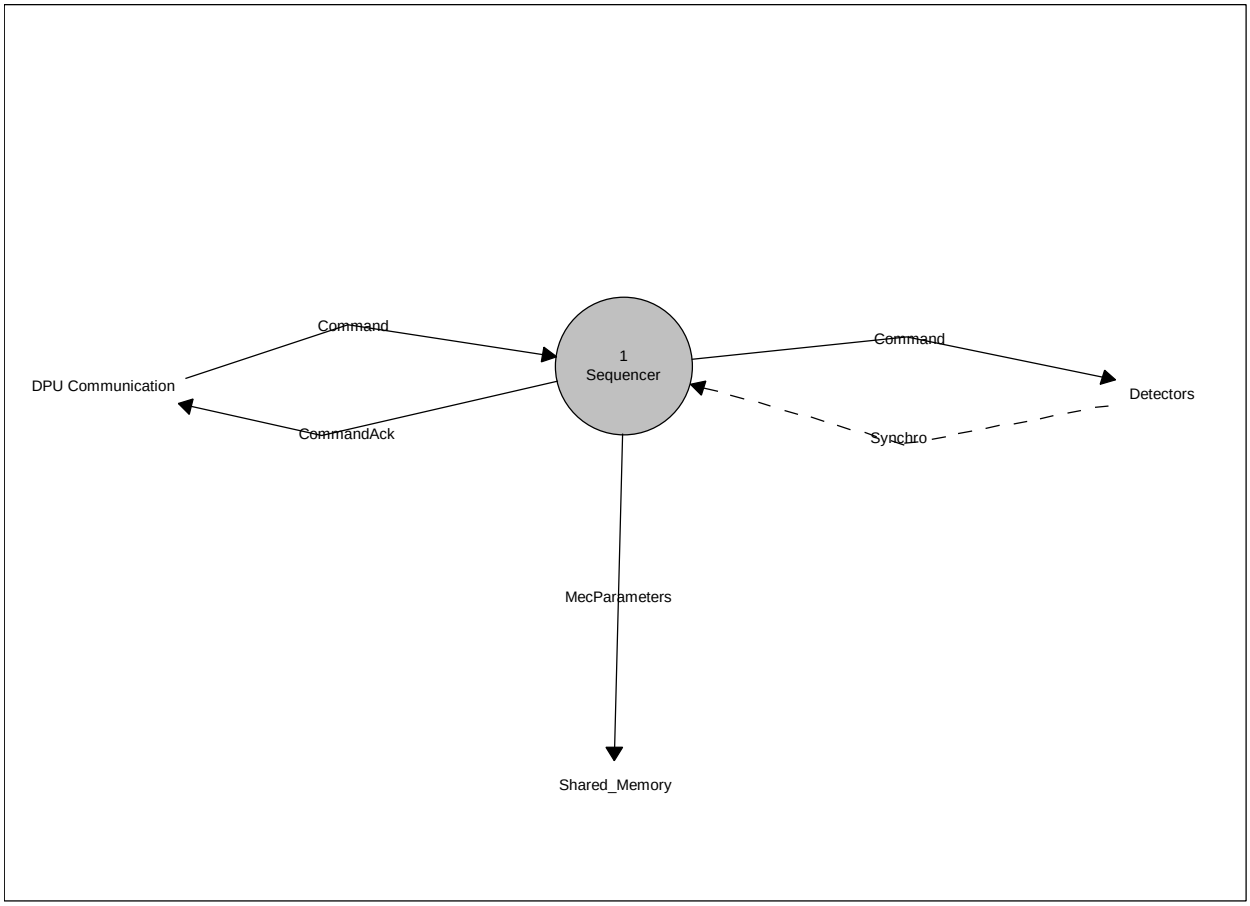
This group is made of :

- The **ISR MEC Servo**, see section 6.6
- One task for each of the black bodies (calibration sources) , see section 6.7.
- One task for the temperature sensor monitoring, see section 6.8



5.1.6 Command Handler

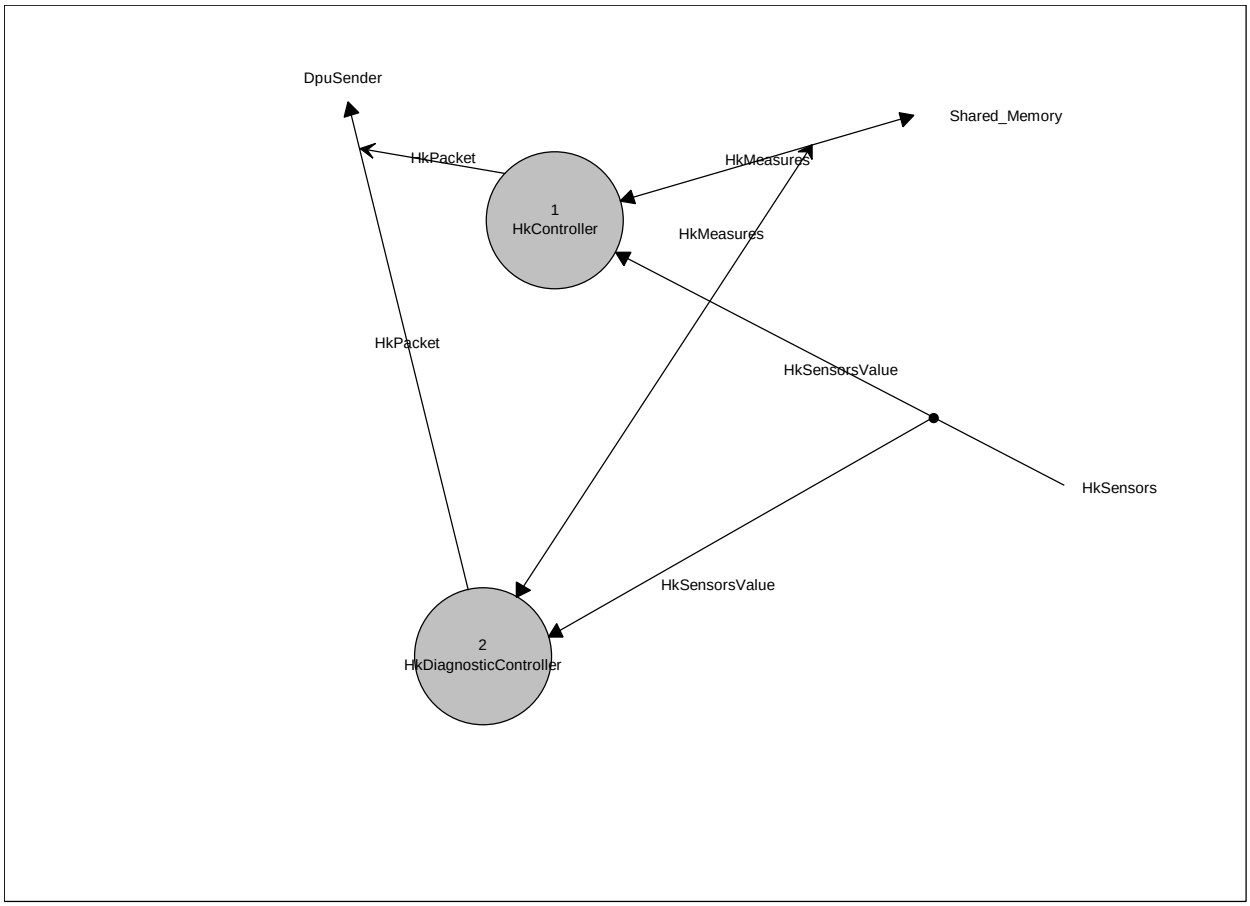
The *Command Handler* shall be implemented by a single task: the Sequencer. It is presented in section 6.



5.1.7 HK Handler

The *HK Handler* shall be implemented by a group of two tasks managing all the housekeeping.

- **HkController** shall handle the nominal housekeeping. It is presented in section 6.4.
- **HkDiagController** shall handle the diagnostic housekeeping. It is presented in section 6.5.



6 Software Design

This section will present briefly each of the task that will implement the on-board software. We will only give a small description of the functionality of each of the tasks and a few words about its implementation. The commands, the housekeeping and the interface are already documented in other documents [AD7][AD8][AD12] and the information will not be duplicated here.

The details of the implementation will be documented directly in the source code.

6.1 The Sequencer

This task manages the interpretation and execution of trigger commands and sequences of commands

According to the commands :

- some will be executed directly in this task;
- some will be dispatched to other task (DecControllers, MecControllers, HkController);

The Sequencer shall be able to interprete and execute all trigger commands defined in [AD12]

The Sequencer also performs the initialization of the application:

- timer creation
- install interrupt handlers

Note : the 1355 links are initialized in the task dedicated to the SMCS handling (section 6.14)

6.1.1 Implementation of the Sequencer main function

The sequencer is a cyclic task that is waiting for messages on its FIFO. Messages can be sent either by the DPU Receiver tasks as it receives a trigger command from the DPU or by one of the detector receiver tasks as it receives an end of ramp readout (spectroscopy) or a readout (photometry).

Executing a command means:

- Extract its ID and parameters
- Map the ID to a function containing the implementation of the command
- Call this function
- At the end of its execution, this function shall send an ACK/NACK (only if it is a trigger command outside a sequence).

6.2 DpuSender

This task is in charge of all the communication from DEC/MEC to DPU. It receives information from the DpuReceiver, the Sequencer and the HkControllers.

2 kinds of message can be sent :

- Command PACK/NACK issued by the DpuReceiver (for load/dump/check/write command) or by the Sequencer (for trigger commands).
- HkPacket issued by the HkControllers.

6.2.1 Implementation of DPU Sender main function

Each time one of the tasks must send a packet to the DPU, it writes a request in a FIFO (DpuSenderFifo). On its side, the DPU Sender reads the requests on the FIFO, formats the packets and sends them on the 1355 link to the DPU.

The DPU Sender is a cyclic task that waits for messages in its FIFO. As soon as it has received a message, it analyzes it and performs these actions:

6.2.1.1 Commands PACK/NACK

Format the packet and write it on the link to DPU.

6.2.1.2 HK Packets (nominal and diagnostic)

Format the packet, copy the data from the HkPacketBuffer into the packet, signal that the buffer is available and write the packet on the link to DPU.

6.2.1.3 DUMP ACK and DUMP ACK INTERMED

A dump command might generate more than one packet. Each time the DPU receiver has filled the dump buffer, the buffer is copied in the packet, the buffer is unlocked and the packet is sent.

6.3 DpuReceiver

DpuReceiver is in charge of all the communication from the DPU to DEC/MEC.

2 kinds of message can be received :

- Load/Dump/Check/Write commands. These commands are directly executed inside the DpuReceiver task. Addresses are NOT checked to prevent writing in not authorised area.
- Trigger commands. These commands are forwarded to the Sequencer.

6.3.1 Implementation of DPU Receiver main function

The DPU Receiver is a cyclic task that is always listening on the link 1355 connected to DPU. As soon as a packet is received, its header is analysed and the following actions are performed :

6.3.1.1 Trigger commands

Trigger ID, SID and parameters are extracted and then forwarded to the sequencer that will execute it.

6.3.1.2 DUMP commands

Length, memID and address are extracted. Memory to be dumped is split into 51 words blocks. For each block, the checksum is computed, the memory is copied into a temporary buffer and a message is sent to the DPU sender who will format the packet and send it to DPU. Once the packet has been received by DPU, DPU sender unlocks the temp buffer and DPU receiver can go on with the next block.

If the temp buffer is not unlocked within 1 sec (i.e because DPU was not ready to receive the packet), the error ERR_DPU_RECEIVER_TIME_OUT_ON_DUMP_BUFFER will be generated and the dump operation will stop.

6.3.1.3 LOAD commands

Length, memID (note: Only DRAM is accessible in the HLSW), checksum and address are extracted. A checksum is computed on the data received; if it is different from the checksum in the packet a NACK is sent to DPU sender that will forward it to DPU. Then, the data are copied into the destination area (inside a critical section to prevent other tasks to use these data during the copy). A checksum is computed on the data copied; if it is different from the checksum in the packet, a NACK is sent but the copied data remains in memory.

6.3.1.4 CHECK commands

Length, memID, checksum and address are extracted. Checksum is computed on the memory to be checked and then compared to the checksum in the packet. If they are different, a NACK is sent otherwise, a PACK is sent.

6.3.1.5 WRITE commands

ParamID, size and checksum are extracted. A checksum is computed on the data received; if it is different from the checksum in the packet a NACK is sent. The paramID is mapped on a memory address and the data are copied (inside a critical section to prevent other tasks to use these data during the copy). A checksum is computed on the data copied; if it is different from the checksum in the packet, a NACK is sent but the copied data remains in memory.

6.4 HkController

HkController is in charge of presenting to the DPU, at regular interval, a list of measurements taken at specific points in the instruments, and a table of status indicators for important instrument functions.

For this purpose, it collects the HK measures generated by the other tasks.

The content of HK Packets is predefined (see [AD12]) and can not be modified. It contains most of the available measurements. The frequency at which HK packet must be generated is fixed at 0.5Hz.

6.4.1 Implementation of HkController main function

The HkController is a cyclic task triggered by a timer that is started at the HLSW start-up.

The task is waiting for a semaphore to be signalled. Then, it waits for the hkBuffer to be unlocked by DPU Sender (this is done as soon as the previous packet has been received by DPU). Then, all acquisitions are performed; for most of the measures, it is as simple as copying a variable into the buffer. Since the error status of all tasks has been copied into the buffer, these status are reset to be able to signal new errors. Then, a message is sent to DPU Sender to send the HK packet to DPU.

6.5 HkDiagController

HkDiagController is in charge of presenting to the DPU, at regular interval, a list of measurements taken at specific points in the instruments, and a table of status indicators for important instrument functions.

For this purpose, it collects the HK measures generated by the other tasks.

The content of HkDiagPackets is defined by the HkDiagList that contains the ID of each of the measures required. This list is sent through a write command.

The frequency at which HkDiagPacket must be generated can be defined by the DPU in the trigger command *DMC_START_DIAG_HK*. For some special values of the frequency, the HkDiag acquisition shall be synchronised with the readouts of the DEC or BOLC.

6.5.1 Implementation of HkDiagController main function

The HkDiagController is a cyclic tasks triggered by a semaphore. The semaphore can either be signalled by a timer when user requests a fixed period or by other events such as a readout reception (it is then signalled by one of the detector receiver tasks).

Measures are stored in a temporary buffer. Each time the buffer reaches the maximum size of a hk packet, it is sent to DPU (via the DPU Sender) and the tasks starts to fill a new buffer.

Since the OBSID, BBID and TIME are used in the header of the packet to time stamp the measures, a new packet must be issued each time one of these parameters is changed.

6.6 MEC ISR Servo

The MEC ISR Servo is a separate piece of software ("low level", implemented in an Interrupt Service Routine) controlling the position of the chopper, the grating and the filter wheels.

The routine also contains a trajectory generator to implement the optimal move between set-points values.

The routine also manages the housekeeping data acquisition for the measurements done by MEC hardware (excluding the DEC and BOLC which have their own housekeeping) and the access to the hardware (except for the timing FPGA parameters)

The code will be validated by the hardware development team and supplied to the software team.

Note :

Since this task must be executed at very regular interval (and it must be tightly synchronized with readouts), it will be triggered by an interrupt generated by a hardware timer. So, the task will be implemented in an interrupt routine and will not be a 'task' as defined by Virtuoso. However, we will still use the denomination of task.

The hardware timer will be part of the logic implemented in the timing FPGA.

The nominal frequency will be 8192 Hz, but lower frequencies may be used in fallback modes.

6.6.1 Implementation of MEC ISR Servo

First of all, the HK acquisitions are performed. At each execution of the ISR, 4 channels are sampled: the chopper position, the 2 Hall sensors of the grating and sequentially one of the 48 analog HK channels.

Then, the ISR executes each of the mechanism control loop (if they are enabled). For each of them, it means :

- Compute the current setpoint (trajectory generator)
- Compute the command
- Write the command on the DACs

Furthermore, the ISR must synchronize the grating and chopper movement with the readouts. For this purpose, it must wait for the synchronization signal coming from the detectors, and wait for a programmable period before starting to modify the setpoint.

6.7 Calibration source controller

There are 2 Calibration source controllers, each one controlling the temperature of the corresponding source.

The calibration source controller is a cyclic task awaken every 500ms (the timing is computed by the MEC ISR Servo such that the task is awaken only when a set of measures are available)

The acquisition of the voltages and current related to the calibration sources are performed in the MEC ISR Servo. The computation of the resistor value is performed in this task.

If the PID is enabled, the task computes the output voltage to be applied to the source.

If the output voltage is lower than 1% of the full range, the task enters a "measure only" mode: No voltage is applied to the source such that it can cool down except for one small pulse every 20 seconds to be able to measure the resistor value.

If the output voltage is bigger, the task enters a "heating" mode. The output voltage is applied alternatively positive and negative to cancel the offset errors.

6.8 Temperature Sensors controller

There is temperature sensors controller task that monitors the 7 FPU temperature sensors.

This task is a cyclic task awaken every 500ms (the timing is computed by the MEC ISR Servo such that the task is awaken only when a set of measures are available)

This task slowly increases the current injected in the temperature sensors. As soon as one of the temperature sensors has reached 2mV, the current is inverted to have a positive and negative measure such that the offset errors are cancelled. Then, it continues to increase the current until all temperature sensors have been measured. Then, a new cycle starts again.

6.9 DecController

DecController is in charge of the communication from the DEC/MEC to the DEC. Most of the commands are sent to the DEC through a 1355 link except for the switch on/off commands that are sent through the MIM extension board. All the commands are issued by the Sequencer through a FIFO.

There is one DecController for each of the DEC.

6.9.1 Implementation of DEC Controller main function

The DEC Controller is a cyclic task that is always waiting for messages on its FIFO. These messages are issued by the Sequencer. The message received is interpreted and translated into messages that are sent to the DEC via the 1355 links or via MIM extension board.

The details of each of the commands are presented in the DDD.

6.10 DecReceiver

The DecReceiver is in charge of all the communication from the DEC to the DEC/MEC.

It receives raw data packets from the DEC. These packets contain Housekeeping information that are copied into the Housekeeping memory area. The science data are sent to the PacketEncoder.

There is one DecReceiver for each of the DEC.

6.10.1 Implementation of the DEC Receiver

The Dec Receiver is a cyclic task that is always listening for packets on its 1355 link. Once a packet is received, these actions should be performed:

- If a DEC detector simulator has been started, the science data should be replaced by the simulated one.
- If the DEC is the synchronization source for the sequencer, the sequencer should be notified when the last readout of a ramp has been received.
- If it has been configured so, the science packet should be forwarded to the PacketEncoder.

6.11 BolController

BolController is in charge of the communication from the DEC/MEC to the BOLC. All the commands are sent to the BOLC through a 1355 link. Note that the switch on/off is carried out by the spacecraft. All the commands are issued by the Sequencer through a FIFO.

Commands are composed of a 4-bit target identifier, a 4 bit parameter identifier; and an 8 bit parameter value. Commands are not interpreted inside DEC/MEC software; they are forwarded "as is" from DPU to BOLC.

6.11.1 Implementation of the BolController

The BolController is a cyclic task that is waiting for messages on its FIFO. These messages are always issued by the Sequencer and contain a command that must be forwarded to the BOLC.

6.12 BolReceiver

The BolReceiver is in charge of all the communication from the BOLC to the DEC/MEC.

It receives raw data packets from the BOLC. These packets contain Housekeeping information that are copied into the Housekeeping memory area. The science data are sent to the PacketEncoders.

6.12.1 Implementation of the BolReceiver

The BolReceiver is a cyclic task waiting for messages on its link 1355. Everytime a packet is received, these actions should be carried out:

- If a BOLC simulator has been started, the science data should be replaced by the simulated data
- If the packet is the first one of a readout and if the BOLC is the synchronization source for the sequencer, the sequencer should be notified.
- If the packet is a housekeeping packet, its content should be copied into the dedicated memory area.

6.13 PacketEncoder

The PacketEncoder is in charge of the communication from DEC/MEC to SPU.

It receives science data from the DecReceiver and from the BolReceiver. PacketEncoder adds Housekeeping information to the science data before sending the packet to the SPU.

There is one PacketEncoder for each of the SPU

6.13.1 Implementation of PacketEncoder

The PacketEncoder is a cyclic task waiting for messages on its FIFO. Messages are issued by the DecReceivers and the BolReceiver. The messages contain a pointer to a buffer that needs to be sent to SPU and the type of packet.

6.14 1355 Drivers

In the previous sections, we have seen that many tasks use the 1355 link to exchange data with other sub-units. In this section we will explain the organisation of the 1355 drivers.

The drivers shall be split in 3 parts:

- an interrupt routine that shall be triggered by the SMCS interrupt line
- a very fast task (SmcsDrvISR) that shall be driven by the interrupt to handle read/write operation
- a slower task (SmcsState) that shall handle the state-machine of the link

To avoid concurrent write-access on the SMCS chip registers, it has been decided that only one task shall write in the registers. The interrupt routine shall only read the interrupt pending register. SmcsDrvISR shall have a read and write access on all the registers and SmcsState shall use only information computed by SmcsDrvISR.

6.14.1 The interrupt routine

It shall be as simple as possible. It shall read the interrupt pending register, store the information in a memory location that shall be used by SmcsDrvISR, reset the interrupt and wake-up SmcsDrvISR.

6.14.2 SmcsDrvISR

It shall be implemented as a cyclic task awaken by the interrupt routine. This task shall handle 3 events:

- Write completed. It shall forward the event to the task that ordered the write operation

- Read completed. It shall forward the event to the task that ordered the read operation
- Parity/disconnect error. It shall signal the error in the housekeeping. If the error occurred on the link to DPU, it shall signal it to SmcsState.

6.14.3 SmcsState

This task shall initialize the communication link:

- Reset the SMCS chips
- Install interrupt handlers
- Start all links except those connected to DECs that shall be started only when DECs are switched on.

It shall be implemented as a cyclic task triggerd by:

- a timer every second to check the status of the links, or
- SmcsDrvISR when it has detected some error, or
- By any task requesting to start/stop a link 1355.

6.15 Software requirements – Component traceability matrix

Software Requirement	Component/remark
DECMEC-SR-1	N/A
DECMEC-SR-2	Sequencer
DECMEC-SR-3	SmcsState
DECMEC-SR-4	Sequencer + HkController
DECMEC-SR-5	SmcsState
DECMEC-SR-6	During nominal operation, DMC does not use any permanent storage that might be affected by an unexpected switch-off. However, switching off the DMC while it is writing in EEPROM (to apply a patch) may result in corrupted EEPROM content.
DECMEC-SR-7	DMC always use the same start-up procedure whatever the status was before switch-off. However, if the DMC has been switched-off while writing in EEPROM, it might not be able to execute nominally and it is recommended to write again the EEPROM via the SUSW.
DECMEC-SR-8	HkController
DECMEC-SR-9	HkController + DpuSender
DECMEC-SR-10	HkController
DECMEC-SR-11	HkDiagController
DECMEC-SR-12	DpuReceiver
DECMEC-SR-13	HkDiagController
DECMEC-SR-14	DpuReceiver + Sequencer
DECMEC-SR-15	DpuReceiver
DECMEC-SR-16	DpuReceiver + [AD7]
DECMEC-SR-17	DpuReceiver + [AD7]
DECMEC-SR-18	DpuReceiver + [AD7]
DECMEC-SR-19	DpuReceiver
DECMEC-SR-20	DpuReceiver + Sequencer
DECMEC-SR-21	DpuReceiver
DECMEC-SR-22	Sequencer
DECMEC-SR-23	Sequencer
DECMEC-SR-24	Sequencer
DECMEC-SR-25	Sequencer
DECMEC-SR-26	Sequencer
DECMEC-SR-27	Sequencer + DecReceiver + BolReceiver
DECMEC-SR-28	Sequencer + [AD12]
DECMEC-SR-29	Sequencer + [AD12]
DECMEC-SR-30	Sequencer + [AD12]
DECMEC-SR-31	Sequencer

DECMEC-SR-32	Not implemented, this is now a ground responsibility
DECMEC-SR-33	MEC ISR Servo
DECMEC-SR-34	Sequencer + [AD12]
DECMEC-SR-35	MEC ISR Servo
DECMEC-SR-36	MEC ISR Servo
DECMEC-SR-37	MEC ISR Servo
DECMEC-SR-38	MEC ISR Servo
DECMEC-SR-39	MEC ISR Servo + Sequencer
DECMEC-SR-40	MEC ISR Servo
DECMEC-SR-41	MEC ISR Servo
DECMEC-SR-42	MEC ISR Servo + Sequencer
DECMEC-SR-43	MEC ISR Servo + Sequencer
DECMEC-SR-44	MEC ISR Servo
DECMEC-SR-45	MEC ISR Servo
DECMEC-SR-46	MEC ISR Servo + Sequencer
DECMEC-SR-47	Calibration source controller
DECMEC-SR-48	Calibration source controller
DECMEC-SR-49	MEC ISR Servo
DECMEC-SR-50	MEC ISR Servo
DECMEC-SR-51	MEC ISR Servo
DECMEC-SR-52	MEC ISR Servo + Sequencer
DECMEC-SR-53	DEC Controller
DECMEC-SR-54	DEC Controller + Sequencer
DECMEC-SR-55	DEC Controller + Sequencer
DECMEC-SR-56	DEC Receiver
DECMEC-SR-57	DEC Receiver + Sequencer
DECMEC-SR-58	DEC Receiver + HK Controller
DECMEC-SR-59	DEC Receiver + Packet Encoder
DECMEC-SR-60	DEC Receiver + Packet Encoder
DECMEC-SR-61	DEC Receiver
DECMEC-SR-62	DEC Receiver
DECMEC-SR-63	BOL Controller
DECMEC-SR-63	BOL Controller
DECMEC-SR-64	BOL Receiver
DECMEC-SR-66	BOL Receiver + Packet Encoder
DECMEC-SR-67	BOL Receiver + Sequencer
DECMEC-SR-68	BOL Receiver
DECMEC-SR-69	BOL Receiver
DECMEC-SR-70	SmcsDrvIsr + SmcsState
DECMEC-SR-71	SmcsDrvIsr + SmcsState
DECMEC-SR-72	SmcsDrvIsr + SmcsState
DECMEC-SR-73	N/A

DECMEC-SR-74	N/A
DECMEC-SR-75	N/A
DECMEC-SR-76	N/A
DECMEC-SR-77	See [AD12]
DECMEC-SR-78	See [AD12]
DECMEC-SR-79	N/A
DECMEC-SR-80	N/A
DECMEC-SR-81	N/A
DECMEC-SR-82	N/A