Title: **Software Design Document**

Document No: IN-IM-TUB-SDD-001

Issue: 7.0 draft

Date: June 2000

Prepared by: R. Volkmer
E. Göhler
M. Stuhlinger
P. Risse

Checked by:

Approved by:

CHANGE - RECORD

Issue	Date	Sheet	Description of change
1.0	June	all	First issue
4.0	October		Timing, Table handling, Typos
4.3	October		Chapter 6 and 7: TM and TC Updated
5.0	January 99	10	Automatism
		17	TM handling
		22	HEPI context saving
		24	Mode diagram
		26	Patch and Dump
		30	Specification of Histograms
		35	On event messages
		37	ADA Tasks
		39	Memory Mapping
		43	Mode Dispatcher
		44	Science Mode
		87	HK
		90	Additional HK
6.0	August 99	All	SDD separated into SRD(chapter 1,2,6,7) and SDD (chapter 3-5)
7.0	June 2000		All (new structure)
			File: sdd-ad.doc Last change: 02.06.2000 17:37

TABLE OF CONTENTS

TABLE OF FIGURES

TABLE OF TABLES

1 INTRODUCTION

1.1 SCOPE

The scope of this document is to describe the design of the software part of the IBIS IMAGER Data Handling System (DHS), written in ADA 83.

1.2 CONTENT

This document contains the general design of the IBIS instrument application software (IASW) which runs on the data processing engines (nominal and redundandunt) of IBIS. It describes the structure and the internal interfaces of the IASW. It does **not** describe the specification nor the implementation of the algorithms of the IASW.

The document is articulated in 4 sections. The first section is the introduction in which is the scope, the description, definition and acronyms and the references included.
Section 2 contains the software architecture in respect of tasks/interfaces and general requirements.
In section 3 the software design is presented by specifying each software module separately in functional and formal terms.
The appendix in section 4 contains parameter structures, the list of telemetry packets and telecommands.

1.3 APPLICABLE AND REFERENCE DOCUMENTS

1.3.1 APPLICABLE DOCUMENTS

AD.1: EID-A rev5
AD.2: DPE HW Design Decription, INT-DD-CRS-0001,Is.1

1.3.2 REFERENCE DOCUMENTS

RD 1: HEPI Interface Description, IN-IM-TUB-TN/EL-018, Is. 4.1
RD 2: IBIS Communication Protocol Definition, IN-IM-TUB-ICD-01, Is. 1
RD 3: HEPI Design Description, IN-IM-TUB-DES-001, Is 5.1
RD 4: Software I/F Control Document, INT-IC-GMV-0001 Is. 3
RD 5: Integral Packet structure Definition, INT-RP-AI-0030, Is.04
RD 6: IBIS IASW User manual, IN-IM-TUB-UM-001, Is. 1.4
RD 7: Reference Document for the TLD ADA compiler MIL-STD-1750 Target, Rev 2G, 15 March 1996
RD 8: Reference Document for the TLD macro assembler MIL-STD-1750 Target, Rev 2F, 15 March 1996
RD 9: Reference Document for the TLD extended memory linker MIL-STD-1750 Target, Rev 3J, 15 March 1996

2 SOFTWARE ARCHITECTURE

2.1 TERMS

2.1.1 TASK

The IBIS IASW consists of some processes running in an preemptive multitasking environment supported by the CSSW. These processes are refered in following as **tasks.** Event though they mostly fullfill a specific function the term **task** should not be applied as this function but the entity of modules running as a single process.

2.1.2 TYPES OF MODULES

The IBIS IASW consists of modules which fall under different categories:

1. **Interface modules** which has to form the link between CSSW and data processing parts of the IASW. They are not necessarily part of a specific task but may be used from different task.
2. **Task modules** which own only one specific task.
3. **Shared modules** which neither form a interface nor belong to a specific task. These modules provide different tasks with internal algorithms and data.

2.1.3 TASK ENTRY POINTS

Each task is started by calling a procedure called the **task entry point**. This procedure contains a **task loop** which must not end.

2.2 GENERAL REQUIREMENTS

2.2.1 PROGRAMMING LANGUAGE

The SW shall be coded in ADA 83, TLD COMPILER Version V-5.7.L.
Time critical functions could be written in assembler Code according 1750A assembler code with TLD ASSEMBLER.

2.2.2 RESOURCE REQUIREMENTS

CPU time:	12095 out of 125 per RTC
Memory space:	ROM size: 48 kWord
	RAM 960 kWord

One Word consists of 16 Bit.

TC handling: 8 per seconds
TM generation: 136 packets / 8sec nominal case (184 packets / 8sec max.)

2.3 MULTITASKING

2.3.1 IASW USES MULTITASKING

The IBIS IASW uses multitasking to both reakt on specific events and process data at same time.data at same time.

2.3.2 TYPE OF MULTITASKING USED

2.3.2.1 Priority

The IASW runs in priority based preemptive multitasking.
Each task has its own priority.
Different priorities will ensure that the behaviour does not depend on the timing granularity of the multitasking system but only on the necessity to process certain events.
Round-robin multitasking feature therefore will not be used.

2.3.2.2 Priority allocation

The priority of the tasks must be set under following constraints:

1. Commandability and collection of status information has highest priority.
2. Cyclic tasks should have high priority (to avoid exceeding their period).
3. Tasks running longer than 21 msec (1 TM window) per cycle must have a priority less than 7.

Requirement 3 is ensures that the CSSW will not be blocked from sending the TM properly which is done with the TM_CYCLE-Task running at priority 7. Refer RD 4, 7.1.
This ensures also that no watchdog event may occur because the DPE watchdog register is reseted by the TM_CYCLE task.

2.3.2.3 Cyclic tasks

Tasks may be cyclic or acyclic depending whether they are using the CSSW service of defining a period or not.

2.3.2.4 Cyclic tasks exceeding period

Cyclic tasks must not exceed their period (Refer RD 4, 3.2.3.1.3.8). If this may happen the cyclic task should become acyclic by deleting its period (refer RD 4, 3.2.3.1.310).

For safety reasons all tasks which have to perform a periodic job should be made cyclic because the CSSW checks whether these tasks exceed their period or not (RD 4, 3.2.3.1.3.8).

2.3.2.5 Period of cyclic tasks

All cyclic tasks have a period of 1 RTC.

The service of defining a period of multiple RTC's will not be used because there is no task which may wait for a long time before performing its job. The only task which may use the period would be the HK collecting task but just this one is synchronized with a BCP1 H/W event to avoid phase shift.

2.3.2.6 Dead-Lock Prevention

Under no circumstances a situation must occur that tasks A waits for an event from task B while task B waits for an event from task A, or any wider wait-event cycle may happen for a set of tasks. To avoid this situation different techniques were developed. The simplest approach is as follows:

The tasks must be designed that:

1. For all events any task may wait for exists a common, non-cyclic chain of inter-task communications.
2. No task (except the one with lowest priority) performs active polling.
3. The CPU budget is set so that event the task with lowest priority will become ready periodically.

The condition 2 and 3 ensures that tasks may block each other because a high-priority task polls actively for a flag/event which will be sent from a low-priority task which never will become running.

2.3.3 LIST OF TASKS

Task Name	Task ID	Priority	Cyclic
Main Task	1	3	No
TC Task	2	11	No
HBR A Task	3	8	No
HK Task	4	12	Yes (but without period)
Histogram Task	5	10	Yes (may become acyclic)
Read Histogram Task	6	4	Yes (may become acyclic)

Table 1: List of IASW Tasks

2.3.4 DESCRIPTION OF IASW TASKS

Each task will be described as a general entity in terms of behaviour, timing and resource requirement.

2.3.4.1 Main Task

2.3.4.1.1 Functionality

The main task defines in which mode the IASW run (RD 6, 2.2).
According current mode the task performs:

Standby Mode	• Context save/restore automatism reactions (RD 6, • HV VETO off automatism reaction
Diagnostic Mode	• Reading PPM from HEPI • Goto Standby mode automatism reaction
Scienctific Mode	• Initialize HEPI • Set HEPI science functions of PPM • Read PPM from HEPI, link them down • Goto Standby mode automatism reaction • Resynchronize HEPI due wrong/low data rata

Table 2: List of Main task functionality

2.3.4.1.2 Initialisation

2.3.4.1.2.1 Connection to events:

The main task will connect to following CSSW events:

- **BCP4** (HEPI synchronisation)
- **LSL End of Transmission** (LSSL access)
- **LSL End of Reception** (LSSL access)
- **LSL Parity Error** (LSSL access)
- **LSL Overrun Error** (LSSL access)
- **Timer A** (Waiting 10msec)

2.3.4.1.2.2 Init state

The main task will start when CSSW goes to nominal mode.

2.3.4.1.2.3 Task entry point

The task entry point will be the procedure INIT in module MODE_DISPATCHER.

2.3.4.1.3 Timing behaviour

The main task is not cyclic. Data processing in diagnostic/scientific mode will be performed by polling the input buffer which will result in maximum CPU request.
If no data are available the main task shall wait 10msec till polling for new data.

In standby mode a wait of 1 RTC between two operations shall be performed.

2.3.4.1.4 Interfaces

2.3.4.1.4.1 Inter-Task communication:

2.3.4.1.4.1.1 Main task recepts:

- The **TC Task** will set the mode and will distribute the status according broadcast packet.
- **HK Task** will read the mode and event processingcounters.
- **HK Task** will set the HEPI resynchronisation flag in case of low/bad event rate.
- **HBR A TASK** information that a buffer is filled with PPM's.

2.3.4.1.4.1.2 Main task sends:

- Start/Stop of processing histograms to **Histogram Task**.
- Start/Stop of reading histograms to the **Read Histogram Task**
- Signal to **HBR A Task** to start reading of PPM's from HEPI.

2.3.4.1.4.2 Interface Modules

TBW.

2.3.4.2 TC Task / processing of Telecommands

2.3.4.2.1 Functionality

The TC task waits for telecommands distributed from CSSW. Each recognized telecommand will be processed provided it is correct in parameter range and size and is allowed in current mode.

2.3.4.2.2 Used types of telecommands

Name	Type	Subtype
Pulse_Command	2	1
Start_Task	5	1
Stop_Task	5	2
Load_Task_Parameter	5	3
Report_Task_Parameter	5	4
Mode_Transition	5	5
Report_Specific_TM	9	1
Enable_Specific_TM	9	4
Disable_Specific_TM	9	5

Table 3: List of used telecommand types/subtypes

2.3.4.2.3 Mode dependence

The acceptance of a TC depends on its type/subtype combination which may be allowed/denied in a certain mode. No other criteria is used to accept/reject telecommands depending on mode.

Mode	Type	Subtype	Allowed
Standby	2	1	Yes
	5	1	Yes
	5	2	Yes
	5	3	Yes
	5	4	Yes
	5	5	Yes
	9	1	Yes
	9	4	Yes
	9	5	Yes
Diagnostic	2	1	Yes
	5	1	Yes
	5	2	Yes
	5	3	Yes
	5	4	Yes
	5	5	Yes
	9	1	Yes
	9	4	Yes
	9	5	Yes
Science Standard	2	1	No
	5	1	No
	5	2	No
	5	3	No
	5	4	No
	5	5	Yes
	9	1	No
	9	4	No
	9	5	No
Science PPM	2	1	No
	5	1	No
	5	2	No
	5	3	No
	5	4	No
	5	5	Yes
	9	1	No
	9	4	No
	9	5	No
Polarimetry	2	1	No
	5	1	No
	5	2	No
	5	3	No
	5	4	No
	5	5	Yes
	9	1	No

	9	4	No
	9	5	No

Table 4: List of telecommands allowed in which mode

2.3.4.2.4 Initialisation

2.3.4.2.4.1 Connection to events:

The tc task will connect to following CSSW events:

- **LSL End of Transmission** (LSSL access)
- **LSL End of Reception** (LSSL access)
- **LSL Parity Error** (LSSL access)
- **LSL Overrun Error** (LSSL access)

2.3.4.2.4.2 Init state

The TC task will start when CSSW goes to nominal mode.

2.3.4.2.4.3 Task entry point

The task entry point will be the procedure START in module TC_DECODER.

2.3.4.2.5 Timing behaviour

The TC task has almost highest priority and is asynchronous. The task waits passive to receive telecommands from CSSW and will wake up only when a telecommand is pending. Because of rejection of telecommands in science mode the most CPU budget may be spent in standby/diagnostic mode only.

2.3.4.2.6 TC Timeout

The maximum duration waiting for telecommands before a timeout is reported is the maximum allowable RTC ticks value.

2.3.4.2.7 Interfaces

2.3.4.2.7.1 Inter-Task communication:

2.3.4.2.7.1.1 TC task recepts:

No signal.

2.3.4.2.7.1.2 TC task sends:

- Mode to **Main Task**
- Broadcast information **to Main Task, Histogram Task, Read Histogram Task, HK Task**.
- Start/Stop of processing histograms to **Histogram Task**.
- Start/Stop of reading histograms to the **Read Histogram Task.**

- Automatism state to **Main Task, Histogram Task, Read Histogram Task, HK Task.**

2.3.4.2.7.2 Interface Modules

TBW.

2.3.4.3 HBR A Task

2.3.4.3.1 Functionality

The **HBR A Task** initiates reading PPM events from HEPI via HBR A to DPE and checks the correct reception. Also the PPM buffer handling is managed by this task to allow reading and processing of PPM's in parallel.

2.3.4.3.2 Initialisation

2.3.4.3.2.1 Connection to events:

The HBR A task will connect to following CSSW events:

- **HSSL End of Reception** (HSSL access)
- **HSSL FIFO Overflow error** (HSSL access)

2.3.4.3.2.2 Init state

The HBR A task will start when CSSW goes to nominal mode.

2.3.4.3.2.3 Task entry point

The task entry point will be the procedure in INITIALISE module HBR_A_IF.

2.3.4.3.3 Timing behaviour

The task is asynchronous and will be activated when the main task asks for PPM. The CPU budget of this task is low for only start will be triggered and a wait to stop event occurs. This may happen in diagnostic/scientific modes only. In standby mode the task will not be used.

2.3.4.3.4 Interfaces

2.3.4.3.4.1 Inter-Task communication:

2.3.4.3.4.1.1 HBR A task recepts:

- S/W event from **Main Task** when buffer of PPM is requested.
- **Main Task** sets flag that buffer is read out.

2.3.4.3.4.1.2 HBR A task sends:

- Sets flag when buffer is available for **Main Task**.

2.3.4.3.4.2 Interface Modules

TBW.

2.3.4.4 Histogram Task

2.3.4.4.1 Functionality

The histogram task manages the starting and stopping of histogram integration in HEPI for PiCsIT events. Also this task aktivates reading of last integrated histogram which will be performed with the Read Histogram Task.

Integration of histograms will be performed in scientific mode only. The conditions that a histogram what type soever will be integrated are defined in RD 6, sect. 3.2.5.1.

2.3.4.4.2 Initialisation

2.3.4.4.2.1 Connection to events:

The Histogram Task will connect to following CSSW events:

- **LSL End of Transmission** (LSSL access)
- **LSL End of Reception** (LSSL access)
- **LSL Parity Error** (LSSL access)
- **LSL Overrun Error** (LSSL access)

2.3.4.4.2.2 Init state

The Histogram Task will not start when CSSW goes to nominal mode.
The task will start when the main task goes to scientific mode.
The task will stop himself when main task leaves scientific mode.

2.3.4.4.2.3 Task entry point

The task entry point will be the procedure NITIALISE in module HBR_B_IF.

Remark: The module name HBR B IF was chosen to reflect the similarity of HEPI interface lines HBR A and B (PPM/histogram blocks). During design process it turned out that processing of histograms needs two tasks for one must track the timing while integrating while the other task reads out and processes the histogram blocks. The modul name remained but its functionallity is not any more the access of the HBR B line.

2.3.4.4.3 Timing behaviour

The Histogram Task is cyclic with a period of 1 RTC (125msec).

In each period the task checks whether one of the start/stop conditions occurred and performs the actions (acessing HEPI to enable/disable certain histogram buffers). During the access of HEPI the task is made non-cyclic by deleting its period to avoid a task overrun condition.

2.3.4.4.4 Interfaces

2.3.4.4.4.1 Inter-Task communication:

2.3.4.4.4.1.1 Histogram task recepts:

- A start/stop task command from **Main Task** when entering/leaving scientific mode to start/stop.
- The satellite status from **TC Task** via a broadcast packet (needed to determine point/slew state and on-target flag).

2.3.4.4.4.1.2 Histogram task sends:

- A signal to the **Read Histogram Task** to start reading of indicated buffers.

2.3.4.4.4.2 Interface Modules

TBW.

2.3.4.5 Read Histogram Task

2.3.4.5.1 Functionality

The Read Histogram Task performs the actual reading of blocks from HEPI, compressing them and link the data down as telemetry packets.
The conditions for reading histogram blocks for the different histogram types are defined in RD 6, sect. 3.2.5.3.
Both types of histograms - science histograms (CsI single/multiple and polarimetry histograms) and calibration histograms – are read blockwise. Because the telemetry rate should be constant the reading and processing of histogram blocks is distributed over the entire reading process of a histogram. According RD 6, 3.2.5.3.1, the reading time shall be the expected integration time of the following histogram (of same type) less a time margin **TBD**.

Reading histogram blocks contains following steps:

1. Send a request histogram block to HEPI with the block address.
2. Wait at least **TBD** msec.
3. Check, whether the block is available.
4. Read the histogram block via HBR B.
5. Compress the block, link down the telemetry containing its data.

2.3.4.5.2 Initialisation

2.3.4.5.2.1 Connection to events:

The Read Histogram Task will connect to following CSSW events:

- **LSL End of Transmission** (LSSL access)
- **LSL End of Reception** (LSSL access)
- **LSL Parity Error** (LSSL access)
- **LSL Overrun Error** (LSSL access)
- **HSSL End of Reception** (HSSL access)
- **HSSL FIFO Overflow error** (HSSL access)

2.3.4.5.2.2 Init state

The Read Histogram Task will start when CSSW goes to nominal mode.

2.3.4.5.2.3 Task entry point

The task entry point will be the procedure INITIALISE in module READ_HISTOGRAM.

2.3.4.5.3 Timing behaviour

The read histogram task is cyclic with a period of 1 RTC (125 msec).
The scheme is as follows:

- Check, whether block is needed to be read out (according time/last block read)
- If so, send block request command to HEPI
- wait till next period
- Read/process block.

The main CPU budget is the processing of blocks which may last up to 65msec.

2.3.4.5.4 Interfaces

2.3.4.5.4.1 Inter-Task communication:

2.3.4.5.4.1.1 Read Histogram task recepts:

- A signal from **Histogram Task/TC Task** when a certain histogram should be read out.
- The satellite status from **TC Task** via a broadcast packet (needed to determine the hand-over flag).

2.3.4.5.4.1.2 Read Histogram task sends:

No signal.

2.3.4.5.4.2 Interface Modules

TBW.

2.4 INTERFACE MODULES

Interface modules are those modules whose routines belong not to a single task only and/or who supply services to either access external devices or share common informations/algorithms.

2.4.1 CONSIDERATIONS

2.4.1.1 Reentrability

If some interface routines will be accesses from different task which run in parallel it must be made shure that the functional result of this routine does not change when it is called at same time from different tasks. To achieve this different approaches may be used:

1. If a resource is needed which allows only one access a time a semaphore concept must be used.
2. No routine should set global variables (except as defined in 1.).
3. Reading global variables will be allowed if the read process is a single atomic access.

2.4.1.2 Mutual exclusion

2.4.1.2.1 Semaphores

Access to single access resources will be controlled by following semaphore mechanisms:

1. Blocking will be controlled by an binary semaphore variable which will be read and set/unset in a crititcal region.
2. If the resource is blocked (semaphore variable is set) the task waits for a specific “free” message which will be sent when the resource is unblocked.
3. Also the task ID of the blocking task will be stored.

According general principles a message is sufficient to control the blocking mechanism by sending the message each time the resource will become unblocked. The additional variables are used to ensure that the design is implemented properly by checking:

- *The block-function will not be called by the same task in nested situations.*
- *The resource does not remain blocked by checking whether the same task is blocking the resource twice.*
- *It could be checked by the interface modules that a specific task indeed blocks the given resource.*

2.4.1.2.2 Access by an announce/confirm mechanism

For specific resources (access of LSSL in Main Task) it is necessary that the task needs not wait till the resource becomes free. For this a different approach will be implemented:

- The task announces an access to the resource (LSSL) which sets a flag.
- If the resource becomes unblocked the annoucing task gets access to the announced resource.
- The task check this situation by polling a function (confirm) which returns true when the access to the resource is granted.

2.4.2 TYPES OF INTERFACES

2.4.2.1 Detector Interfaces

TBW

2.4.2.2 TM Interface

TBW

2.4.2.3 Status Informations

TBW

2.4.2.4 Tables

TBW

2.4.2.5 Algorithms

TBW

2.5 MEMORY MAPPING

The 1750A structure allows 64KWord each data and instructions at same time. To enhance the memory space it is possible to reprogramm the memory controller in units of 4kWord pages. The mapping of physical and logical pages could be set as desired.
The TLD ADA compiler/linker supports the enhanced memory by swapping the set of accessible pages each time. Changing this set (also refered as ***adress state****) the is time consuming so the change of addressstates should occur as seldom as possible. See also RD 9, sect 3.4.*

2.5.1 OVERVIEW

2.5.1.1 Data exchange between different address states

If the address state will change no variables are accessible except those parts of data which are allocated in pages which do not change physically when changing the address state (address state transition). The address states are numbered, starting from 0.

2.5.1.1.1 Global variables

No global variables must be accessed across different address states.

Exception:
Global variables which must be common for different address states may be defined in module IASW_COMMON which will be allocated at the same physical page for all address states.

2.5.1.1.2 Transfer of data between address states

1. The stack must be allocated at common physical pages for both source and destination addess states for all address state transitions.
2. All variables transfered between different address states and are accessed via a VAR clause must be allocated on stack .
3. All variables transfered between different address states and greater than 1 Word must be allocated on stack.
4. Arguments of procedures which hase size 1 Word ore are scalar types need not be allocated on stack.

These Variables are stored in the 1750A register and remain as is during address state transition.

2.5.1.2 Allocation of modules in different address states

2.5.1.2.1 Stack allocation of tasks

Each task has its own stack. The stack must be allocated on physical pages which are visible in all address states the task will run at.

2.5.1.2.2 Adress state 0 modules

1. Time critical code of IASW and entire CSSW code is running in AS (address state) 0.
2. All task entry points are in AS 0.

2.5.1.2.3 Address state 1 modules : Histogram processing

All histogram controlling/processing routines are allocated in AS 1.

2.5.1.2.4 Address state 2 modules: Tables

All context tables are allocated in AS 2.

2.5.1.2.5 Address state 3 modules: ISGRI Energy correction

The ISGRI energy correction table and its algorithm is allocated in AS 3.

2.6 ERROR HANDLING

2.6.1 EXCEPTIONS

Under no circumstances ADA-Exceptions should be allowed to leave the outmost scope of a single ADA-task because then a runtime routine containing a breakpoint will be reached. This would result in stopping the entire IASW.
So each exception must be handled at least at task scope or in embedded scopes. The way it will be done is that each major function (which one depends on timing/fail save conditions) shall handle exceptions it must deal with. So both should be done:

- *The exception is reported (On Event message)*
- *The software reacts on the exception so it is in a stable state.*

2.6.1.1 Task exception handler

Each task must maintain an <u>exception handler</u> which catches <u>all</u> exceptions within the task loop.

2.6.1.2 Function exception handler

1. Each function which is called when resources are blocked must maintain an exception handler.
2. Each function which is called within a critical context should maintain an exception handler.
3. In case of speed reasons a function may leave out an exception handler provided a failure will have no sideeffects.

2.6.1.3 Reactions on exceptions:

2.6.1.3.1 On-event message

If an exception was catched the exception handler shall send an On-event Message which contains:

- An unique exception ID which defines location or reason of exception raise.
- Type of exception (if common exception).
- Task ID when exception occurred.
- Logical address of exception variable.

<u>Example:</u>

```
PROCEDURE A is
BEGIN

EXCEPTION
     WHEN   Constraint_Error_Exception  ⇒
            Exception_Handler.Constraint_OEM (23);
```

```
    WHEN   others ⇒
              Exception_Handler.Exception_OEM (23);
END A;
```

2.6.1.3.2 Retry/Stop in exception case

In most cases it is sufficient to continue with the caller which will continue processing.

2.6.2 STACK OVERFLOW

2.6.2.1 Static stack check

For each task the maximal stack consumption shall be verified at software integration time.

For each task the stack size should be set at maximum stack consumption plus 100 Word margin.
The TLD compiler supports the stack analysis provided no procedure is recursive and the parameter size is always well defined. Recursion shall not be used in the IBIS IASW, so this may not bother. The parameter size is not always strict defined because commands which are sent to an external device may have different length. The maximum allowable length is 64 Word. To have some margins (about 10%) used by Astres the value above is defined.

2.6.2.2 Stack check at run time

Stack overflow will be checked at run time periodically by the CSSW.

2.6.3 TASK OVERRUN

Periodic tasks are checked by CSSW.

3 DETAILED DESIGN DESCRIPTION

3.1 USED H/W-S/W INTERFACE

Working as a real time data evaluating and controling instrument the IBIS DPE has to access both the instrument as the OBDH bus. This will be achieved by serial lines, analog lines and relais lines to the instrument devices and on the other hand by means to access the OBDH bus to supply Telemetry data and to receive Telecommands.

3.1.1 ACCESS VIA CSSW ONLY

The IASW shall use the CSSW only to access any hardware beyond the DPE.
The interfaces used by IASW will be described in following sections. For CSSW terms refer [RD 5].

3.1.2 TELECOMMANDS

Telecommands shall be collected with CSSW function **GET_TC** only.

3.1.2.1 TC Buffer size

The telecommand buffer size shall be able to keep 8 unprocessed telecommands.
Telemetry
All telemetry data except essential housekeeping shall be given to CSSW function **PUT_SCIENCE_TM.**

3.1.2.2 Housekeeping

Essential housekeeping shall be given CSSW function **PUT_IASW_HK_TM**.

3.1.2.3 On-Event Messages

The IASW may in case of specific events generate on-event-messages. These messages shall be delivered to CSSW function **PUT_ON_EVENT_MESSAGE** which fills the on-event message into the CSSW OEM-Buffer. IASW must take care that this buffer does not overflow.

3.1.2.4 Interface Communication Buffer Status

The IASW must care that the CSSW buffer does not overflow. This will be checked with the status of the buffer in the CSSW which keeps the outgoing telemetry packets except essential housekeeping.
The status is checked with the CSSW function **GET_ICB_STATE.**

3.1.3 LOW SPEED SERIAL LINE (LSSL)

The low speed serial line is used to transmit commands to and read configuration data from detector electronic devices.

3.1.3.1 Devices

There are 11 electronic devices of IBIS which could be accessed via the low speed serial line [RD 2]. They are referenced as *peripheral instruments* also.

3.1.3.1.1 HEPI

Single device for data evaluation.
LSL Line 0, address not used (0).

3.1.3.1.2 ISGRI

8 modules (MCE) to read out ISGRI CdTe data.
LSL Line 3, address lines 0,1,4:

MCE	Address
1	194
2	226
3	210
4	242
5	193
6	225
7	211
8	241

3.1.3.1.3 PICSIT

Single device to read out PICSIT CsI data.
LSL Line 1, address not used (0).

3.1.3.1.4 VETO

Single device to control veto shield.
LSL Line 2, address not used (0).

3.1.3.2 initialising LSSL

IASW will use the CSSW functions **SET_LOW_SPEED_LINE_MODE** and **SET_TRS_LINE** to define an asynchronous, high level protocol.

3.1.3.3 Starting LSSL Transmission

When starting a LSSL transmission the CSSW functions

- **CONFIGURE_LOW_SPEED_LINE,**
- **SET_TRANSMITTER_ADDRESS,**
- **ENABLE_LOW_SPEED_LINE**
- **TRANSMIT_LSL_MESSAGE**

will be used to define the channel and address of the transmission according the receiving device. The line will be enabled and the data transmitted as being stored in transmit buffer.

The baud rate will be set at 16kBit/sec [RD 2].

3.1.3.4 Starting LSSL Reception

When starting a LSSL reception the CSSW functions

- **CONFIGURE_LOW_SPEED_LINE,**
- **SET_TRANSMITTER_ADDRESS,**
- **ENABLE_LOW_SPEED_LINE**
- **RECEIVE_LSL_MESSAGE**

will be used to define the channel and address of the transmission according the sending device. The line will be enabled and the data received stored in given buffer.

The baud rate will be set at 16kBit/sec [RD 2].

3.1.3.5 resetting LSSL

The LSSL will be reseted using the CSSW function **SET_LOW_SPEED_LINE_MODE** with first synchronous mode, then asynchronous mode.
This will provoke an CSSW internal reset of the low speed serial line.
[RD 5, sect. 3.2.4.3.3.1]

<u>3.1.4 HIGH SPEED SERIAL LINE (HSSL)</u>

The high speed serial line is used to transmit scientific data from HEPI to DPE.

3.1.4.1 Initialising HSSL

The HSSL shall be initialised with the CSSW functions **RESET_SERIAL_LINE_MODULE** and **CONFIGURE_HIGH_SPEED_LINE** with DMA interlace 32.

3.1.4.2 Starting Reception

The HSSL reception calls CSSW functions **SELECT_HIGH_SPEED_LINE** (A/B) and **RECEIVE_HSL_MESSAGE.**

3.1.4.3 Resetting HSSL

The HSSL shall be reseted with the CSSW function **RESET_SERIAL_LINE_MODULE.**

3.1.4.4 HSSL A

The high speed serial line A shall be used to read single photon event data from HEPI [RD 1].

3.1.4.5 HSSL B

The high speed serial line B shall be used to read histogram data from HEPI [RD 1].

3.1.5 ANALOG CHANNEL ACQUISITION

Analog values shall be converted with CSSW function **START_ANALOG_ACQUISITION** and read out with function **READ_LAST_ ANALOG_ ACQUIRED_VALUE.**

3.1.5.1 Bilevel Channel ACQUISITION

Some analog channels shall be declared as bilevel with CSSW function **CREATE_A_BILEVEL,** which will define these channels to contain only single bit information (high/low).

3.1.5.2 Analog channels

Analog values will be fetched from following peripheral instruments:

Peripheral Instrument	Channel	Bilevel	Use
VETO	0	No	VECU 1
VETO	1	No	VECU 2
VETO	2	No	PMT
PICSIT	3	No	PICSIT voltage
VETO	4	Yes	Watchdog enabled
PICSIT	5	Yes	Watchdog enabled

3.1.6 RELAIS PULSES

Additionally to the serial lines and analog lines the IASW may access single lines which will send/receive relais pulses. There are 8 channels which could be used for relais send/receive.

3.1.6.1 Sending relais pulses

Relais pulses are sent with CSSW function **SWITCH_RELAIS**. All channels will be accessible via telecommand.

3.1.6.2 Reading Relais status

Following relais lines will be used for getting information from HEPI:

- LINE 0 : Single photon events are available to be read via HSSL A.
- Line 1: Histogram blocks are available to be read via HSSL B.

It shall be possible to read any relais status signals via telecommand.

3.1.6.3 Send/Receive Relais signals independend

There is no functional correlation between the relais lines used for sending and receiving signals.

3.1.7 ON-BOARD TIME

The on board time will be fetched with CSSW function **GET_CURRENT_OBT.** In case the time need not be very accurate the function **GET_LAST_FROZEN_OBT** may be used.

3.1.7.1 BCP4

HEPI uses the 1 second broadcast pulse (BCP) 2/4 to define a synchronized time to timestamp all photon events supplied by ISGRI/PICSIT. For a time reconstruction on ground the starting time of each measurement must be known with high accuracy . This time will be the first broadcast pulse after a detector reset command which will be read from IASW.

For detector time synchronisation the last BCP4 time will be read.

3.1.8 HARDWARE EVENTS (INTERRUPTS)

The IASW allocates following hardware events:

- **BCP1** (Housekeeping synchronisation)
- **BCP4** (Detector synchronisation)
- **LSL End of Transmission** (LSSL access)
- **LSL End of Reception** (LSSL access)
- **LSL Parity Error** (LSSL access)
- **LSL Overrun Error** (LSSL access)
- **HSL End of Message** (HSSL access)
- **HSL FIFO Full Error** (HSSL access)
- **Timer A** (Waiting short time)
- **End of Analog Acquisition** (Analog access)

3.1.8.1 Interrupt allocation

The IASW may replace the standard CSSW interrupt handlers in case of timing problems. If the interrupts are used by parts of the CSSW it must be made shure that the H/W events are redistibuted to all parts of the DPE S/W system.

3.2 LIST OF MODULES

Name	Type	Adress state	Task
ADD_HK	Task Module	0	HK Task
AMPLITUDE_CORRECTION	Task Module	3	Main Task
ANALOG_IF	Interface Module	0	HK Task
BCP4_HANDLER	Interface Module	0	
BCPK_HANDLER	Interface Module	0	TC Task, Main

			Task, Histogram Task, Read Histogram Task
COMPUTE_CHECKSUM	Interface Module	0	Main Task, TC Task
DETECTOR_INTERFACE	Interface Module	0	TC Tasks, Main Task, HK Task, Histogram Task, Read Histogram Task
DIAGNOSTIC	Task Module	0	Main Task
DIAGNOSTIC_MODE	Task Module	0	Main Task
ECR	Interface Module	0	Main Task, HK Task, TC Task
FAST_TR	Transit routine	0	
HANDLE_PIXEL_NOISY	Task Module	0	HK Task
HBR_A_IF	Task Module	0	HBR A Task
HBR_B_IF	Task Module	1	Histogram Task
HEPI	Interface Module	0	Main Task, HK Task, TC Task, Histogram Task, Read Histogram Task
HEPI_CONTEXT_TABLE	Interface Module	2	Main Task, TC Task
HISTOGRAM_CONFIG	Task Module	1	Histogram Task
HISTOGRAM_OP	Task Module	1	Read Histogram Task
HK	Task Module	0	HK Task
HSL_INTERFACE	Interface Module	0	HBR A Task, Read Histogram Task
IASW	IASW ENTRY POINT	0	N/A
IASW_COMMON	Interface Module	0	N/A
IASW_STACKS	Interface Module	0	HK Task, Histogram Task, Read Histogram Task
INSTRUMENT_CONTEXT	Interface Module	0	Main Task, TC Task
ISGRI	Interface Module	0	Main Task, TC Task, HK Task
ISGRI_CONTEXT_TABLE	Interface Module	2	Main Task, TC Task
LINE_MANAGER	Interface Module	0	Main Task, HK Task, HBR A Task, TC Task, Histogram Task, Read Histogram Task
LSL_INTERFACE	Interface Module	0	Main Task, HK

			Task, TC Task, Histogram Task, Read Histogram Task
MODE_DISPATCHER	Task Module	0	Main Task
OBT_OPERATIONS	Interface Module	0	Main Task, TC Task, Histogram Task, Read Histogram Task
OBT_ASM	Interface Module	0	Main Task, TC Task, Histogram Task, Read Histogram Task
OEM_INTERFACE	Interface Module	0	Main Task, HBR A Task, HK Task, TC Task, Histogram Task, Read Histogram Task
PICSIT	Interface Module	0	Main Task, TC Task, HK Task
PICSIT_CONTEXT_TABLE	Interface Module	2	
READ_HISTOGRAM	Task Module	1	Read Histogram Task
S1	Task Module	0	Main Task
S2	Task Module	0	Main Task
S3	Task Module	0	Main Task
S4	Task Module	0	Main Task
S5	Task Module	1	Read Histogram Task
S6	Task Module	1	Read Histogram Task
S7	Task Module	1	Read Histogram Task
S8	Task Module	0	Main Task
SCIENCE_FUNCTIONS	Interface Module	0	Main Task, Histogram Task, Read Histogram Task
SCIENCE_HK	Interface Module	0	Main Task, HK Task, TC Task, Histogram Task, Read Histogram Task
SCIENCE_MODE	Task Module	0	Main Task
SQRT	Interface Module	0	Main Task

TABLE_HANDLER	Interface Module	2	TC Task
TASK_PARAMETER	Task Module	0	TC Task
TASKS	INITIAL MODULE	0	N/A
TC_DECODER	Task Module	0	TC Task
TM_INTERFACE	Interface Module	0	Main Task, HK Task, TC Task, Read Histogram Task
TRANSPARENT	Task Module	0	Main Task
VETO	Interface Module	0	Main Task, TC Task, HK Task
VETO_CONTEXT_TABLE	Interface Module	2	Main Task, TC Task

Table 5: List of all IASW modules

3.2.1 LINE RESOURCE MANAGEMENT

Different Tasks need access of the HSL or LSL Device. Hence it follows the problem, that one Task wants to read from the Device during a other Task is reading. To avoid this shared resource conflict each HSL or LSL accessing Task must use the **LINE MANAGEMENT PROTOCOL**: (High and Low speed Line are considered in the same manner, as a **LINE**)

1. Bevore using the Line it must be checked whether the Line is free. If so then the Line might be used. If not the task must wait till the Line becomes free.
2. When a task has access to the Line it must block the Line. Operation 1.), 2.) must be atomic operations in respect of multitasking, i.e. must not be interrupted by the task scheduler.
3. If a certain Task used the Line it must unblock the Line and inform waiting tasks that the Line has become free.
4. A polling approach with high priority should be maintained, i.e. a certain taks must have the possibility to look whether the Line has become unblocked and then get access to it (Main task event loop).

3.2.1.1 Context tables

All context tables are located at AS 2.

Description	Size (word)	Start address (physical)	Start address (logical)
HEPI	10240	16#1A000#	16#4000#
VETO	64	16#1E000#	16#8000#
PICSIT	320	16#1F000#	16#9000#
ISGRI	10752	16#20000#	16#A000#

Remarks:
ISGRI table consists of 8 single ISGRI context tables, one per MCE. Each single ISGRI MCE context table is 21 blocks a 64 words large.
HEPI context above defines the LUT, but HEPI also contains a certain context in the REGISTER TYPE II which is stored in DPE and upload each time when a science mode is started. The REGISTER TYPE II has a size of 35 byte (+ 1 spare).
The table above declares table allocation in blocks. The used table size might be smaller according length value of last block. (see: IBIS COMMUNICATION PROTOCOL DEFINITION).

3.3 DESCRIPTION OF THE SW MODULES

3.3.1 MODULE ADD_HK

- **List of Contents:**
  Procedure GET_ADD_HK

- **Global Variables:** VETO_SPECTRA_PARAMETER.
  Structure:

component	length (bit)	Description	Default
SP_CAL	1	Flag to define whether spectra is in concidence with (1) or without (0) conincidence of calibration	0
SPEHK_PER	15	spectra integration time in sec.	300
SPEHK_INF		defines whether (1) or not (0) channels will increase. If not, SPEHK_CH defines channel to integrate into.	0
SPEHK_CH		channel to integrate spectra into.	0

VETO_SPECTRA_PARAMETER is sent as is to veto when new spectra will be integrated. IASW will use the SPEHK_PER-component only to determine when to read the spectra next time.

Description

The module ADD_HK collects additional housekeeping H3, H4 from ISGRI, PICSIT and VETO. Each time the function GET_ADD_HK is called (which must be done periodically each 8 seconds) it will be checked whether it is time to collect new data from the detectors. The time is:

1. ISGRI additional HK is read each 32 sec = 4 HK cycles.
2. PICSIT additional HK is read each 32 sec = 4 HK cycles.
3. VETO additional HK period depends on the time period of the variable VETO_SPECTRA_PARAMETER. SPEHK_PER which may be changed via telecommand.

3.3.1.1 Procedure GET_ADD_HK

- **Input:** Electrical configuration register ECR: Defines which detector is switched on.
- **Output:** Returns true when successfull collected additional housekeeping H3, H4.

- **Called by modules:** HK, each 8 seconds.
- **Description:**

The function increases an internal counter which defines when to read out which type of additional HK. Provided it is time to read an H3 data packet first PICSIT will be read, then ISGRI (with increasing MCE number) will be read out. After successfull reading the data read is copied into the telemetry packet H3 and sent next cycle (next call of function GET_ADD_HK).
If the ECR for given detector is off (PICSIT or ISGRI MCE to be read out) no access to the detector will performed. The H3 packet contains a flag at the end of the telemetry packet which indicates that the content of the H3 packet is empty.

Veto HK will be read out according the period defined in global variable VETO_SPECTRA_PARAMETER. This will be done in following steps:

1. Set internal veto integration time at period defined in variable VETO_SPECTRA_PARAMETER.
2. Send to VETO the content of veto_spectra_parameter with command V_SPEHKPAR.
3. Wait the period plus 1 HK margin.
4. Read VETO spectra into H4 packet, send TM H4.

- **Constraints:** The function must be called each 8 sec.
- **Efficiency:** Not applicable
- **Verification Requirement:** TBD

3.3.2 MODULE AMPLITUDE_CORRECTION

- **List of Contents:**

  Function RISETIME_CORRECTION return Amplitude_correction.
  T_Risetime_range
  Function GAIN_CORRECTION return Float
  Function FIRST_STEP return Amplitude_correction.T_Unsigned_4bit
  Function SECOND_STEP return System.Unsigned
  Function COMPTON_AMP_CORR return System.Unsigned
  Procedure ISGRI_AMP_CORR
  Function READ_GAIN_OFFSET_TABLE return Table_Handler.T_Block
  Procedure WRITE_GAIN_OFFSET_TABLE
  Function READ_FINAL_TABLE return Table_Handler.T_Block
  Procedure WRITE_FINAL_TABLE

- **Global Variables:** None

- **Description:**

This module contains all routines to perform the 2-step correction of a CdTe amplitude according the following equations:

First step: Gain and Offset

This step needs values out of a 16 kword lookup table named GAIN_OFFET_TABLE. The size of this table is 256 blocks of 128 bytes. Each block could be set via TC(5,3)(TID=32, FID=0) and read out via TC(5,3)(TID=32, FID=0).
For every CdTe pixel (Y,Z) there are four 4-bit Integer values g_h, o_h, g_t o_t (range –8..+7). The four values for one pixel are stored inside a single word (same memory address). All values are initialised with zero.

Structure of the GAIN_OFFSET_TABLE: 128 x 128 x 4 4-bit Integer

Pixel	g_h (4 bit)	o_h (4 bit)	g_t (4 bit)	o_t (4 bit)
Y, Z=0 (1 word)	0	0	0	0
Y, Z=1 (1 word)	0	0	0	0
... (1 word)	0	0	0	0
Y=127, Z=127	0	0	0	0

Table 6: Amplitude correction GAIN_OFFSET_TABLE

This lookup table values are parameters for the following calibration functions.

	Resulting ranges:

$gain\_t = 1 + g\_t$ $offset\_t = -1 + (6 * o\_t)$ $gain\_h = 2 + (0{,}1 * g\_h)$ $offset\_h = -4 + (4 * o\_h)$	$-7 \le gain\_t \le 8$ $-49 \le offset\_t \le 41$ $1{,}2 \le gain\_h \le 2{,}7$ $-36 \le offset\_h \le 24$

At the begin of the first step a linear correction of the origin detector values for rise-time and pulse height is performed. Using this results the type of charge loss curve is determined.

$T = (risetime * gain\_t) + offset\_t$ $H = (pulseheight * gain\_h) + offset\_h$ $E1 = \frac{T_0 * H}{T_0 - T}$	Allowed ranges: $0 \le T \le 127$ $0 \le H \le 5550{,}9$ $0 \le E1 \le 15$

with T: linear corrected input value rise-time.
H: linear corrected input value of pulse height.
T_0 (constant): 127 = maximum value of rise-time.
E1: type of charge loss curve, table entry value for second step.

Second step: Charge Loss Curves

The charge loss curves give the ratio between the energy and the pulse height as a function of the rise-time. The 2 kword lookup table FINAL_TABLE keeps 16 bit shape parameters of the h(E,T) relationship. The size of this table is 32 blocks of 128 bytes. Each block could be set via TC(5,3)(TID=32, FID=1) and read out via TC(5,3)(TID=32, FID=1).

The corrected pulse height H out of the first step multiplied by shape parameter S(E1,T) result in the corrected amplitude in units of [keV].

$E2 = H * S(E1,T) * 1{,}8\exp(-4)$	Resulting range: $0 \le E2 \le 65480$

Because E2 must stay in range of 16 bit System.Unsigned (0..65535) and with maximum value of H = 5550,9 the maximum allowed factor = 11,8. The allowed range for shape parameters inside the FINAL_TABLE is $0 \le S(E1,T) \le 65535$. So an additional calibration factor (1,8exp(-4)) is used to map the table value range of (0..65535) to the range (0..11,8).

Structure of the FINAL_TABLE:

E1 \ T	0	1	...	127
0	$S_{0,0}$	$S_{0,1}$	...	$S_{0,127}$

1	$S_{1,0}$	$S_{1,1}$	...	$S_{1,127}$
...	...	...	...	...
15	$S_{15,0}$	$S_{15,1}$	...	$S_{15,127}$

Table 7: Amplitude correction FINAL_TABLE

All values S are initialised with "11111". Together with the calibration factor of 1,8exp(-4) the initialisation values correspond to a factor = 2.

3.3.2.1 Function RISETIME_CORRECTION return Amplitude_Correction.T_Risetime_range

- **Input:** pulse height, rise time, Y-position, Z-position of an event type Hex(24,A4, A8)
- **Output:** Corrected rise time
- **Called by modules:** AMPLITUDE_CORRECTION
- **Description:**

This module internal function performs the linear rise time correction of the CdTe amplitude correction. It needs two parameter g_t, o_t out of the 16 kword lookup table GAIN_OFFSET_TABLE. This two parameters are used for the calibration functions:

$gain\_t = 1 + g\_t$ $offset\_t = -1 + (6 * o\_t)$	Resulting ranges: $-7 \le gain\_t \le 8$ $-49 \le offset\_t \le 41$

Then the linear correction of the rise time is performed according to:

$$T = (risetime * gain\_t) + offset\_t$$

The allowed range for the corrected rise time is $0 \le T \le 127$. If the result of the calculation is negative T will be set to zero. If the result of the calculation is greater than 127 T will be set to 127.
All calculations inside this function are done in floating points. The corrected risetime value T is returned as system unsigned.

- **Constraints:** None.
-

- **Efficiency:** Not applicable.

- **Verification Requirement:** TBD

3.3.2.2 Function GAIN_CORRECTION return Float

- **Input:** pulse height, rise time, Y-position, Z-position of an event type Hex(A4, A8)

- **Output:** Linear corrected pulse height

- **Called by modules:** AMPLITUDE_CORRECTION

- **Description:**

This module internal function performs the linear pulse height correction of the CdTe amplitude correction. It needs two parameter g_h, o_h out of the 16 kword lookup table GAIN_OFFSET_TABLE. This two parameters are used for the calibration functions:

$gain\_h = 2 + (0{,}1 * g\_h)$ $offset\_h = -4 + (4 * o\_h)$	Resulting ranges: $1{,}2 \le gain\_h \le 2{,}7$ $-36 \le offset\_h \le 24$

Then the linear correction of the rise time is performed according to:

$$H = (pulseheight * gain\_h) + offset\_h$$

The allowed range for the corrected pulse height is $0 \le T \le 5550{,}9$. If the result of the calculation is negative H will be set to zero. The value 5550,9 is the maximum value which is possible for H (pulse height = 2047; gain_h = 2.7 and offset_h = 24).
All calculations inside this function are done in floating points. The linear corrected pulse height value H is returned as floating point.

- **Constraints:** None.

- **Efficiency:** Not applicable

- **Verification Requirement:** TBD

3.3.2.3 Function FIRST_STEP return Amplitude_Correction.T_Unsigned_4bit

- **Input:** linear corrected pulse height, linear corrected rise time,
- **Output:** charge loss curve selection parameter E1
- **Called by modules:** AMPLITUDE_CORRECTION
- **Description:**

This module internal function performs the first step of the CdTe amplitude correction. It needs the linear corrected pulse height H and the linear corrected rise time T of the event.

$E1 = \frac{T_0 * H}{T_0 - T}$	Resulting range: $0 \le E1 \le 15$

with T: linear corrected input value rise-time.
H: linear corrected input value of pulse height.
T_0 (constant): 127 = maximum value of rise-time.
E1: type of charge loss curve, table entry value for second step.

All calculations inside this function are done in floating points. The charge loss curve selection parameter E1 is returned as system unsigned.

- **Constraints:** None
- **Efficiency:** Not applicable
- **Verification Requirement:** TBD

3.3.2.4 Function SECOND_STEP return System.Unsigned

- **Input:** charge loss curve selection parameter E1, linear corrected rise time, linear corrected pulse height
- **Output:** Corrected CdTe amplitude in [keV]
- **Called by modules:** AMPLITUDE_CORRECTION

- **Description:**

This module internal function performs the second step of the CdTe amplitude correction. It needs the parameter S(E1,T) out of the 2 kword lookup table FINAL_TABLE. The result of this function is the corrected CdTe amplitude in keV.

$E2 = H * S(E1,T) * 1{,}8\exp(-4)$	Resulting range: $0 \le E2 \le 65480$

Because E2 must stay in range of 16 bit System.Unsigned (0..65535) and with maximum value of H = 5550,9 the maximum allowed factor = 11,8. The allowed range for shape parameters inside the FINAL_TABLE is $0 \le S(E1,T) \le 65535$. So an additional calibration factor (1,8exp(-4)) is used to map the table value range of (0..65535) to the range (0..11,8).

- **Constraints:** None.
- **Efficiency:** Not applicable
- **Verification Requirement:** TBD

3.3.2.5 Function COMPTON_AMP_CORR return System.Unsigned

- **Input:** pulse height, rise time, Y-position, Z-position of an Compton single event type Hex(A4) or Compton multiple event type Hex(A8)
- **Output:** Corrected CdTe amplitude
- **Called by modules:** S3
- **Description:**

This global function calls all the internal functions which are necessary to perform the CdTe amplitude correction. The sequence of function calls is:

RISETIME_CORRECTION
GAIN_CORRECTION
FIRST_STEP
SECOND_STEP

- **Constraints:** None.
- **Efficiency:** Not applicable

- **Verification Requirement:** TBD

3.3.2.6 Procedure ISGRI_AMP_CORR

- **Input:** pulse height, rise time, Y-position, Z-position of a CdTe PPM event type Hex(24) and the high rise time threshold
- **Output:** Corrected rise time and corrected amplitude
- **Called by modules:** S1
- **Description:**

This global function calls all the internal functions which are necessary to perform the CdTe amplitude correction similar to function COMPTON_AMP_CORR. In addition the procedure ISGRI_AMP_CORR contains a risetime filter for CdTe PPM events. If the corrected rise time is greater than the high rise time threshold (the CdTe PPM event will not pass the rise time filter of module S1 and it will not be linked down), then the rest of the amplitude correction must not be performed to save CPU time. The sequence of function calls is:

RISETIME_CORRECTION
Only if the corrected rise time is smaller than the high rise time threshold:
GAIN_CORRECTION
FIRST_STEP
SECOND_STEP

- **Constraints:** None
- **Efficiency:** Not applicable
- **Verification Requirement:** TBD

3.3.2.7 Function READ_GAIN_OFFET_TABLE return Table_Handler.T_Block

- **Input:** Block_ID
- **Output:** 128 byte block of the GAIN_OFFSET_TABLE

- **Called by modules:** TABLE_HANDLER

- **Description:**

It must be possible to read out the values of the GAIN_OFFSET_TABLE. It is not possible to read out a single value, but one block with length of 128 byte. For this reason the whole table is mapped into blocks of 128 bytes. Each block of the mapped table has got its own Block_ID. Each block could be read out via TC(5,3)(TID=32, FID=0).
This function reads out the block with number Block_ID of the GAIN_OFFSET_TABLE.

- **Constraints:** None

- **Efficiency:** Not applicable

- **Verification Requirement:** TBD

3.3.2.8 Procedure WRITE_GAIN_OFFSET_TABLE

- **Input:** Block_ID, block of 128 byte containing new values for the GAIN_OFFSET_TABLE

- **Output:** None

- **Called by modules:** TABLE_HANDLER

- **Description:**

It must be possible to assign new values to the GAIN_OFFSET_TABLE. It is not possible to assign a single value, but one block with length of 128 byte. For this reason the whole table is mapped into blocks of 128 bytes. Each block of the mapped table has got its own Block_ID. Each block could be set via TC(5,3)(TID=32, FID=0). This procedure overwrites the block with number Block_ID of the GAIN_OFFSET_TABLE.

- **Constraints:** None

- **Efficiency:** Not applicable

- **Verification Requirement:** TBD

3.3.2.9 Function READ_FINAL_TABLE return Table_Handler.T_Block

- **Input:** Block_ID
- **Output:** 128 byte block of the FINAL_TABLE
- **Called by modules:** TABLE_HANDLER
- **Description:**

It must be possible to read out the values of the FINAL_TABLE. It is not possible to read out a single value, but one block with length of 128 byte. For this reason the whole table is mapped into blocks of 128 bytes. Each block of the mapped table has got its own Block_ID. Each block could be read out via TC(5,3)(TID=32, FID=1). This function reads out the block with number Block_ID of the FINAL_TABLE.

- **Constraints:** None
- **Efficiency:** Not applicable
- **Verification Requirement:** TBD

3.3.2.10 Procedure WRITE_FINAL_TABLE

- **Input:** Block_ID, block of 128 byte containing new values for the FINAL_TABLE
- **Output:** None
- **Called by modules:** TABLE_HANDLER
- **Description:**

It must be possible to assign new values to the FINAL_TABLE. It is not possible to assign a single value, but one block with length of 128 byte. For this reason the whole table is mapped into blocks of 128 bytes. Each block of the mapped table has got its own Block_ID. Each block could be set via TC(5,3)(TID=32, FID=1). This procedure overwrites the block with number Block_ID of the FINAL_TABLE.

- **Constraints:** None
- **Efficiency:** Not applicable
- **Verification Requirement:** TBD

3.3.3 MODULE ANALOG_IF

- **List of Contents:**
  Functions:
  FETCH_ANALOG_VAL,
  FETCH_BILEVEL_VAL

- **Global Variables:** None.

- **Exceptions:**

3.3.3.1 Function FETCH_ANALOG_VAL

- **Input:** CHANNEL - the analog channel to read from according RD 4.
- **Output:** Returns the converted analog value if successfully done, otherwise 0.
- **Called by modules: HK**
- **Description:**

1. starts a conversion of given channel
2. waits till conversion is over
3. reads out and returns converted value

- **Constraints:**

Keep care that the function is not called when in other tasks an analog conversion is still running.
The channel must not be configured as a BILEVEL
Before calling the event END_OF_ANALOG_ACQUISITION_SIGNAL must be connected to the HW event END_OF_ANALOG_ACQUISITION (in module TASKS).
The analog value mask must be configured false.

- **Efficiency:** Not applicable
- **Verification Requirement:** TBD

3.3.3.2 Function FETCH_BILEVEL_VAL

- **Input:** CHANNEL - the analog channel to read from according RD 4.
- **Output:**

Returns the converted analog bilevel value if successfully done, otherwise 0.

- **Called by modules:** HK
- **Description:**

- starts a bilevel conversion of given channel
- waits till conversion is over

- reads out and returns converted value.

- **Constraints:**

Keep care that the function is not called when in other tasks an analog conversion is still running. The channel must be configured as a BILEVEL.
Before calling the event END_OF_ANALOG_ACQISITION_SIGNAL the HW event END_OF_ANALOG_ACQISITION must be connected (in module TASKS).
The analog value mask must be configured false.

- **Efficiency:** Not applicable
- **Verification Requirement:** TBD

3.3.4 MODULE BCP4_HANDLER

- **List of Contents:**
  Procedure INITIALISE
  Procedure ENABLE
  Procedure DISABLE

- **Global Variables:** None.

- **Description**

This module contains an enhanced bcp4 interrupt handler (interrupt #11) which could be disabled/ enabled by calling a procedure.
The new interrupt handler should prevent the occurence of the LSSL overrun error.

3.3.4.1 Procedures INITIALISE

- **Input:** None
- **Output:** None
- **Called by modules:**

-IASW

- **Description:**

Installs internal bcp4 interrupt handler with mask 0.
This new handler checks whether being enabled and, if not, skips sending of bcp4 events. Per default the BCP4-handling is enabled.

- **Constraints:**

Should be used to replace the CSSW BCP4 interrupt (int 11) handler. The new interrupt handler could be enabled/disabled with procedures below.
The procedure should be installed after the cssw has started up.

- **Efficiency:** Not applicable

- **Verification Requirement:** TBD

3.3.4.2 Procedure Enable

- **Input:** None
- **Output:** None
- **Called by modules:**

-SCIENCE_MODE
-DIAGNOSTIC_MODE

- **Description:**

Enables handling of BCP4-interrupt (INT 11). Sets flag true which enables BCP4 handling.

- **Constraints:**

May be called after the bcp4 interrupt was disabled. Should be called before using HEPI resync procedure which waits for BCP4.

- **Efficiency:** Not applicable
- **Verification Requirement:** TBD

3.3.4.3 Procedure Disable

- **Input:** None
- **Output:** None
- **Called by modules:**

-SCIENCE_MODE
-DIAGNOSTIC_MODE

- **Description:**

Disables handling of BCP4-interrupt (INT 11). Sets flag false which enables BCP4 handling.

- **Constraints:**

May be called after the bcp4 interrupt was disabled. Should not be called before using HEPI resync procedure which waits for BCP4.

- **Efficiency:** Not applicable
- **Verification Requirement:** TBD

3.3.5 MODULE BCPK_HANDLER

- `Procedure SET_BCPK`

- Procedure UPDATE_BCPK
- Procedure SET_POINTING
- Procedure POINTING
- Procedure ON_TARGET
- Procedure HAND_OVER
- Procedure SAVE_CONTEXT
- Procedure VETO_OFF
- Procedure ISGRI_BIAS_OFF-when
- Procedure SLEW_TIME
- Procedure GET_LAST_POINTING_START
- Procedure GET_LAST_SLEW_START
- Procedure GET_POINTING_ID
- Procedure WRITE_IREM_TABLE
- Procedure READ_IREM_TABLE
- Procedure PROCEDURE

3.3.5.1 Global Variables

3.3.5.1.1 POINT_SLEW_AUTOMATISM

defines how to react when satellite status changes point/slew. It does:

POINT-SLEW AUTOMATISM is ON:
- The function POINTING reports the satellite state being in pointing(true) or slew (false) according the broadcast package contents.
- The function SLEW_TIME reports (as OBT) the start of next slew

POINT-SLEW AUTOMATISM is OFF:
- The function POINTING reports the last setting of point/slew withthe procedure SET_POINTING (s.b.).
- The function SLEW_TIME must not be called when POINT-SLEW AUTOMATISM is off. If doing so an exception EX_NO_SLEW_TIME will be raised.

Remark: This variable is global but should be set only byload task parameter access.

3.3.5.1.2 ECLIPSE_AUTOMATISM

defines how to react when satellite enters eclipse mode.
ECLIPSE_AUTOMATISM is ON:
Each time the broadcast package contains an eclipse flag it affectsthe functions GOTO_STANDBY and SAVE_CONTEXT (s.b.).The result : the IASW goes to eclipse mode,i.e. stop science mode, save context.
ECLIPSE_AUTOMATISM is OFF:The broadcast package eclipse flag will NOT affect GOTO_STANDBY andSAVE_CONTEXT functions.
Remark: This variable is global but should be set only byload task parameter access.

3.3.5.1.3 RADIATION_BELT_AUTOMATISM

defines how to react when satellite enters the radiation belt.
RADIATION_BELT_AUTOMATISM is ON:
Each time the broadcast package contains an radiation belt flag it affectsthe functions GOTO_STANDBY and VETO_OFF (s.b.).The result : the IASW goes to standby mode,i.e. stop science mode.
RADIATION_BELT_AUTOMATISM is OFF:The broadcast package radiation belt flag will NOT affect GOTO_STANDBYfunction.
Remark: This variable is global but should be set only byload task parameter access.

3.3.5.1.4 EMERGENCY_OFF_AUTOMATISM

defines how to react when emergency switch off mode (ESAM) or imminent switch offbroadcast flag is set.
EMERGENCY_OFF_AUTOMATISM is ON:
Each time the broadcast package contains a true ESAM flagor an imminent switch off flag the functions GOTO_STANDBY(s.b.) returns TRUE.The result : the IASW goes to standby mode,i.e. stop science mode.
EMERGENCY_OFF_AUTOMATISM is OFF:The broadcast package ESAM flag will NOT affect GOTO_STANDBYfunction.
Remark: This variable is global but should be set only byload task parameter access.

3.3.5.1.5 RADIATION_MONITOR_AUTOMATISM

defines how to react when a radiation monitor countrate exceeds its threshold.
RADIATION_MONITOR_AUTOMATISM is ON:
Each time broadcast package radiation monitor flag exceeds its thresholdit affects the functions GOTO_STANDBY and VETO_OFF (s.b.).The result : the IASW goes to standby mode,i.e. stop science mode, and veto is switched off.
RADIATION_MONITOR_AUTOMATISM is OFF:The broadcast package radiation monitor entries does not affectIASW.
Remark: This variable is global but should be set only byload task parameter access.

3.3.5.1.6 RADIATION_MONITOR_1_THRESHOLD

the threshold of the radiation monitor 1 (electron count rate) above which the instruments are switched off.default : at upper most value.
The variable is defined global to be read out by the housekeeping taskAND to be written into by tc_decoder/task parameter only

3.3.5.1.7 RADIATION_MONITOR_2_THRESHOLD

the threshold of the radiation monitor 2 (proton count rate 1) above which the instruments are switched off.default : at upper most value.
The variable is defined global to be read out by the housekeeping taskAND to be written into by tc_decoder/task parameter only

3.3.5.1.8 RADIATION_MONITOR_3_THRESHOLD

the threshold of the radiation monitor 3 (proton count rate 2) above which the instruments are switched off.default : at upper most value.
The variable is defined global to be read out by the housekeeping taskAND to be written into by tc_decoder/task parameter only

3.3.5.2 Exceptions

3.3.5.2.1 EX_NO_SLEW_TIME

this exception will be raised if it is tried to get the next slew time though none is availablebecause of POINT-SLEW AUTOMATISM is off.

3.3.5.3 Procedure SET_BCPK

3.3.5.3.1 Input

BCPK_TC - the broadcast package received as a telecommand. Length must be according BCPKT-Document. (34 words)

3.3.5.3.2 Use/Constraints

This function will be called from TC decoder which- received a TC containing the broadcast package.

3.3.5.3.3 Description

- Check length, if not correct -> OEM
- Stores bcpk received
- Set internal flags according content.
- Derive additional information as e.g. slew time.
- Calls procedure UPDATE_BCPK.

3.3.5.4 Procedure UPDATE_BCPK

3.3.5.4.1 Use/Constraints

This function should be called periodically to ensure last broadcast packet received will be evaluated properly. This concerns especially entry/exit of radiation belt and eclipse condition which is on-board time dependend (it must be decided according current OBT whether the satellite is in or out eclipse/ radiation belt).

3.3.5.4.2 Description

- defines whether being in eclipse/radiation belt
- changes mode to standby when necessary.

3.3.5.5 Procedure SET_POINTING

3.3.5.5.1 Input

ON - Flag that shows whether we are in pointing (TRUE) or slew (FALSE) mode.

3.3.5.5.2 Use/Constraints

This function should be called when the point/slew- status is changed manually (eg. via a TC) when point-slew automatism is off. It doesn't have any effect if point-slew automatism is on.

3.3.5.5.3 Description

If point slew automatism is off -> set internal flag reported in function POINTING.

3.3.5.6 Procedure POINTING

3.3.5.6.1 Output

Returns : TRUE when satellite is in pointing status FALSE when satellite is in slew status.

3.3.5.6.2 Use/Constraints

Call when the information whether satellite is in pointing mode needed.

3.3.5.6.3 Description

A Pointing flag is set by broadcast packages so the accuracy of the flag is about 8 sec. This flag is returned. We are in pointing mode if a.) ACC mode is inertial pointing mode b.) AOCS sub mode is IPM c.) The pointing-ID is not zero (then we are in slew)- Remark: We may be in pointing mode though not stable pointing In that case this function returns true while the On-Target-function returns false.

3.3.5.7 Procedure ON_TARGET

3.3.5.7.1 Output

Returns : TRUE when satellite is on target FALSE when not

3.3.5.7.2 Use/Constraints

Call when information of satellite on target status needed. Usefull for histogram integration when integration must be stopped (and resumed later) because of non-stable pointing.

3.3.5.7.3 Description

A On-Target flag is set by broadcast packages so the accuracy of the flag is about 8 sec. This flag is returned. It returns true when we are in IPM mode and on target flag is set.

3.3.5.8 Procedure HAND_OVER

3.3.5.8.1 Output

Returns : TRUE when satellite changes stations FALSE when not (save downlinking of data)

3.3.5.8.2 Use/Constraints

Call when information about savety to link down data is needed.Usefull for histogram integration when histogram packets are read out and must be linked down while none of them should be lost (resulting in image corruption for it might be compressed).

3.3.5.8.3 Description

A Hand-Over flag is set by broadcast packages so the accuracy of the flag is about 8 sec. This flag is returned.

3.3.5.9 Procedure SAVE_CONTEXT

3.3.5.9.1 Output

returns TRUE when context of HEPI,ISGRI,PICSIT,VETO must be saved now. returns FALSE when not.

3.3.5.9.2 Use/Constraints

This function should be called when entering standby in cause of goto_standby procedure. If it returns TRUE the context of all instruments should be saved. If it turnes to FALSE the context should be restored.

3.3.5.9.3 Description

returns (eclipse entered OR imminent switch off)

3.3.5.10 Procedure VETO_OFF

3.3.5.10.1 Output

returns TRUE when veto high voltage must be turned off. returns FALSE when not.

3.3.5.10.2 Use/Constraints

This function should be called when standby is entered because of GOTO_STANDBY function. When TRUE the veto high voltage must be switched off.

3.3.5.10.3 Description

returns (ESAM OR (radiation > threshold) OR radiation belt entry OR imminent switch off)

3.3.5.11 Procedure ISGRI_BIAS_OFF-when

3.3.5.11.1 Output

returns TRUE when isgri bias must be set to zero. off. returns FALSE when not.

3.3.5.11.2 Use/Constraints

This function should be called in standby mode due to GOTO_STANDBY function. When TRUE ISGRI bias should be set to zero.

3.3.5.11.3 Description

returns (ESAM OR (radiation > threshold) OR Eclipse entry OR radiation belt entry OR imminent switch off)

3.3.5.12 Procedure SLEW_TIME

3.3.5.12.1 Output

RETURNS : on-board time when the satellite will enter slew mode.

3.3.5.12.2 Use/Constraints

Might be called if a roughly estimation of the start of next slew mode is desired. THIS FUNCTION MUST NOT BE CALLED WHEN POINT-SLEW AUTOMATISM IS OFF And: The slew time is wrong if we are not in pointing mode

3.3.5.12.3 Description

The next slew time is set by the broadcast packages. So this information is updated each 8 sec. The slew time is computed by following assumptions: a.) The pointing will be recogniced as above. b.) The length of pointing is declared in the broadcast information pointing_duration. The function will return the sum of start of pointing- and its duration. If Point-Slew automatism is of an exception EX_NO_SLEW_TIME will be raised.

3.3.5.13 Procedure GET_LAST_POINTING_START

3.3.5.13.1 Output

Entry time (OBT) of last pointing. Returns 0 if still not occurred.

3.3.5.13.2 Use/Constraints

Might be called to get information about pointing/slew (Science HK).

3.3.5.13.3 Description

Each time the bcpk reports entering pointing the current on board time will be stored and reported by this function.

3.3.5.14 Procedure GET_LAST_SLEW_START

3.3.5.14.1 Output

Entry time (OBT) of last slew. Returns 0 if still not occurred.

3.3.5.14.2 Use/Constraints

Might be called to get information about pointing/slew (Science HK).

3.3.5.14.3 Description

Each time the bcpk reports entering slew mode the current on board time will be stored and reported by this function.

3.3.5.15 Procedure GET_POINTING_ID

3.3.5.15.1 Output

Returns pointing number and revolution number as defined by last broadcast package.

3.3.5.15.2 Use/Constraints

Might be called to get information about the pointing which is/was performed (science hk)

3.3.5.15.3 Description

Extracts pointing ID as set by set_bcpk.

3.3.5.16 Procedure WRITE_IREM_TABLE

3.3.5.16.1 Input

Block - the values could not be sent single but inside a block of 64 word. the first three words of this block contain the new values

3.3.5.16.2 Use/Constraints

This function will be called from TABLE HANDLER which itself receiced the block

3.3.5.16.3 Description

- Copy first three words of Block into the Radiation Monitor tresholds variavbles

3.3.5.17 Procedure READ_IREM_TABLE

3.3.5.17.1 Output

Returns - the values could not be sent single but inside a Block of 64 word. the first three words of this block contain the new values

3.3.5.17.2 Use/Constraints

This function will be called from TABLE HANDLER which itself returns the block

3.3.5.17.3 Description

- Copy the three radiation monitor treshold values into the first three words of a Block with 64 words length

3.3.5.18 Procedure PROCEDURE

3.3.6 MODULE COMPUTE_CHECKSUM

3.3.6.1 Global Variables

3.3.6.2 Exceptions

3.3.7 MODULE DETECTOR_INTERFACE

- Procedure SEND_CMD
- Procedure READ
- Procedure GET_LAST_SUB_I_ADDR

3.3.7.1 Global Variables

3.3.7.2 Exceptions

3.3.7.3 Procedure SEND_CMD

3.3.7.3.1 Input

- SUB_I_ADDR : the sub instrument address, consisting of a channel and a address field.
- SEQ_COUNT : the sequence counter to be sent. This counter reflects the sended command number and must- be increased each time when a command is sent. Bit 14 shows whether it is an external (0) or internal (1) command. Each instrument interface must take care to increase the sequence counter.
- CMD : the buffer to link to given instrument.- The transmitted word number is derived from the length of this buffer.

3.3.7.3.2 Output

OK : Returns true if succesfully transmitted command.

3.3.7.3.3 Use/Constraints

Call this procedure to send a sequence of words as a command to the peripheral instrument defined by the sub_instrument address. The sequence counter must be either:
- the sequence counter as received from GET_TC function (a 13 bit value) or
- the value 16#2000# (bit 2 which is the 14-th bit from LSB set) to show that the detector interface must use its own TC counter. After sending the command (after which the procedure will return) the ok flag shows whether transfer was successfully. THE LENGTH OF COMMAND MUST BE AT LEAST 3 WORDS THERE IS NO INTERNAL BLOCKING OF THE LSL

3.3.7.3.4 Use/Constraints

THE LINE MANAGER MUST BE USED BY THE CALLER OF THIS PROCEDURE

3.3.7.3.5 Description

- Sets the apid and the sequence count in the command buffer.
- starts lbr transmission at address/channel according- sub-instrument address.
- Waits till transmission is over or timeout occurred.-

3.3.7.4 Procedure READ

3.3.7.4.1 Input

- SUB_I_ADDR : the sub instrument address, consisting of a channel and a address field.
- SEQ_COUNT : the sequence counter to be sent. This counter reflects the sended command number and must- be increased each time when a command is sent. Bit 14 shows whether it is an external (0) or internal (1) command. Each instrument interface must take care to increase the sequence counter.
- CMD : the buffer to link to given instrument.- The transmitted word number is derived from the length of this buffer. It must contain the command which provokes a response.
- RESP_LEN : The length of the expected response in words. Warning The buffer must be at least contain RESP_LEN words

3.3.7.4.2 Output

- OK : Returns true if succesfully transmitted command.a
- RESPONSE : The buffer in which the received values must be written into.

3.3.7.4.3 Use/Constraints

Call this procedure to send a sequence of words as a command to the peripheral instrument defined by the sub_instrument address. The sequence counter must be either:
- the sequence counter as received from GET_TC function (a 13 bit value) or
- the value 16#2000# (bit 2 which is the 14-th bit from LSB set) to show that the detector interface must use its own TC counter. After sending the command and receiving responding words the procedure will return and show in ok whether- being successfully. THE LENGTH OF COMMAND MUST BE AT LEAST 3 WORDS THERE IS NO INTERNAL BLOCKING OF THE LSL

3.3.7.4.4 Use/Constraints

THE LINE MANAGER MUST BE USED BY THE CALLER OF THIS PROCEDURE

3.3.7.4.5 Description

- Sets the apid and the sequence count in the command buffer.
- starts lbr transmission at address/channel according- sub-instrument address.
- sets response buffer at receive buffer.
- Waits till transmission is over or timeout occurred.-
- Waits till reception is over or timeout occurred. -

3.3.7.5 Procedure GET_LAST_SUB_I_ADDR

3.3.7.5.1 Output

Returns : instrument address of last on-board command sent.

3.3.7.5.2 Use/Constraints

For HK - call to get last instrument accessed LSL.- Should be called BEFORE fetching HK from instruments because otherwise it is the ID of last HK command.

3.3.7.5.3 Description

Each time the Detector interface is accessed it stores the Sub-instrument-address of its command if the sequence counter shows it was an internal command. This address is returned.

3.3.8 MODULE ECR

3.3.8.1 Global Variables

3.3.8.2 Exceptions

3.3.9 MODULE HANDLE_PIXEL_NOISY

3.3.9.1 Global Variables

3.3.9.2 Exceptions

3.3.10 MODULE HBR_A_IF

- Procedure BUFFER_AVAILABE
- Procedure INITIALISE
- Procedure GET_BUFFER_RANGE
- Procedure RESET

3.3.10.1 Global Variables

3.3.10.1.1 BUFFER

The buffer for reading in events.
MUST NOT BE WRITTEN IN

3.3.10.2 Exceptions

3.3.10.2.1 NO_BUFFER_AVAILABLE

exception raised when calling GET_BUFFER though no buffer is available.

3.3.10.3 Procedure BUFFER_AVAILABE

3.3.10.3.1 Output

boolean, true, if an buffer could be fetched

3.3.10.3.2 Use/Constraints

: Must be called before getting events to be shure there are any.

3.3.10.3.3 Description

Checks, if buffer is empty. If so -> try to read from other buffer (if it is filled). In that case start to read in new data from HEPI.

3.3.10.4 Procedure INITIALISE

3.3.10.4.1 Use/Constraints

Must be called when the system is initialised.

3.3.10.4.2 Description

The HBR_A-task reads in a block of 1kW which is split into single events. While on block supplies the events the other one is filled by the HSL.- again if any events are available by HEPI. The buffer is filled only when
- empty
- not used for supplying events.

3.3.10.5 Procedure GET_BUFFER_RANGE

3.3.10.5.1 Output

The range of here global declared buffer which migth be used for read out. The supplied buffer range is declared empty after read out. The range consists of: -
- START_INDEX - the index (inclusive) of first event
- END_INDEX - the index (inclusive) of last event

3.3.10.5.2 Use/Constraints

If running a science mode GET_BUFFER_RANGE should called periodically as often as the BUFFER_AVAILABLE Flag is true. If so the global declared buffer could be e used in given index ranges for event processing. CONSTRAINTS : GET_BUFFER_RANGE MUST NOT BE CALLED IF NO BUFFER IS AVAILABLE Otherwise an exception will be raised.

3.3.10.5.3 Description

Defines according buffer state (empty or filled) which buffer is to be read out. Declares a block to be empty when read out.

3.3.10.6 Procedure RESET

3.3.10.6.1 Use/Constraints

call this procedure when to skip events residing in buffers.

3.3.10.6.2 Description

declares both buffers as empty (read out)

3.3.11 MODULE HBR_B_IF

3.3.11.1 Global Variables

3.3.11.2 Exceptions

3.3.12 MODULE HEPI

- Procedure HBR_A_DATA_AVAILABLE
- Procedure HBR_B_DATA_AVAILABLE
- Procedure COMMAND_HEPI
- Procedure READ_HEPI
- Procedure HEPI_STATUS_SHADOW
- Procedure INITIALISE
- Procedure SYNCHRONIZE
- Procedure GET_LAST_SYNC_OBT
- Procedure GET_SEQ_COUNT

3.3.12.1 Global Variables

3.3.12.1.1 HEPI_TYPE_II_VAR

the hepi type II register. This is a global variable which must be set ONLY by TCdecoder. It might be read from everywhere else.

3.3.12.2 Procedure HBR_A_DATA_AVAILABLE

3.3.12.2.1 Use/Constraints

Call to check whether new data could be read in via HBR A.

3.3.12.2.2 Description

reads mRTU 0 status.

3.3.12.3 Procedure HBR_B_DATA_AVAILABLE

3.3.12.3.1 Use/Constraints

Call to check whether new data could be read in via HBR B (histograms).

3.3.12.3.2 Description

reads mRTU 1 status.

3.3.12.4 Procedure COMMAND_HEPI

3.3.12.4.1 Input

HEPI_COMMAND - The command to be send to HEPI. PARAMETER - The parameter for that command.

3.3.12.4.2 Output

: ok is true when transfer successfully done

3.3.12.4.3 Use/Constraints

Call the procedure with given command ID and parameter for that command. NO PARAMETER CHECK WILL BE DONE THERE IS NO INTERNAL BLOCKING OF THE LSL

3.3.12.4.4 Use/Constraints

THE LINE MANAGER MUST BE USED BY THE CALLER OF THIS PROCEDURE

3.3.12.4.5 Description

Command_Hepi transmits the command via the LSL to HEPI. The control will return when transmission is over. The length of the command is set according the length of the parameter array. The internal hepi status register variable is set by this procedure when setting status register with H_S_HEPI_ST or resetting HEPI with H_HEPI_RES or H_STR_SYNC.

3.3.12.5 Procedure READ_HEPI

3.3.12.5.1 Input

HEPI_COMMAND - The read command to be sent to HEPI PARAMETER - The parameter necessary for this command

3.3.12.5.2 Output

HEPI_RESPONSE : The data read out from HEPI. This array must be set properly at the expected data size from HEPI. ok : true when transfer successfully done

3.3.12.5.3 Use/Constraints

Call Read_Hepi with a command which requests data from HEPI, and necessary parameters. The procedure will return the hepi response data according the command sent into response buffer. NO PARAMETER CHECK WILL BE DONE ALSO NO COMMAND ID CHECK IS APPLIED THERE IS NO INTERNAL BLOCKING OF THE LSL

3.3.12.5.4 Use/Constraints

THE LINE MANAGER MUST BE USED BY THE CALLER OF THIS PROCEDURE

3.3.12.5.5 Description

Command_Hepi transmits the command via the LSL to HEPI. Then the HEPI will send the data requested- via LSL to DPE . The control will return to caller when transmission is over. The length of the command to send is derived from the parameter block length; that of the response from the response block length.

3.3.12.6 Procedure HEPI_STATUS_SHADOWRETURNS: Contents of HEPI status register which is last sent to HEPI.

3.3.12.6.1 Use/Constraints

Should be called when a certain bit of the hepi status must be changed but the whole status is to be sent to HEPI. Keep care of different tasks reading this hepi status and setting their bits

3.3.12.6.2 Description

Each times the status is sent to HEPI it is stored also in this hepi module. This value is readable by this function which returns the 'shadow' value of the master HEPI status register. The internal hepi status register variable is set by the COMMAND_HEPI-procedure when setting status register with H_S_HEPI_ST or resetting HEPI with H_HEPI_RES or H_STR_SYNC.

3.3.12.7 Procedure INITIALISE

3.3.12.7.1 Use/Constraints

Call this procedure when HEPI needs to be initialised, i.e. in the very beginning of THE CALLER MUST NOT BLOCK LSL

3.3.12.7.2 Description

- performs an init pulse.
- wait 5 seconds
- sends a init command to HEPI via LSL
- wait 5 seconds
- sends a reset command to HEPI via LSL

3.3.12.8 Procedure SYNCHRONIZE

3.3.12.8.1 Use/Constraints

Call this procedure when HEPI needs to be synchronized, i.e. when starting science/dia- gnostic mode or HEPI lost synchronisation. The calling task must connect the BCP4-HW event at BCP4_EVENT declared above

3.3.12.8.2 Description

- Waits for the next BCP4-event (occuring each second)-
- sends a synchronize-command to HEPI
- waits till next BCP4-event
- change sync-onboard time according bcp4-time (readable via GET_LAST_SYNC_OBT)

3.3.12.9 Procedure GET_LAST_SYNC_OBT

3.3.12.9.1 Output

Returns: the on board time of last HEPI synchro- nisation performed with procedure SYNCHRONIZE.

3.3.12.9.2 Use/Constraints

Call this for report in science_hk module.

3.3.12.9.3 Description

returns stored on board time of last synchronisation set with procedure SYNCHRONIZE. Returns zero if not set.

3.3.12.10 Procedure GET_SEQ_COUNT

3.3.12.10.1 Output

Returns: 13-bit value containing current number of commands sent to hepi. Value wraps around at 2^13.

3.3.12.10.2 Use/Constraints

In HK for setting the HEPI sequence count field. returns 13-Bit value

3.3.12.10.3 Description

Reflects counter status.

3.3.13 MODULE HEPI_CONTEXT_TABLE

3.3.14 MODULE HISTOGRAM_CONFIG

3.3.14.1 Global Variables

3.3.14.2 Exceptions

3.3.15 MODULE HISTOGRAM_OP

- Procedure FIRST_READ_ADDRESS
- Procedure NEXT_READ_ADDRESS
- Procedure SWITCH_INTEGRATION
- Procedure CLEAR_INTEGRATION_BUFFER
- Procedure BUFFER2BLOCKS
- Procedure ORDER_HISTOGRAM_BLOCK
- Procedure READ_HISTOGRAM_BLOCK-

3.3.15.1 Global Variables

3.3.15.2 Exceptions

3.3.15.2.1 NO_BLOCK_AVAILABLE

exception raised when calling READ_HISTOGRAM_BLOCK though no data block available(checked with HEPI HBR_B_DATA_AVAILABLE).

3.3.15.3 Procedure FIRST_READ_ADDRESS

3.3.15.3.1 Input

Buffer - the buffer of which the first read address is to be determined.

3.3.15.3.2 Output

Returns the address of first buffer which must be read- out.

3.3.15.3.3 Use/Constraints

Call this function when for given buffer a read out is to be started and the first address is to be defined. THE BUFFER MUST NOT BE EMPTY AND AT LEAST ONE BUFFER MUST BE SET TRUE

3.3.15.3.4 Description

Returns according toggle and buffer switched on the first address.

3.3.15.4 Procedure NEXT_READ_ADDRESS

3.3.15.4.1 Input

- Buffer : The histogram buffer to be taken into account.
- Address: The histogram address of which the next is to be determined.

3.3.15.4.2 Output

Address : The address which will be set at the predator of previous address. Buffer: The buffer will be set empty when last address of buffer turned on is reached. And: Each buffer read out will be disabled.

3.3.15.4.3 Use/Constraints

Call this procedure when for given histogram buffer and address already read the following address is needed for read out. Also a end of buffer check will be done to set buffer empty flag. When a certain - histogram is read out its flag will be set at FALSE.

3.3.15.4.4 Description

- adds a single 1024 kByte block to address.
- if larger than current buffer to be read out -> check whether following is not empty. start there, if so.
- set empty flag true when at end of all buffers.
- set flag of buffer already read out at FALSE.

3.3.15.5 Procedure SWITCH_INTEGRATION

3.3.15.5.1 Input

- ON - defines whether to turn on or off integration
- BUFFER - defines which buffer in HEPI to turn on/off-

3.3.15.5.2 Output

Returns true when successfully done.

3.3.15.6 Procedure CLEAR_INTEGRATION_BUFFER

3.3.15.6.1 Input

- TOGGLE : defines the toggle buffer to clear buffer buffer in.

- SCIENCE : if true the science buffers (CsI single and multiple or Polarimetry are the same memory area, therefore cleared in total), otherwise the calibration- memory is cleared.

3.3.15.6.2 Output

returns true when successfully done.

3.3.15.7 Procedure BUFFER2BLOCKS

3.3.15.7.1 Input

HISTOGRAM_BUFFER - the buffer to be read out and whose number of blocks is needed.n

3.3.15.7.2 Output

Returns number of blocks contained in given buffer. If the buffer is empty or no buffer selected the function returns 0.

3.3.15.7.3 Use/Constraints

May be used to get the total number of blocks needed to read a certain type of histograms.

3.3.15.7.4 Description

- checks for type of histogram
- if not empty return block number for selected histogram

3.3.15.8 Procedure ORDER_HISTOGRAM_BLOCK

3.3.15.8.1 Input

- ADDRESS - the memory address in HEPI to read from

3.3.15.8.2 Output

- OK - flag to show whether successfully done.

3.3.15.8.3 Use/Constraints

Call this function when a histogram block is requested to be read out but still nothing in FIFO. The histogram reading should be done in following order: 1.) Order a histogram block with ORDER_HISTOGRAM_BLOCK 2.) After certain time (~ 1msec) check whether data are available with HEPI function HBR_B_DATA- AVAILABLE. If false -> wait longer. 3.) When data are available -> use READ_HISTOGRAM- block to get the data from HEPI via HBR B. KEEP TRACK OF BLOCKS HELD IN HEPI FIFO - DO NOT ORDER MORE THAN 7 BLOCKS WITHOUT READING OUT

3.3.15.8.4 Description

Sends HEPI a LSL command with given address to put histogram block into HBR B FIFO.

3.3.15.9 Procedure READ_HISTOGRAM_BLOCK-

3.3.15.9.1 Input

- ADDR_STATE - the address state the output buffer is residing at.

3.3.15.9.2 Output

- OK - flag to show whether successfully done.
- HISTOGRAM_DATA - the data to be read out.

3.3.15.9.3 Use/Constraints

Call this function when a histogram block is already in FIFO (by procedure ORDER_HISTOGRAM_BLOCK) and available to be read out. The histogram reading should be done in following order: 1.) Order a histogram block with ORDER_HISTOGRAM_BLOCK 2.) After certain time (~ 1msec) check whether data are available with HEPI function HBR_B_DATA- AVAILABLE. If false -> wait longer. 3.) When data are available -> use READ_HISTOGRAM- block to get the data from HEPI via HBR B. KEEP TRACK OF BLOCKS HELD IN HEPI FIFO - DO NOT TRY TO READ FROM AN EMPTY HEPI FIFO. THIS MEANS: DO NOT CALL THIS PROCEDURE WHEN HBR_B_DATA_AVAILABLE RETURNS FALSE IF DOING SO AN EXCEPTION

3.3.15.9.4 Description

Blocks HBR B, reads in HBR B data into given histogram data block and returns result of reading.

3.3.16 MODULE HK

- Procedure INITIALISE

3.3.16.1 Global Variables

3.3.16.1.1 NOISY_PIXEL_AUTOMATISM

defines whether to perform check of PICSIT noisy pixel.
NOISY_PIXEL_AUTOMATISM is ON:When generating HK the PICSIT HK data field is checked anda possibly over-countrate pixel is switched off.
NOISY_PIXEL_AUTOMATISM is OFF:PICSIT HK are not touched.

Remark: This variable is global but should be set only by load task parameter access.

3.3.16.1.2 MIN_VALID_RATE_THRESHOLD

the minimal event rate of valid events which must occur in science mode when HEPI must not be resynchronized. The event rate is defined by number of events per 8 seconds (housekeeping period).

3.3.16.1.3 MAX_INVALID_RATE_THRESHOLD

the maximal event rate of invalid events which might occur in science mode before HEPI will be resynchronized. The event rate is defined by the numberof events per 8 seconds (housekeeping period).

3.3.16.2 Exceptions

3.3.16.3 Procedure INITIALISE

3.3.16.3.1 Use/Constraints

Must be called when the system is initialised.

3.3.16.3.2 Description

The HK task waits HK_DELAY RTC's from last BCP1 and starts then to read in the Hk from detectors, then acquire the IASW HK data and send it to CSSW. Finally a sort of pixel monitoring will be performed.

3.3.17 MODULE HSL_INTERFACE

3.3.17.1 Global Variables

3.3.17.2 Exceptions

3.3.18 MODULE IASW_COMMON

3.3.18.1 Global Variables

3.3.18.1.1 LUT_VERSION_NO

the current version number of all look-up tables used within the IASW for data processing.The number could be changed with a patch. It must not be changedby any software access. It's address is $2000, AS 0, operandpages. (see link file also)

3.3.18.1.2 CONTEXT_VERSION_NO

the current version number of all context tables used for peripheral instruments.The number could be changed with a patch. It must not be changedby any software access. It's address is $2001, AS 0, operandpages. (see link file also).

3.3.18.2 Exceptions

3.3.19 MODULE IASW_STACKS

3.3.19.1 Global Variables

3.3.19.2 Exceptions

3.3.20 MODULE INSTRUMENT_CONTEXT

- Procedure CSF

- Procedure CRF
- Procedure SET_CSF
- Procedure SET_CRF
- Procedure DOWNLOAD_CONTEXT
- Procedure UPLOAD_CONTEXT

3.3.20.1 Global Variables

3.3.20.2 Exceptions

3.3.20.3 Procedure CSF

3.3.20.3.1 Output

Returns: a list of flags which are true when context of certain instrument was saved successfully in DPE.

3.3.20.3.2 Use/Constraints

Call for report in housekeeping.

3.3.20.3.3 Description

Reports flags inside module. A context save was successfull if 1.) transfer did not fail 2.) Checksum computed after transfer both of DPE and instrument does not differ. The flags are set by procedure SET_CSF.

3.3.20.4 Procedure CRF

3.3.20.4.1 Output

Returns: a list of flags which are true when context of certain instrument was uploaded successfully from DPE into instruments

3.3.20.4.2 Use/Constraints

Call for report in housekeeping.

3.3.20.4.3 Description

Reports flags inside module. A context restore was successfull if 1.) transfer did not fail 2.) Checksum computed after transfer both of DPE and instrument does not differ. The flags are set by procedure SET_CRF.

3.3.20.5 Procedure SET_CSF

3.3.20.5.1 Input

FLAG - which instrument belongs to flag to be set. TO - State of flag which has to be set: TRUE : context save successfull. FALSE: context save non-successfull. A context save was is considered to be successfull if:- 1.) transfer did not fail 2.) Checksum computed after transfer both of DPE and instrument does not differ.

3.3.20.5.2 Use/Constraints

Must be used by context saving procedures only which have to obey definition above.

3.3.20.5.3 Description

sets flags inside module to be reported by CSF.

3.3.20.6 Procedure SET_CRF

3.3.20.6.1 Input

FLAG - which instrument belongs to flag to be set. TO - State of flag which has to be set: TRUE : context restore successfull. FALSE: context restore non-successfull. A context restore was is considered to be successfull if: 1.) transfer did not fail 2.) Checksum computed after transfer both of DPE and instrument does not differ.

3.3.20.6.2 Use/Constraints

Must be used by context restoring procedures only which have to obey definition above.

3.3.20.6.3 Description

sets flags inside module to be reported by CRF.

3.3.20.7 Procedure DOWNLOAD_CONTEXT

3.3.20.7.1 Use/Constraints

Call this procedure when the context of all instruments must be saved in DPE (Eclipse).

3.3.20.7.2 Description

- for each instrument download table into DPE
- if successfully done -> compare CRC
- set context saving flags according result.
- if false -> generate on event message.
- set context restore flags (CRF) false.

3.3.20.8 Procedure UPLOAD_CONTEXT

3.3.20.8.1 Use/Constraints

Call this procedure when the context of all instruments must be restored from DPE to instruments. (after Eclipse)

3.3.20.8.2 Description

- for each instrument upload table into DPE if context save flag for the instrument is true.
- if successfully done -> compare CRC
- set context restoring flags according result.
- if false -> generate on event message.

3.3.21 MODULE ISGRI

- Procedure SWITCH_BIAS_OFF
- Procedure COMMAND_ISGRI
- Procedure READ_ISGRI
- Procedure GET_SEQ_COUNT

3.3.21.1 Global Variables

3.3.21.2 Exceptions

3.3.21.3 Procedure SWITCH_BIAS_OFF

3.3.21.3.1 Use/Constraints

Must be called when radiation increases or context- is saved.

3.3.21.3.2 Description

Sets ISGRI to standby and sends ISGRI bias 0 commands according ECR state.

3.3.21.4 Procedure COMMAND_ISGRI

3.3.21.4.1 Input

- ISGRI_COMMAND - the command to be send to ISGRI according command ID above.
- PARAMETER - The parameter necessary for this command
- MCE - the Module Control Electronics number

3.3.21.4.2 Output

: ok is true when transfer successfully done

3.3.21.4.3 Use/Constraints

Call the procedure with given command ID and parameter for that command to set given MCE NO PARAMETER CHECK WILL BE DONE THERE IS NO INTERNAL BLOCKING OF THE LSL

3.3.21.4.4 Use/Constraints

THE LINE MANAGER MUST BE USED BY THE CALLER OF THIS PROCEDURE The ISGRI Dump or HK command MUST NOT be used with this procedure for these commands return a response.

3.3.21.4.5 Description

Command_Isgri transmits the command via the LSL to ISGRI. The control will return when transmission is over. The length of the command is set internally according parameter block length.

3.3.21.5 Procedure READ_ISGRI

3.3.21.5.1 Input

ISGRI_COMMAND - The read command to be sent to ISGRI PARAMETER - The parameter necessary for this MCE - the Module Control Electronics number command

3.3.21.5.2 Output

ISGRI_RESPONSE : The data read out from ISGRI. This array must be set properly at the expected data size from ISGRI. ok : true when transfer successfully done

3.3.21.5.3 Use/Constraints

Call Read_Isgri with a command which requests data from ISGRI, and necessary parameters. The procedure will return the isgri response data according the command sent into response buffer. NO PARAMETER CHECK WILL BE DONE ALSO NO COMMAND ID CHECK IS APPLIED THERE IS NO INTERNAL BLOCKING OF THE LSL

3.3.21.5.4 Use/Constraints

THE LINE MANAGER MUST BE USED BY THE CALLER OF THIS PROCEDURE

3.3.21.5.5 Description

Command_Isgri transmits the command via the LSL to ISGRI into given MCE. Then the ISGRI will send the data requested via LSL to DPE . The control will return to caller when transmission is over. The length of the command to send is derived from the parameter block length; that of the response from the response block length.

3.3.21.6 Procedure GET_SEQ_COUNT

3.3.21.6.1 Output

Returns: 13-bit value containing current number of commands sent to any ISGRI MCE. Value wraps around at 2^13.

3.3.21.6.2 Use/Constraints

In HK for setting the ISGRI sequence count field. returns 13-Bit value

3.3.21.6.3 Description

Reflects counter status.

3.3.22 MODULE ISGRI_CONTEXT_TABLE

3.3.22.1 Global Variables

3.3.22.2 Exceptions

3.3.23 MODULE LINE_MANAGER

- Procedure INITIALISE
- Procedure LSL_IS_BLOCKED
- Procedure GET_LSL
- Procedure RELEASE_LSL
- Procedure GET_HSL
- Procedure RELEASE_HSL
- Procedure ANNOUNCE_LSL
- Procedure LSL_CONFIRMED

3.3.23.1 Global Variables

3.3.23.2 Exceptions

3.3.23.2.1 EX_NESTED_BLOCKING

for a certain task double allocation of a line (lsl/hsl) must not occur for the task would blockentirely. Therefore an exception will be raised.The exception remains at line manager task.

3.3.23.3 Procedure INITIALISE

3.3.23.3.1 Use/Constraints

Must be called when starting up.

3.3.23.3.2 Description

Connects internal interrupt routine Handle_HSSL to interrupt 13 which sends a signal to task which blocked the HSL.

3.3.23.4 Procedure LSL_IS_BLOCKED

3.3.23.4.1 Output

RETURNS Boolean value: FALSE when LSL is currently not blocked, TRUE when LSL is blocked.

3.3.23.4.2 Use/Constraints

This function could be used for checking before accessing the low speed line whether the line is actually blocked.

3.3.23.4.3 Description

Returns module flag LSL_Blocked.

3.3.23.5 Procedure GET_LSL

3.3.23.5.1 Use/Constraints

This procedure (or ANNOUNCE_LSL,s.b.) must be called before the LSL is to be used.

3.3.23.5.2 Description

IF lsl free THEN block lsl ELSE wait till lsl free.

3.3.23.6 Procedure RELEASE_LSL

3.3.23.6.1 Use/Constraints

This procedure must be called after a LSL transfer- was finished (successfully or not). The transfer end will be signalled by events. It is allowed to call this procedure even when not a GET_LSL is performed; so the LSL access state could be reseted by this procedure (eg. in exception cases).

3.3.23.6.2 Description

Sends a message to all tasks which are waiting for the lsl free and releases blocking flag. If a former announcement was performed by another task this task gets the LSL. If the current task made the announcement it will be cleared.

3.3.23.7 Procedure GET_HSL

3.3.23.7.1 Use/Constraints

This procedure must be called before the HSL is to be used.

3.3.23.7.2 Description

IF hsl free THEN block hsl ELSE wait till hsl free.

3.3.23.8 Procedure RELEASE_HSL

3.3.23.8.1 Use/Constraints

This procedure must be called after a HSL transfer- was finished (successfully or not). The transfer end will be signalled by events.

3.3.23.8.2 Description

Sends a message to all tasks which are waiting for the hsl free and releases blocking flag.

3.3.23.9 Procedure ANNOUNCE_LSL

3.3.23.9.1 Description

The serial line manager stores the task id of the one who called announce_lsl and gives access to the lsl for this task when the currently lsl using

3.3.23.10 Procedure LSL_CONFIRMED

3.3.23.10.1 Use/Constraints

Call this function to inquire whether a lsl announcement was confirmed.

3.3.23.10.2 Description

returns true when calling task has the lsl access.

3.3.24 MODULE LSL_INTERFACE

3.3.24.1 Global Variables

3.3.24.2 Exceptions

3.3.25 MODULE MODE_DISPATCHER

3.3.25.1 Global Variables

3.3.25.2 Exceptions

3.3.26 MODULE OBT_OPERATIONS

3.3.26.1 Global Variables

3.3.26.2 Exceptions

3.3.27 MODULE OEM_INTERFACE

- Procedure REPORT_OEM
- Procedure EXCEPTION_OEM
- Procedure PERIPHERAL_OEM
- Procedure GET_OEM_STATE

3.3.27.1 Global Variables

3.3.27.2 Exceptions

3.3.27.3 Procedure REPORT_OEM

3.3.27.3.1 Input

- OEM_ID : The on event message ID which defines on event message class and identifier.
- OEM_PARAM : The 6 bit parameter in first field of on event message.
- MESSG1 : first word of on event message.
- MESSG2 : second word of on event message.

3.3.27.3.2 Use/Constraints

Should be called in defined situations with OEM ID according this context; Parameter should be set to provide information fo the on event.

3.3.27.3.3 Use/Constraints

EXCEPTIONS ANOTHER ON-EVENT FUNCTION MUST BE USED.

3.3.27.3.4 Description

Writes on event according oem_id and parameter into if it is not full. Otherwise set on event flag. On event message counter will be increased.

3.3.27.4 Procedure EXCEPTION_OEM

3.3.27.4.1 Input

Exception ID

3.3.27.4.2 Use/Constraints

Call this function each time when an exception was trapped with a unique Identifier according Exception_ID.

3.3.27.5 Procedure PERIPHERAL_OEM

3.3.27.5.1 Input
ON event message field containing given on event message.

3.3.27.5.2 Use/Constraints
Call this function when receiving a housekeeping packet from external instruments (HEPI, ISGRI, PICSIT, VETO) with given On event message.

3.3.27.6 Procedure GET_OEM_STATE

3.3.27.6.1 Output
Returns : record T_OEM_STATE which contains:
- A flag OEM_SKIPPED which is true when on event messages are tried to generate but the OEM buffer is already full. This flag will be reset when-calling this function (this is explicitely a SIDE EFFECT)
- OEM_COUNTER - the number of generated on event messages modulo maximum.

3.3.27.6.2 Use/Constraints
Should be called from housekeeping task when generating HK-TM to insert the OEM status. KEEP CARE OF RESETING OEM_SKIPPED FLAG BY CALLING THIS FUNCTION

3.3.27.6.3 Description
A counter is increased each OEM generated. The OEM_SKIPPED flag is set each time a on event message could not be written into the ICB buffer. It will be reset by calling this functioin.

3.3.28 MODULE PICSIT

- Procedure COMMAND_PICSIT
- Procedure READ_PICSIT
- Procedure GET_SEQ_COUNT

3.3.28.1 Global Variables

3.3.28.2 Exceptions

3.3.28.3 Procedure COMMAND_PICSIT

3.3.28.3.1 Input
PICSIT_COMMAND - The command to be send to PICSIT PARAMETER - The parameter for that command.

3.3.28.3.2 Output
: ok is true when transfer successfully done

3.3.28.3.3 Use/Constraints
Call the procedure with given command ID and parameter for that command. NO PARAMETER CHECK WILL BE DONE THERE IS NO INTERNAL BLOCKING OF THE LSL

3.3.28.3.4 Use/Constraints

THE LINE MANAGER MUST BE USED BY THE CALLER OF THIS PROCEDURE

3.3.28.3.5 Description

Command_Picsit transmits the command via the LSL to PICSIT. The control will return when transmission is over. The length of the command is set according the length of the parameter array.

3.3.28.4 Procedure READ_PICSIT

3.3.28.4.1 Input

PICSIT_COMMAND - The read command to be sent to PICSIT PARAMETER   - The parameter necessary for this command

3.3.28.4.2 Output

PICSIT_RESPONSE : The data read out from PICSIT. This array must be set properly at the expected data size from PICSIT. ok   : true when transfer successfully done

3.3.28.4.3 Use/Constraints

Call Read_Picsit with a command which requests data from PICSIT, and necessary parameters. The procedure will return the picsit response data according the command sent into response buffer. NO PARAMETER CHECK WILL BE DONE ALSO NO COMMAND ID CHECK IS APPLIED   THERE IS NO INTERNAL BLOCKING OF THE LSL

3.3.28.4.4 Use/Constraints

THE LINE MANAGER MUST BE USED BY THE CALLER OF THIS PROCEDURE

3.3.28.4.5 Description

Command_Picsit transmits the command via the LSL to PICSIT. Then the PICSIT will send the data requested via LSL to DPE . The control will return to caller when transmission is over. The length of the command to send is derived from the parameter block length; that of the response from the response block length.

3.3.28.5 Procedure GET_SEQ_COUNT

3.3.28.5.1 Output

Returns: 13-bit value containing current number of commands sent to PICSIT. Value wraps around at 2^13.

3.3.28.5.2 Use/Constraints

In HK for setting the PICSIT sequence count field. returns 13-Bit value

3.3.28.5.3 Description

Reflects counter status.

3.3.29 MODULE PICSIT_CONTEXT_TABLE

3.3.29.1 Global Variables

3.3.29.2 Exceptions

3.3.30 MODULE READ_HISTOGRAM

- Procedure WAIT
- Procedure INITIALISE
- Procedure BUFFER_IS_EMPTY
- Procedure START
- Procedure STOP
- Procedure RESUME
- Procedure RESUME_TIMED
- Procedure DISCARD_HISTOGRAMS
- Procedure GET_BLOCK_RATE
- Procedure GET_BLOCK_INFO
- Procedure GET_FILLED_AREA

3.3.30.1 Global Variables

3.3.30.2 Exceptions

3.3.30.2.1 EX_BUFFER_NOT_EMPTY

this exception will be raised when it will be tried to start reading out a certain histogrambuffer though the former is not read out entirely.

3.3.30.3 Procedure WAIT

3.3.30.3.1 Use/Constraints

Call when using Read Histogram Blocks and wait for next RTC.

3.3.30.3.2 Use/Constraints

BE USED IN READ_HISTOGRAM_TASK ONLY

3.3.30.3.3 Description

Waits for next RTC or occurrence of Stop signal.

3.3.30.4 Procedure INITIALISE

3.3.30.4.1 Use/Constraints

Must be called by the init-read-task routine. This is the entry point of the read histogram task.

3.3.30.4.2 Description

- Waits till start signal occurs.
- When occurs -> perform read histogram task by fetching blocks from HEPI and process them internally by linking data down as TM.
- Terminate itself when stop signal occurs
- Starts where it stopped after RESUME (below)

3.3.30.5 Procedure BUFFER_IS_EMPTY

3.3.30.5.1 Input

HISTOGRAM_TYPE - the type of histogram to be checked (science or calibration)

3.3.30.5.2 Output

Returns true if last started buffer of given histogram type is empty, otherwise false.

3.3.30.5.3 Use/Constraints

Should be called before starting read out of new histogram buffer to avoid read out interferences.

3.3.30.5.4 Description

Returns empty-Value of histogram buffer according given type.

3.3.30.6 Procedure START

3.3.30.6.1 Input

- HISTOGRAMS : The histogram buffers which are inte- grated and now ready to be read out. The buffer might be of type SCIENCE or CALIBRATION.
- READ_TIME : the time (in OBT units = 2^-19 sec) the reading of given buffers should last. From this time the period will be computed.

3.3.30.6.2 Use/Constraints

This function should be called when integration of a certain buffer (science or calibration) has finished. Then the read histogram task starts reading out of the buffer. READ TASK MUST HAVE FINISHED, I.E. READ BUFFER MUST BE EMPTY. OTHERWISE AN EXCEPTION WILL BE RAISED.

3.3.30.6.3 Description

If current read buffer of given type is empty it will- be overridden by the argument HISTOGRAMS. Otherwise exception BUFFER_NOT_EMPTY will be raised. The the read out time will be computed and finally read out task be signalled to start reading out.

3.3.30.7 Procedure STOP

3.3.30.7.1 Use/Constraints

Call when histogram acquisition must stop but later on continue (with Procedure RESUME). Especially when leaving science mode but desiring to continue later on.

3.3.30.7.2 Description

Signals the read task to terminate itself.

3.3.30.8 Procedure RESUME

3.3.30.8.1 Use/Constraints

Call this procedure after STOP was called and rest of histograms should be read out. Do not call when STOP was not performed.

3.3.30.8.2 Description

Starts histogram reading task again.

3.3.30.9 Procedure RESUME_TIMED

3.3.30.9.1 Input

- READ_TIME : the time (in OBT units = 2^-19 sec) the reading of given buffers should last. From this time the period will be computed.

3.3.30.9.2 Use/Constraints

Call this procedure after STOP was called and rest of histograms should be read out in given time (the time is the upper margin). Do not call when STOP was not performed.

3.3.30.9.3 Description

- Sets reading time for both science/calibration histograms.
- Starts histogram reading task again.

3.3.30.10 Procedure DISCARD_HISTOGRAMS

3.3.30.10.1 Use/Constraints

Call this procedure when current read histogram should not be read out due non-valid data. Could be used when new histogram should be read out imminent.

3.3.30.10.2 Description

sets all histograms to be read empty and forces immediate read by reseting read counters.

3.3.30.11 Procedure GET_BLOCK_RATE

3.3.30.11.1 Output

Returns positive number of long integer. Defines number- of blocks read via HBR B.

3.3.30.11.2 Use/Constraints

Must be called periodically by HK function to read current rate of blocks read via HBR B.

3.3.30.11.3 Description

A counter is increased each time a block is success- fully read out. The counter will be reseted after returning the value. The counter also will be reseted when read histogram task is started.

3.3.30.12 Procedure GET_BLOCK_INFO

3.3.30.12.1 Output

- HBR_B_BLOCK_CAL : current block number read of calibration histogram.
- HBR_B_BLOCK_POLAR: current block number read of polarimetry histogram.
- HBR_B_BLOCK_SINGLE: current block number read of CsI single histogram.
- HBR_B_BLOCK_MULTI: current block number read of CsI single histogram.

3.3.30.12.2 Use/Constraints

To report histogram read block status in Science HK.

3.3.30.12.3 Description

Extracts block numbers read for histograms according address currently used for reading.

3.3.30.13 Procedure GET_FILLED_AREA

3.3.30.13.1 Output

Returns 8-bit number representing the histogram buffers as follows: 0 - CsI single, buffer 0 1 - CsI multiple, buffer 0 2 - Polarimetry, buffer 0 3 - CsI calibration histogram buffer 0 4 - CsI single, buffer 1 5 - CsI multiple, buffer 1 6 - Polarimetry, buffer 1 7 - CsI calibration histogram, buffer 1

3.3.30.13.2 Use/Constraints

May be used for HK information.

3.3.30.13.3 Description

Defines, which science/calibration buffer currently are not empty. These are the buffers also which is read from.

3.3.31 MODULE S1 (CDTE PHOTON BY PHOTON)

- **List of Contents:**

  Procedure PROCESS_EVENT
  Procedure CLOSE_TM
  Procedure RESET
  Function GET_SEQUENCE_COUNTER return
  CSSW_IF_TYPES.T_Source_Seq_Count.

- **Global Variables:**

HIGH_RISETIME_THRESHOLD: threshold value for CdTe PPM risetime selection. The default value of HIGH_RISETIME_THRESHOLD = 128 (default value corresponds to no selection).
Could be set via TC(5,3)(TID=16, FID=0). Could be read out via TC(5,4)(TID=16, FID=0). Parameter is reported in additional housekeeping TM packet H2.

CDTE_E_MAX: threshold value for CdTe PPM amplitude selection. The default value of CDTE_E_MAX = 65535 (in units of [keV], default value corresponds to no selection).
Could be set via TC(5,3)(TID=16, FID=1). Could be read out via TC(5,4)(TID=16, FID=1). Parameter is reported in additional housekeeping TM packet H2.

3.3.31.1 Procedure PROCESS_EVENT

- **Input:** 80 bit event of type Hex(24) according to HEPI Interface Document from module HBR_A_IF
- **Output:** TM(1,0) packet to module TM_INTERFACE
- **Called by modules:** SCIENCE_MODE
- **Description:**

This procedure contains all working steps to pack a CdTe photon by photon event into a TM packet. First it is checked whether the scientific function S1 (CdTe PPM) has been restarted by module TM_INTERFACE. In case of restart a dummy event is written into the TM packet.
Then the amplitude correction function AMPLITUDE_CORRECTION. ISGRI_AMP_CORR is called to correct the CdTe event amplitude and risetime. It is checked whether the corrected risetime and amplitude are smaller than the corresponding threshold values. The limiting values CDTE_E_MAX and HIGH_RISETIME_THRESHOLD are configurable via TC(5,3), could be read out via TC(5,4) and can be verified via the additional IASW HK (H2). If the corrected risetime and amplitude are in defined ranges the event is packed into 40 bits and written into the TM packet. From the first event of the TM packet the 24 MSB of 32 bit time information are written into the data field header. The 8 LSB are written into the time field of the first TM event record. So the whole 32 bit time information of the first event of a TM packet can be reconstructed. From all other events inside the TM packet only the 8 LSB of the time information are carried over.

<table>
<tr><td colspan="3">Amplitude (11 bit)</td><td>MSB Risetime (7 bit)</td></tr>
<tr><td>Rt.
LSB</td><td colspan="2">Y_position (7 bit)</td><td>Z-position (7 bit)</td></tr>
<tr><td colspan="2">Event time (8 bit = 8 LSB of 32 bit)</td><td colspan="2"></td></tr>
</table>

<table>
<tr><td colspan="2"></td><td colspan="3">Event time (8 bit = 8 LSB of 32 bit)</td></tr>
<tr><td colspan="3">Amplitude (11 bit)</td><td>MSB</td><td>Risetime (7 bit)</td></tr>
<tr><td>Rt.
LSB</td><td>Y_position (7 bit)</td><td colspan="3">Z-position (7 bit)</td></tr>
</table>

Figure 1: Formats of 40 bit event records

If the time difference between the previous event and the current event is more than the maximum value of the 8 bit time field of the TM event record then a 40 bit dummy event record will be written into the TM packet before the current event. The dummy record has got the same structure like the TM event record, but all values are set to a defined pattern except the time field. The time field keeps a dummy counter of 8 bit length which shows how often the 8 bit event time field has overflowed.

<table>
<tr><td colspan="3">Amplitude (11 bit) = 2047</td><td>MSB</td><td>Risetime (7 bit) = 0</td></tr>
<tr><td>Rt.
LSB</td><td>Y_position (7 bit) = 127</td><td colspan="3">Z-position (7 bit) = 0</td></tr>
<tr><td colspan="2">Dummy counter</td><td colspan="3"></td></tr>
</table>

<table>
<tr><td colspan="2"></td><td colspan="3">Dummy counter</td></tr>
<tr><td colspan="3">Amplitude (11 bit) = 2047</td><td>MSB</td><td>Risetime (7 bit) = 0</td></tr>
<tr><td>Rt.
LSB</td><td>Y_position (7 bit) = 127</td><td colspan="3">Z-position (7 bit) = 0</td></tr>
</table>

Figure 2: Formats of 40 bit dummy records

The maximum value allowed for this dummy counter is Hex(FF-2). The value Hex(FF-1) for the dummy counter is used as flag that the data processing is restarted, the maximum value Hex(FF) is used as a flag that the counter itself has overflowed. In this case of counter overflow two dummy records are written into the TM packet before the current event. The first dummy keeps the counter with the flag value Hex(FF). The second dummy keeps the 32 bit time information of the current event inside the event record partitions of *Amplitude, Risetime, Y_Pos* and *Z_Pos*. The dummy counter inside the time field is set to Hex(FF). If the TM packet is filled it will be given to the TM_INTERFACE module by calling the procedure TM_INTERFACE.PUT_TM .

- **Constraints:** None.
- **Efficiency:** Not applicable
- **Verification Requirement:** TBD

3.3.31.2 Procedure CLOSE_TM

- **Input:** None
- **Output:** TM(1,0) packet to module TM_INTERFACE
- **Called by modules:** SCIENCE_MODE
- **Description:**

When ending the science mode the partial filled TM(1,0) packet must be filled with defined values Hex(FFFF) and sent away via the procedure TM_INTERFACE.PUT_TM out of the module TM_INTERFACE. If the current TM packet contains no event the TM packet is not filled and is not sent away.

- **Constraints:** None.
- **Efficiency:** Not applicable
- **Verification Requirement:** TBD

3.3.31.3 Procedure RESET

- **Input:** None
- **Output:** None
- **Called by modules:** SCIENCE_MODE
- **Description:**

When starting the science mode the variables for the current position inside the TM packet of the processing procedure S1.PROCESS_EVENT must be reset to initialisation values to make sure that the routine begin with correct start values.

- **Constraints:** None
- **Efficiency:** Not applicable
- **Verification Requirement:** TBD

3.3.31.4 Function GET_SEQUENCE_COUNTER return CSSW_IF_TYPES.T_Source_Seq_Count

- **Input:** None
- **Output:** Sequence Counter of the current TM(1,0) packet
- **Called by modules:** SCIENCE_HK
- **Description:**

This function returns the sequence counter of the TM(1,0) packet, which currently is to fill by procedure S1.PROCESS_EVENT. That means the returned value is the sequence counter of the next TM(1,0) packet that will be given to the TM_INTERFACE.

- **Constraints:** None
- **Efficiency:** Not applicable
- **Verification Requirement:** TBD

3.3.32 MODULE S2 (CDTE CALIBRATION)

- **List of Contents:**

  Procedure PROCESS_EVENT
  Procedure CLOSE_TM
  Procedure RESET
  Function GET_SEQUENCE_COUNTER return
  CSSW_IF_TYPES.T_Source_Seq_Count.

- **Global Variables:** None.

3.3.32.1 Procedure PROCESS_EVENT

- **Input:** 80 bit event of type Hex(34) according to HEPI Interface Document from module HBR_A_IF
- **Output:** TM(2,0) packet to module TM_INTERFACE
- **Called by modules:** SCIENCE_MODE
- **Description:**

This procedure contains all working steps to pack a CdTe calibration event into a TM packet.
First it is checked whether the scientific function S2 (CdTe calibration) has been restarted by module TM_INTERFACE. In that case one 32 bit dummy event consistent out of Hex(FFFF) is written into the TM packet. Then the CdTe calibration event is compressed to 32 bit and written into a TM(2,0) packet.

Amplitude (11 bit)			MSB Risetime (7 bit)
Rt. LSB	Y_position (7 bit)	Z-position (7 bit)	

Figure 3: Formats of 32 bit event records

The time information of the event is not considered. Only the 24 MSB of the 32 bit time information of the first event of the TM packet are written into the data field header. If the TM packet is filled it will be given to the TM_INTERFACE module by calling the procedure TM_INTERFACE.PUT_TM.

- **Constraints:** None
- **Efficiency:** Not applicable
- **Verification Requirement:** TBD

3.3.32.2 Procedure CLOSE_TM

- **Input:** None
- **Output:** TM(2,0) packet to module TM_INTERFACE
- **Called by modules:** SCIENCE_MODE
- **Description:**

When ending the science mode the partial filled TM(2,0) packet must be filled with defined values Hex(FFFF) and sent away via the procedure TM_INTERFACE.PUT_TM out of the module TM_INTERFACE. If the current TM packet contains no event the TM packet is not filled and is not sent away.

- **Constraints:** None.
- **Efficiency:** Not applicable
- **Verification Requirement:** TBD

3.3.32.3 Procedure RESET

- **Input:** None
- **Output:** None
- **Called by modules:** SCIENCE_MODE
- **Description:**

When starting the science mode the variable for the current position inside the TM packet of the processing procedure S2.PROCESS_EVENT is reset to initialisation value to make sure that the routine begins with correct start value.

- **Constraints:** None
- **Efficiency:** Not applicable
- **Verification Requirement:** TBD

3.3.32.4 Function GET_SEQUENCE_COUNTER return CSSW_IF_TYPES.T_Source_Seq_Count

- **Input:** None
- **Output:** Sequence Counter of the current TM(2,0) packet
- **Called by modules:** SCIENCE_HK
- **Description:**

This function returns the sequence counter of the TM(2,0) packet, which currently is to fill by procedure S2.PROCESS_EVENT. That means the returned value is the sequence counter of the next TM(2,0) packet that will be given to the TM_INTERFACE.

- **Constraints:** None
- **Efficiency:** Not applicable
- **Verification Requirement:** TBD

3.3.33 MODULE S3 (COMPTON)

- **List of Contents:**
  Procedure PROCESS_S_EVENT,
  Procedure PROCESS_M_EVENT
  Procedure CLOSE_S_TM,
  Procedure CLOSE_M_TM
  Procedure RESET
  Function GET_S_SEQUENCE_COUNTER return System.Unsigned
  Function GET_M_SEQUENCE_COUNTER return System.Unsigned
  Function C_SELECT return Boolean

- **Global Variables:**

E_COMPTON_THRESHOLD: threshold value for performing compton computation. The default value of E_COMPTON_THRESHOLD = 1000 (in units of [keV]).
Could be set via TC(5,3)(TID=16, FID=2). Could be read out via TC(5,4)(TID=16, FID=2). Parameter is reported in additional housekeeping TM packet H2.

MINIMUM_COMPTON_ANGLE: threshold value for compton selection. The default value of MINIMUM_COMPTON_ANGLE = 0,968 (corresponds to cos(14,5°).
Could be set via TC(5,3)(TID=16, FID=6). Could be read out via TC(5,4)(TID=16, FID=6). Parameter is reported in additional housekeeping TM packet H2.

S_CALIBRATION_FACTOR: scaling factor of CsI single event amplitude to get the amplitude in units of keV. The default value of S_CALIBRATION_FACTOR = 5,0.
Could be set via TC(5,3)(TID=16, FID=4). Could be read out via TC(5,4)(TID=16, FID=4). Parameter is reported in additional housekeeping TM packet H2.

M_CALIBRATION_FACTOR: scaling factor of CsI multiple event amplitude to get the amplitude in units of keV. The default value of M_CALIBRATION_FACTOR = 10,0.
Could be set via TC(5,3)(TID=16, FID=5). Could be read out via TC(5,4)(TID=16, FID=5). Parameter is reported in additional housekeeping TM packet H2.

3.3.33.1 Procedures PROCESS_S_EVENT

- **Input:** 160 bit event of type Hex(A4, B4) according to HEPI Interface Document from module HBR_A_IF.

- **Output:** TM(3,0) packet to module TM_INTERFACE

- **Called by modules:** SCIENCE_MODE

- **Description:**

This procedures contains all working steps to pack a compton single photon by photon event or a compton single calibration event into a TM packet. First it is

checked whether the scientific function S3.0 (Compton PPM/calibration) has been restarted by module TM_INTERFACE. In case of restart a dummy event is written into the TM packet.
Then it is checked whether the event is a PPM or a calibration event. For compton single calibration events there are no selection mechanisms. If the not scaled CsI event amplitude of a compton single PPM event is not higher than the threshold value E_COMPTON_THRESHOLD then the CsI amplitude is scaled using S_CALIBRATION_FACTOR to keV and the amplitude correction function AMPLITUDE_CORRECTION.COMPTON_AMP_CORR is called to correct the CdTe event amplitude. Then it is checked whether the input event is a real compton event by calling the compton selection function C_SELECT. The function C_SELECT returns a boolean value, true means real compton event, false means no compton event. If the event is no real compton event then it is rejected and not processed. If the event is a real compton event then it is compressed to 64 bits and written into a TM packet. The event format is the same for compton single PPM events and compton single calibration events. Calibration events are signed with calibration flag field = 0, photon by photon events are signed with calibration flag field = 1. From the first event of the TM packet the 24 MSB of 32 bit time information are written into the data field header. The 8 LSB are written into the time field of the first TM event record. So the whole 32 bit time information of the first event of a TM packet can be reconstructed.

CsI amplitude (8 MSB of 10 bit)		CdTe amplitude (8 MSB of 11 bit)	
CdTe risetime (7 bit)	CsI Y-position (6 bit)	MSB CsI Z-	
position (6 bit)	CdTe Y-position (7 bit)	MSB CdTe Z-position (7 bit)	
LSB	CF	Dummy counter (6 bit)	Event time (8 LSB out of 32 bit)

Figure 4: Formats of 64 bit event records

From all other events of the TM packet only the 8 LSB of the time information are carried over. If the time difference between the previous event and the current event is more than the maximum value of the 8 bit time field of the TM event record then the dummy field of the event record keeps a counter of 6 bit length which shows how often the 8 bit event time field has overflowed. The maximum value allowed for this dummy counter is Hex(3F-2). The value Hex(3F-1) for the dummy counter is used as flag that the data processing is restarted, the maximum value Hex(3F) is used as a flag that the counter itself has overflowed. In this case a dummy record is written into the TM packet before the current event. The dummy keeps the 32 bit time information of the current event inside the first 32 bit of the event record. The dummy field is set to maximum value Hex(3F), the calibration flag is set to 1 and all other bits are set to zero.

MSB	32 bit event time (16 MSB)

32 bit event time (16 LSB) LSB			
16 bit set to zero			
0	1	Dummy counter = 62 (6 bit)	8 bit set to zero

Figure 5: Formats of 64 bit dummy records

If the TM packet is filled it will be given to the TM_INTERFACE module by calling the procedure TM_INTERFACE.PUT_TM .

- **Constraints:** None.
- **Efficiency:** Not applicable
- **Verification Requirement:** TBD

3.3.33.2 Procedures PROCESS_M_EVENT

- **Input:** 160 bit event of type Hex(A8) according to HEPI Interface Document from module HBR_A_IF.
- **Output:** TM(3,1) packets to module TM_INTERFACE
- **Called by modules:** SCIENCE_MODE
- **Description:**

The performed sequences are similar to description of procedure PROCESS_S_EVENT. There are no compton multiple calibration events, the calibration flag field is always set to 1. The event format and dummy format is the same as in procedure PROCESS_S_EVENT. The scaling factor for CsI multiple amplitude to get amplitude in keV is M_CALIBRATION_FACTOR.

- **Constraints:** None.
- **Efficiency:** Not applicable
- **Verification Requirement:** TBD

3.3.33.3

3.3.33.4 Procedures CLOSE_S_TM / CLOSE_M_EVENT

- **Input:** None

- **Output:** TM(3,0) / TM(3,1) packets to module TM_INTERFACE
- **Called by modules:** SCIENCE_MODE
- **Description:**

When ending the science mode the partial filled TM(3,0) / TM(3,1) packets must be filled with defined values Hex(FFFF) and sent away via the procedure PUT_TM out of the module TM_INTERFACE. If the current TM packets contain no event the TM packet is not filled and is not sent away.

- **Constraints:** None.
- **Efficiency:** Not applicable
- **Verification Requirement:** TBD

3.3.33.5 Procedure RESET

- **Input:** None
- **Output:** None
- **Called by modules:** SCIENCE_MODE
- **Description:**

When starting the science mode variables for the current position inside the TM packet of the processing procedures S3.PROCESS_S/M_EVENT are reset to initialisation values to make sure that the routines begin with correct start values.

- **Constraints:** None.
- **Efficiency:** Not applicable
- **Verification Requirement:** TBD

3.3.33.6 Functions GET_S3_HK_DATA return T_S3_Hk_Data

- **Input:** None

- **Output:** Sequence Counter of the current TM(3,0) / TM(3,1) packet
- **Called by modules:** SCIENCE_HK
- **Description:**

This functions returns the sequence counters of the TM(3,0) / TM(3,1) packets, which currently are to fill by procedures S3.PROCESS_S/M_EVENT. That means the returned values are the sequence counters of the next TM(3,0) / TM(3,1) packet that will be given to the TM_INTERFACE. Type T_S3_Hk_Data is a global record type built of two components of CSSW_IF_TYPES.T_Source_Seq_Count.

- **Constraints:** None
- **Efficiency:** Not applicable
- **Verification Requirement:** TBD

3.3.33.7 Function C_SELECT return Boolean

- **Input:** CsI amplitude scaled to [keV], CsI Y/Z-positions, corrected CdTe amplitude, CdTe Y/Z-positions
- **Output:** Boolean: true = real compton - process event, false = no compton - reject event
- **Called by modules:** S3
- **Description:**

First all input parameter are converted from integers into floats. To take account the gaps between the detector modules, ISGRI is imagined as a matrix with 134x130 elements and PICsIT as a matrix of 67x65 elements. The conversions of the positions is done as followed. The multiplication by 2 is to put CsI pixel in units of CdTe pixels.

$Y(CdTe) = ((Y_{in}(CdTe) / 64) * 2) + Y_{in}(CdTe)$ $\qquad$ $Y(CsI) = ((Y_{in}(CsI) / 32) + Y_{in}(CsI)) * 2$

$Z(CdTe) = ((Z_{in}(CdTe) / 32) * 2) + Z_{in}(CdTe)$ $\qquad$ $Z(CsI) = ((Z_{in}(CsI) / 16) + Z_{in}(CsI)) * 2$

There is no algorithm to calculate the exact distance of the pixels in y - z plane. All position calculations are done in units of CdTe pixels. The units of CdTe pixel is calculated to 4,6mm taking in account the - size of a CdTe pixel = 4x4mm^2 and

- 0,6mm distance between two pixels.

The distance between the two detector layers (10 cm) is therefore 21,739 units of CdTe pixels.

For the compton selection the formula of doc. IN-IB-Sap-RP-045 (P.Laurent, 09/98) is implemented.:

$cDC1 = E_{CdTe} * mc^2 / (E_{CsI} / (E_{CsI} + E_{CdTe}))$
$cDC2 = E_{CsI} * mc^2 / (E_{CdTe} / (E_{CsI} + E_{CdTe}))$
$rDC = sqrt[(Y_{CsI} - Y_{CdTe})^2 + (Z_{CsI} - Z_{CdTe})^2]$

$oDC1 = \{H*(1-cDC1) + rDC * sqrt[cDC1 * (2-cDC2)]\} / sqrt[rDC^2+H^2]$
$oDC2 = \{rDC*sqrt[cDC2*(2*cDC2)] – H*(1*cDC2)\} / sqrt[rDC^2+H^2]$

If [(oDC1<1) and (oDC1>c_{min})] OR [(oDC2<1) and (oDC2>c_{min})] then Compton=True

with
cDC1, CdC2 = 1 – cosine of compton angles (1: first interaction in CdTe,
2: first interaction in CsI)
rDC = geometrical projected distance between the 2 points of interaction.
ODC1, oDC2 = reconstructed source angle (1: first interaction in CdTe,
2: first interaction in CsI)
mc^2 = 511 keV
H = distance between CdTe-layer and CsI-layer.
sqrt[] = square root (imported from IASW module SQRT)
c_{min} = cosine of the acceptance angle

To calculate the square roots the algorithm out of *TLD standard RTX module MATH* is used.
This square root algorithm is explained in description of IASW module SQRT.

- **Constraints:** None
- **Efficiency:** Not applicable
- **Verification Requirement:** TBD

3.3.34 MODULE S4 (CSI PHOTON BY PHOTON)

- **List of Contents:**
  Procedure PROCESS_SINGLE_EVENT,

Procedure PROCESS_MULTIPLE_EVENT
Procedure CLOSE_SINGLE_TM,
Procedure CLOSE_MULTIPLE_TM
Procedure RESET
Function GET_S4_HK_DATA return T_S4_Hk_Data.

- **Global Variables:**

3.3.34.1 Procedures PROCESS_SINGLE_EVENT / PROCESS_MULTIPLE_EVENT

- **Input:** 80 bit event of type Hex(64, 68) according to HEPI Interface Document from module HBR_A_IF

- **Output:** TM(4,0) / TM(4,1) packets to module TM_INTERFACE

- **Called by modules:** SCIENCE_MODE

- **Description:**

This procedures contain all working steps to pack a CsI single/multiple photon by photon event into a TM packet. First it is checked whether the scientific function S4.0 (CsI single PPM)/ S4.1 (CsI multiple PPM) has been restarted by module TM_INTERFACE. In case of restart a dummy event is written into the TM packet. Then the CsI single/multiple PPM event is compressed to 32 bits and written into a TM packet. From the first event of the TM packet the 24 MSB of 32 bit time information are written into the data field header. The 8 LSB are written into the time field of the first TM event record. So the whole 32 bit time information of the first event of a TM packet can be reconstructed. From all other events of the TM packet only the 8 LSB of the time information are carried over.

<table>
<tr><td colspan="2">Amplitude (10 bit)</td><td colspan="2">Y-Position (6 bit)</td></tr>
<tr><td>DF (2 bit)</td><td>Z-Position (6 bit)</td><td colspan="2">Event time (8 LSB out of 32 bit)</td></tr>
</table>

Figure 6: Formats of 32 bit event records

For CsI single PPM event the dummy flag field (DF) is always set to zero.
If the time difference between the previous event and the current event is more than the maximum value of the 8 bit time field of the TM event record then a 32 bit dummy event record will be written into the TM packet before the current event. The dummy record has got the same structure like the TM event record, but Amplitude and both y/z-position partitions are set to zero and the dummy flag field is set from zero to Hex(3). The time field keeps a counter of 8 bit length which shows how often the 8 bit event time field has overflowed.

Amplitude (10 bit) = 0		Y-Position (6 bit) = 0
DF = 3	Z-Position (6 bit) = 0	Dummy Counter

Figure 7: Formats of 32 bit dummy records

The maximum value allowed for this dummy counter is Hex(FF-2). The value Hex(FF-1) for the dummy counter is used as flag that the data processing is restarted, the maximum value Hex(FF) is used as a flag that the counter itself has overflowed. In this case two dummy records are written into the TM packet before the current event. The first dummy keeps the dummy flag field with Hex(3) and the counter with the flag value Hex(FF). The second dummy keeps the 32 bit time information of the current event inside the event record. If the TM packet is filled it will be given to the TM_INTERFACE module by calling the procedure TM_INTERFACE.PUT_TM.

- **Constraints:** None
- **Efficiency:** Not applicable
- **Verification Requirement:** TBD

3.3.34.2 Procedures CLOSE_SINGLE_TM / CLOSE_MULTIPLE_TM

- **Input:** None
- **Output:** TM(4,0) / TM(4,1) packets to module TM_INTERFACE
- **Called by modules:** SCIENCE_MODE
- **Description:**

When ending the science mode the partial filled TM(4,0) / TM(4,1) packets must be filled with defined values Hex(FFFF) and sent away via the procedure PUT_TM out of the module TM_INTERFACE. If the current TM packets contain no event the TM packet is not filled and is not sent away.

- **Constraints:** None.
- **Efficiency:** Not applicable
- **Verification Requirement:** TBD

3.3.34.3 Procedure RESET

- **Input:** None
- **Output:** None
- **Called by modules:** SCIENCE_MODE
- **Description:**

When starting the science mode variables for the current position inside the TM packet of the processing procedures PROCESS_SINGLE/MULTIPLE_EVENT are reset to initialisation values to make sure that the routines begin with correct start values.

- **Constraints:** None.
- **Efficiency:** Not applicable
- **Verification Requirement:** TBD

3.3.34.4 Functions GET_S4_HK_DATA return T_S4_Hk_Data

- **Input:** None
- **Output:** Sequence Counter of the current TM(4,0) / TM(4,1) packet
- **Called by modules:** SCIENCE_HK
- **Description:**

This functions returns the sequence counters of the TM(4,0) / TM(4,1) packets, which currently are to fill by procedures S4.PROCESS_SINGLE/MULTIPLE_EVENT. That means the returned values are the sequence counters of the next TM(4,0) / TM(4,1) packet that will be given to the TM_INTERFACE. Type T_S4_Hk_Data is a global record type built of two components of CSSW_IF_TYPES.T_Source_Seq_Count.

- **Constraints:** None
- **Efficiency:** Not applicable

- **Verification Requirement:** TBD

3.3.35 MODULE S5 (CSI-CALIBRATION)

- **List of Contents:**
  Procedure PROCESS_HISTOGRAM
  Procedure CLOSE_TM
  Procedure RESET
  Function GET_SEQUENCE_COUNTER return
  CSSW_IF_TYPES.T_Source_Seq_Count.

- **Global Variables:** None.

- **Exceptions:** EX_ADDRESS_NOT_CONTINUOUS

3.3.35.1 Procedures PROCESS_HISTOGRAM

- **Input:** Interface
  - one block (1026 byte) of histogram data interface according to HEPI Document from module READ_HISTOGRAM,
  - Histogram ID,
  - Start time of integration [OBT],
  - Integration time [seconds]

- **Output:** TM(5,0) packets to module TM_INTERFACE

- **Called by modules:** READ_HISTOGRAM

- **Description:**

The input data block consists out of 2 byte of block address and 1024 byte histogram data. This data corresponds to 1024 histogram cells each with size of one byte. First it is checked whether the start address of the histogram data block is continuous to previous blocks. If it was the first block, the check is omitted. If the address is not continuous the current TM packet is closed and sent away and the exception EX_ADDRESS_NOT_CONTINUOUS is raised.

S5 histogram TM packets have a 5 word data field header. The data field header will be filled before the first histogram data cell is written into the TM packet. The TM packet data field header contains all information about the histogram integration and the memory address of the histogram data:

Type/Subtype (8 bit) = Hex(50)	MSB Start time of histogram integration
Start time of histogram integration [seconds] (24 bit) LSB	
Real integration time of the histogram [seconds]	
MSB Start address of the reference histogram data cell (first data cell inside the TM packet)	
Start address of reference cell (24 bit) LSB	Histogram ID

Figure 8: Format of 80 bit histogram data field header.

Compression algorithm:

The compression algorithm tries to reduce the size of a 8 bit histogram cell into a 4 bit TM packet data cell using a "count difference method". The TM packet data field is built out of an array [1..848] of 4 bit data cells. When starting a new TM packet all data cells are initialised with flag value (-8).

The 24 bit histogram memory address given inside the TM packet data field header is the memory address of the first histogram cell which is written into the first two data cells of the TM packet. This first two TM packet data cells are equivalent to a uncompressed 8 bit histogram cell containing the origin count values for the corresponding pixel and energy channel. For the second histogram cell the difference of counts to the first histogram cell is calculated. This Δ(counts) of [histogram cell (1) – histogram cell (2)] is written as a 4 bit integer value into third TM packet data cell. The forth data cell contains the Δ(counts) of [histogram cell (2) – histogram cell (3)], the fifth data cell contains the Δ(counts) of [histogram cell (3) – histogram cell (4)] and so on. For Δ(counts) the range –7..+7 of a 4 bit integer type is used. The value –8 is used as a flag.

Δ(counts)	-8	-7	-6	-5	-4	-3	-2	-1	0	1	2	3	4	5	6	7
Bit code	1000	1001	1010	1011	1100	1101	1110	1111	0000	0001	0010	0011	0100	0101	0110	0111

Table 8: Bitcode of 4 bit integer values

If Δ(counts) is not in range of –7..+7 then the flag value (-8) is written into the TM packet data cell followed by the origin 8 bit histogram cell. The following example shows the working method of the histogram compression algorithm:

Histogram cells (origin data from HEPI) (8 bit)				TM packet data cells(4 bit)	
23	21	22	First → data cell = 8 bit keeping the origin	23 (8 bit)	-2

16	19	14		+1	-6	+3
9	2	2		-5	-7	0
24	25	23		-8	24 (8 bit)	
18	20	17		+1	-2	-5
17				+2	-3	0

Figure 9: Example of histogram compression algorithm.

The last 3 data cells of the TM packet are reserved for a position counter, therefore the maximum number of available data cells is 845. The value inside this position counter indicates the first data cell which contains no data. If the position counter = 846 all data cells are valid, if position counter = 845 then the last data cell is invalid.

To make sure that no histogram TM packet will be lost during short timed high data rate to ICB TM sub buffer the procedure repeats up to ten times the sending of the TM packet to the TM_INTERFACE in intervals of 125 milliseconds. If the TM packet could not be sent an on-event message is generated that the TM packet is lost.
As long as the satellite is changing ground stations the processing and sending of histogram TM packets is interrupted.

- **Constraints:** None
- **Efficiency:** Not applicable
- **Verification Requirement:** TBD

3.3.35.2 Procedures CLOSE_TM

- **Input:** None
- **Output:** TM(5,0) packets to module TM_INTERFACE
- **Called by modules:** READ_HISTOGRAM
- **Description:**

When finishing the histogram read out and processing the partial filled TM(5,0) packets sent away to the module TM_INTERFACE. If the current TM packets contain no data the TM packet is not sent away.

- **Constraints:** None.
- **Efficiency:** Not applicable

- **Verification Requirement:** TBD

3.3.35.3 Procedure RESET

- **Input:** None
- **Output:** None
- **Called by modules:** READ_HISTOGRAM
- **Description:**

When starting the histogram read out and processing variables of the processing procedure PROCESS_HISTOGRAM are reset to initialisation values to make sure that the routines begin with correct start values.

- **Constraints:** None.
- **Efficiency:** Not applicable
- **Verification Requirement:** TBD

3.3.35.4 Function GET_SEQUENCE_COUNTER return CSSW_IF_TYPES.T_Source_Seq_Count

- **Input:** None
- **Output:** Sequence Counter of the current TM(5,0) packet
- **Called by modules:** SCIENCE_HK
- **Description:**

This function returns the sequence counter of the TM(5,0) packet, which currently is to fill by procedure S5.PROCESS_HISTOGRAM. That means the returned value is the sequence counter of the next TM(5,0) packet that will be given to the TM_INTERFACE.

- **Constraints:** None

- **Efficiency:** Not applicable

- **Verification Requirement:** TBD

3.3.36 MODULE S6 (CSI POLARIMETRIC)

- **List of Contents:**
  Procedure PROCESS_HISTOGRAM
  Procedure CLOSE_TM
  Procedure RESET
  Function GET_SEQUENCE_COUNTER return
  CSSW_IF_TYPES.T_Source_Seq_Count.

- **Global Variables:** None.

- **Exceptions:** EX_ADDRESS_NOT_CONTINUOUS

3.3.36.1 Procedures PROCESS_HISTOGRAM

- **Input:** - one block (1026 byte) of histogram data interface according to HEPI Interface Document from module READ_HISTOGRAM,
  - Histogram ID,
  - Start time of integration [OBT],
  - Integration time [seconds]

- **Output:** TM(6,0) packets to module TM_INTERFACE

- **Called by modules:** READ_HISTOGRAM

- **Description:**

The input data block consists out of 2 byte of block address and 1024 byte histogram data. This data corresponds to 1024 histogram cells each with size of one byte. First it is checked whether the start address of the histogram data block is continuous to previous blocks. If it was the first block, the check is omitted. If the address is not continuous the current TM packet is closed and sent away and the exception EX_ADDRESS_NOT_CONTINUOUS is raised.

S6 histogram TM packets have a 5 word data field header. The data field header will be filled before the first histogram data cell is written into the TM packet. The TM packet data field header contains all information about the histogram integration and the memory address of the histogram data:

Type/Subtype (8 bit) = Hex(60)	MSB Start time of histogram integration
Start time of histogram integration [seconds] (24 bit) LSB	
Real integration time of the histogram [seconds]	
MSB Start address of the reference histogram data cell (first data cell inside the TM packet)	
Start address of reference cell (24 bit) LSB	Histogram ID

Figure 10: Format of 80 bit histogram data field header.

Compression algorithm:
The compression algorithm tries to reduce the size of a 8 bit histogram cell into a 4 bit TM packet data cell using a “count difference method”. The TM packet data field is built out of an array [1..848] of 4 bit data cells. When starting a new TM packet all data cells are initialised with flag value (-8).
The 24 bit histogram memory address given inside the TM packet data field header is the memory address of the first histogram cell which is written into the first two data cells of the TM packet. This first two TM packet data cells are equivalent to a uncompressed 8 bit histogram cell containing the origin count values for the corresponding pixel and energy channel. For the second histogram cell the difference of counts to the first histogram cell is calculated. This Δ(counts) of [histogram cell (1) – histogram cell (2)] is written as a 4 bit integer value into third TM packet data cell. The forth data cell contains the Δ(counts) of [histogram cell (2) – histogram cell (3)], the fifth data cell contains the Δ(counts) of [histogram cell (3) – histogram cell (4)] and so on. For Δ(counts) the range –7..+7 of a 4 bit integer type is used. The value –8 is used as a flag.

Δ(counts)	-8	-7	-6	-5	-4	-3	-2	-1	0	1	2	3	4	5	6	7
Bit code	1000	1001	1010	1011	1100	1101	1110	1111	0000	0001	0010	0011	0100	0101	0110	0111

Table 9: Bitcode of 4 bit integer values

If Δ(counts) is not in range of –7..+7 then the flag value (-8) is written into the TM packet data cell followed by the origin 8 bit histogram cell. The following example shows the working method of the histogram compression algorithm:

Histogram cells (origin data from HEPI) (8 bit)				TM packet data cells(4 bit)	
23	21	22	First → data cell = 8 bit keeping the origin	23 (8 bit)	-2

16	19	14		+1	-6	+3
9	2	2		-5	-7	0
24	25	23		-8	24 (8 bit)	
18	20	17		+1	-2	-5
17				+2	-3	0

Figure 11: Example of histogram compression algorithm.

The last 3 data cells of the TM packet are reserved for a position counter, therefore the maximum number of available data cells is 845. The value inside this position counter indicates the first data cell which contains no data. If the position counter = 846 all data cells are valid, if position counter = 845 then the last data cell is invalid.

To make sure that no histogram TM packet will be lost during short timed high data rate to ICB TM sub buffer the procedure repeats up to ten times the sending of the TM packet to the TM_INTERFACE in intervals of 125 milliseconds. If the TM packet could not be sent an on-event message is generated that the TM packet is lost.
As long as the satellite is changing ground stations the processing and sending of histogram TM packets is interrupted.

- **Constraints:** None
- **Efficiency:** Not applicable
- **Verification Requirement:** TBD

3.3.36.2 Procedures CLOSE_TM

- **Input:** None
- **Output:** TM(6,0) packets to module TM_INTERFACE
- **Called by modules:** READ_HISTOGRAM
- **Description:**

When finishing the histogram read out and processing the partial filled TM(6,0) packets is sent away to the module TM_INTERFACE. If the current TM packets contain no data the TM packet is not sent away.

- **Constraints:** None.
- **Efficiency:** Not applicable

- **Verification Requirement:** TBD

3.3.36.3 Procedure RESET

- **Input:** None
- **Output:** None
- **Called by modules:** READ_HISTOGRAM
- **Description:**

When starting the histogram read out and processing variables of the processing procedure PROCESS_HISTOGRAM are reset to initialisation values to make sure that the routines begin with correct start values.

- **Constraints:** None.
- **Efficiency:** Not applicable
- **Verification Requirement:** TBD

3.3.36.4 Function GET_SEQUENCE_COUNTER return CSSW_IF_TYPES.T_Source_Seq_Count

- **Input:** None
- **Output:** Sequence Counter of the current TM(6,0) packet
- **Called by modules:** SCIENCE_HK
- **Description:**

This function returns the sequence counter of the TM(6,0) packet, which currently is to fill by procedure S5.PROCESS_HISTOGRAM. That means the returned value is the sequence counter of the next TM(6,0) packet that will be given to the TM_INTERFACE.

- **Constraints:** None

- **Efficiency:** Not applicable

- **Verification Requirement:** TBD

3.3.37 MODULE S7 (SPECTRAL IMAGING, CSI SINGLE/MULTIPLE HISTOGR.)

- **List of Contents:**
  Procedure PROCESS_S_HISTOGRAM
  Procedure PROCESS_M_HISTOGRAM
  Procedure CLOSE_S_TM
  Procedure CLOSE_M_TM
  Procedure RESET
  Function GET_SEQUENCE_COUNTER return
  CSSW_IF_TYPES.T_Source_Seq_Count.

- **Global Variables:** None.

- **Exceptions:** EX_ADDRESS_NOT_CONTINUOUS

3.3.37.1 Procedures PROCESS_S_HISTOGRAM / PROCESS_M_HISTOGRAM

- **Input:** - one block (1026 byte) of histogram data interface according to HEPI Interface Document from module READ_HISTOGRAM,
  - Histogram ID,
  - Start time of integration [OBT],
  - Integration time [seconds]

- **Output:** TM(7,0) / TM(7,1) packets to module TM_INTERFACE

- **Called by modules:** READ_HISTOGRAM

- **Description:**

The input data block consists out of 2 byte of block address and 1024 byte histogram data. This data corresponds to 1024 histogram cells each with size of one byte. First it is checked whether the start address of the histogram data block is continuous to previous blocks. If it was the first block, the check is omitted. If the address is not continuous the current TM packet is closed and sent away and the exception EX_ADDRESS_NOT_CONTINUOUS is raised.

S7 histogram TM packets have a 5 word data field header. The data field header will be filled before the first histogram data cell is written into the TM packet. The TM packet data field header contains all information about the histogram integration and the memory address of the histogram data:

Type/Subtype (8 bit) = Hex(70 / 71)	MSB Start time of histogram integration
Start time of histogram integration [seconds] (24 bit) LSB	
Real integration time of the histogram [seconds]	
MSB Start address of the reference histogram data cell (first data cell inside the TM packet)	
Start address of reference cell (24 bit) LSB	Histogram ID

Figure 12: Format of 80 bit histogram data field header.

Compression algorithm:
The compression algorithm tries to reduce the size of a 8 bit histogram cell into a 4 bit TM packet data cell using a “count difference method”. The TM packet data field is built out of an array [1..848] of 4 bit data cells. When starting a new TM packet all data cells are initialised with flag value (-8).
The 24 bit histogram memory address given inside the TM packet data field header is the memory address of the first histogram cell which is written into the first two data cells of the TM packet. This first two TM packet data cells are equivalent to a uncompressed 8 bit histogram cell containing the origin count values for the corresponding pixel and energy channel. For the second histogram cell the difference of counts to the first histogram cell is calculated. This Δ(counts) of [histogram cell (1) – histogram cell (2)] is written as a 4 bit integer value into third TM packet data cell. The forth data cell contains the Δ(counts) of [histogram cell (2) – histogram cell (3)], the fifth data cell contains the Δ(counts) of [histogram cell (3) – histogram cell (4)] and so on. For Δ(counts) the range –7..+7 of a 4 bit integer type is used. The value –8 is used as a flag.

Δ(counts)	-8	-7	-6	-5	-4	-3	-2	-1	0	1	2	3	4	5	6	7
Bit code	1000	1001	1010	1011	1100	1101	1110	1111	0000	0001	0010	0011	0100	0101	0110	0111

Table 10: Bitcode of 4 bit integer values

If Δ(counts) is not in range of –7..+7 then the flag value (-8) is written into the TM packet data cell followed by the origin 8 bit histogram cell. The following example shows the working method of the histogram compression algorithm:

<table>
<tr><td colspan="3">Histogram cells (origin data from HEPI) (8 bit)</td><td rowspan="7">First →
data cell = 8 bit keeping the origin count value</td><td colspan="3">TM packet data cells(4 bit)</td></tr>
<tr><td>23</td><td>21</td><td>22</td><td colspan="2">23 (8 bit)</td><td>-2</td></tr>
<tr><td>16</td><td>19</td><td>14</td><td>+1</td><td>-6</td><td>+3</td></tr>
<tr><td>9</td><td>2</td><td>2</td><td>-5</td><td>-7</td><td>0</td></tr>
<tr><td>24</td><td>25</td><td>23</td><td>-8</td><td colspan="2">24 (8 bit)</td></tr>
<tr><td>18</td><td>20</td><td>17</td><td>+1</td><td>-2</td><td>-5</td></tr>
<tr><td>17</td><td></td><td></td><td>+2</td><td>-3</td><td>0</td></tr>
</table>

Figure 13: Example of histogram compression algorithm.

The last 3 data cells of the TM packet are reserved for a position counter, therefore the maximum number of available data cells is 845. The value inside this position counter indicates the first data cell which contains no data. If the position counter = 846 all data cells are valid, if position counter = 845 then the last data cell is invalid.

To make sure that no histogram TM packet will be lost during short timed high data rate to ICB TM sub buffer the procedure repeats up to ten times the sending of the TM packet to the TM_INTERFACE in intervals of 125 milliseconds. If the TM packet could not be sent an on-event message is generated that the TM packet is lost.
As long as the satellite is changing ground stations the processing and sending of histogram TM packets is interrupted.

- **Constraints:** None
- **Efficiency:** Not applicable
- **Verification Requirement:** TBD

3.3.37.2 Procedures CLOSE_S_TM / CLOSE_M_TM

- **Input:** None
- **Output:** TM(7,0) / TM(7,1) packets to module TM_INTERFACE
- **Called by modules:** READ_HISTOGRAM
- **Description:**

When finishing the histogram read out and processing the partial filled TM(7,0) / TM(7,1) packets are sent away to the module TM_INTERFACE. If the current TM packets contain no data the TM packets are not sent away.

- **Constraints:** None.
- **Efficiency:** Not applicable
- **Verification Requirement:** TBD

3.3.37.3 Procedure RESET

- **Input:** None
- **Output:** None
- **Called by modules:** READ_HISTOGRAM
- **Description:**

When starting the histogram read out and processing variables of the processing procedure PROCESS_S/M_HISTOGRAM are reset to initialisation values to make sure that the routines begin with correct start values.

- **Constraints:** None.
- **Efficiency:** Not applicable
- **Verification Requirement:** TBD

3.3.37.4 Functions GET_S7_HK_DATA return T_S7_Hk_Data

- **Input:** None
- **Output:** Sequence Counter of the current TM(7,0) / TM(7,1) packet
- **Called by modules:** SCIENCE_HK
- **Description:**

This functions returns the sequence counters of the TM(7,0) / TM(7,1) packets, which currently are to fill by procedures S7.PROCESS_S/M_EVENT. That means the returned values are the sequence counters of the next TM(7,0) / TM(7,1) packet that will be given to the TM_INTERFACE. Type T_S7_Hk_Data is a global record type built of two components of CSSW_IF_TYPES.T_Source_Seq_Count.

- **Constraints:** None
- **Efficiency:** Not applicable
- **Verification Requirement:** TBD

3.3.38 MODULE S8 (SPECTRAL TIMING)

- **List of Contents:**
  Procedure PROCESS_EVENT
  Procedure CONFIGURE_COMPRESSION
  Function REPORT_COMPRESSION return Integer
  Procedure CLOSE_TM
  Procedure RESET
  Function GET_SEQUENCE_COUNTER return System.Unsigned

- **Global Variables:**

FREQUENCY: integration time of spectral timing histograms in units [61,035 µsec.] This variable is set automatically by IASW when starting science mode. It is translated out of the hardware register HEPI register type II.

- **Exceptions:** None

3.3.38.1 Procedure PROCESS_EVENT

- **Input:** 160 bit event of type Hex(E6) according to HEPI Interface Document from module HBR_A_IF
- **Output:** TM(8,0) packet to module TM_INTERFACE
- **Called by modules:** SCIENCE_MODE
- **Description:**

This procedure contains all working steps to pack a spectral timing histogram event into a TM packet. The 32 bit time information of the first event of the TM packet is written into the data field header. The event is compressed and written into the TM packet. If the TM packet is filled it will be given to the TM_INTERFACE module by calling the procedure TM_INTERFACE.PUT_TM.

Data field header:
The TM(8,0) has got a data field header with size of 48 bit. Next to a type/subtype field there is the cell status flag with size of 1 byte. It indicates how much and which of the 8 spectral timing histogram cells / energy channels are active. Then there is the 32 bit time information of the first histogram event inside the TM packet and because the integration time is known the event times of all spectral timing histograms inside the TM packet can be reconstructed.

Type/subtype (8 bit) = Hex(80)	Cell status flag (8 bit)
MSB Event time of first spectral timing histogram inside the TM packet (32 bit)	
Event time of first spectral timing histogram inside the TM packet (32 bit) LSB	

Figure 14: Data field header of TM(8,0).

Compression algorithm:
The HEPI output are 8 histogram cells each with 8 bit length. Instead of using 8 bits for each histogram cell it is sufficient to reduce the length of each histogram cell to 4 bits. In case that a 8 bit cell histogram cell could not be reduced to a 4 bit cell, the whole 8 bit cell is copied into the TM packet behind a flag cell. The flag is a 4 bit cell with the maximum value of Hex(F). Therefor the maximum value allowed for compressed 4 bit histogram cells is Hex(E). All cells of one histogram are packed into the same TM packet. If there is no more place at the end of a TM packet for the complete histogram, the end of data is marked with flag Hex(F02), the TM packet is filled with flags Hex(F) and sent away. Then the histogram is packed into a new TM packet.
Proceed from an event rate of 28000 events per second the whole histogram keeps about 28 events using minimal integration time of 0,976 mseconds. Then the average value of each histogram cell is about 4. For a random distribution this 4 bit cells are able to keep values with 5 sigma deviation.

Histograms not in time frequency:
The time difference between two following histograms is FREQUENCY. If the time difference of two following spectral timing events is not the defined FREQUENCY, then behind a flag Hex(F00) the 32 bit absolute time information of the event is written into the TM packet, followed by the compressed histogram. The written absolute time is the new time base for the following spectral timing events. The flag is the not possible cell sequence Hex(F00). This flag is unequivocal because behind a flag Hex(F) must follow a 8 bit cell including a value greater than Hex(E). The 32 bit absolute time must not be word / byte aligned. The defined FREQUENCY for the spectral timing events could be verified by the additional IASW HK (H2).

Data processing was restarted:

If the data processing was disabled by TM_INTERFACE because of exceeded ICB TM sub buffer thresholds and is restarted again an additional flag Hex(F01) is written into the TM packet.

- **Constraints:** None
- **Efficiency:** Not applicable
- **Verification Requirement:** TBD

3.3.38.2 Procedure CONFIGURE_COMPRESSION

- **Input:** Cell status flag.
- **Output:** None.
- **Called by modules:** TASK_PARAMETER
- **Description:**

This procedure configures the selection of active spectral timing histogram cells inside procedure S8.PROCESS_EVENT.
A spectral timing histogram consist out of 8 energy channels / histogram cells. It is possible to define how much and which of 8 histogram cells are to process using a cell status flag of 1 byte length. This selection of active histogram cells of spectral timing histograms can be set via TC(5,3)(TID=16, FID=10) and reported via TC(5,4)(TID=16, FID=10). The cell status flag is also part of the TM(8,0) data field header. The MSB (C7) inside the cell status flag corresponds to the highest energy channel, the LSB (C0) corresponds to the least energy channel. Per default only cell C0 and C2 are active histogram cells.

Histogram Cell	C7	C6	C5	C4	C3	C2	C1	C0
Cell status	1=active, 0 = OFF	1=active, 0 = OFF	1=active, 0 = OFF	1=active, 0 = OFF	1=active, 0 = OFF	1=active, 0 = OFF	1=active, 0 = OFF	1=active, 0 = OFF

Table 11: Status flag of spectral timing histogram

- **Constraints:** None
- **Efficiency:** Not applicable
- **Verification Requirement:** TBD

3.3.38.3 Function REPORT_COMPRESSION return Integer

- **Input:** None.
- **Output:** Cell status flag.
- **Called by modules:** TASK_PARAMETER
- **Description:**

This procedure reports the selection of active spectral timing histogram cells inside procedure S8.PROCESS_EVENT.
A spectral timing histogram consist out of 8 energy channels / histogram cells. It is possible to define how much and which of 8 histogram cells are to process using a cell status flag of 1 byte length. This selection of active histogram cells of spectral timing histograms can be set via TC(5,3)(TID=16, FID=10) and reported via TC(5,4)(TID=16, FID=10). The cell status flag is also part of the TM(8,0) data field header. The MSB (C7) inside the cell status flag corresponds to the highest energy channel, the LSB (C0) corresponds to the least energy channel. Per default only cell C0 and C2 are active histogram cells.

Histogram Cell	C7	C6	C5	C4	C3	C2	C1	C0
Cell status	1=active, 0 = OFF	1=active, 0 = OFF	1=active, 0 = OFF	1=active, 0 = OFF	1=active, 0 = OFF	1=active, 0 = OFF	1=active, 0 = OFF	1=active, 0 = OFF

Table 12: Status flag of spectral timing histogram

- **Constraints:** None
- **Efficiency:** Not applicable
- **Verification Requirement:** TBD

3.3.38.4 Procedure CLOSE_TM

- **Input:** None
- **Output:** TM(8,0) packet to module TM_INTERFACE
- **Called by modules:** SCIENCE_MODE

- **Description:**

When ending the science mode the partial filled TM(8,0) packet must be filled with defined values (zeros) and sent away via the procedure PUT_TM out of the module TM_INTERFACE. The end of data is marked with flag Hex(F02). This flag is unequivocal because behind a flag Hex(F) must follow a 8 bit cell including a value greater than Hex(E). If there is no more place inside the TM packet to write the flag Hex(F02) the TM packet is just filled with flags Hex(F). If the current TM packet contains no data the TM packet is not sent away.

- **Constraints:** None.
- **Efficiency:** Not applicable
- **Verification Requirement:** TBD

3.3.38.5 Procedure RESET

- **Input:** None
- **Output:** None
- **Called by modules:** SCIENCE_MODE
- **Description:**

When starting the science mode variables for the current position inside the TM packet of the processing procedure PROCESS_EVENT are reset to initialisation values to make sure that the routine begin with correct start values.

- **Constraints:** None
- **Efficiency:** Not applicable
- **Verification Requirement:** TBD

3.3.38.6 Function GET_SEQUENCE_COUNTER return System.Unsigned

- **Input:** None
- **Output:** Sequence Counter of the current TM(8,0) packet

- **Called by modules:** SCIENCE_HK

- **Description:**

This function returns the sequence counter of the TM(8,0) packet, which currently is to fill by procedure S8.PROCESS_EVENT. That means the returned value is the sequence counter of the next TM(8,0) packet that will be given to the TM_INTERFACE.

- **Constraints:** None

- **Efficiency:** Not applicable

- **Verification Requirement:** TBD

3.3.39 MODULE SCIENCE_FUNCTIONS

3.3.39.1 Global Variables

3.3.39.2 Exceptions

3.3.40 MODULE SCIENCE_HK

3.3.40.1 Global Variables

3.3.40.2 Exceptions

3.3.41 MODULE SCIENCE_MODE

3.3.41.1 Global Variables

3.3.41.2 Exceptions

3.3.42 MODULE SQRT

3.3.42.1 Global Variables

3.3.42.2 Exceptions

3.3.43 MODULE TABLE_HANDLER

3.3.43.1 Global Variables

3.3.43.2 Exceptions

3.3.44 MODULE TASK_PARAMETER

3.3.44.1 Global Variables

3.3.44.2 Exceptions

3.3.45 MODULE TASKS

3.3.45.1 Global Variables

3.3.45.2 Exceptions

3.3.46 MODULE TC_DECODER

- Procedure START
- Procedure EXECUTED_TC
- Procedure REJECTED_TC

3.3.46.1 Global Variables

3.3.46.1.1 LAST_TC

last received (valid or invalid) telecommand:

3.3.46.2 Exceptions

3.3.46.3 Procedure START

3.3.46.3.1 Use/Constraints

must be called by the tc starting task stump.

3.3.46.3.2 Description

Gets TC, checks parameters, supplies the tc + parameter to other tasks and generates on-events for not recognized/rejected TC.

3.3.46.4 Procedure EXECUTED_TC

3.3.46.4.1 Output

Number of successfully performed Telecommands. If the Telecommand number exceeds T_TC_COUNTER'LAST it will be reset at zero.

3.3.46.4.2 Use/Constraints

No restrictions.

3.3.46.4.3 Description

Each executed telecommand increases a counter which value will be returned.

3.3.46.5 Procedure REJECTED_TC

3.3.46.5.1 Output

Number of rejected Telecommands If this number exceeds T_TC_COUNTER'LAST it will be reset at zero.

3.3.46.5.2 Use/Constraints

No restrictions.

3.3.46.5.3 Description

Each rejected or non-successfully executed telecommand increases a counter which value will be returned.

3.3.47 MODULE TM_INTERFACE

3.3.47.1 Global Variables

3.3.47.2 Exceptions

3.3.48 MODULE TRANSPARENT

3.3.48.1 Global Variables

3.3.48.2 Exceptions

3.3.49 MODULE VETO

- Procedure SWITCH_VETO_OFF
- Procedure COMMAND_VETO
- Procedure READ_VETO
- Procedure GET_SEQ_COUNT

3.3.49.1 Global Variables

3.3.49.2 Exceptions

3.3.49.3 Procedure SWITCH_VETO_OFF

3.3.49.3.1 Use/Constraints

Must be called in emergency case when veto high voltage is to be turned off. LSL MUST BE BLOCKED

3.3.49.3.2 Description

Sends Veto standby command.

3.3.49.4 Procedure COMMAND_VETO

3.3.49.4.1 Input

VETO_COMMAND - The command to be send to VETO. PARAMETER - The parameter for that command.

3.3.49.4.2 Output

: ok is true when transfer successfully done

3.3.49.4.3 Use/Constraints

Call the procedure with given command ID and parameter for that command. NO PARAMETER CHECK WILL BE DONE THERE IS NO INTERNAL BLOCKING OF THE LSL

3.3.49.4.4 Use/Constraints

THE LINE MANAGER MUST BE USED BY THE CALLER OF THIS PROCEDURE

3.3.49.4.5 Description

Command_Veto transmits the command via the LSL to VETO. The control will return when transmission is over. The length of the command is set according the length of the parameter array.

3.3.49.5 Procedure READ_VETO

3.3.49.5.1 Input

VETO_COMMAND - The read command to be sent to VETO PARAMETER - The parameter necessary for this command

3.3.49.5.2 Output

VETO_RESPONSE : The data read out from VETO. This array must be set properly at the expected data size from VETO. ok : true when transfer successfully done

3.3.49.5.3 Use/Constraints

Call Read_Veto with a command which requests data from VETO, and necessary parameters. The procedure will return the veto response data according the command sent into response buffer. NO PARAMETER CHECK WILL BE DONE ALSO NO COMMAND ID CHECK IS APPLIED THERE IS NO INTERNAL BLOCKING OF THE LSL

3.3.49.5.4 Use/Constraints

THE LINE MANAGER MUST BE USED BY THE CALLER OF THIS PROCEDURE

3.3.49.5.5 Description

Command_Veto transmits the command via the LSL to VETO. Then the VETO will send the data requested- via LSL to DPE . The control will return to caller when transmission is over. The length of the command to send is derived from the parameter block length; that of the response from the response block length.

3.3.49.6 Procedure GET_SEQ_COUNT

3.3.49.6.1 Output

Returns: 13-bit value containing current number of commands sent to veto. Value wraps around at 2^13.

3.3.49.6.2 Use/Constraints

In HK for setting the VETO sequence count field. returns 13-Bit value

3.3.49.6.3 Description

Reflects counter status.

3.3.50 MODULE VETO_CONTEXT_TABLE

3.3.50.1 Global Variables

3.3.50.2 Exceptions

4 ANNEX

4.1.1 GENERAL DESCRIPTION OF IBIS IASW TC

TC TABLE: TBW

Table 13: Table of accepted IASW TC

4.1.2 IASW BROADCAST PACKET PROCESSES

A] On-Board generated Data:

Flag	Description	IBIS IASW Reaction
A1	OTF: On Target Flag	Start/Stop of Histogram Integration
A2	ESAM Emergency Safe Acquisition	Go to standby, VETO HV off
A3	AOCS Modes:	
	3.1 Inactive Mode	n.u.
	3.2 Standby Mode	n.u
	3.3 ISA Initial Sun Acquisition	n.u
	3.4 SSA Sun Sensor Acquisition	n.u
	3.5 STA Star Tracker Acquisition	n.u.
	3.6 IPM Inertial Pointing Mode	Science Mode
	3.7 TCM Thrust Control Mode	n.u.
A4	AOCS Submodes	used
A5	Radiation Monitor Count Rate #1	Go to stand by, Switch off VETO HV
A6	Radiation Monitor Count Rate #2	Go to stand by, Switch off VETO HV
A7	Radiation Monitor Count Rate #3	Go to stand by, Switch off VETO HV

B] Orbital Data

B1	Radiation Belts Entry Time	Switch to Standby, VETO HV off
B2	Radiation Belts Exit Time	used
B3	Eclipse Entry time	Switch to Standby, Save peripheral context
B4	Radiation Monitor Data Validity Flag	monitored
B5	Eclipse Exit Time	Restore of peripheral context

C] Operational Data

C 1	SPI Telemetry Share	n.u.

C 2	IBIS Telemetry Share	n.u.
C 3	JEM-X1 Telemetry Share	n.u.
C 4	JEM-X2 Telemetry Share	n.u.
C 5	OMC Telemetry Share	n.u.
C 6	All Instrument Switch-off Imminent	switch to standby, VETO HV off, save peripheral context
C 7	Ground Station Hand-Over Flag	Stop sending histograms
C 8	Pointing ID	pointing/slew switching
C 9	Pointing Duration	pointing/slew switching