SPACE SYSTEMS FINLAND	DEBIE DPU SW	REF : DEB-SSF-DD-001 ISSUE : 1.4 DATE : 31.8.1999

DOCUMENT TYPE: Design

TITLE:

DEBIE DPU SOFTWARE DESIGN DOCUMENT

	FUNCTION	NAME	DATE	SIGNATURE
PREPARED BY	SW Engineer	Ville Sipinen		
CHECKED BY	SW Engineer	Harri Paloheimo		
APPROVED BY	QA Engineer	Niklas Holsti		

DOCUMENT IDENTIFICATION	WBS Nr NONE	KEYWORDS DEBIE, design

DOCUMENT STATUS SHEET

Issue	Date	Modifications	Reason for change / Comments
1.0	21.11.1997	All	First issue, for proposal.
1.1	11.5.1998		Preliminary architectural design
		Section 1.1	Scope updated
		Section 1.2	Added description for chapters 4 and 5
		Section 1.6	Added following acronymes and abbreviations: HW, SW, SSF
		Section 2.1	Deleted sampling of leading edge FIFO from the list of DEBIE functions and added measurement of rise time and delays between trigger signals to it
		Section 2.1.1	Deleted leading edge FIFO reading from the list of high level services of HW/SW interface and added rise time measurement ot it.
		Section 2.1.2	Deleted telecommand possibillity for mode transition from Init to Standby from Table 1.
		Section 2.1.3	Deleted handling of leasing edge FIFO from the list of particle hit actions and added measurement of rise time and delays between trigger signals to it.
		Section 2.16	Replaced 10s dead line by 60s dead line for calculating checsum from code memory
		Section 2.1.7	Deleted TBC
		Section 2.1.9	Updated the list of types of telemetry
		Section 2.2.2	Deleted references to leading edge FIFO
		Section 2.2.5	Updated text to reflect new 60s dead line for code memory checksum calculations
		Section 2.2.8	Deleted rise time algorithm from possible problems
		Section 2.3	Deleted rise time algorithm for leading edge measurements from the list of open issues and added the definition of events to be counted to it
		Section 3.1	Added arrow from Housekeeping Memory to Acquisition Task to Figure 1.
		Section 3.1.1	Deleted reading of leading edge FIFO from the list of Acquisition task activities and added reading of rise time and measurement of delayd between trigger signals to it

DOCUMENT STATUS SHEET

Issue	Date	Modifications	Reason for change / Comments
		Section 3.1.3	Updated text to show that TC and TM interrupts are separate.
		Section 3.1.4	Splitted TM/TC interrupt service to TM and TC interrupt services
		Section 3.1.5	Housekeeping Memory must be visible also to the Acquisition task
		Section 3.2	Deleted TBC from the reference to Keil RTX 51 real time kernel
		Chapters 4 and 5	Added new chapters
1.2	17.9.1998	Section 1.1	This issue presents the Prototype design (and plans for final SW)
		Section 1.2	Added new chapters to overview
		Section 1.3	Updated the reference number of [AD1] and added [AD4]
		Section 1.4	Added Reference Documents 3, 4 and 5
		Section 2	Added not about differences to the Prototype SW requirements
		Section 2.1	Deleted partition of measurement data to raw a processed data and DPU temperature measurements.
		Section 2.1.1	Deleted DPU temperature measurement
		Section 2.1.2	Replaced Init mode with DPU Self Test
		Table 1	Updated table: transition from Standby to DPU Self Test (fromer Init) is not allowed anymore.
		Section 2.1.3	Deleted restarting of FIFO.
		Section 2.1.4	Deleted hit rate calculation and updated the event classification
		Section 2.1.5	Updated the section
		Section 2.1.6	Updated the voltage and temperature monitoring
		Section 2.1.7	Updated the section to concern only Sensor Unit Self Test
		Section 2.1.8	Replaced Classification Channels telecommand group with Time Window definition commands. Noted additional timing requirements for some TC and multi-word TC.

DOCUMENT STATUS SHEET

Issue	Date	Modifications	Reason for change / Comments
		Section 2.1.9 Section 2.2.3	Clarified the handling of TC during TM.
		Section 2.2	Deleted calculation of program memory cecksum from the critical areas.
		Section 2.2.2	Changed the lower limit of the delay of the Peak Detector reading to 100 .. 300 s.
		Section 2.2.5 (in previous issue)	Deleted section concerning program memory checksum
		Section 2.2.5 (2.2.6 in previous issue)	Updated section.
		Section 2.2.7 (2.2.8 in previous issue)	Deleted calculation of program memory checksum from summary of possible problems.
		Section 2.3	Deleted event counting and classification from open issues
		Section 3	Replaced DEBIE Test Software with Prototype Software. Corrected spelling of Matlab to MATLAB.
		Section 3.1	Replaced Self Test / Calibration with DPU Self Test. Noted that ISRs are shown in the figure.
		Figure 1	TM and TC interrupts separated
		Section 3.1.1	Updated the Acquisition task main activities
		Section 3.1.2	Replaced Self Test / Calibration with DPU Self Test and added note about simplified implementation in the Prototype SW. Updated list of actions.
		Section 3.1.3	Confirmed that TC format check is done in TC ISR
		Section 3.1.4	Deleted Health Monitoring loop timer ISR from the list of interrupt services (internal to RTX-51). Confirmed use of mailboxes.
		Section 3.1.5	Deleted distinction of processed and raw measurement data. Added discussion of Science TM freezing and buffering.
		Section 3.2	Updated the list of RTX services to which DHI has to provide interface to. Omitted “two layer” structure of DHI.
		Figure 2	Deleted distinction of Higher Level and Lower Level of DHI and DTS from the figure
		Chapter 4	Added new chapter

DOCUMENT STATUS SHEET

Issue	Date	Modifications	Reason for change / Comments
		Chapter 5 (chapter 4 in issue 1.1)	Updated the dynamic architecture, added descriptions of the interrupt services. Removed description of the UNIX test environment in favour of the Verification Plan document. Corrected spelling of Matlab to MATLAB.
		Chapter 6 (Chapter 5 in issue 1.1)	TM and TC interrupts separated in Hood diagrams.
		Chapter 7 and 8	Added new chapters
1.3	2.2.1999	1.1	This issue presents the design of the first version of the Flight SW.
		1.2	Chapters 7 and 8 are empty in this document version, because the design is not yet completely implemented.
		1.3 & 1.4	Added writers of applicable and reference documents.
		1.4	Added SW User Manual to reference documents
		2	Deleted obsolete references to Prototype SW
		2.1.1	Low level HW services implemented typically by macros defined in C-header files. Deleted non existing HW interface service ‘clock reading’ (clock is a SW counter).
		Table 1	DPU Self Test can be entered also from Standby and Acquisition modes due to a telecommand.
		2.1.7	SU Self Test is performed for each SU independently.
		2.1.8	Calibration TC replaced with SU Self Test TC, added Error Reset
		2.2.6	No separate TC necessary to modify reference checksum after code patch.
		2.3	There are now no major open issues at present.
		3.1	Telemetry / Telecommand task is replaced with Telecommand Execution task. The Acqusition Task activity in different modes is described more precisely.
		Figure 1	Task interfaces updated to correspond to current design.
		3.1.1	Classification algorithms are defined in [AD1].

DOCUMENT STATUS SHEET

Issue	Date	Modifications	Reason for change / Comments
		3.1.2	Mailboxes and semaphores do not need to be initialized. Description of wait state of the Health Monitoring task and processor idle state is updated.
		3.1.3	Telecommand Execution task is only partly responsible for transmission of telemetry. The state transitions related to telemetry sessions are made by TC and TM Interrupt services.
		3.1.4	TC interrupt sends TC word, not only TC code.
		3.1.5	Semaphores will probably not be used.
		3.2	Deleted sempahore operations from the list of used services. Added attach interrupt, wait for interrupt and wait for time period.
		3.3	It is TBC if MATLAB and DMI will be used at all.
		Table 2	Deleted os_send_token
		4.4	Semaphores are not used, but shared resources are protected by disabling interrupts.
		4.5.1	Call trees will be generated using the “cflow” tool.
		Figure 3	Updated the figure
		5.2	Housekeeping and operational telemetry is replaced with TM register telemetry.
		Figure 4	Deleted enabling of TC response
		5.3	Handling of the mail from TC Interrupt updated. Handling of “TM_Ready” message updated.
		5.4	Health Monitoring Task description updated
		5.6	Quality formula is defined in [AD1].
		Chapters 7 and 8	Header files not included in this document version, because the design is not yet completely implemented.
1.4	TBD	See below	Updated for DEBIE Flight SW delivery
		1.2	Chapters 7 and 8 are not empty in this document version
		1.5	There are no definitions

DOCUMENT STATUS SHEET

Issue	Date	Modifications	Reason for change / Comments
		1.6	Deleted DMI from the list of acronyms and abbreviations.
		2.1.2	Sensor Units are not necessarily switched Off in Standby mode.
		2.2.5	Hit trigger budget limit is 20 hits / 10s
		3	Deleted remark about DMI
		3.1.2	Idle state used (deleted other option and TBD)
		3.1.2	Mentioned SU self test as a part of health monitoring action.
		3.3	Removed DEBIE MATLAB Interface section.
		4.3	RTX timer interrupt is no longer lower priority interrupt
		5	Dynamic architecture figures are not going to be exctended anymore
		5.4.1	New section which describes SU self test action as a part of the health monitoring cycle.
		Figure 7	Figure modified to better describe the health monitoring loop cycles.
		Figure 8	Added SelfTest_SU and SelfTestChannel procedures into the health monitoring function hierarchy.
		Figure 9	Refined that self test in power-up timing is DPU self test. Replaced incorrect 100 ms with 800 us.
		Figure 10	Added description of Hit Budgeting
		Chapers 7 and 8	Added current DAS and DHI headers
		Document wide	New company logo.

TABLE OF CONTENTS

TABLE OF CONTENTS

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1. Introduction

1.1 Scope

This document presents the design of an on-board software for the DEBIE instrument. DEBIE is a standard instrument for monitoring space debris and meteoroids in near Earth orbit.

The DEBIE DPU software runs on an 80C32 processor in the DEBIE Data Processor Unit (the DPU). The software configures the DEBIE detectors, records impact events and transmits the records to the spacecraft main computer. The software is controlled by telecommands issued from the spacecraft main computer.

The present issue of this document presents the design of the first version of the Flight Software.

1.2 Overview

Section 2 summarises the requirements for the DEBIE.

Section 3 describes the architectural overview.

Section 4 describes the general design principles.

Section 5 describes the dynamic architecture.

Section 6 describes the static architecture.

Section 7 describes the DAS header files.

Section 8 describes the DHI header files.

1.3 Applicable Documents

[AD1] DEBIE Requirements Specification, DEB-FIN-RS-001
Patria Finavitec Systems

[AD2] DEBIE TM/TC Interface Control Document, DEB-FIN-IC-001
Patria Finavitec Systems

[AD3] DEBIE HW/SW Interface Control Document, DEB-FIN-IC-002
Patria Finavitec Systems

[AD4] DEBIE DPU Software Requirements Document, DEB-SSF-RS-001
Space Systems Finland Ltd

1.4 Reference Documents

[RD1] DEBIE Software Development Plan, DEB-SSF-PL-001
Space Systems Finland Ltd

[RD2] MHS C51 Programmer's Guide and Instruction Set
Matra MHS

[RD3] RTX-51, RTX-251 User's Guide 05.96
Keil Software

[RD4] C51 Compiler User's Guide 01.97
Keil Software

[RD5] DEBIE DPU SW Design and Coding Standard, DEB-SSF-ST-001
Space Systems Finland Ltd

[RD6] DEBIE DPU SW User Manual, DEB-SSF-MA-001
Space Systems Finland Ltd

1.5 Definitions

None.

1.6 Acronyms and Abbreviations

ADC	Analog to Digital Converter
DAS	DEBIE Application Software
DEBIE	Debris In Orbit Evaluator
DHI	DEBIE Hardware Interface
DPU	Data Processing Unit
DTS	DEBIE Test Software
EGSE	Electrical Ground Support Equipment
FIFO	First In First Out
HK_TM	HouseKeeping TeleMetry
HW	Hardware
ISR	Interrupt Service Routine
RAM	Random Access Memory
RTX	RealTime eXecutive (in this case Keil RTX-51 for 8051)
SSF	Space Systems Finland
SU	Sensor Unit
SW	Software
TBC	To Be Confirmed
TBD	To Be Defined
TC/TM	TeleCommand / TeleMetry

2. DEBIE Requirements

This chapter summarises the requirements for the DEBIE DPU SW.

2.1 DEBIE DPU SW Requirement Summary

The functions for the DEBIE DPU SW based on the DEBIE Requirement Specification [AD1] are:

- Interface to the DEBIE HW
- Functional mode transitions
- Detection and counting of incoming particles
- Sampling of peak detector measurements during particle hits
- Measurement of rise time and delays between trigger signals
- Measurement of the Sensor Unit temperatures
- Classification of the detected particle hits
- Storage of the measurement data
- Health monitoring of SU temperatures and secondary supply voltages
- Self Test / Calibration
- Refreshing of the watchdog
- Calculation of a checksum from the memory
- Error status recording
- Reception and execution of telecommands from the spacecraft
- Transmission of all measurement and status to the spacecraft via telemetry

The following subsections describe each function in more detail.

2.1.1 Interface to the DEBIE Hardware

Interface to the DEBIE hardware will be isolated in a separate software entity as described in the Software Development Plan [RD1]. It will be composed from two layers: one providing higher level DEBIE specific hardware services and the other low level 80C32 processor control services. The low level services are implemented typically by macros defined in C-header files.

The higher level services will provide services needed for controlling DEBIE hardware including:

- Sensor Unit power control
- Rise time measurement
- Peak detector reading
- Trigger limit setting
- SU temperature measurement
- Secondary supply voltage measurement
- TM/TC interface

The lower level services used by the higher level services include for example:

- Read processor port
- Write processor port
- Write processor port pin
- Initialize internal timer
- Start internal timer
- Stop internal timer
- Set processor to idle state
- Write processor register
- Read processor register
- Read memory location
- Write memory location
- Soft reset
- Hard reset

2.1.2 Functional modes

There shall be four functional modes in DEBIE: Off, DPU Self Test, Standby and Acquisition. Permissible transitions between these modes and valid transition causes are described below:

Table 1: Functional modes and transitions

Source mode	Destination mode			
	Off	DPU Self Test	Standby	Acquisition
Off	NA	power on	not allowed	not allowed
DPU Self Test	power off	NA	automatic	not allowed
Standby	power off	telecommand	NA	telecommand
Acquisition	power off	telecommand	telecommand	NA

In Off mode DEBIE is not operational.

In DPU Self Test mode a short self test sequence is performed and the resulting status shall be stored into local memory. In this mode DEBIE shall not perform acquisition functions (particle hit measurements) nor transmit any science data to the spacecraft. After completion of the initialization sequence the Standby mode shall be entered automatically.

In Standby mode housekeeping, operational and science data shall be sent to the spacecraft on request. In this mode no acquisition functions shall be executed. Health Monitoring Function shall be executed periodically in this mode.

In the Acquisition mode all kinds of telemetry data are sent to the spacecraft on request and Health Monitoring and acquisition functions shall be active.

Telecommands shall be received and executed when appropriate in all modes except in Off mode.

2.1.3 Particle Hit Detection and Actions

The DEBIE hardware triggers the processor when a particle hits a Sensor Unit. The DEBIE software shall count these triggers. When the particle hit trigger is received the following actions shall be done:

- Peak detector outputs of each plasma and piezoelectric sensor shall be sampled
- Pulse rise time and delays between trigger signals shall be measured
- A time tag shall be associated to the measurement data
- The Sensor Unit temperatures shall be measured or retrieved from Health Monitoring Function

The threshold level of each trigger signal and sensor shall be adjustable independently with specific telecommands.

2.1.4 Processing of the Measurement Data

The measurement data shall be processed as follows:

- Total number of particle hits for each Sensor Unit shall be counted
- The detected particle hits shall be classified into different classes according to signal amplitude or co-incidence of signals from different sensors.
- Events of different class on each Sensor Unit shall be counted to provide information of the background noise

2.1.5 Storage of the Measurement data

A quality number shall be calculated for each event and attached to the event record.

If the science data memory is not full the new event record shall be stored in the next free place.

If the science data memory is full the event record or records with the lowest quality number is searched and if the quality number of the new event is higher or equal, the lowest quality event record - or if there are several events with that lowest quality, the oldest of them - shall be replaced with the new event record.

2.1.6 Health Monitoring

The state of DEBIE shall be continuously monitored. The status of secondary supply voltages shall be monitored at least once in 10 seconds by reading a binary HW fault indicator. The SU temperatures shall be measured at least once in 60 seconds. The secondary supply voltages shall be measured at least once in 180 seconds.

In case any one of the supply voltage outputs to the Sensor Units is detected to be short circuited, that output shall be switched off at once. If any one of the Sensor Units is found to be overheated, all supply voltages to that Sensor Unit shall be switched off at once. In both cases the corresponding error status bit in the error register shall be set.

DEBIE includes a watchdog timer to ensure proper software execution. The watchdog timer shall be reset regularly by the software.

A checksum from the RAM used as program memory shall be calculated regularly. DEBIE shall not execute any piece of code whose checksum is older than 60 seconds. To ensure compliance to this requirement all memory containing executable code shall be checked at least once in less than 60 seconds.

When the checksum does not match, a checksum failure counter shall be incremented and a soft reset function preserving the error status and failure counters shall be executed.

2.1.7 SU Self Test and Calibration

It shall be possible by specific telecommands to run a self test/calibration for each Sensor Unit independently.

The self test shall include the checking and the measurement of secondary supply voltage ouputs for the Sensor Unit being self tested.

The self test shall include a calibration sequence where a calibration pulse is generated to each detector chain one by one. The measured data will be used on ground to calibrate each peak detector and pulse edge detector output.

2.1.8 Reception and Execution of Telecommands

Validity of any new telecommand received shall be checked. If a false command or parity error is detected the false command along with its address and parity shall be latched to the Command Status Register, the Error Status bit in the Error Status Register shall be set and the command shall not be executed. An accepted command shall be latched to the Command Status Register with the Error Status bit cleared and the Time Tag for the received command shall be latched to the Command Time Tag Register.

False telecommands are commands with false telecommand address and/or false telecommand code. The valid telecommand addresses are described in the TC/TM Interface Control Document [AD2].

The following types of telecommands exist:

- Functional mode switching commands
- Power control commands
- SU Self Test command
- Soft reset command

- Memory handling commands
- Clock handling commands
- Trigger thresholds setting commands
- Classification levels setting commands
- Time window definition commands
- Telemetry request commands
- Error reset command

Telecommands separated by more than 10 ms shall be accepted, others shall be rejected. Some telecommand sequences (eg. Set Time) and some multi-word telecommands (eg. Write Memory) have additional timing requirements.

The response to a telecommand shall be available at most 1 ms after telecommand reception, except during Science or Memory Dump telemetry, when all telecommands shall be rejected.

2.1.9 Telemetry

The following types of telemetry shall be sent to the spacecraft when requested:

- Science Telemetry (requested by Send Science Data File TC)
- TM Register Telemetry (requested by Send Status Register TC)
- Memory Dump (requested by Read Memory TC)

During Science or Memory Dump telemetry transmission no telecommands shall be executed. During TM Register Telemetry, reception of a telecommand shall end the telemetry; the telecommand shall be handled and executed as usual.

2.2 Critical Areas

The following are critical areas in the DEBIE software:

- refreshing watchdog
- timing of measurement data acquisition
- timing of telecommand reception
- timing of telemetry data stream
- interference between Health Monitoring and Acquisitions
- interference between program Memory Write telecommands and checksums

These aspects of the software and possible problems related to them are described in more detail in following subsections.

2.2.1 Watchdog

There should not be any problems in refreshing the watchdog itself, but there must be enough margin to the deadline of the watchdog refreshing that any non-fatal overruns cannot block the software so much that the deadline is missed.

2.2.2 Timing of Measurement Data Acquisition

The peak detector output must be read at least 100 .. 300 µs (depending on the channel) and at most 2 ms after the trigger signal.

DEBIE must be able to detect and measure particle hits coming once in a second. This means that measurement data shall be read and processed in less than a second from the trigger signal.

2.2.3 Timing of the Telecommand Reception

The Command Status Register and Error Status Register shall be updated at most 1 ms after the reception of a telecommand.

DEBIE must be able to accept telecommands separated by more than 10 ms, unless the first telecommand is Science or Memory DumpTM request, when the next telecommand shall be accepted only after the requested telemetry data is transmitted.

This means that all telecommands, except TM requests, shall be executed in less than 10 ms.

2.2.4 Timing of the Telemetry Data Stream

DEBIE must allow chaining TM accesses with 1 ms interval, one access triggering the read action for the next. This means that the software must be able to write the next word of the requested telemetry in less than 1 ms after the previous one is transmitted to the spacecraft.

2.2.5 Interference Between Health Monitoring and Acquisitions

When a particle hit occurs during Health Monitoring measurements, the Hit Trigger task pre-empts the Health Monitoring task and measures the Peak Detector outputs using the same ADC as the Health Monitoring. This disturbs the Health Monitoring measurements, making its ADC results invalid. A global “hit occurred” flag detects this situation, and the Health Monitoring measurement has to be repeated.

If the particle hits come too often, the Health Monitoring measurements cannot be peformed. Therefore the DEBIE DPU Software has to set a limit on the number of hit triggers that will be handled during one Health Monitoring cycle. This limit is currently 20 hits per Health Monitoring period, which is 10s.

2.2.6 Interference Between Memory Write and Checksums

The memory write telecommands sent during checksum calculations must not cause a false checksum error. A false checksum error could be signalled if a checksum of modified code is compared against the old reference value, or if the new reference value is compared against a checksum of the unmodified code. Either could result from an interleaving of the checksumming and memory update functions.

The simplest way to prevent the false checksum errors is to ignore the possible differences between calculated checksum and reference value at the first check following the code update and update the reference checksum value during memory patch telecommand execution.

2.2.7 Summary of Possible Problems

Although there seem to be no very complex actions, some of the requirements may lead to considerable performance problems.

The 2 ms limit for reading the peak detector output together with the 1 ms limit for the response for a telecommand reception or telemetry word reading may prove to be too hard a requirement for the 80C32 processor. Much of the related actions can be forced into the interrupt services, since task switches may take too long time. However the execution time of interrupt services themselves should not be too long as it increases the response time of the software to other interrupts.

2.3 Open Issues

There are no major open issues at present.

3. DEBIE Architecture Overview

The DEBIE software will include two main parts:

- DEBIE Application Software (DAS)
- DEBIE Hardware Interface (DHI)

A subset of the Application Software, intended for testing the DEBIE prototype, is called the Prototype Software.

The connections between these parts and EGSE and the hardware are described in the Software Development Plan [RD1].

3.1 DEBIE Application Software

This part of the software is responsible for the main part of the activities. It will implement all DEBIE functions: acquisition function, health monitoring function and telemetry/telecommand function. It uses services of the DEBIE Hardware Interface to achieve this.

There will be three tasks in the DEBIE Application Software (DAS):

- Acquisition task
- Health Monitoring task
- Telecommand Execution task

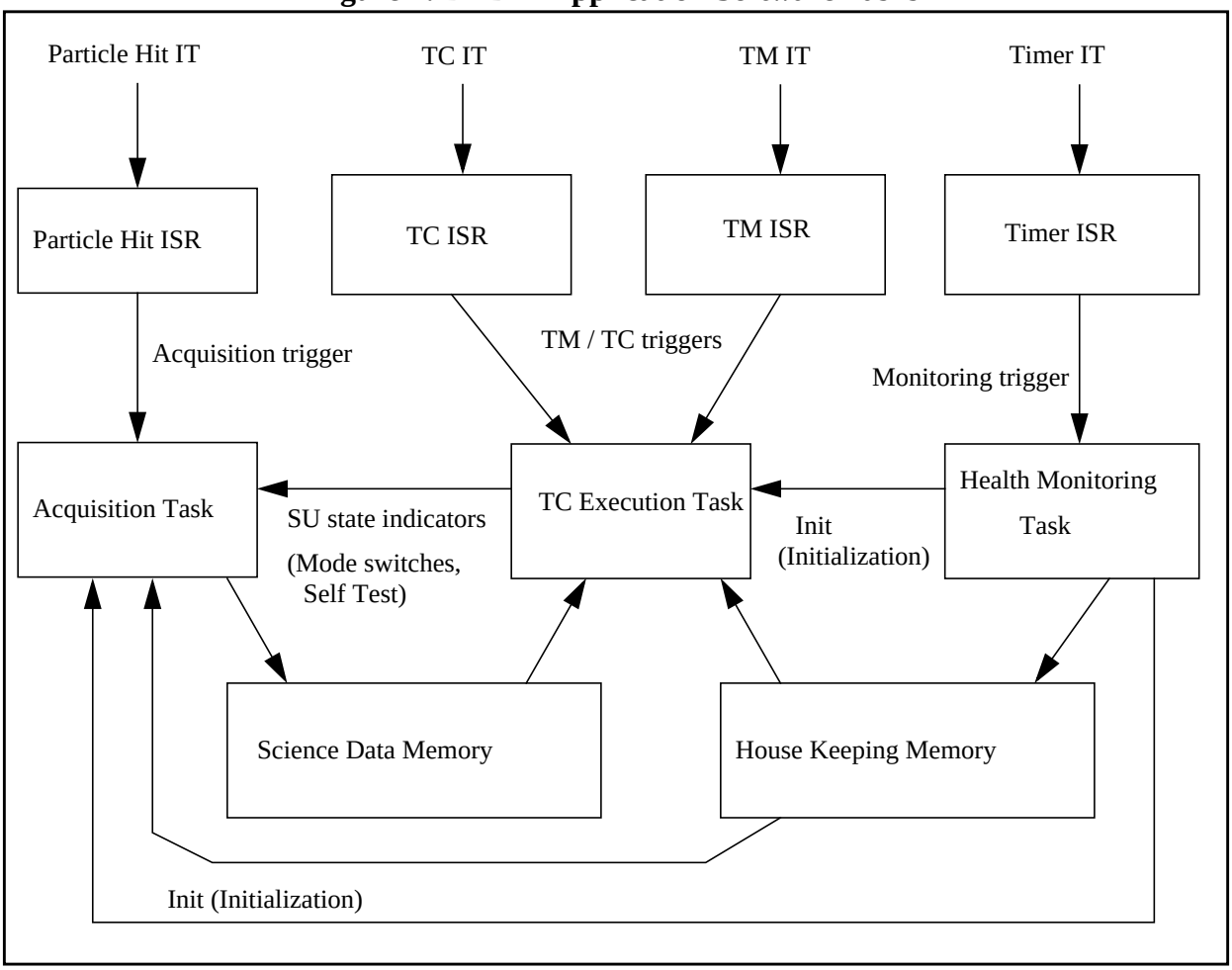
The Telecommand Execution task shall be active in every mode in which the DEBIE is active. This means all other modes than the Off mode.

The Health Monitoring task shall be the first task activated after starting the DEBIE software. It shall first execute the necessary software initialization and execute DPU Self Test function and after that transition to Standby mode. In the Standby and Acquisition modes it will execute the Health Monitoring functions.

The Acquisition task shall be active only in the Acquisition mode - or to be more precise, it will process and store Event Data only in the Acquisition mode, except when the Self Test sequence is performed to some Sensor Unit.

The following figure shows the tasks, the triggering interrupts and their Interrupt Service Routines (ISRs).

Figure 1: DEBIE Application Software Tasks



3.1.1 Acquisition Task

This task is responsible for the acquisition functions of the DEBIE software. It will be triggered by the Particle Hit interrupt service.

There will be the following main activities for this ISR and task:

- reading of peak detector output and rise time
- reading of delays between trigger signals
- classification of the measurement data
- storing the measurement data

The rise time and peak detector output are read by using the services of the DEBIE Hardware Interface.

The algorithms for the classification of the measurement data are defined in the [AD1].

The measurement data are stored in the memory as described in the section 2.1.5 on page 5.

After the measurement activities are done the task will be set in the waiting state until the next Particle Hit awakens it again.

3.1.2 Health Monitoring Task

This task is responsible for initialization of the DEBIE and the Health Monitoring Functions described in the section 2.1.6 on page 5.

This task contain initialization code, run only once after power up or soft reset, and a monitoring loop run periodically as long as the DEBIE is active.

The initialization code shall execute the following actions:

- Initialize necessary global variables depending on hard or soft reset
- Initialize other tasks
- Initialize interrupt service routines
- Call DPU Self Test function

The Health Monitoring loop shall be triggered by an interrupt service of one of the processor's internal timers and executes the following actions:

- Refresh the watchdog
- Measure Sensor Unit temperatures and secondary supply voltages
- Check for interference from the acquisition function, repeat measurements when needed
- Switch off short circuited voltages or voltages to overheated Sensor Units
- Store housekeeping data in the memory
- Calculate checksum from the memory holding executable code.
- SU self test execution, when corresponding TC is received.

In the Prototype Software only Sensor Unit temperatures are measured.

After these actions the Health Monitoring task shall wait until the timer awakens it. If there are no other task running, the RTX will put processor in idle state until some interrupt occurs.

3.1.3 Telecommand Execution Task

This task is responsible for reception and execution of the telecommands and partly for transmission of telemetry. After a telecommand is received its validity is checked. The software shall check at least the address and code of the telecommand. This is done in the TC interrupt service.

When a valid telecommand is received and response given to the spacecraft, the corresponding action is done in less than 10 ms unless the received telecommand is a telemetry request. Possible types of telecommands are listed in section 2.1.8 on page 6.

When a telemetry request is received, the Telecommand Interrupt Service changes the state of this task to the telemetry transmission state where TM interrupts are interpreted as requests to send the next word of telemetry. This state lasts until all requested telemetry is sent to the spacecraft or the telemetry session started with Send Status Register TC is stopped with a new telecommand. The state of the task is changed back to the telecommand handling state by Telecommand or Telemetry Interrupt service.

After the received telecommand is rejected or executed (including transmission of one word of requested telemetry for TM request telecommands) this task is set to the waiting state until the next TM (after last telemetry word) or TC interrupt awakens it.

3.1.4 Interrupt Services

The DAS will implement the following interrupt services:

- Particle Hit interrupt service
- TM interrupt service
- TC interrupt service

These interrupt services shall trigger the corresponding tasks (see above sections) by sending messages to designated mailboxes. Mailboxes carry information from the interrupt service to the corresponding task, for example the TC word from the TC interrupt service to the TM/TC task.

3.1.5 Shared Memory

Because of limitations of the kernel messaging services and the processor and memory resources, shared memory must be used to communicate some data between tasks.

At least the science data memory holding measurement data and housekeeping data memory need to be in shared memory. The science data must be visible to the Acquisition task and to the TM/TC task. The housekeeping data must be visible to the Health Monitoring task, Acquisition task and to the TM/TC task.

Accesses to the shared memory areas shall be protected. Mutual exclusion can be ensured by preventing task switches. Task switches must not be prevented for longer periods, because then all activity of the other tasks would be blocked.

A particular problem is posed by the Science Telemetry, during which the Science File in memory must be frozen to preserve the integrity of the checksums and event counters. If a particle hit occurs during Science Telemetry, the new data must be buffered and not placed in the Science File until the telemetry is done.

3.2 DEBIE Hardware Interface

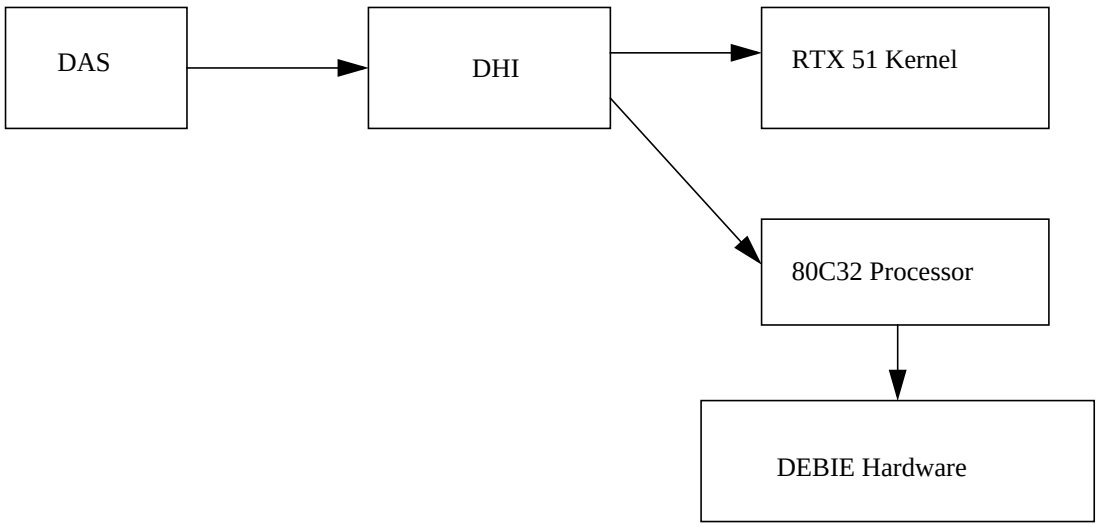
This part of the software is reponsible for providing hardware interface services listed in the section 2.1.1 on page 3. It shall also provide an interface to the Keil RTX 51 Real Time Kernel including at least the following services:

- Create task
- Send message to mailbox
- Read message from mailbox
- Attach interrupt
- Wait for interrupt
- Wait for time period

There shall be no tasks in this software entity, but only re-entrant service routines called by operations in either the DEBIE Application Software or DEBIE Test Software.

The following figure illustrates the role of the DEBIE Hardware Interface.

Figure 2: Role of DEBIE Hardware Interface



4. General System Design

This chapter describes general design principles used for development of DEBIE DPU SW in Keil PK51 environment.

4.1 The 80C32 Processor

In this section, we briefly describe the main features of the DEBIE DPU processor, the 80C32, to give some background information for the design choices.

4.1.1 Registers

The 80C32 instruction set uses eight 8-bit general registers, one 8-bit Accumulator register, an 8-bit Stack Pointer register, a 16-bit Data Pointer register, a 16-bit Program Counter register, and a Program Status Word containing various condition flags etc.

The eight general registers are duplicated in four *register banks*. A background task and an interrupt service can use different banks to reduce the time needed to save and restore registers. A field in the Program Status Word selects the current bank.

4.1.2 Memory

The memory space is divided into several regions as follows:

- *code*: program memory (up to 64 K Bytes; cannot be written to)
- *data*: directly accessible (128 bytes) internal data memory; the first 32 bytes overlay the four general register banks
- *idata*: indirectly accessible (256 bytes) internal data memory; the first 128 bytes overlay the “data” area; contains the hardware stack pointed to by the Stack Pointer register
- *bdata*: bit-addressable (16 bytes) internal data memory; overlays a part of the “data” area
- *xdata*: external data memory (64 KBytes); accessed through the Data Pointer register
- *pdata*: paged external data memory (256 bytes); usually overlays the first 256 bytes of the “xdata” area.

The “data” area can be accessed with immediate (absolute) addressing instructions. The “idata” area can be accessed only indirectly through any general register. The “xdata” area can be accessed only indirectly through the Data Pointer register.

The addresses 128 .. 255, when used as immediate addresses, refer to Special Function Registers such as interrupt controls, memory-mapped I/O registers and the like.

4.1.3 Instructions

Only the Accumulator register is capable of all arithmetic operations. The instruction set contains mainly 8-bit instructions. All immediate operands are 8 bits except for the branching and calling instructions (operand = code address) and the instruction that loads the Data Pointer (operand = xdata address).

The only instructions using 16-bit arithmetic are the Data Pointer increment instruction and the multiply operation, which yields a 16-bit product of two 8-bit numbers.

No floating point data types or operations are defined by the processor, but software libraries exist for this.

4.1.4 Memory Models

The Keil C51 compiler supports the following memory models [RD4]:

- SMALL: all variables, by default, reside in the internal data memory (idata) of the processor.
- COMPACT: all variables, by default, reside in one page (pdata) of external RAM. This memory model can accommodate a maximum of 256 bytes of variables.
- LARGE: all variables, by default, reside in external data memory (xdata). With this memory model all of the external memory (up to 64 KBytes) can be used to store variables, but memory access is inefficient and generates more code than SMALL and COMPACT.

In spite of selected default memory model each variable can be explicitly assigned to specific memory space with the corresponding keywords [RD4], eg. “idata” or “xdata”.

RTX-51 requires that reentrant functions are compiled in the COMPACT memory model and non-reentrant functions in SMALL or LARGE models. Because the SMALL model is not sufficient for DEBIE, the non-reentrant functions have to be compiled in the LARGE model.

If possible all local variables used by non-reentrant functions should be placed in internal memory with “data” or “idata” keyword. Priority for data or idata placement should be given to variables accessed frequently in performance-critical routines.

4.1.5 Interrupts

Six interrupts are available: External Interrupts 0 and 1; Timer 0 and Timer 1 Overflow Interrupts; Timer 2 Overflow-or-capture Interrupt, and the Serial Port Receive-or-Transmit Interrupt. Each interrupt source can be individually enabled (unmasked) or disabled (masked). There is also a common enable/disable control that affects all interrupt sources.

Interrupt service order is defined by interrupt priority level (high or low) and interrupt priority within level (fixed ordering).

The interrupt priority level controls the nesting of interrupt services: a high-level interrupt can interrupt the service of a low-level interrupt. The application program can assign any interrupt source to the high level or the low level.

Interrupt priority within level decides which of several simultaneous interrupts on the same level is serviced first; there is a fixed order of the six sources.

4.2 Use of RTX Kernel

A full version of the Keil RTX 51 real time kernel is used in the DEBIE DPU software. This kernel takes care of task management and communication, protection of shared resources and other responsibilities of a real time kernel relevant for the DEBIE software.

The following table lists the RTX functions which are to be used and their purpose. See [RD3] for detailed description of the functions.

Table 2: Used RTX functions

Name	Purpose
os_start_system	Initialize RTX-51 and start first task
os_create_task	Create new task and place it in list of ready tasks
os_enable_isr	Enable an interrupt source
os_disable_isr	Disable an interrupt source
oi_set_int_masks	Set interrupt mask bits
oi_reset_int_masks	Reset interrupt mask bits
os_attach_interrupt	Attach interrupt to the current task.
os_wait	Wait for event
os_send_message	Send a message to a mailbox from a task
isr_send_message	Send a message to a mailbox from an interrupt service
os_set_slice	Set the system time interval

4.3 Interrupt Management

TM and TC interrupts are handled by C-functions declared with “interrupt” key word or if necessary by assembler routines. Below is an example declaration of a C-function used for interrupt service (see [RD4] for details):

```
void ExampleISR (void) interrupt 1 using 3
/* Handles interrupt number 1 (Timer 0 Overflow) */
/* Uses register bank 3 */
```

The interrupt service routines use a dedicated register bank, different from the banks used by the tasks.

Interrupt service routines send messages to the tasks with the RTX function “isr_send_message”.

The hit trigger interrupt service is handled by a RTX51 “fast task”. This task is attached to the hit trigger interrupt with “os_attach_interrupt” function. The priority of this task is 3 and it uses a dedicated register bank, different from the banks used by other tasks and interrupt service routines. Messages to the Acquisition task are sent with the RTX function “os_send_message”.

The TM, TC and RTX timer interrupts are HW-connected (via programmable interrupt vectors) ISRs and cannot be preempted. Because the hit trigger interrupt is serviced by a “fast task” and not a HW-connected ISR, the TM , TC and RTX timer interrupts can preempt the hit trigger interrupt service. Thus, the preemption hierarchy is the following:

- ordinary tasks can be preempted by
- the Hit trigger interrupt, which can be preempted by
- the TM, TC and RTX timer interrupts, which are mutually non-preempting.

4.4 Task Management

All “real” tasks in the DEBIE software are “standard” RTX tasks. RTX “fast tasks” are used only for the hit trigger interrupt service (see [RD3] for definitions of standard and fast tasks). All RTX standard tasks use the register bank 0.

Tasks wait for messages to their mailboxes with RTX function “os_wait” and send messages to other tasks with “os_send_message” if needed.

Accesses to shared resources are protected by disabling interrupts. When access to a shared resource is needed the interrupts are disabled by clearing processor’s interrupt master enable/disable bit and when a shared resource is not needed anymore, that interrupt master bit is set (see [RD2]).

4.5 Stack Sizes

4.5.1 Standard task stacks

All standard tasks share a common stack area in the internal RAM of the processor. During task switches the contents of this area are copied to the task context area of the previous task in external RAM and the new stack contents are copied from the task context area of the next task from the external RAM [RD3].

The size of the standard task stack area can be configured in the RTX configuration. The required stack size is the maximum stack size used by any standard task plus stack space required by C51 interrupt service routines (see section 4.5.3).

Because of limited size of internal RAM available for stack areas normal functions compiled with Keil C51 compiler do not use stack for storing parameters and local variables. The stack is used only for storing return addresses. Therefore the required size of a stack can be determined from the call tree. The call trees will be generated using the Unix “cflow” tool.

4.5.2 Fast task stacks

The “fast tasks” have their own stack area. There is only one fast task in the DEBIE DPU Software: the hit trigger interrupt service task. The size of this stack can also be determined from the call tree (see above).

The size of the fast task stack can be configured in the RTX configuration.

4.5.3 Interrupt service stacks

The interrupt service routines (defined as C51 functions) use the stack of the interrupted task. At least ACC, B, DPH, DPL and PSW registers are stored if they are used in the service routines. Because only one C51 ISR can be running at a time, 7 bytes (5 for registers, 2 for return address) of additional space must be reserved for task stacks. If some other registers are stored in stack, the space required for them must be added also [RD3].

Space required from the longest call tree from any ISR must be added also. Only the “isr_send_message” function should be called from any ISR.

4.5.4 Re-entrant Functions

Normal C51 functions must not be used simultaneously by several tasks or interrupt functions. Any function that should be able to be called from several task simultaneously must be declared as “reentrant”. Below is an example of declaration of reentrant function (see [RD4] for details):

```
void ExampleReentrant (void) reentrant
```

RTX-51 supports reentrant fuctions only in the COMPACT memory model (see section 4.1.4). Each task contains a separate reentrant stack configurable in size. Reentrant functions may be used in combination with non-reentrant functions of the SMALL and LARGE models. Simultaneous use of reentrant and non-reentrant functions in COMPACT model is not allowed [RD3].

To minimize, or eliminate if possible, the required reentrant stack all parameters should be passed in processor registers and no local variables should be used in reentrant functions if possible.

Because the amount of parameters that can be passed in registers is quite limited, a pointer to a structure holding the parameters should be passed when many parameter values have to be passed to a reentrant function.

The need for local variables in reentrant functions can be avoided if the calling non-reentrant function reserves a work-area for the reentrant function and passes a pointer to it as a parameter. The declaration of a reentrant function in DEBIE software could look like following:

```
#define WORK_REENTRANT_SERVICE 10

void ReentrantService (
        struct ServiceParams_t *Params,
        int WorkArea[WORK_REENTRANT_SERVICE]
        ) reentrant
```

4.6 Portability of DEBIE Application Software

In order to maintain portability of the DEBIE Application Software the RTX functions must not be called directly from the DAS and the keywords of the language extensions of the Keil C51 compiler must not be written directly in the source code of DAS.

The RTX functions are called via the DEBIE Hardware Interface, which contains also the interface to the RTX, except in the C51 interrupt service routines the RTX functions are called via specific macros defined depending on the target environment.

The keywords specific to the Keil compiler are used via macros that are defined depending on the target environment. For the version suitable for native workstation environment these macros can be empty.

For example a task function declared with these macros could look like following:

```
#define TASK_DEFINITION _task_ 0 _priority_ 1

void TaskMain (void) TASK_DEFINITION
{
        /* some code here */
}
```

Note that the definitions of the macros should be in different files that can be selected depending on the environment.

See coding standard in [RD5] for actual rules to define these macros.

5. Dynamic Architecture

In the following figures the interrupt services and tasks of the DEBIE Application Software are described in outline.

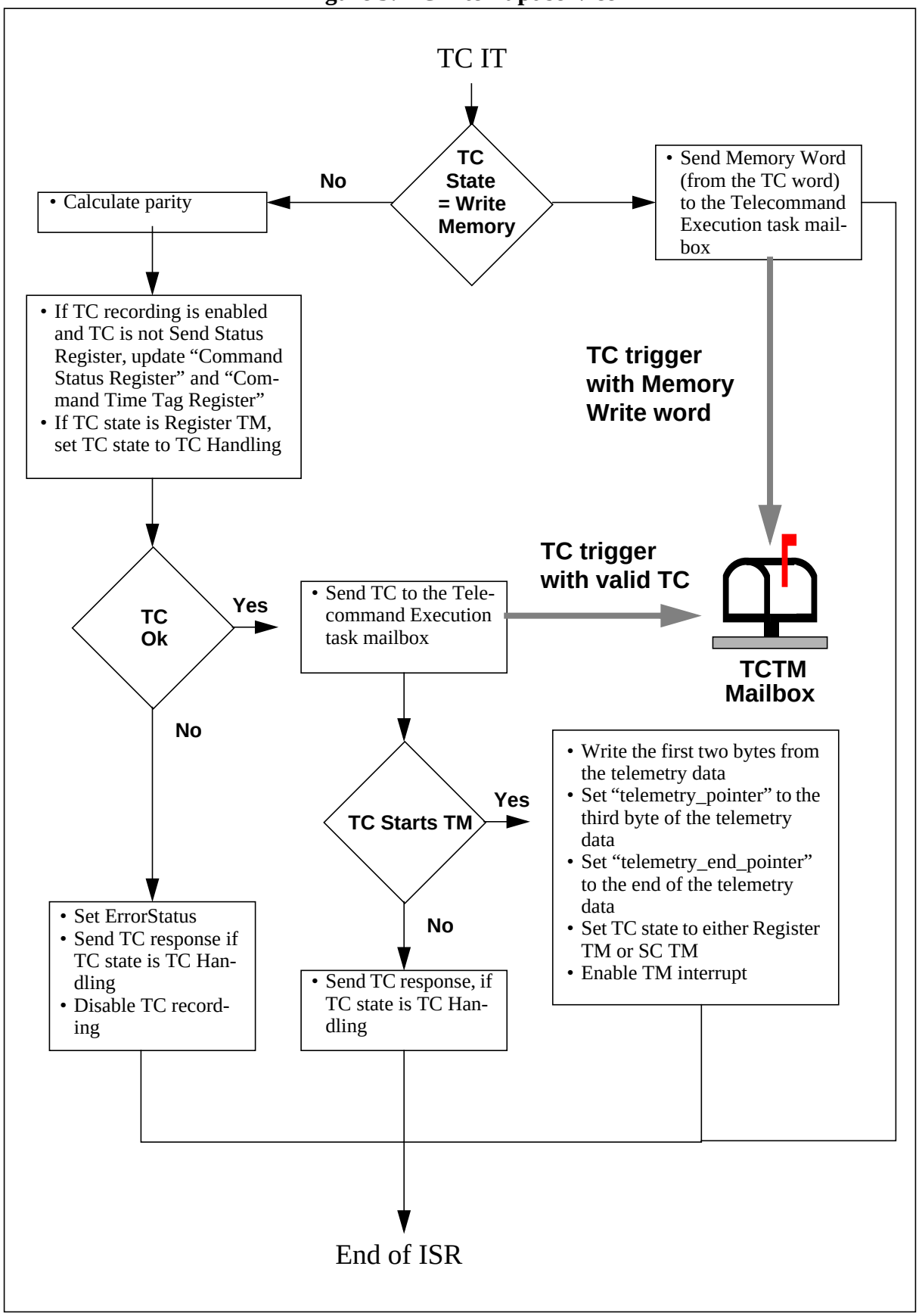
5.1 TC Interrupt Service

In the Keil PK51 environment the interrupt service is implemented as a C51 interrupt service routine (see chapter 4 for details).

The validity of the received telecommand is checked. The parity bit, telecommand address and telecommand code are checked. When an error is detected, The response is stored in the TM registers. In the Keil environment this is done by writing values to registers in the DEBIE TM hardware via specific macros.

The TC interrupt service sends the received telecommand to the Telecommand Execution Task. In the Keil environment the “isr_send_message” function is called via a specific macro.

Figure 3: TC interrupt service



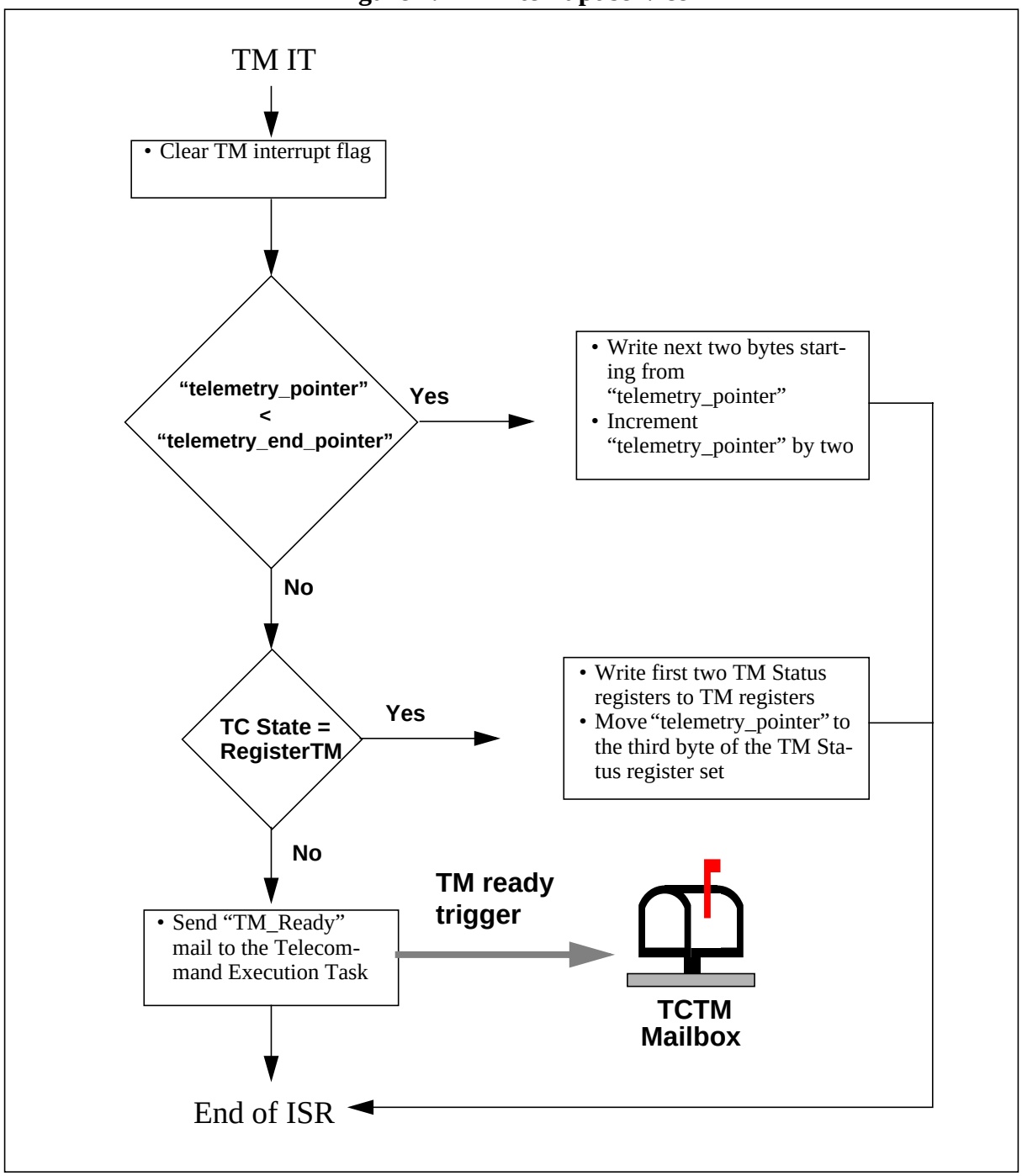
5.2 TM Interrupt Service

The TM interrupt service is implemented in the same way as the TC interrupt service. However, the TM interrupt is enabled only during the transmission of TM register telemetry, Science telemetry, or Memory Dump telemetry; it is disabled for the transmission of the TM response to normal TCs.

When the TM interrupt occurs (and is enabled), the next two bytes from the specified buffer are written to the TM hardware registers. This is handled in the same way as the response to the TC interrupts in the TC interrupt service.

If the requested telemetry was Science Telemetry or the contents of a memory block and the last bytes of that telemetry were sent, a “TM_Ready” message is sent to the Telecommand Execution Task in the way described above in the section 5.1.

Figure 4: TM interrupt service



5.3 Telecommand Execution Task

In the Keil environment this task is defined in the way described in the section 4.4 and with the RTX function "os_create_task", which is called via a specific function in the DEBIE Hardware Interface.

The Telecommand Execution task loop is described in the figure 5.

The execution of Telecommand Execution task operations starts when some mail comes from either the TC Interrupt or TM Interrupt handler. The next actions depend on the source of the received mail as follows.

If the mail is caused by a TC Interrupt, the TC State defines whether a new Telecommand is to be executed, a new TC word belongs to the memory patch sequence, the LSB of Read or Write Memory block is to be received, or sending of TM Register is to be stopped. For clarity and lack of space these actions are not described in detail in the figure 5.

If the mail is caused by a TM Interrupt (“TM_Ready” message), the TC State is set to TC_Handling, the TM interrupt is disabled. If the TC state was Science TM, the Science Data File is made empty, the Event Counters are cleared, and any buffered events received during the Science TM are entered in the Science Data File.

Table 3 describes the meaning of each TC state and figure 6 describes the transitions between TC states.

All telecommands except telemetry requests must be executed in less than 10 ms. The immedate response to the telecommands are sent by the TC interrupt handler in less than 1 ms.

Figure 5: Telecommand and Telemetry task loop

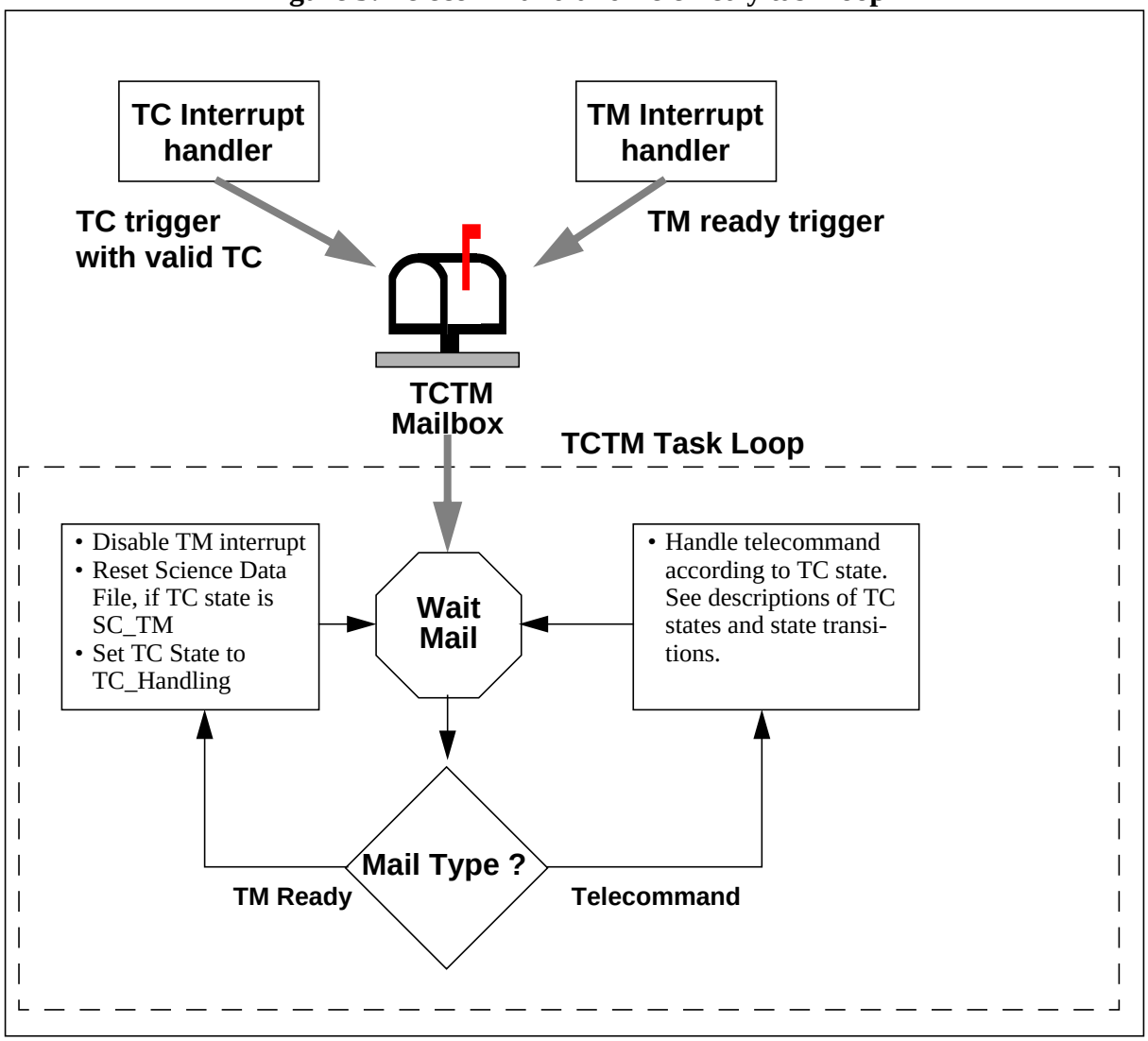
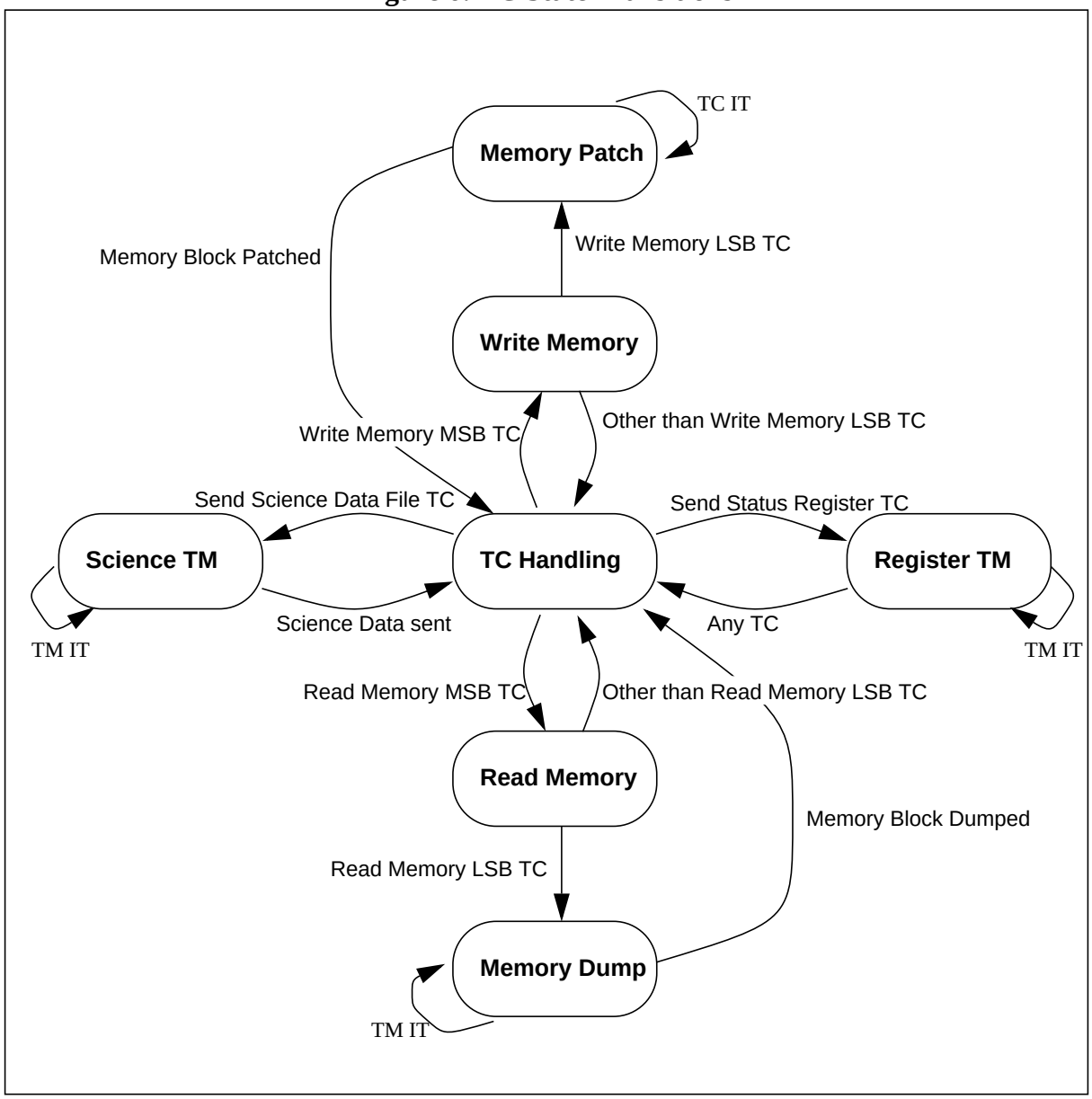


Table 3: Descriptions of the TC states

TC State	Description
TC Handling	In this state most of the telecommands are executed. This is the default TC state.
Science TM	In this state the Science Data File is sent to the telemetry.
Register TM	In this state the TM status registers are sent to the telemetry.
Memory Dump	In this state 32 bytes of data memory are sent to the telemetry.
Memory Patch	In this state 32 bytes of data or code memory are patched, and then for the Write Program Memory sequence, the action defined by the last TC word of the “Program Memory Patch Command Sequence” is performed (action is “continue normally” for the Write Data Memory sequence).
Read Memory & Write Memory	These are auxiliary states for Memory Dump and Memory Patch, where the MSB of the memory block concerned is defined and the LSB is expected to be defined by the next telecommand.

Figure 6: TC State Transitions



5.4 Health Monitoring Task

This is the first task to be started. This is done in the Keil environment with the “os_start_system” function, which is called via a specific function in the DEBIE Hardware Interface.

The Health Monitoring task is described in the figures 7 and 8. Note that the figures apply to the final software and in the Prototype Software only Sensor Unit temperatures are measured.

Initialization is done before entering the task loop. InitSystem() function creates necessary RTX tasks and sets the system clock interval. Also the Self Test sequence is executed before the loop.

In the loop the DPU time in the Housekeeping Telemetry buffers (TM registers) is advanced and the Monitor() function is called. After that the task waits for a time-out. In the Keil environment the time-out is implemented with the RTX-51 time interval event. When the time interval elapses the loop is continued from the beginning.

The secondary power supply voltages are measured once in 180 seconds and the sensor unit temperatures are measured once in 60 seconds. If a particle hit occurs during health monitoring measurements the measurement is repeated. The amount of repetitions is limited depending on the measurement to prevent the program from getting stuck in a loop. Once a valid result is obtained, it is stored in the Housekeeping Telemetry buffer (TM registers). If a valid result is not obtained for a temperature measurement, the corresponding sensor unit is switched off.

Low (+/- 5V) and high (+/- 50V) supply voltages are monitored against short circuiting once in 10 seconds. If a low voltage output is suffering from a short circuit, all sensor units are switched off. If a high voltage output is suffering from a short circuit, only the corresponding sensor unit is switched off. If a temperature measurement indicates that a Sensor Unit is overheated, all power supply lines to it are switched off.

Checksum from the next part of the code memory is calculated. The size of the part of memory whose checksum is to be calculated in one cycle of the task execution is set so that the checking of the whole of the code memory takes 60 seconds. If an anomaly is detected in the code memory soft reset is performed i.e the program code is copied from PROM to RAM but the Failure Status and counters are preserved, and DPU Self Test mode is entered.

If an anomaly is enountered during the health monitoring, the Error, Mode and SU status registers, related to a given case, are updated in the Housekeeping Telemetry buffers (see [AD2] and [RD6] for details).

The task is triggered every second, and internal counters indicate the 10, 60 and 180 second periods; this counter manipulation is omitted from the figure for clarity.

5.4.1 SU Self Test

SU Self test execution [AD3] is a part of health monitoring cycle. The SU self test action is initiated with specific TC. SU self test start up duration depends on health monitoring cycle phase, and takes no more than 10 seconds after received TC to start. SU self test takes two health monitoring cycles, i.e. 2 seconds.

In the first cycle, the SU self test action reads V_DOWN and HV status registers, measures +/-5V and +/-50V power supplies and measures temperature 1 and 2 values. If V_DOWN is active, the sensor unit is switched off and corresponding SU error is set into the error register and the LV supply error bit is set in status register of the tested SU. SU self test is aborted . If the HV status register bit is active, the corresponding SU error is set into the error status register and the HV supply error bit is set in the SU status register. In power supply measuring if any of the measured voltages is out of limit, the corresponding supply error bit is set. Temperature values are compared against high limit values. If the temperature exceeds the limit the SU is switched off and the corresponding temperature error bit is set. The SU self test is aborted.

In the second cycle all the sensor channels, plasma 1e/2e/1i and piezo 1/2, are tested. Channels are tested one at the time. The threshold levels for the channels not under the test are set to maximum value, and the test threshold level is set for the channel to be tested. The test pulse trigger interrupt is caused by changing the DAC channel register value from 0 level to test pulse level and back to 0 level. The test pulse level is channel specific. For the plasma 2e channel this test pulse trigger interrupt is done explicitly by setting the processor's interrupt pending bit, because this channel is not able to generate a hit trigger interrupt.

Figure 7: Health Monitoring task

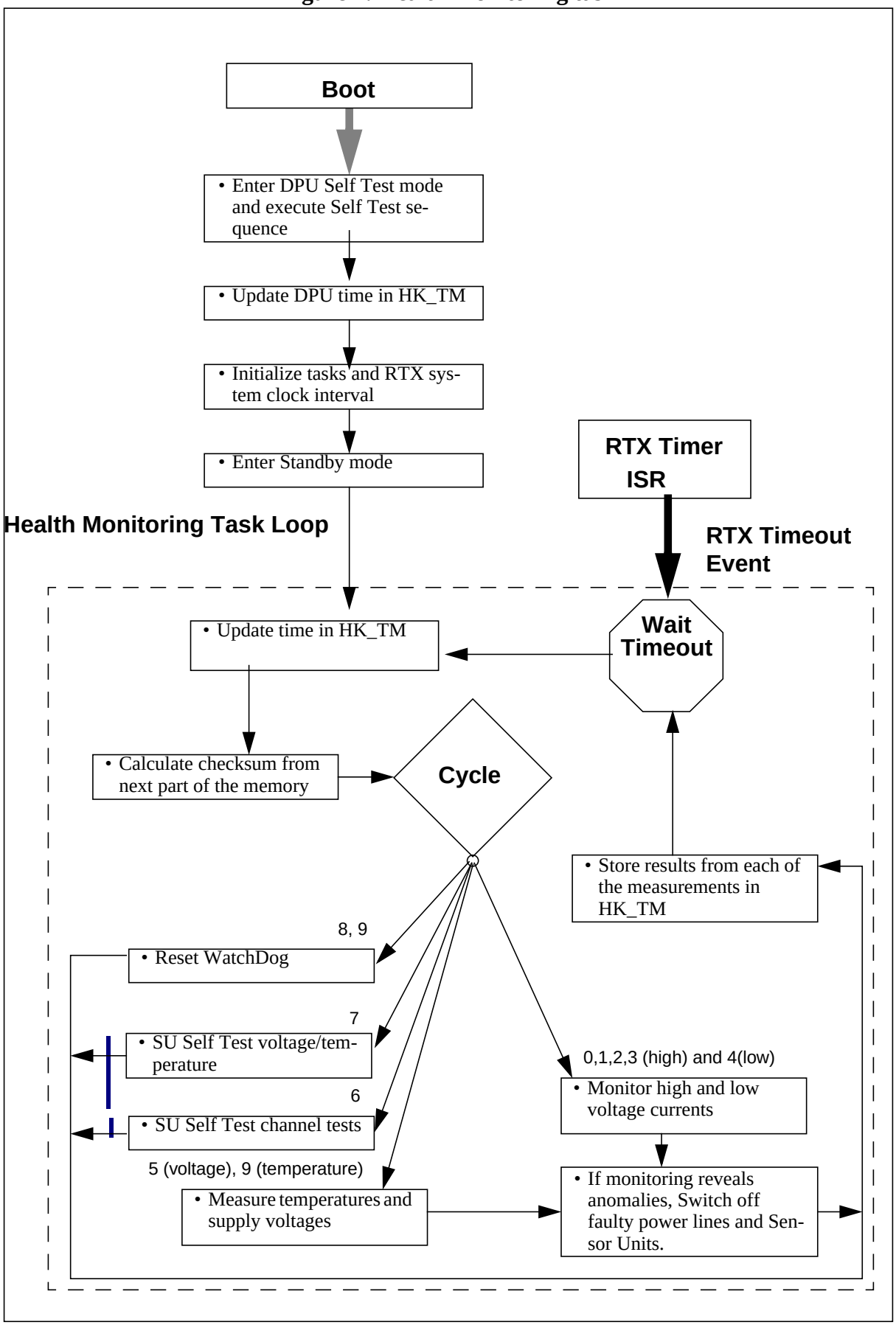


Figure 8: Health monitoring function hierarchy

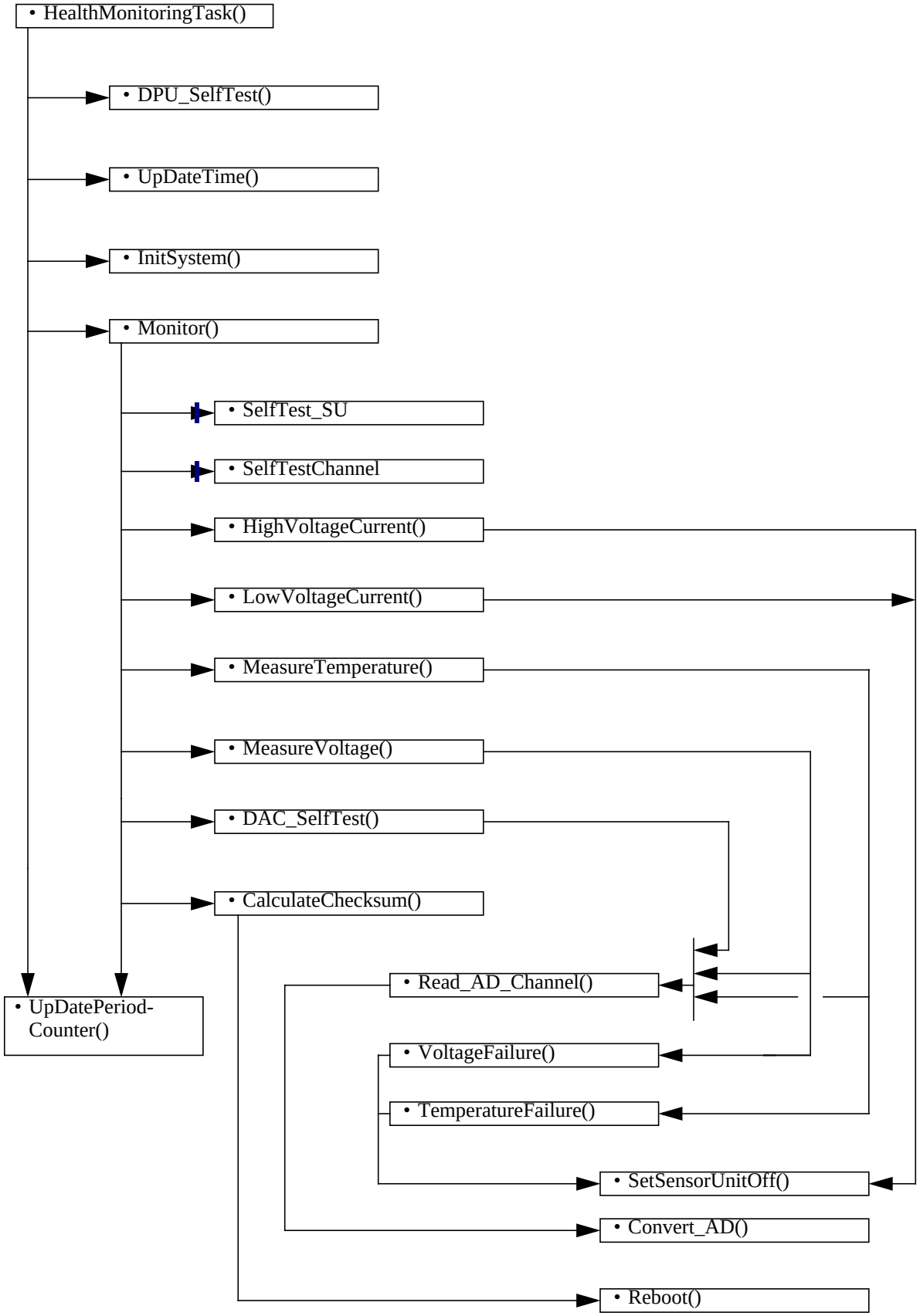
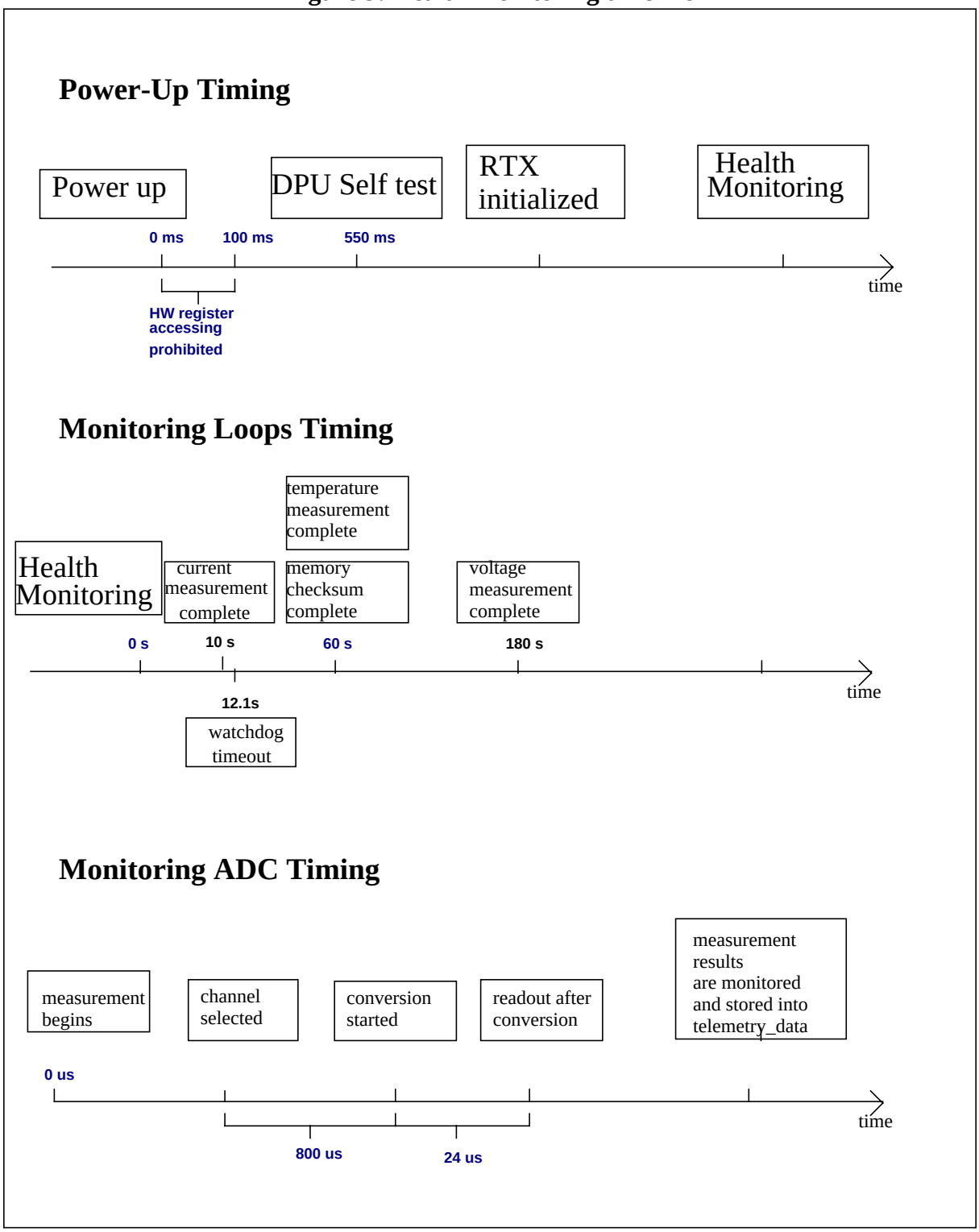


Figure 9: Health Monitoring timeline

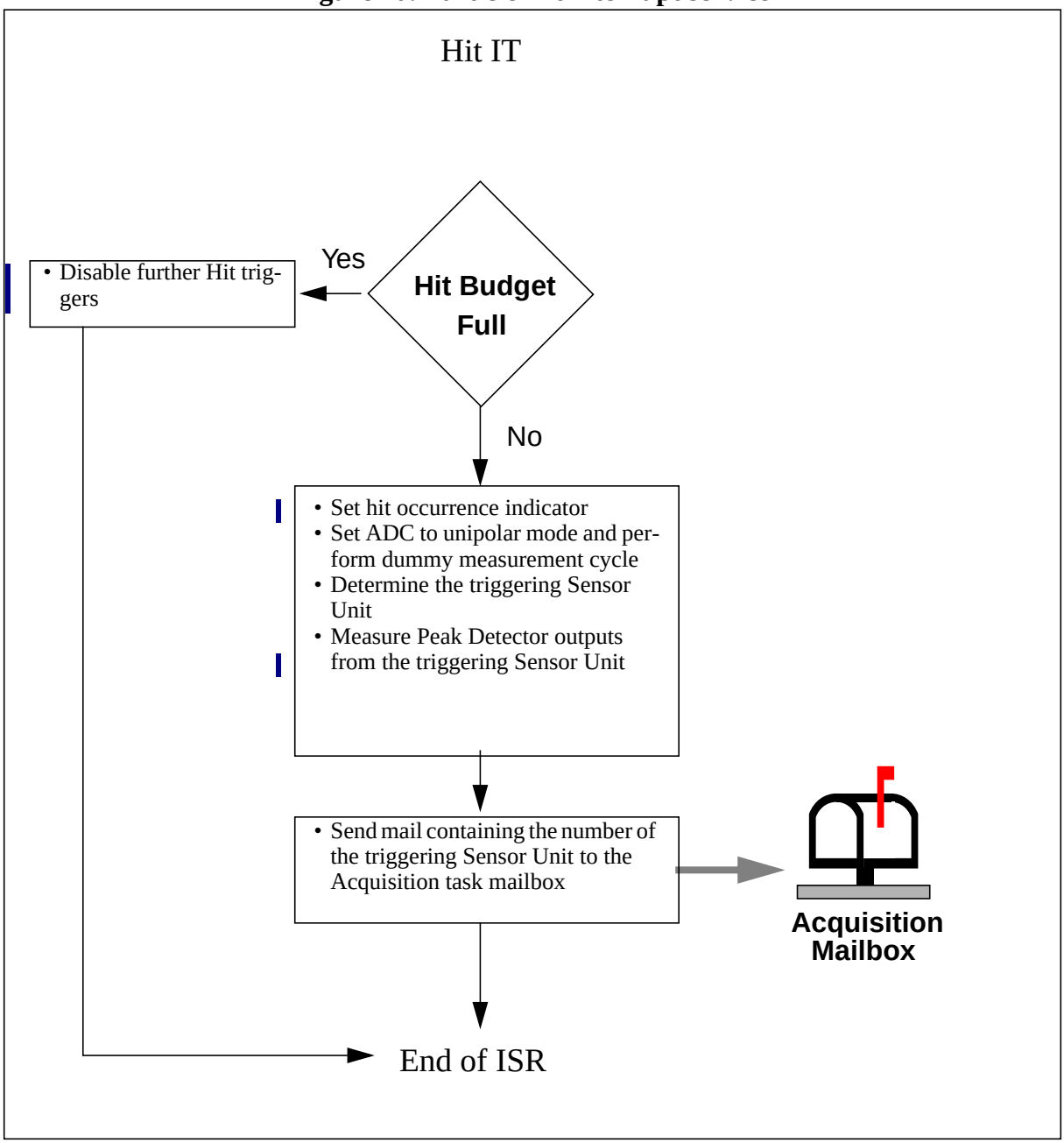


5.5 Hit Trigger Interrupt Service

In the Keil environment this interrupt service is implemented by a RTX-51 fast task (see chapter 4 for details).

The five analog outputs of Peak Detectors from the triggering Sensor Unit are measured and stored in a temporary buffer. Then a message holding the number of the triggering Sensor Unit is sent to the Acquisition Task. In the Keil enviroment the “os_send_message” function is called via a specific function in the DEBIE Hardware Interface.

Figure 10: Particle Hit interrupt service



5.6 Acquisition Task

The Acquisition task loop is described in the figure 11.

When some mail comes to the Acquisition task mailbox, the Peak Detector outputs sampled by the interrupt service are fetched. Then the Pulse Rise Time and the delays between trigger signals are measured, the measurement time is read and the Sensor Unit temperatures fetched from the Housekeeping Telemetry registers.

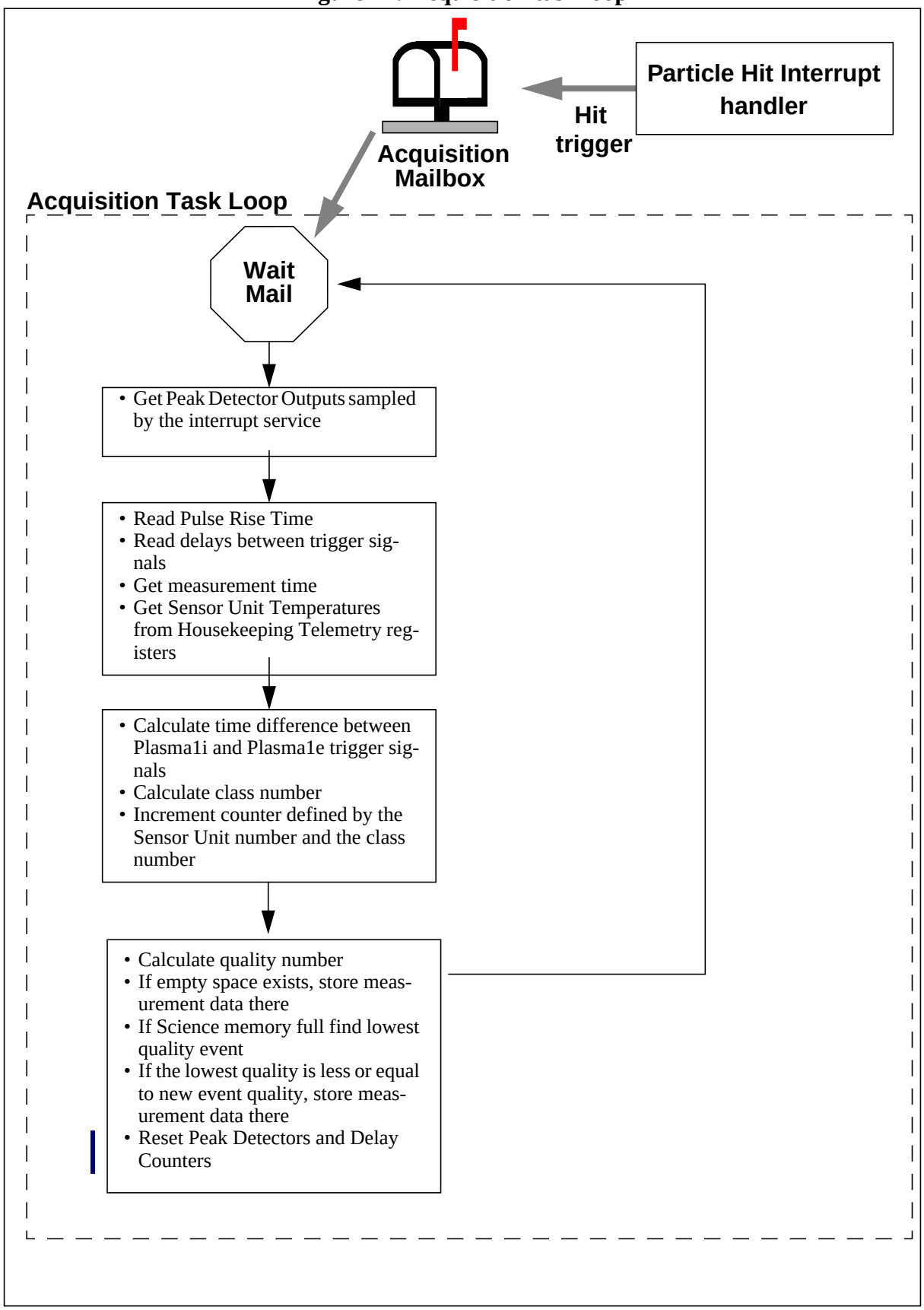
The time difference between the Plasma1i and Plasma1e trigger signals are calculated.

From the measurement data the class number of the event is determined by comparing the amplitude of each measured sensor signal to classification threshold levels and comparing the relative timing of the sensor signals to timing windows. This class number and the number of the triggering Sensor Unit defines an index for the event counter which is incremented.

If the event was triggered by Sensor Unit being self tested, a quality number is set to its maximum value. Otherwise the quality number of the trigger event is calculated with the formula defined in [AD1].

When the Event data is gathered the Event record holding it is stored - or at least attempted to be stored - in the Science Data File in the way described in section 2.1.5.

Figure 11: Acquisition task loop



6. Static Architecture

The following chapters are extracted from the HoodNICE database used for the architectural design of the DEBIE DPU Software. There are two top level objects DEBIE Application Software (DAS) and DEBIE HW/SW Interface (DHI) and their child objects, and one environmental object RTX_51 (real time kernel) described in these chapters.

The descriptions are concentrated to the graphical descriptions of the objects and the interfaces between them. Also the operations provided by each object are outlined by describing operation sets provided by different objects. Individual operations are not defined in the HoodNICE database in order to avoid need for additional maintenance work during the detailed design phase. The HoodNICE is going to be used only during the architectural design.

The static architecture described here presents the final software. The Prototype SW is slightly different. For example the Science Data and the operations meant for handling it (presented by operation set AcquisitionData) are moved to Telemetry object in the Prototype Software, because the Classification object is a null object in the Prototype Software.

6.1 Environment Object : RTX_51

6.1.1 Problem definition

6.1.2 Formalization of the solution

6.1.2.1 Object Description Skeleton

```
object RTX_51 is environment passive
    description
    implementation_constraints
    provided_interface
        none

end_object RTX_51
```

6.1.2.2 Generated spec code

6.1.2.3 Generated body code

6.2 Object : DEBIE

6.2.1 Problem definition

6.2.2 Formalization of the strategy

6.2.2.1 Identification of objects

DAS

Kind ACTIVE

DHI

Kind PASSIVE

6.2.2.2 Identification of operations

Operation Dictionary

HANDLE_TRIGGER_IT

Provided by DAS

HANDLE_TM_IT

Provided by DAS

HANDLE_TC_IT

Provided by DAS

MAIN_TASK

Provided by DAS

Operation Set Dictionary

DPU_Control

Provided by DHI

TmTcControl

Provided by DHI

TaskControl

Provided by DHI

InterruptControl

Provided by DHI

MessageControl

Provided by DHI

AD_Conversion

Provided by DHI

SU_Control

Provided by DHI

6.2.2.3 Grouping operations and objects

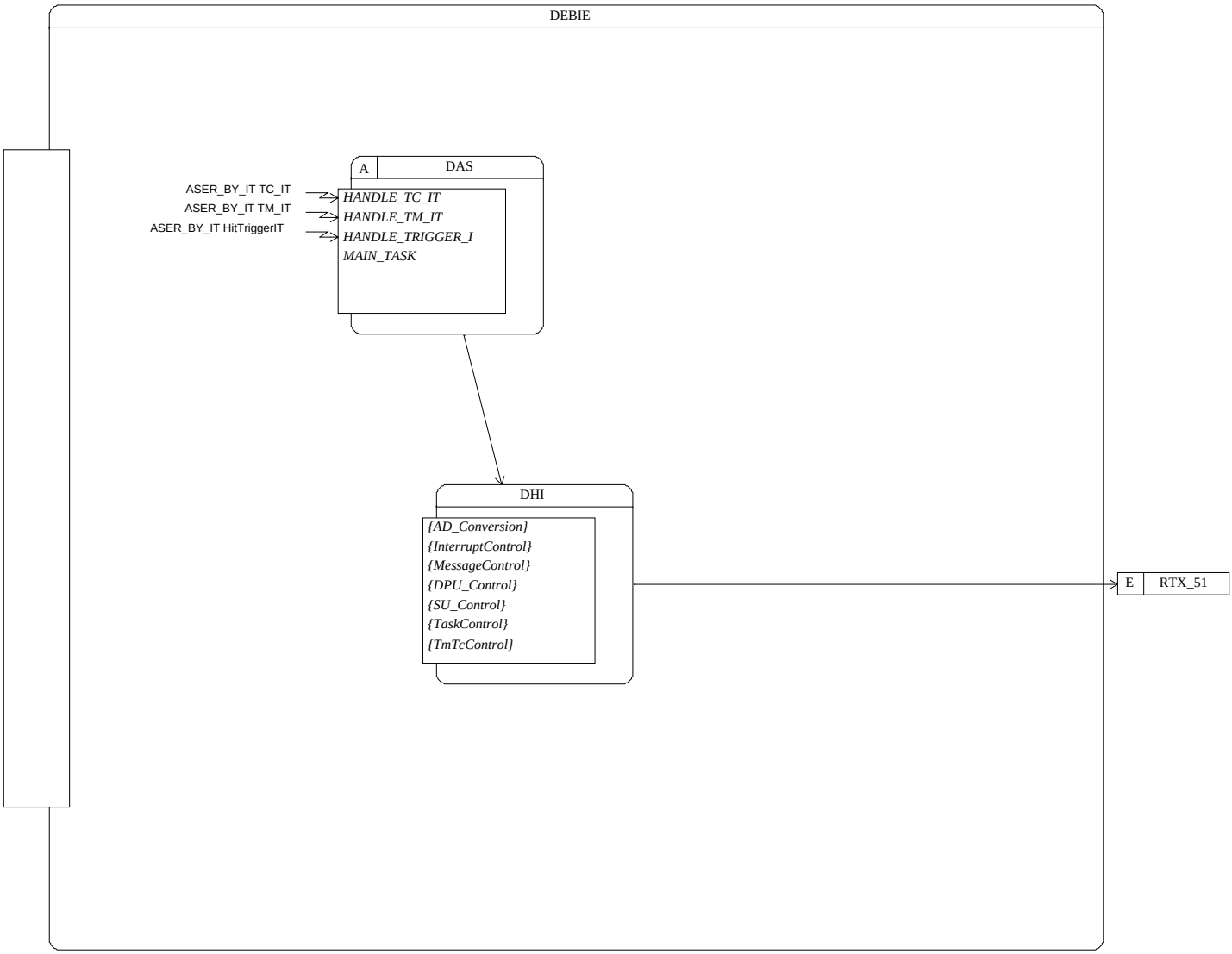
DAS

HANDLE_TC_IT
HANDLE_TM_IT
HANDLE_TRIGGER_IT
MAIN_TASK

DHI

AD_Conversion
InterruptControl
MessageControl
DPU_Control
SU_Control
TaskControl
TmTcControl

6.2.2.4 Graphical description



6.2.3 Formalization of the solution

6.2.3.1 Generated spec code

```
procedure DEBIE is
begin
    null;
end;
```

6.3 Object : DAS

6.3.1 Problem definition

6.3.2 Formalization of the strategy

6.3.2.1 Identification of objects

Acquisitions
Kind ACTIVE
TmTcInterface
Kind ACTIVE
HealthMonitoring
Kind ACTIVE

6.3.2.2 Identification of operations

Operation Dictionary

HANDLE_TRIGGER_IT
Provided by Acquisitions
HANDLE_TM_IT
Provided by TmTcInterface
HANDLE_TC_IT
Provided by TmTcInterface
HealthMonitoringTask
Provided by HealthMonitoring
TelecommandExecutionTask
Provided by TmTcInterface
AcquisitionTask
Provided by Acquisitions

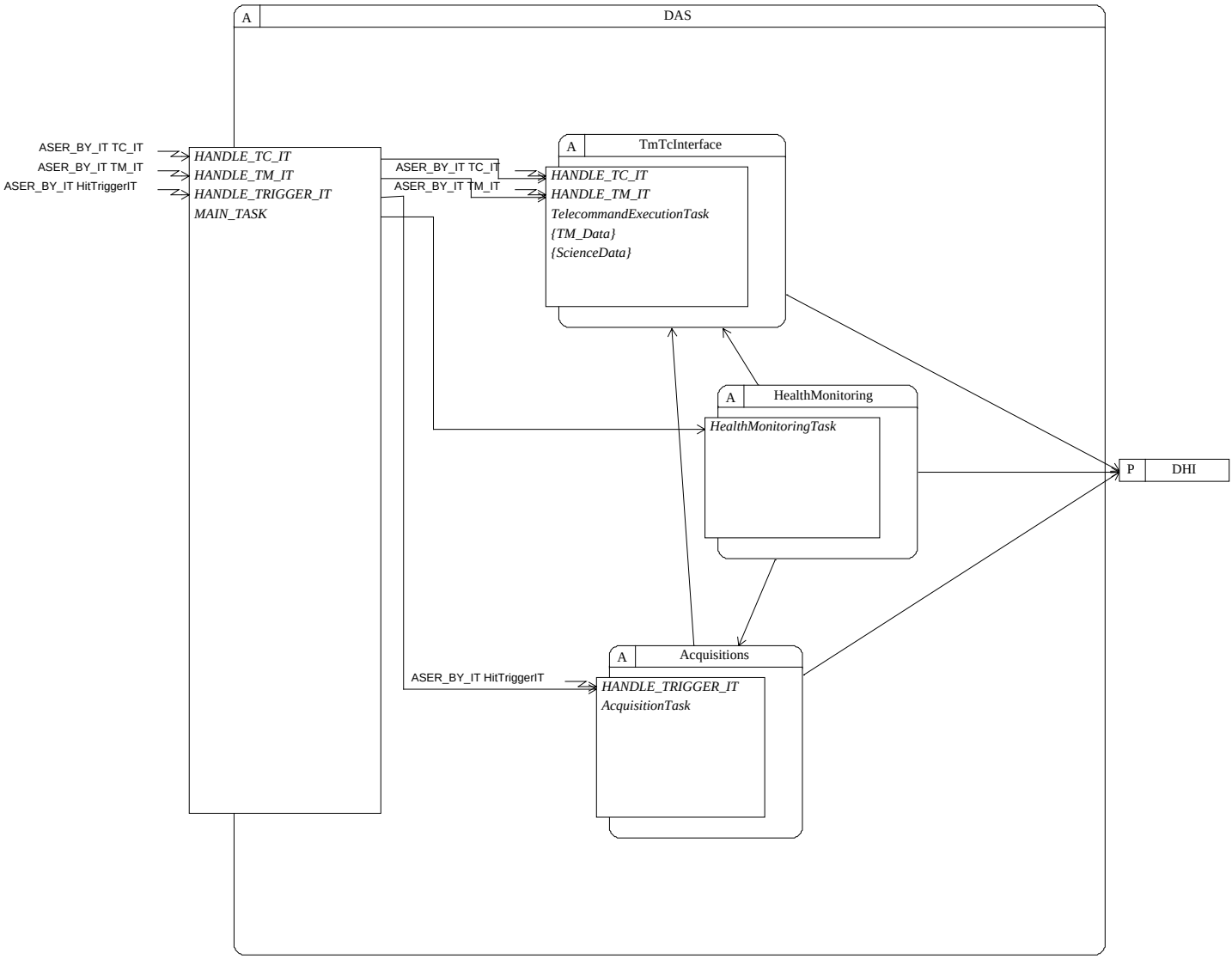
Operation Set Dictionary

TM_Data
Provided by TmTcInterface
ScienceData
Provided by TmTcInterface

6.3.2.3 Grouping operations and objects

Acquisitions
HANDLE_TRIGGER_IT
AcquisitionTask
TmTcInterface
HANDLE_TC_IT
HANDLE_TM_IT
TelecommandExecutionTask
TM_Data
ScienceData
HealthMonitoring
HealthMonitoringTask

6.3.2.4 Graphical description



6.3.3 Formalization of the solution

6.3.3.1 Object Description Skeleton DAS

object DAS **is active**
description
implementation_constraints
provided_interface
operations
HANDLE_TC_IT ;
HANDLE_TM_IT ;
HANDLE_TRIGGER_IT ;
MAIN_TASK ;
object_control_structure
description
pragma generate_obcs_task ;

```
        constrained_operations
            HANDLE_TM_IT  constrained_by  aser_by_it TM_IT;
            HANDLE_TRIGGER_IT  constrained_by  aser_by_it HitTriggerIT;
            HANDLE_TC_IT  constrained_by  aser_by_it TC_IT;
    required_interface
        object  DHI ;
            operation_sets
                TmTcControl ;
                TaskControl ;
                SU_Control ;
                DPU_Control ;
                MessageControl ;
                InterruptControl ;
                AD_Conversion ;

    internals
        objects
            Acquisitions ;
            TmTcInterface ;
            HealthMonitoring ;
        operations
            HANDLE_TC_IT  implemented_by TmTcInterface.HANDLE_TC_IT;
            HANDLE_TM_IT  implemented_by TmTcInterface.HANDLE_TM_IT;
            HANDLE_TRIGGER_IT  implemented_by Acquisitions.HANDLE_TRIGGER_IT;
            MAIN_TASK  implemented_by HealthMonitoring.HealthMonitoringTask;
end_object DAS
```

6.3.3.2 Generated spec code

6.3.3.3 Generated body code

6.4 Object : Acquisitions

6.4.1 Problem definition

6.4.2 Formalization of the strategy

6.4.2.1 Identification of objects

Measurements
Kind ACTIVE
Classification
Kind PASSIVE

6.4.2.2 Identification of operations

Operation Dictionary

HANDLE_TRIGGER_IT
Provided by Measurements
AcquisitionTask
Provided by Measurements

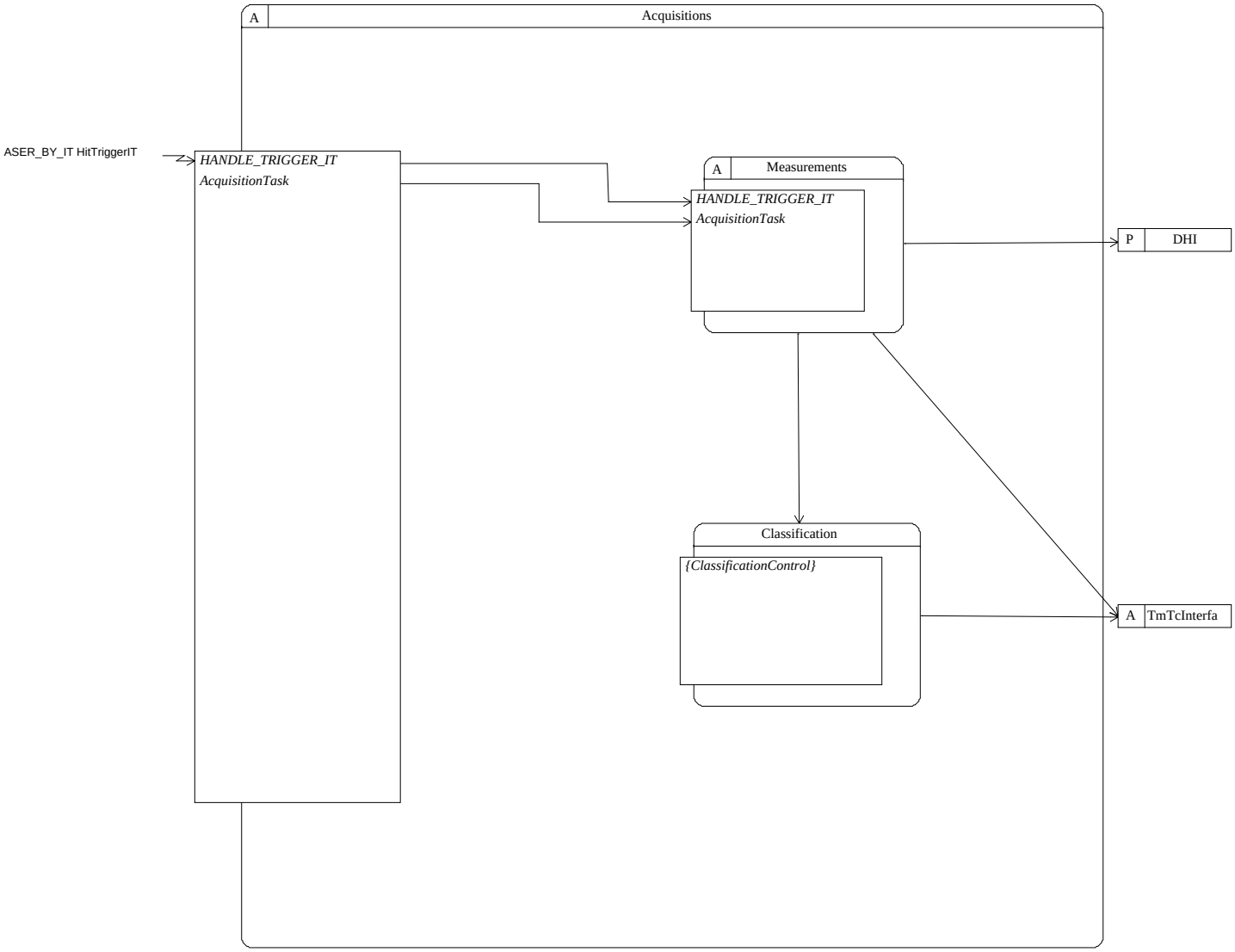
Operation Set Dictionary

ClassificationControl
Provided by Classification

6.4.2.3 Grouping operations and objects

Measurements
HANDLE_TRIGGER_IT
AcquisitionTask
Classification
ClassificationControl

6.4.2.4 Graphical description



6.4.3 Formalization of the solution

6.4.3.1 Object Description Skeleton Acquisitions

object Acquisitions **is active**
description
implementation_constraints
provided_interface
operations
HANDLE_TRIGGER_IT ;
AcquisitionTask ;
object_control_structure
description
pragma generate_obcs_task ;
constrained_operations
HANDLE_TRIGGER_IT **constrained_by aser_by_it** HitTriggerIT;
required_interface

```
        object DHI ;
            operation_sets
                AD_Conversion ;
                MessageControl ;
                TaskControl ;
                InterruptControl ;
                SU_Control ;
        object TmTcInterface ;
            operation_sets
                ScienceData ;
                TM_Data ;

    internals
        objects
            Measurements ;
            Classification ;
        operations
            HANDLE_TRIGGER_IT  implemented_by Measurements.HANDLE_TRIGGER_IT;
            AcquisitionTask  implemented_by Measurements.AcquisitionTask;
end_object Acquisitions
```

6.4.3.2 Generated spec code

6.4.3.3 Generated body code

6.5 Object : Measurements

6.5.1 Problem definition

6.5.2 Formalization of the solution

6.5.2.1 Object Description Skeleton Measurements

```
object Measurements is active
    description
    implementation_constraints
    provided_interface
        operations
            HANDLE_TRIGGER_IT ;
            AcquisitionTask ;
    object_control_structure
        description
            pragma generate_obcs_task ;
    required_interface
        object Classification ;
            operation_sets
                ClassificationControl ;
        object DHI ;
            operation_sets
                AD_Conversion ;
                MessageControl ;
                SU_Control ;
                TaskControl ;
                InterruptControl ;
        object TmTcInterface ;
            operation_sets
                TM_Data ;

    internals
    object_control_structure
        description
    operation_control_structures

    operation HANDLE_TRIGGER_IT
        description
    end_operation HANDLE_TRIGGER_IT

    operation AcquisitionTask
        description
    end_operation AcquisitionTask

end_object Measurements
```

6.5.2.2 Generated spec code

6.5.2.3 Generated body code

6.6 Object : Classification

6.6.1 Problem definition

6.6.2 Formalization of the solution

6.6.2.1 Object Description Skeleton Classification

```
object Classification is passive
    description
    implementation_constraints
    provided_interface
        operation_sets
            ClassificationControl ;
    required_interface
        object TmTcInterface ;
            operation_sets
                ScienceData ;

    internals
end_object Classification
```

6.6.2.2 Generated spec code

6.6.2.3 Generated body code

6.7 Object : TmTcInterface

6.7.1 Problem definition

6.7.2 Formalization of the strategy

6.7.2.1 Identification of objects

TmTcHandler
Kind ACTIVE
Telemetry
Kind ACTIVE

6.7.2.2 Identification of operations

Operation Dictionary

HANDLE_TC_IT
Provided by TmTcHandler
HANDLE_TM_IT
Provided by Telemetry
TelecommandExecutionTask
Provided by TmTcHandler

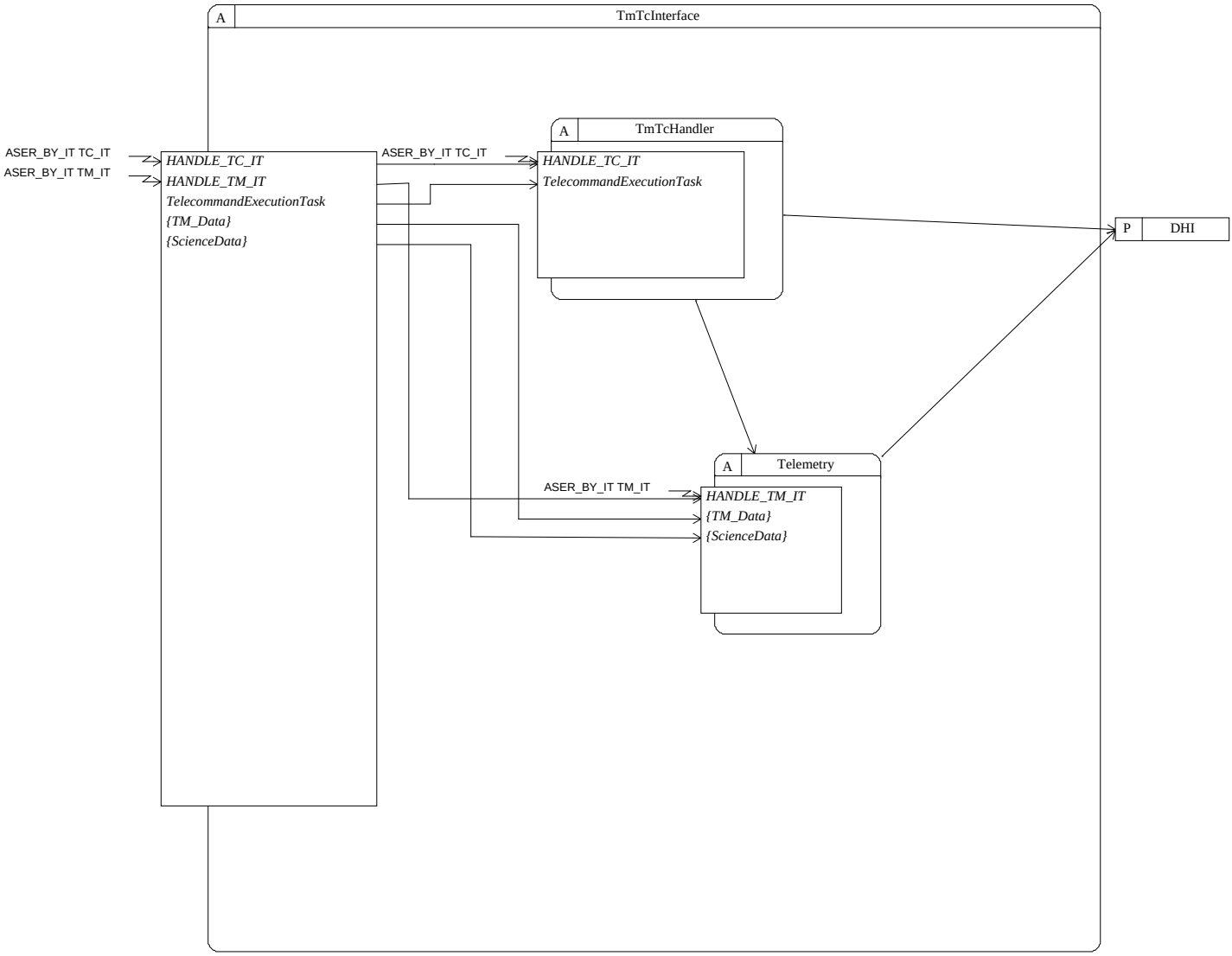
Operation Set Dictionary

TM_Data
Provided by Telemetry
ScienceData
Provided by Telemetry

6.7.2.3 Grouping operations and objects

TmTcHandler
HANDLE_TC_IT
TelecommandExecutionTask
Telemetry
HANDLE_TM_IT
TM_Data
ScienceData

6.7.2.4 Graphical description



6.7.3 Formalization of the solution

6.7.3.1 Object Description Skeleton TmTcInterface

object TmTcInterface **is active**
description
implementation_constraints
provided_interface
operations
HANDLE_TC_IT ;
HANDLE_TM_IT ;
TelecommandExecutionTask ;
operation_sets
TM_Data ;
ScienceData ;
object_control_structure

```
        description
            pragma generate_obcs_task ;
        constrained_operations
            HANDLE_TM_IT constrained_by aser_by_it TM_IT;
            HANDLE_TC_IT constrained_by aser_by_it TC_IT;
    required_interface
        object DHI ;
            operation_sets
                MessageControl ;
                SU_Control ;
                TmTcControl ;
                DPU_Control ;
                InterruptControl ;
                TaskControl ;

    internals
        objects
            TmTcHandler ;
            Telemetry ;
        operations
            HANDLE_TC_IT implemented_by TmTcHandler.HANDLE_TC_IT;
            HANDLE_TM_IT implemented_by Telemetry.HANDLE_TM_IT;
            TelecommandExecutionTask implemented_by TmTcHandler.TelecommandExecutionTask;
        operation_sets
            TM_Data implemented_by Telemetry.TM_Data;
            ScienceData implemented_by Telemetry.ScienceData;
end_object TmTcInterface
```

6.7.3.2 Generated spec code

6.7.3.3 Generated body code

6.8 Object : TmTcHandler

6.8.1 Problem definition

6.8.2 Formalization of the solution

6.8.2.1 Object Description Skeleton TmTcHandler

```
object TmTcHandler is active
    description
    implementation_constraints
    provided_interface
        operations
            HANDLE_TC_IT ;
            TelecommandExecutionTask ;
    object_control_structure
        description
            pragma generate_obcs_task ;
        constrained_operations
            HANDLE_TC_IT constrained_by aser_by_it TC_IT;
    required_interface
        object Telemetry ;
            operation_sets
                TM_Data ;
                ScienceData ;
        object DHI ;
            operation_sets
                TmTcControl ;
                InterruptControl ;
                MessageControl ;
                DPU_Control ;
                SU_Control ;
                TaskControl ;

    internals
    object_control_structure
        description
    operation_control_structures

    operation HANDLE_TC_IT
        description
    end_operation HANDLE_TC_IT

    operation TelecommandExecutionTask
        description
    end_operation TelecommandExecutionTask

end_object TmTcHandler
```

6.8.2.2 Generated spec code

6.8.2.3 Generated body code

6.9 Object : Telemetry

6.9.1 Problem definition

6.9.2 Formalization of the solution

6.9.2.1 Object Description Skeleton Telemetry

```
object Telemetry is active
    description
    implementation_constraints
    provided_interface
        operations
            HANDLE_TM_IT ;
        operation_sets
            TM_Data ;
            ScienceData ;
    object_control_structure
        description
            pragma generate_obcs_task ;
        constrained_operations
            HANDLE_TM_IT constrained_by aser_by_it TM_IT;
    required_interface
        object DHI ;
            operation_sets
                TmTcControl ;
                MessageControl ;

    internals
    object_control_structure
        description
    operation_control_structures

    operation HANDLE_TM_IT
        description
    end_operation HANDLE_TM_IT

end_object Telemetry
```

6.9.2.2 Generated spec code

6.9.2.3 Generated body code

6.10 Object : HealthMonitoring

6.10.1 Problem definition

6.10.2 Formalization of the solution

6.10.2.1 Object Description Skeleton HealthMonitoring

```
object HealthMonitoring is active
    description
    implementation_constraints
    provided_interface
        operations
            HealthMonitoringTask ;
    object_control_structure
        description
            pragma generate_obcs_task ;
    required_interface
        object DHI ;
            operation_sets
                AD_Conversion ;
                InterruptControl ;
                DPU_Control ;
                SU_Control ;
                TaskControl ;
        object TmTcInterface ;
            operation_sets
                TM_Data ;
                ScienceData ;
            operations
                TelecommandExecutionTask ;
        object Acquisitions ;
            operations
                AcquisitionTask ;

    internals
    object_control_structure
        description
    operation_control_structures

    operation HealthMonitoringTask
        description
    end_operation HealthMonitoringTask

end_object HealthMonitoring
```

6.10.2.2 Generated spec code

6.10.2.3 Generated body code

6.11 Object : DHI

6.11.1 Problem definition

6.11.2 Formalization of the strategy

6.11.2.1 Identification of objects

HW_Interface
Kind PASSIVE
RTX_Interface
Kind PASSIVE

6.11.2.2 Identification of operations

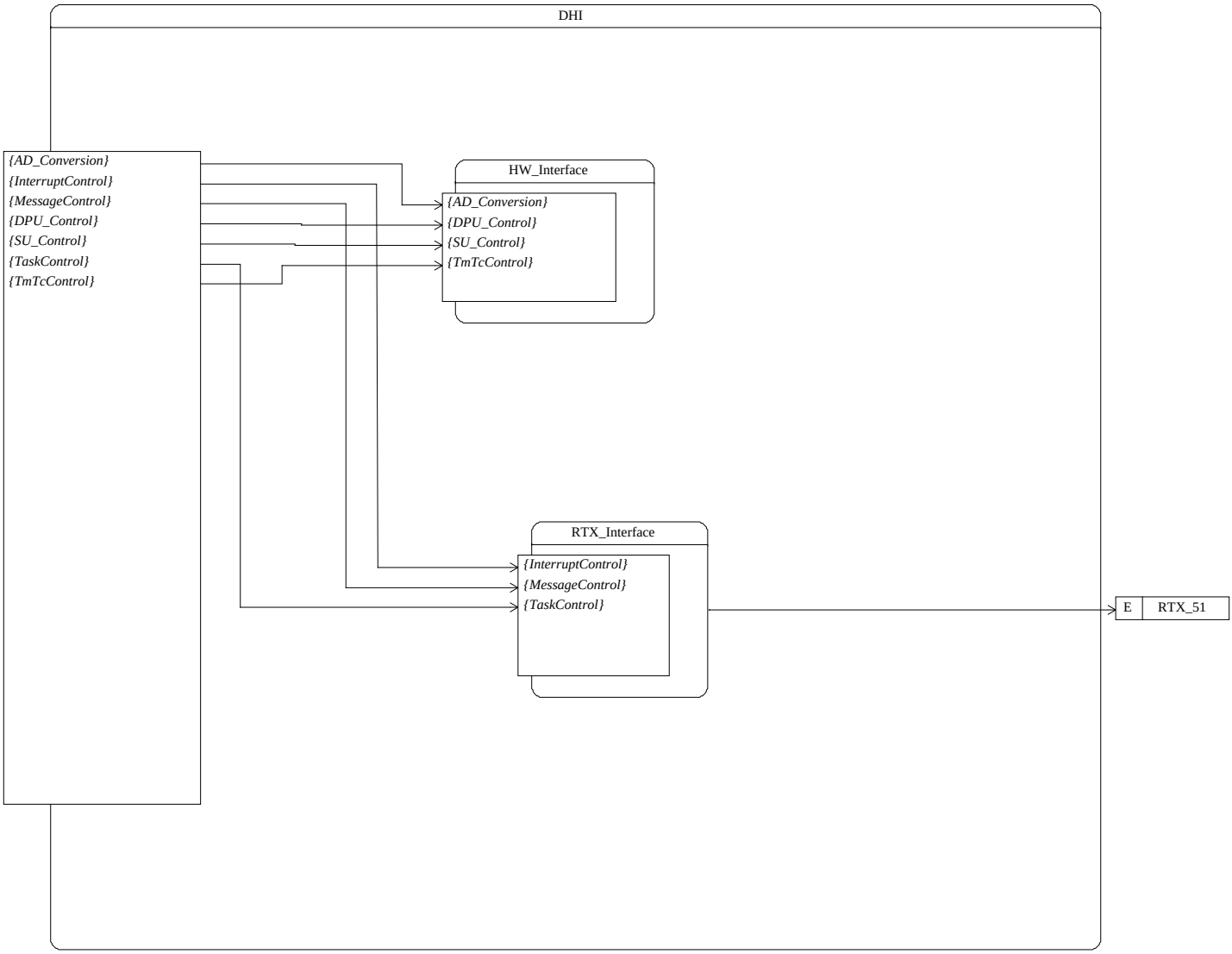
Operation Set Dictionary

TmTcControl
Provided by HW_Interface
SU_Control
Provided by HW_Interface
DPU_Control
Provided by HW_Interface
AD_Conversion
Provided by HW_Interface
TaskControl
Provided by RTX_Interface
MessageControl
Provided by RTX_Interface
InterruptControl
Provided by RTX_Interface

6.11.2.3 Grouping operations and objects

HW_Interface
AD_Conversion
DPU_Control
SU_Control
TmTcControl
RTX_Interface
InterruptControl
MessageControl
TaskControl

6.11.2.4 Graphical description



6.11.3 Formalization of the solution

6.11.3.1 Object Description Skeleton DHI

object DHI **is passive**
description
implementation_constraints
provided_interface
operation_sets
AD_Conversion ;
InterruptControl ;
MessageControl ;
DPU_Control ;
SU_Control ;
TaskControl ;
TmTcControl ;

```
    required_interface
        object RTX_51 ;
            none

    internals
        objects
            HW_Interface ;
            RTX_Interface ;
        operation_sets
            AD_Conversion implemented_by HW_Interface.AD_Conversion;
            InterruptControl implemented_by RTX_Interface.InterruptControl;
            MessageControl implemented_by RTX_Interface.MessageControl;
            DPU_Control implemented_by HW_Interface.DPU_Control;
            SU_Control implemented_by HW_Interface.SU_Control;
            TaskControl implemented_by RTX_Interface.TaskControl;
            TmTcControl implemented_by HW_Interface.TmTcControl;
end_object DHI
```

6.11.3.2 Generated spec code

6.11.3.3 Generated body code

6.12 Object : HW_Interface

6.12.1 Problem definition

6.12.2 Formalization of the solution

6.12.2.1 Object Description Skeleton HW_Interface

```
object HW_Interface is passive
    description
    implementation_constraints
    provided_interface
        operation_sets
            AD_Conversion ;
            DPU_Control ;
            SU_Control ;
            TmTcControl ;
    required_interface
        none

    internals
end_object HW_Interface
```

6.12.2.2 Generated spec code

6.12.2.3 Generated body code

6.13 Object : RTX_Interface

6.13.1 Problem definition

6.13.2 Formalization of the solution

6.13.2.1 Object Description Skeleton RTX_Interface

```
object RTX_Interface is passive
    description
    implementation_constraints
    provided_interface
        operation_sets
            InterruptControl ;
            MessageControl ;
            TaskControl ;
    required_interface
        object RTX_51 ;
            none

    internals
end_object RTX_Interface
```

6.13.2.2 Generated spec code

6.13.2.3 Generated body code

7 DAS Header Files

This chapter contains the header source files for the DEBIE Application Software module.

These listings are taken from the reference directory of the PVCS archive of DAS. The PVCS tool is used as a configuration management tool.

7.1 class.h

```
/*-----------------------------------------------------------------------------
/*                                                                           --
/*    Copyright (C) 1998 : Space Systems Finland Ltd.                        --
/*                                                                           --
/*-----------------------------------------------------------------------------
/*
/*    System Name:   DEBIE DPU SW
/*
/*      $Workfile:   class.h  $
/*        $Author:   paloheim  $
/*          $Date:   Fri Jun 04 14:47:38 1999  $
/*      $Revision:   1.6  $
/*           $Log:   /home/share/fileserver/projects/debie/design/code/DAS/PVCS/
class.h,p  $
 *
 *    Rev 1.6   Fri Jun 04 14:47:38 1999   paloheim
 * SMR_214: Code memory patch execution command check
 * SNCR_186:
 *
 *    Rev 1.5   Thu Apr 08 22:49:24 1999   holsti
 * SMR_184: Correct Init_SU_Settings and simplify Boot
 * SNCR_203:
 *
 *    Rev 1.4   Thu Apr 08 18:42:38 1999   paloheim
 * SMR_183: Initialised classification parameters after power-up
 * SNCR_201:
 *
 *    Rev 1.3   Fri Mar 26 11:26:26 1999   ville
 * SMR_170: Correct classification bits associated to Piezo channels
 * SNCR_177
 *
 *    Rev 1.2   Mon Jan 18 11:50:48 1999   paloheim
 * SMR_114: Problems due to multiple inclusion prevented
 * SNCR_118
 *
 *    Rev 1.1   Wed Dec 02 11:26:54 1998   ville
 * SMR_097: Classification index calculation ficed
 * SNCR_106
 *
 *    Rev 1.0   Thu Oct 22 16:31:18 1998   ville
 * Initial revision.
/*
/*- * ----------------------------------------------------------------------------*/

/* Part of : DAS */



#ifndef CLASS_H
#define CLASS_H

#include "tm_data.h"
```

```
#define MAX_AMPLITUDE_TERM 5
/* Maximum value for an amplitude term in the quality formula. */
/* Valid range: 1 - 255. */

#define DEFAULT_COEFF 5
/* Default  value for classification coefficient        */
/* adjustable with telecommands. Gives maximum allowed  */
/* (5) amplitude term with maximum amplitude with this  */
/* formula. If amplitudes are going to be smaller, the  */
/* amplitude can be amplified by setting greater value  */
/* to the quality coefficient. Minimum amplification is */
/* 1/5 and maximum 50.                                  */
/* Valid range 1 - 255. */

#define AMPLITUDE_DIVIDER ((DEFAULT_COEFF * 16.0) / MAX_AMPLITUDE_TERM)
/* Divider for an amplitude term in the quality formula.      */
/*  16 = maxumum value for the rough 2 based logarithm of the */
/*       signal amplitude in the quality formula.             */

#define PLASMA_1_PLUS_CLASS  0x80
#define PLASMA_1_MINUS_CLASS 0x40
#define PLASMA_2_PLUS_CLASS  0x08
#define PIEZO_1_CLASS        0x20
#define PIEZO_2_CLASS        0x10
/* Classification index mask values for signal amplitudes */
/* above the classification levels.                       */

#define PLASMA_1_PLUS_TO_PIEZO_CLASS  0x02
#define PLASMA_1_MINUS_TO_PIEZO_CLASS 0x01
#define PLASMA_1_PLUS_TO_MINUS_CLASS  0x04
/* Classification index mask values for delays inside the */
/* time windows.                                          */

extern void InitClassification(void);
/* Inits classification thresholds and coefficients. */

extern void ClassifyEvent(event_record_t EXTERNAL *new_event);
/* Classifies event and calculates the quality number. */

extern void Init_SU_Settings (SU_settings_t EXTERNAL *set);
/* Sets the default values for classification parameters. */

#endif
```

7.2 classtab.h

```
/*-----------------------------------------------------------------------------
/*                                                                           --
/*    Copyright (C) 1998 : Space Systems Finland Ltd.                        --
/*                                                                           --
/*-----------------------------------------------------------------------------
/*
/*    System Name:   DEBIE DPU SW
/*
/*      $Workfile:   classtab.h  $
/*        $Author:   paloheim  $
/*          $Date:   Mon Jan 18 11:53:20 1999  $
/*      $Revision:   1.1  $
/*           $Log:   /home/share/fileserver/projects/debie/design/code/DAS/PVCS/
classtab.h,p  $
 *
 *    Rev 1.1   Mon Jan 18 11:53:20 1999   paloheim
 * SMR_114: Problems due to multiple inclusion prevented
 * SNCR_118
 *
 *    Rev 1.0   Thu Oct 22 16:31:46 1998   ville
 * Initial revision.
/*
/*- * ---------------------------------------------------------------------------*/

/* Part of : DAS */

#ifndef CLASSTAB_H
#define CLASSTAB_H

extern unsigned char EXTERNAL event_class[256];

#endif
```

7.3 health.h

```
/*-----------------------------------------------------------------------------
/*                                                                           --
/*    Copyright (C) 1998 : Space Systems Finland Ltd.                        --
/*                                                                           --
/*-----------------------------------------------------------------------------
/*
/*    System Name:   DEBIE DPU SW
/*
/*      $Workfile:   health.h  $
/*        $Author:   ville  $
/*          $Date:   Tue Aug 24 09:43:10 1999  $
/*      $Revision:   1.10  $
/*           $Log:   /home/share/fileserver/projects/debie/design/code/DAS/PVCS/
health.h,p  $
 *
 *    Rev 1.10   Tue Aug 24 09:43:10 1999   ville
 * SMR_238: Corrected TEMP 2 high limit value
 * SNCR_262
 *
 *    Rev 1.9   Tue Aug 17 15:04:10 1999   paloheim
 * SMR_232: Corrected errors in voltage measurement and error clearing
 * SNCR_219:
 * SNCR_221:
 *
 *    Rev 1.8   Fri Aug 06 11:58:04 1999   ville
 * SMR_222: Time to be sent to telemetry frozen
 * SNCR_251
 *
 *    Rev 1.7   Tue Jun 01 12:37:20 1999   paloheim
 * SMR_213: Commented reentrancy, edited comments and updated SW
 * SNCR_198:
 * SNCR_171:
 * SNCR_186:
 * SNCR_238: no scp
 * SNCR_154: no scp
 *
 *    Rev 1.6   Thu May 13 13:20:18 1999   paloheim
 * SMR_202: DPU and SU self test sequences implemented
 * SNCR: none
 *
 *    Rev 1.5   Wed May 12 14:09:08 1999   holsti
 * SMR_201: Fast memory test and other reset updates.
 * SNCR_215:
 *
 *    Rev 1.4   Thu Feb 11 15:33:16 1999   paloheim
 * SMR_136: Use of error and mode status bits corrected
 * SNCR_134:
 * SNCR_136:
 * SNCR_137:
 *
 *    Rev 1.3   Mon Jan 25 14:25:32 1999   paloheim
 * SMR_128: CalculateChecksum edited
 * SNCR_115
```

```
 * SNCR_117
 *
 *    Rev 1.2   Mon Jan 18 11:51:08 1999   paloheim
 * SMR_114: Problems due to multiple inclusion prevented
 * SNCR_118
 *
 *    Rev 1.1   Tue Jan 05 16:09:20 1999   paloheim
 * SMR_110: Health monitoring updated
 *
 *    Rev 1.0   Fri Aug 14 11:03:04 1998   ville
 * Initial revision.
/*
/*- * ----------------------------------------------------------------------*/

/* Part of : DAS */

#ifndef HEALTH_H
#define HEALTH_H

#include "keyword.h"
#include "dpu_ctrl.h"
#include "su_ctrl.h"

#define CH_SELECTED 1
#define CH_NOT_SELECTED 0

#define RESULT_OK 1
#define CONVERSION_ACTIVE 0
#define HIT_OCCURRED 2

#define CONVERSION_STARTED 1

#define TEST_OK 1

/* Health monitoring round identification numbering. */

#define HEALTH_COUNT    9
/* Health Monitoring loop count. */

#define VOLTAGE_COUNT   17
/* Voltage Measurement loop count. */

#define TEMP_COUNT      5
/* Temperature measurement loop count. Its value must equal or greater than  */
/* NUM_SU, because its value defines the SU whos temperatures are to be      */
/* measured.                                                                 */

#define CHECK_COUNT     59
/* Checksum loop count. */

#define MAX_TEMP_1      0xFA
#define MAX_TEMP_2      0xF7
/* Maximum temperature (0xFA = 90 C and 0xF7 = 85C) for a Sensor Unit. */
```

```
#define CHECK_SIZE      547
/* Checksum is counted for code memory 547 bytes per check round. */

#define CODE_MEMORY_END        0x7FFF
/* The last code memory address to be checked in function */
/* 'CalculateChecksum'.                                   */
/* 'CODE_MEMORY_END'  should have a value smaller         */
/* than 2^16 - 1. Otherwise it will affect a 'for'        */
/* loop in 'CalculateChecksum' function in a way          */
/* that makes this loop infinite.                         */

#define MAX_CHECKSUM_COUNT 59
#define MIN_CHECKSUM_COUNT 0
/* Limiting values used in function 'CalculateChecksum'. */

typedef unsigned char channel_t;
typedef unsigned int  adc_unsigned_t;
typedef signed int adc_sign_t;

extern EXTERNAL unsigned char confirm_hit_result;
extern EXTERNAL dpu_time_t internal_time;

extern void SetSoftwareError(unsigned char error) compact reentrant;
extern void ClearSoftwareError(void);
extern void SetModeStatusError(unsigned char mode_status_error) compact reentrant;
extern void ClearModeStatusError(void);

extern void SetMode(DEBIE_mode_t mode) compact reentrant;
extern DEBIE_mode_t GetMode(void);
extern void Clear_SU_Error(void);
extern void Set_SU_Error(sensor_index_t SU_index, unsigned char SU_error);
extern void SetErrorStatus(unsigned char error_source);
extern void ClearErrorStatus(void);
extern void Clear_RTX_Errors(void);


extern void Boot (void);


#endif
```

7.4 kernobj.h

```
/*-----------------------------------------------------------------------------
/*                                                                           --
/*    Copyright (C) 1998 : Space Systems Finland Ltd.                        --
/*                                                                           --
/*-----------------------------------------------------------------------------
/*
/*    System Name:   DEBIE DPU SW
/*
/*      $Workfile:   kernobj.h  $
/*        $Author:   paloheim  $
/*          $Date:   Mon Jan 18 11:48:32 1999  $
/*      $Revision:   1.2  $
/*           $Log:   /home/share/fileserver/projects/debie/design/code/DAS/PVCS/
kernobj.h,p  $
 *
 *    Rev 1.2   Mon Jan 18 11:48:32 1999   paloheim
 * SMR_114: Problems due to multiple inclusion prevented
 * SNCR_118
 *
 *    Rev 1.1   Tue Aug 04 11:32:36 1998   ville
 * SMR_005: Overlapping definitions removed
 * SNCR_005
 *
 *    Rev 1.0   Tue Aug 04 09:56:32 1998   ville
 * Initial revision.
/*
/*- * ----------------------------------------------------------------------------*/

/* Definitions of kernel objects (eg. task and mailbox numbers) */

/* Part of : DAS */

/* Task numbers */

#ifndef KERNOBJ_H
#define KERNOBJ_H

#define HEALTH_MONITORING_TASK     0
#define TC_TM_INTERFACE_TASK       1
#define ACQUISITION_TASK           2

#define HIT_TRIGGER_ISR_TASK       3

/* Task priorities */

#define HEALTH_MONITORING_PR       0
#define TC_TM_INTERFACE_PR         1
#define ACQUISITION_PR             2

#define HIT_TRIGGER_PR             3

/* Mailbox numbers */
```

```
#define TCTM_MAILBOX              0
#define ACQUISITION_MAILBOX       1

/* ISR source numbers */

#define TC_ISR_SOURCE             0
#define TM_ISR_SOURCE             2
#define HIT_TRIGGER_ISR_SOURCE    5

#endif
```

7.5 measure.h

```
/*-----------------------------------------------------------------------------
/*                                                                           --
/*    Copyright (C) 1998 : Space Systems Finland Ltd.                        --
/*                                                                           --
/*-----------------------------------------------------------------------------
/*
/*    System Name:   DEBIE DPU SW
/*
/*      $Workfile:   measure.h  $
/*        $Author:   paloheim  $
/*          $Date:   Mon Aug 16 17:29:40 1999  $
/*      $Revision:   1.14  $
/*           $Log:   /home/share/fileserver/projects/debie/design/code/DAS/PVCS/
measure.h,p  $
 *
 *    Rev 1.14   Mon Aug 16 17:29:40 1999   paloheim
 * SMR_227: New SU Self Test sequence
 * SNCR_249:
 * SNCR_256:
 *
 *    Rev 1.13   Tue May 18 23:41:54 1999   holsti
 * SMR_206: SetSensorUnitOff defines its data.
 * SNCR_233:
 *
 *    Rev 1.12   Thu May 13 13:19:50 1999   paloheim
 * SMR_202: DPU and SU self test sequences implemented
 *  SNCR: none
 *
 *    Rev 1.11   Thu Mar 25 09:31:20 1999   paloheim
 * SMR_169: Corrected Peak Detector and Delay counter resets
 * SNCR_174:
 *
 *    Rev 1.10   Thu Feb 18 15:15:22 1999   paloheim
 * SMR_138: Edited reentrant function declarations and line lengths
 * SNCR_139:
 *
 *    Rev 1.9   Mon Feb 15 14:09:56 1999   paloheim
 * SMR_137: SU Self Test Implemented
 * SNCR: none
 *
 *    Rev 1.8   Mon Feb 08 15:43:06 1999   paloheim
 * SMR_132: Hits during one Monitoring period limited
 * SNCR_109:
 *
 *    Rev 1.7   Mon Jan 18 11:54:10 1999   paloheim
 * SMR_114: Problems due to multiple inclusion prevented
 * SNCR_118
 *
 *    Rev 1.6   Mon Jan 11 16:45:52 1999   paloheim
 * SMR_111: Update SU state transitions
 * SNCR_107
 *
 *    Rev 1.5   Tue Jan 05 16:11:10 1999   paloheim
```

```
 * SMR_110: Health monitoring updated
 *
 *    Rev 1.4   Tue Aug 25 15:18:00 1998   ville
 * SMR_040: Task function prototypes replaced etc.
 * SNCR_041
 *
 *    Rev 1.3   Mon Aug 24 10:26:36 1998   ville
 * SMR_036: Pointer to new task passed to CreateTask
 * SNCR_037
 *
 *    Rev 1.2   Thu Aug 20 14:04:22 1998   ville
 * SMR_032: 'reentrant keyword replaced
 * SNCR_033
 *
 *    Rev 1.1   Mon Aug 17 12:43:18 1998   ville
 * SMR_025: Macro renamed
 * SNCR_027
 *
 *    Rev 1.0   Mon Aug 17 10:30:36 1998   ville
 * Initial revision.
/*
/*- * ----------------------------------------------------------------------*/

/* Part of : DAS */
#ifndef MEASURE_H
#define MEASURE_H


#include "kernobj.h"
#include "su_ctrl.h"

/*Maximum number of conversion start tries allowed in the HitTriggerTask*/
#define ADC_MAX_TRIES 25

#define HIT_BUDGET_DEFAULT 20
/* Default limit for the events handled during one Health Monitoring */
/* period. Valid values 1 .. 255.                                    */

#define PEAK_RESET_MIN_DELAY 1
/* Peak detector reset min delay: 1 * 10ms. */

#define COUNTER_RESET_MIN_DELAY 1
/* Delay counter reset min delay: 1 * 10 ms.   */
/* NOTE that specifications would allow delay  */
/* of 1ms, but minimum delay that is possible  */
/* to be generated with RTX is one tick = 10ms */

#define SELF_TEST_DELAY 4
/* This delay equals the length of 4 system cycles. */

/*Sensor Unit numbers*/
#define SU1 1
#define SU2 2
#define SU3 3
```

```
#define SU4 4

/*type definitions*/
typedef enum {
   off_e,               /* SU off state - power is Off.                  */
   start_switching_e,   /* Transition to On state is starting.           */
   switching_e,         /* Transition to On state is started.            */
   on_e,                /* SU on state - power is On.                    */
   self_test_mon_e,     /* Selt Test, Voltage and Temperature monitoring */
   self_test_e,         /* Selt Test, test pulse setup.                  */
   self_test_trigger_e, /* Self test, test pulse handling                */
   acquisition_e        /* Power is On and Hit Events are accepted.      */

} SU_state_t;
   /* From these only off_e, on_e and self_test_e are actual SU states   */
   /* defined in the User Requirements. Those and 'acquisition_e' are    */
   /* the main states between which the SU state transitions are made.   */

typedef struct {
   sensor_number_t SU_number;        /* Sensor Unit number                  */
   SU_state_t SU_state;              /* Sensor unit states can be either On  */
                                     /* or Off.                             */
   SU_state_t expected_source_state; /* Excpected source state of the SU    */
                                     /* state transition.                   */
   unsigned char execution_result;   /* This variable is used to indicate   */
                                     /* execution results.                  */

} sensor_unit_t;

extern SU_state_t EXTERNAL SU_state[4];

extern sensor_number_t EXTERNAL self_test_SU_number;

typedef enum {high_e, low_e} SU_test_level_t;

extern unsigned char EXTERNAL hit_budget;
extern unsigned char EXTERNAL hit_budget_left;


/*function prototypes*/
extern void Switch_SU_State(sensor_unit_t EXTERNAL *SU_setting)
               COMPACT_DATA REENTRANT_FUNC;

extern void Start_SU_SwitchingOn(
               sensor_index_t SU,
               unsigned char EXTERNAL *exec_result)
               COMPACT_DATA REENTRANT_FUNC;

extern void SetSensorUnitOff(
               sensor_index_t SU,
               unsigned char EXTERNAL *exec_result)
               COMPACT_DATA REENTRANT_FUNC;

extern SU_state_t ReadSensorUnit(unsigned char SU_number)
```

```
                    COMPACT_DATA REENTRANT_FUNC;

extern void Update_SU_State(sensor_index_t SU_index)
               COMPACT_DATA REENTRANT_FUNC;

/*pointers to tasks*/

extern void (PROGRAM *hit_task)(void);
extern void (PROGRAM *acq_task)(void);

#endif
```

7.6 tc_hand.h

```
/*-----------------------------------------------------------------------------
/*                                                                           --
/*    Copyright (C) 1998 : Space Systems Finland Ltd.                        --
/*                                                                           --
/*-----------------------------------------------------------------------------
/*
/*    System Name:   DEBIE DPU SW
/*
/*      $Workfile:   tc_hand.h  $
/*        $Author:   ville  $
/*          $Date:   Mon Mar 22 11:39:54 1999  $
/*      $Revision:   1.11  $
/*           $Log:   /home/share/fileserver/projects/debie/design/code/DAS/PVCS/
tc_hand.h,p  $
 *
 *    Rev 1.11   Mon Mar 22 11:39:54 1999   ville
 * SMR_165: Set Quality Coefficient TCs implemented
 *
 *    Rev 1.9   Mon Jan 18 11:50:02 1999   paloheim
 * SMR_114: Problems due to multiple inclusion prevented
 * SNCR_118
 *
 *    Rev 1.8   Mon Jan 11 16:51:56 1999   paloheim
 * SMR_111: Update SU state transitions
 * SNCR_107
 *
 *    Rev 1.7   Mon Jan 04 10:26:14 1999   paloheim
 * SMR_107: New time setting telecommands implemented
 * SNCR_100
 *
 *    Rev 1.6   Wed Dec 30 14:25:44 1998   ville
 * SMR_103: Write Memory telecommands implemented.
 *
 *    Rev 1.5   Wed Dec 02 10:03:36 1998   ville
 * SMR_096: TC timeout, Set Class Level TC, etc.
 * SNCR_101
 *
 *    Rev 1.4   Thu Nov 12 13:05:18 1998   ville
 * SMR_088: Send Status Register TC prevented from aborting Science TM
 * SNCR_093
 *
 *    Rev 1.3   Tue Aug 25 15:16:44 1998   ville
 * SMR_040: Task function prototypes replaced etc.
 * SNCR_041
 *
 *    Rev 1.2   Mon Aug 24 10:27:42 1998   ville
 * SMR_036: Pointer to new task passed to CreateTask
 * SNCR_037
 *
 *    Rev 1.1   Wed Aug 12 13:20:16 1998   ville
 * SMR_020: Added newline at the end of file
 * SNCR_025
 *
```

```
 *    Rev 1.0   Fri Jul 31 10:34:28 1998   ville
 * Initial revision.
/*
/*- * ----------------------------------------------------------------------*/

/* Part of : DAS */

#ifndef TC_HAND_H
#define TC_HAND_H

/* Valid telecommand address codes:                */
/* NOTE that all codes are not yet defined, because */
/* all telecommands are not implemented in the      */
/* Prototype SW.                                    */

#define UNUSED_TC_ADDRESS                     0x00

#define START_ACQUISITION                     0x01
#define STOP_ACQUISITION                      0x02

#define ERROR_STATUS_CLEAR                    0x03

#define SEND_STATUS_REGISTER                  0x05
#define SEND_SCIENCE_DATA_FILE                0x06

#define SET_TIME_BYTE_0                       0x0C
#define SET_TIME_BYTE_1                       0x0D
#define SET_TIME_BYTE_2                       0x0E
#define SET_TIME_BYTE_3                       0x0F

#define SOFT_RESET                            0x09

#define CLEAR_WATCHDOG_FAILURES               0x0A
#define CLEAR_CHECKSUM_FAILURES               0x0B

#define WRITE_CODE_MEMORY_MSB                 0x10
#define WRITE_CODE_MEMORY_LSB                 0x6F
#define WRITE_DATA_MEMORY_MSB                 0x15
#define WRITE_DATA_MEMORY_LSB                 0x6A
#define READ_DATA_MEMORY_MSB                  0x1F
#define READ_DATA_MEMORY_LSB                  0x60

#define SWITCH_SU_1                           0x20
#define SWITCH_SU_2                           0x30
#define SWITCH_SU_3                           0x40
#define SWITCH_SU_4                           0x50

#define SET_SU_1_PLASMA_1P_THRESHOLD          0x21
#define SET_SU_2_PLASMA_1P_THRESHOLD          0x31
#define SET_SU_3_PLASMA_1P_THRESHOLD          0x41
#define SET_SU_4_PLASMA_1P_THRESHOLD          0x51

#define SET_SU_1_PLASMA_1M_THRESHOLD          0x22
#define SET_SU_2_PLASMA_1M_THRESHOLD          0x32
```

```
#define SET_SU_3_PLASMA_1M_THRESHOLD          0x42
#define SET_SU_4_PLASMA_1M_THRESHOLD          0x52

#define SET_SU_1_PIEZO_THRESHOLD              0x23
#define SET_SU_2_PIEZO_THRESHOLD              0x33
#define SET_SU_3_PIEZO_THRESHOLD              0x43
#define SET_SU_4_PIEZO_THRESHOLD              0x53

#define SET_SU_1_PLASMA_1P_CLASS_LEVEL        0x24
#define SET_SU_2_PLASMA_1P_CLASS_LEVEL        0x34
#define SET_SU_3_PLASMA_1P_CLASS_LEVEL        0x44
#define SET_SU_4_PLASMA_1P_CLASS_LEVEL        0x54

#define SET_SU_1_PLASMA_1M_CLASS_LEVEL        0x25
#define SET_SU_2_PLASMA_1M_CLASS_LEVEL        0x35
#define SET_SU_3_PLASMA_1M_CLASS_LEVEL        0x45
#define SET_SU_4_PLASMA_1M_CLASS_LEVEL        0x55

#define SET_SU_1_PLASMA_2P_CLASS_LEVEL        0x28
#define SET_SU_2_PLASMA_2P_CLASS_LEVEL        0x38
#define SET_SU_3_PLASMA_2P_CLASS_LEVEL        0x48
#define SET_SU_4_PLASMA_2P_CLASS_LEVEL        0x58

#define SET_SU_1_PIEZO_1_CLASS_LEVEL          0x26
#define SET_SU_2_PIEZO_1_CLASS_LEVEL          0x36
#define SET_SU_3_PIEZO_1_CLASS_LEVEL          0x46
#define SET_SU_4_PIEZO_1_CLASS_LEVEL          0x56

#define SET_SU_1_PIEZO_2_CLASS_LEVEL          0x27
#define SET_SU_2_PIEZO_2_CLASS_LEVEL          0x37
#define SET_SU_3_PIEZO_2_CLASS_LEVEL          0x47
#define SET_SU_4_PIEZO_2_CLASS_LEVEL          0x57

#define SET_SU_1_PLASMA_1E_1I_MAX_TIME        0x29
#define SET_SU_2_PLASMA_1E_1I_MAX_TIME        0x39
#define SET_SU_3_PLASMA_1E_1I_MAX_TIME        0x49
#define SET_SU_4_PLASMA_1E_1I_MAX_TIME        0x59

#define SET_SU_1_PLASMA_1E_PZT_MIN_TIME       0x2A
#define SET_SU_2_PLASMA_1E_PZT_MIN_TIME       0x3A
#define SET_SU_3_PLASMA_1E_PZT_MIN_TIME       0x4A
#define SET_SU_4_PLASMA_1E_PZT_MIN_TIME       0x5A

#define SET_SU_1_PLASMA_1E_PZT_MAX_TIME       0x2B
#define SET_SU_2_PLASMA_1E_PZT_MAX_TIME       0x3B
#define SET_SU_3_PLASMA_1E_PZT_MAX_TIME       0x4B
#define SET_SU_4_PLASMA_1E_PZT_MAX_TIME       0x5B

#define SET_SU_1_PLASMA_1I_PZT_MIN_TIME       0x2C
#define SET_SU_2_PLASMA_1I_PZT_MIN_TIME       0x3C
#define SET_SU_3_PLASMA_1I_PZT_MIN_TIME       0x4C
#define SET_SU_4_PLASMA_1I_PZT_MIN_TIME       0x5C

#define SET_SU_1_PLASMA_1I_PZT_MAX_TIME       0x2D
```

```
#define SET_SU_2_PLASMA_1I_PZT_MAX_TIME        0x3D
#define SET_SU_3_PLASMA_1I_PZT_MAX_TIME        0x4D
#define SET_SU_4_PLASMA_1I_PZT_MAX_TIME        0x5D

#define SET_COEFFICIENT_1                      0x70
#define SET_COEFFICIENT_2                      0x71
#define SET_COEFFICIENT_3                      0x72
#define SET_COEFFICIENT_4                      0x73
#define SET_COEFFICIENT_5                      0x74

/* State of Telecommand Execution task */

typedef enum {
   TC_handling_e,
   read_memory_e,
   memory_dump_e,
   write_memory_e,
   memory_patch_e,
   register_TM_e,
   SC_TM_e
   } TC_state_t;

extern TC_state_t TC_state;

/*pointer to a task*/
extern void (PROGRAM *TC_task)(void);

/* Functions prototype */
extern void Set_TC_Error(void);

#endif
```

7.7 telem.h

```
/*-----------------------------------------------------------------------------
/*                                                                           --
/*    Copyright (C) 1998 : Space Systems Finland Ltd.                        --
/*                                                                           --
/*-----------------------------------------------------------------------------
/*
/*    System Name:   DEBIE DPU SW
/*
/*      $Workfile:   telem.h  $
/*        $Author:   paloheim  $
/*          $Date:   Wed May 12 13:08:26 1999  $
/*      $Revision:   1.11  $
/*           $Log:   /home/share/fileserver/projects/debie/design/code/DAS/PVCS/
telem.h,p  $
 *
 *    Rev 1.11   Wed May 12 13:08:26 1999   paloheim
 * SMR_201: Fast memory test and other reset updates
 * SNCR_181:
 * SNCR_214:
 * SNCR_215:
 *
 *    Rev 1.10   Mon Jan 18 11:52:42 1999   paloheim
 * SMR_114: Problems due to multiple inclusion prevented
 * SNCR_118
 *
 *    Rev 1.9   Thu Dec 31 10:15:28 1998   ville
 * SMR_104: Read Data memory implemented.
 *
 *    Rev 1.8   Wed Dec 30 14:23:44 1998   ville
 * SMR_103: Write Memory telecommands implented.
 *
 *    Rev 1.7   Tue Dec 29 13:39:42 1998   ville
 * SMR_102: Science Data protected.
 *
 *    Rev 1.6   Tue Nov 17 12:51:08 1998   ville
 * SMR_095: New SW version
 *
 *    Rev 1.5   Mon Nov 16 10:16:40 1998   ville
 * SMR_092: New SW version
 *
 *    Rev 1.4   Fri Oct 09 14:39:58 1998   ville
 * SMR_085: New SW version
 *
 *    Rev 1.3   Mon Oct 05 13:59:16 1998   ville
 * SMR_079: New software version
 *
 *    Rev 1.2   Wed Sep 30 13:40:54 1998   ville
 * SMR_071: New Software Version
 *
 *    Rev 1.1   Tue Sep 08 09:17:42 1998   ville
 * SMR_057: SW version defined
 * SNCR_059
 *
```

```
 *    Rev 1.0   Tue Aug 04 15:21:42 1998   ville
 * Initial revision.
/*
/*- * ----------------------------------------------------------------------*/

/* Part of : DAS */

#ifndef TELEM_H
#define TELEM_H

#ifndef EXTERNAL
   #include "keyword.h"
#endif

/* Special value for TC/TM mail to be used only     */
/* telemetry is ready                               */

#define TM_READY                              0xFFFF

extern unsigned char EXTERNAL *telemetry_pointer;
extern unsigned char EXTERNAL *telemetry_end_pointer;
extern unsigned int  EXTERNAL free_slot_index;

extern unsigned char EXTERNAL read_memory_checksum;
/* Checksum to be sent at the end of Read Memory sequence. */


extern event_record_t EXTERNAL *GetFreeRecord(void);
/* Returns pointer to next free event record from the */
/* event record queue, or pointer to the last record  */
/* od the queue, if the queue is full.                */

extern void ResetEventQueueLength(void);

#endif
```

7.8 tm_data.h

```
/*-----------------------------------------------------------------------------
/*                                                                           --
/*    Copyright (C) 1998 : Space Systems Finland Ltd.                        --
/*                                                                           --
/*-----------------------------------------------------------------------------
/*
/*    System Name:   DEBIE DPU SW
/*
/*      $Workfile:   tm_data.h  $
/*        $Author:   paloheim  $
/*          $Date:   Mon May 31 10:10:12 1999  $
/*      $Revision:   1.22  $
/*           $Log:   /home/share/fileserver/projects/debie/design/code/DAS/PVCS/
tm_data.h,p  $
 *
 *    Rev 1.22   Mon May 31 10:10:12 1999   paloheim
 * SMR_212: Adjusted implementation of Send Status Register
 * SNCR_213:
 *
 *    Rev 1.21   Thu May 13 16:07:58 1999   paloheim
 * SMR_203: RTX-error indication implemented
 * SNCR_227:
 * SNCR_225:
 * SNCR_186:
 *
 *    Rev 1.20   Thu May 13 13:21:34 1999   paloheim
 * SMR_202: DPU and SU self test sequences implemented
 * SNCR: none
 *
 *    Rev 1.19   Thu Apr 08 13:35:22 1999   paloheim
 * SMR_182: Implemented new error indicationg bits
 * SNCR_167:
 * SNCR_196: [no scp]
 *
 *    Rev 1.18   Tue Apr 06 11:14:32 1999   paloheim
 * SMR_174: Removed variable science_tm_time
 * SNCR_189:
 *
 *    Rev 1.17   Tue Mar 16 15:07:50 1999   paloheim
 * SMR_152: Updated MAX_EVENTS
 *  SNCR_156:
 *
 *    Rev 1.16   Tue Mar 09 12:39:20 1999   paloheim
 * SMR_146: TM alignment constraints are taken into account.
 * SNCR_149:
 * SNCR_150:
 *
 *    Rev 1.15   Thu Feb 11 15:26:12 1999   paloheim
 * SMR_136: Use of error and mode status bits corrected
 * SNCR_134:
 * SNCR_136:
 * SNCR_137:
 *
```

```
*    Rev 1.14   Tue Feb 09 17:10:46 1999   paloheim
* SMR_134: DPU Self Test implemented
* SNCR: none
*
*    Rev 1.13   Mon Feb 08 15:40:58 1999   paloheim
* SMR_132: Hits during one Monitoring period limited
* SNCR_109:
*
*    Rev 1.12   Thu Jan 28 14:08:10 1999   paloheim
* SMR_129: Length indicator added to Science telemetry data
* SNCR: none
*
*    Rev 1.11   Mon Jan 18 11:55:02 1999   paloheim
* SMR_114: Problems due to multiple inclusion prevented
* SNCR_118
*
*    Rev 1.10   Tue Jan 05 16:11:28 1999   paloheim
* SMR_110: Health monitoring updated
*
*    Rev 1.9   Sun Jan 03 13:24:28 1999   ville
* SMR_105: Telemetry formats updated.
* SNCR_091
*
*    Rev 1.8   Tue Dec 29 13:39:10 1998   ville
* SMR_102: Science Data protected.
*
*    Rev 1.7   Fri Oct 02 14:54:20 1998   ville
* SMR_076: Only relevant bits from ModeStatus checked
* SNCR_082
*
*    Rev 1.6   Thu Aug 20 12:57:46 1998   ville
* SMR_031: Keil keyword replaced
* SNCR_032
*
*    Rev 1.5   Mon Aug 17 14:13:04 1998   ville
* SMR_027: Variable declarations corrected
* SNCR_28
*
*    Rev 1.4   Wed Aug 12 10:42:08 1998   ville
* SMR_018: Replaced 'unsigned int' with 'unsigned shor int'
* SNCR_023
*
*    Rev 1.3   Thu Aug 06 16:23:56 1998   ville
* SMR_013: Macros redefined
* SNCR_014
*
*    Rev 1.2   Fri Jul 31 09:51:34 1998   ville
* SMR_001: DEBIE mode constant macros defined
* SNCR_002
*
*    Rev 1.1   Tue Jul 28 17:07:50 1998   ville
* Type definitions modified: typedef used for structs
*
*    Rev 1.0   Thu Jul 02 10:54:24 1998   ville
```

```
 * Initial revision.
/*
/*- * ---------------------------------------------------------------------------*/

/* Part of : DAS */

#ifndef TM_DATA_H
#define TM_DATA_H

#ifndef EXTERNAL
   #include "keyword.h"
#endif

#include "dpu_ctrl.h"

#define MAX_EVENTS   1261
/* Size of one event is 26 bytes and MAX_EVENTS should be */
/* 32 768 bytes/ 26 bytes = 1261 (rounded up). This       */
/* ensures that at least 32 kB of memory is used.         */


#define NUM_SU       4
#define NUM_CH 5
/* Number of measurement channels. */
#define NUM_CLASSES  10
#define NUM_TEMP     2
#define NUM_NOT_USED (4 + 0x70 - 0x6A)

#define DPU_SELF_TEST 0
#define STAND_BY      1
#define ACQUISITION   2

#define MODE_BITS_MASK 3

/* Definitions related to error indicating bits in mode status register: */
#define SUPPLY_ERROR                0x80
#define DATA_MEMORY_ERROR           0x40
#define PROGRAM_MEMORY_ERROR        0x20
#define ADC_ERROR                   0x04

/* Definitions related to error indicating bits in SU status register: */
#define HV_SUPPLY_ERROR             0x80
#define LV_SUPPLY_ERROR             0x40
#define TEMPERATURE_ERROR           0x20
#define SELF_TEST_ERROR             0x10
#define HV_LIMIT_ERROR              0x08
#define LV_LIMIT_ERROR              0x04
#define SUPPLY_VOLTAGE_MASK         0x03

/* Used when error indiacting bits are cleared. */



/* Definitions related to error indicating bits in error status register: */
```

```
#define CHECKSUM_ERROR              0x08
#define WATCHDOG_ERROR              0x04
#define OVERALL_SU_ERROR            0xF0
/* Used to indicate error in all of the SUs. */

#define ERROR_STATUS_OFFSET         0x10
/* Used when SU error indicating bit is selected. */


/* Definitions related to error indicating bits in software error register: */
#define MEASUREMENT_ERROR           0x01
#define OS_START_SYSTEM_ERROR       0x02
#define OS_WAIT_ERROR               0x04
#define OS_SET_SLICE_ERROR          0x08


#define NUM_QCOEFF 5
/* Number of Quality Coefficients. */

/* Sensor Unit low power and TC settings : */

typedef struct {
   unsigned char plus_5_voltage;                    /* byte  1 */
   unsigned char minus_5_voltage;                   /* byte  2 */
   unsigned char plasma_1_plus_threshold;           /* byte  3 */
   unsigned char plasma_1_minus_threshold;          /* byte  4 */
   unsigned char piezo_threshold;                   /* byte  5 */
   unsigned char plasma_1_plus_classification;      /* byte  6 */
   unsigned char plasma_1_minus_classification;     /* byte  7 */
   unsigned char piezo_1_classification;            /* byte  8 */
   unsigned char piezo_2_classification;            /* byte  9 */
   unsigned char plasma_2_plus_classification;      /* byte 10 */
   unsigned char plasma_1_plus_to_minus_max_time;   /* byte 11 */
   unsigned char plasma_1_plus_to_piezo_min_time;   /* byte 12 */
   unsigned char plasma_1_plus_to_piezo_max_time;   /* byte 13 */
   unsigned char plasma_1_minus_to_piezo_min_time;  /* byte 14 */
   unsigned char plasma_1_minus_to_piezo_max_time;  /* byte 15 */
} SU_settings_t;

/* TM data registers : */

typedef struct {
   unsigned char        error_status;                     /* reg   0       */
   unsigned char        mode_status;                      /* reg   1       */
   unsigned short int   TC_word;                          /* reg   2 -   3 */
   unsigned long int    TC_time_tag;                      /* reg   4 -   7 */
   unsigned char        watchdog_failures;                /* reg   8       */
   unsigned char        checksum_failures;                /* reg   9       */
   unsigned char        SW_version;                       /* reg  10       */
   unsigned char        isr_send_message_error;           /* reg  11       */
   unsigned char        SU_status[NUM_SU];                /* reg  12 -  15 */
   unsigned char        SU_temperature[NUM_SU][NUM_TEMP]; /* reg  16 -  23 */
   unsigned char        DPU_plus_5_digital;               /* reg  24       */
   unsigned char        os_send_message_error;            /* reg  25       */
```

```
   unsigned char        os_create_task_error;             /* reg  26       */
   unsigned char        SU_plus_50;                       /* reg  27       */
   unsigned char        SU_minus_50;                      /* reg  28       */
   unsigned char        os_disable_isr_error;             /* reg  29       */
   unsigned char        not_used_1;                       /* reg  30       */
   SU_settings_t        sensor_unit_1;                    /* reg  31 -  45 */
   unsigned char        os_wait_error;                    /* reg  46       */
   SU_settings_t        sensor_unit_2;                    /* reg  47 -  61 */
   unsigned char        os_attach_interrupt_error;        /* reg  62       */
   SU_settings_t        sensor_unit_3;                    /* reg  63 -  77 */
   unsigned char        os_enable_isr_error;              /* reg  78       */
   SU_settings_t        sensor_unit_4;                    /* reg  79 -  93 */
   unsigned short int   failed_code_address;              /* reg  94 -  95 */
   unsigned short int   failed_data_address;              /* reg  96 -  97 */
   unsigned short int   SU_hits[NUM_SU];                  /* reg  98 - 105 */
   tm_dpu_time_t        time;                             /* reg 106 - 109 */
   unsigned char        software_error;                   /* reg 110       */
   unsigned char        hit_budget_exceedings;            /* reg 111       */
   unsigned char        coefficient[NUM_QCOEFF];          /* reg 112 - 121 */
   unsigned char        not_used;                         /* reg 122       */

   /* The last register of telemetry data should be 'not_used'.   */
   /* This is necessary for correct operation of telemetry        */
   /* retrieving TCs i.e. number of bytes should be even.         */

}  telemetry_data_t;

extern EXTERNAL telemetry_data_t telemetry_data;

/* Hit trigger event record : */

typedef struct {
   unsigned char      quality_number;    /* byte  0      */
   unsigned char      classification;    /* byte  1      */
   unsigned char      SU_number;         /* byte  2      */
   tm_dpu_time_t      hit_time;          /* byte  3 -  6 */
   unsigned char      SU_temperature_1;  /* byte  7      */
   unsigned char      SU_temperature_2;  /* byte  8      */
   tm_ushort_t        plasma_1_plus;     /* byte  9 - 10 */
   tm_ushort_t        plasma_1_minus;    /* byte 11 - 12 */
   tm_ushort_t        piezo_1;           /* byte 13 - 14 */
   tm_ushort_t        piezo_2;           /* byte 15 - 16 */
   tm_ushort_t        plasma_2_plus;     /* byte 17 - 18 */
   unsigned char      rise_time;         /* byte 19      */
   signed   char      delay_1;           /* byte 20      */
   tm_ushort_t        delay_2;           /* byte 21 - 22 */
   tm_ushort_t        delay_3;           /* byte 23 - 24 */
   unsigned char      checksum;          /* byte 25      */
} event_record_t;

/* Science Data File : */

typedef struct {
   unsigned short int length;
```

```
   unsigned char  event_counter[NUM_SU][NUM_CLASSES];
   unsigned char  not_used;
   unsigned char  counter_checksum;
   event_record_t event[MAX_EVENTS];
}  science_data_file_t;

extern EXTERNAL science_data_file_t science_data;

extern unsigned short int EXTERNAL max_events;
/* This variable is used to speed up certain    */
/* Functional Test by adding the possibility    */
/* to restrict the amount of events.            */
/* It is initialised to value MAX_EVENTS at     */
/* Boot.                                        */

extern void RecordEvent(void);
/* This function increments proper event counter and stores   */
/* the new event record to the science data memory, if there  */
/* is free place or events with lower or equal quality number */


extern void ClearEvents(void);
/* Cleares the event counters and the quality numbers of    */
/* the event records in the science data memory              */

#endif
```

7.9 version.h

```
/*-----------------------------------------------------------------------------
/*                                                                           --
/*    Copyright (C) 1998 : Space Systems Finland Ltd.                        --
/*                                                                           --
/*-----------------------------------------------------------------------------
/*
/*    System Name:   DEBIE DPU SW
/*
/*      $Workfile:   version.h  $
/*        $Author:   paloheim  $
/*          $Date:   Fri Aug 27 14:03:00 1999  $
/*      $Revision:   1.25  $
/*           $Log:   /home/share/fileserver/projects/debie/design/code/DAS/PVCS/
version.h,p  $
 *
 *    Rev 1.25   Fri Aug 27 14:03:00 1999   paloheim
 * SMR_243: Updated version number
 * SNCR: none
 *
 *    Rev 1.24   Fri Aug 27 13:44:36 1999   paloheim
 * SMR_242: Corrected +/-50V error limit handling in SU self test
 * SNCR_265:
 *
 *    Rev 1.23   Thu Aug 26 09:52:38 1999   paloheim
 * SMR_240:  Updated code_checksum value
 * SNCR: none
 *
 *    Rev 1.22   Thu Aug 19 11:20:50 1999   ville
 * SMR_236: Updated code_checksum value
 *
 *    Rev 1.21   Tue Aug 17 13:27:22 1999   paloheim
 * SMR_231: Updated code_checksum value
 * SNCR: none
 *
 *    Rev 1.20   Mon Aug 16 18:13:00 1999   paloheim
 * SMR_229: Updated code_checksum value
 * SNCR: none
 *
 *    Rev 1.19   Mon Aug 16 12:52:52 1999   paloheim
 * SMR_225: Updated code_checksum value
 * SNCR: none
 *
 *    Rev 1.18   Tue Aug 10 16:52:58 1999   paloheim
 * SMR_224 : Updated CODE_CHECKSUM value
 * SNCR: none
 *
 *    Rev 1.17   Sun Jul 25 15:46:18 1999   holsti
 * SMR_221: Reset-class protected from Data RAM test.
 * SNCR_242:
 *
 *    Rev 1.16   Fri Jul 02 12:37:26 1999   paloheim
 *  SMR_218: Corrected errors in health.c
 *  SNCR_239:
```

```
 *  SNCR_240:
 *
 *    Rev 1.15   Thu Jul 01 15:36:00 1999   paloheim
 * SMR_217: Corrected Error in SU Self Testing
 * SNCR_241
 *
 *    Rev 1.14   Wed Jun 16 14:24:52 1999   paloheim
 * SMR_216: Edited HighVoltageCurrent
 *  SNCR_229:
 *
 *    Rev 1.13   Thu Jun 10 14:27:44 1999   paloheim
 * SMR_215: Version update
 * SNCR: none
 *
 *    Rev 1.12   Fri May 21 00:14:30 1999   holsti
 * SMR_208: WaitMail returns execution_result.
 *
 *    Rev 1.11   Tue May 18 23:43:00 1999   holsti
 * SMR_206: SetSensorUnitOff defines its data.
 *
 *    Rev 1.10   Mon May 17 23:05:18 1999   holsti
 * SMR_205: Corrected RTX-51 usage mistages.
 *
 *    Rev 1.9   Sun May 16 09:21:20 1999   holsti
 * SMR_204: RTX interface portable and no calls from interrupts.
 *
 *    Rev 1.8   Thu May 06 17:19:36 1999   holsti
 * SMR_200: Enable Hit Trigger before Counter Reset.
 * SNCR_228:
 *
 *    Rev 1.7   Tue May 04 14:41:26 1999   holsti
 * SMR_199: Fast memory test routines.
 *
 *    Rev 1.6   Fri Apr 30 13:11:48 1999   holsti
 * SMR_196: New SW version.
 *
 *    Rev 1.5   Sat Apr 17 02:49:10 1999   holsti
 * SMR_191: Precedence error in shift-plus corrected.
 * SNCR_212:
 *
 *    Rev 1.3   Thu Apr 08 22:48:52 1999   holsti
 * SMR_184: Correct Init_SU_Settings and simplify Boot
 *
 *    Rev 1.2   Thu Apr 01 11:49:10 1999   ville
 * SMR_174: New SW version
 *
 *    Rev 1.1   Mon Jan 18 11:52:58 1999   paloheim
 * SMR_114: Problems due to multiple inclusion prevented
 * SNCR_118
 *
 *    Rev 1.0   Wed Dec 30 14:21:56 1998   ville
 * Initial revision.
/*
/*- * ------------------------------------------------------------------------*/
```

```
/* Part of : DAS */

#ifndef VERSION_H
#define VERSION_H

#define SW_VERSION 19
/* Software version. Unique for each delivered build of the software. */

#define CODE_CHECKSUM 0x74
/* This must be set so that the checksum calculated from the code */
/* memory becomes zero (this value will then be actually equal to */
/* the code checksum when this value would be zero).              */
/* So the procedure is as follows:                                */
/*   1. Set CODE_CHECKSUM to zero.                                */
/*   2. Compile and link DEBIE into a .hex file.                  */
/*   3. Compute the XOR of the .hex data (using e.g. 'hexor').    */
/*   4. Set CODE_CHECKSUM to the value computed in step 3.        */
/*   5. Recompile and relink DEBIE into a new .hex file.          */
/*   6. Verify that XOR of the new .hex file is zero.             */

#endif
```

8 DHI Header Files

This chapter contains the header source files for the DEBIE Harware Interface software module.

These listings are taken from the reference directory of the PVCS archive of DHI. The PVCS tool is used as a configuration management tool.

8.1 ad_conv.h

```
/*-----------------------------------------------------------------------------
/*                                                                           --
/*    Copyright (C) 1998 : Space Systems Finland Ltd.                        --
/*                                                                           --
/*-----------------------------------------------------------------------------
/*
/*    System Name:   DEBIE DPU SW
/*
/*      $Workfile:   ad_conv.h  $
/*        $Author:   paloheim  $
/*          $Date:   Tue Jun 01 12:35:44 1999  $
/*      $Revision:   1.6  $
/*           $Log:   /home/share/fileserver/projects/debie/design/code/DHI/PVCS/
ad_conv.h,p  $
 *
 *    Rev 1.6   Tue Jun 01 12:35:44 1999   paloheim
 * SMR_213: Commented reentrancy, edited comments and updated SW
 * SNCR_198:
 * SNCR_171:
 * SNCR_186:
 * SNCR_238: no scp
 * SNCR_154: no scp
 *
 *    Rev 1.5   Mon Jan 18 11:45:32 1999   paloheim
 * SMR_114: Problems due to multiple inclusion prevented
 * SNCR_118
 *
 *    Rev 1.4   Fri Oct 02 17:10:22 1998   ville
 * SMR_078: Reading of ADC channel register deleted etc.
 * SNCR_079
 * SNCR_080
 * SNCR_083
 *
 *    Rev 1.3   Tue Aug 11 16:53:00 1998   ville
 * SMR_016: Macro redefined
 * SNCR_019
 *
 *    Rev 1.2   Fri Aug 07 14:17:08 1998   ville
 * SMR_014: Macros added for ADC bipolar/unipolar selection
 * SNCR_016
 *
 *    Rev 1.1   Fri Jul 24 15:03:12 1998   ville
 * Updated during DHI coding
 *
 *    Rev 1.0   Thu Jul 02 10:24:46 1998   ville
 * Initial revision.
/*
/*- * ----------------------------------------------------------------------------*/

/* Macros, functions, types and constants for controlling AD converter */

/* Part of : DHI */
```

```
#ifndef AD_CONV_H
#define AD_CONV_H

#ifndef REG52_DEFINED
   #include
   #define REG52_DEFINED
#endif

#ifndef ABS_ADD_DEFINED
   #include
   #define ABS_ACC_DEFINED
#endif

/* Constant definitions */

#define AD_CHANNELS     0x28
/* Number of possible ADC channels (includes GND channels) */

/* AD converter control register addresses            */
/* NOTE: these registers should be accessed only from */
/* ad_conv.c or macros defined in this file           */

#define  ADC_STATUS      0xFF20
#define  ADC_DATA        0xFF21
#define  START_ADC       0XFF21
#define  RESET_ADC       0xFF20
#define  ADC_CHANNEL     0xFFA0
#define  DAC_OUTPUT      0xFFB7
#define  BP_UP           0x40
#define  BP_DOWN         0xBF

sbit     END_OF_ADC      = P1^2;

/* AD converter control macros */

#define UPDATE_ADC_CHANNEL_REG      XBYTE[ADC_CHANNEL] = ADC_channel_register
#define START_CONVERSION            XBYTE[START_ADC]   = 0
#define GET_RESULT                  XBYTE[ADC_DATA]
#define SET_DAC_OUTPUT(LEVEL)       XBYTE[DAC_OUTPUT]  = (LEVEL)

extern unsigned char xdata ADC_channel_register;
/* Holds value of the ADC Channel HW register       */
/* Is used by Hit Trigger ISR task and Health       */
/* Monitoring task.                                 */
/* Updating must be atomic in the Health Monitoring */
/* task, because Hit Trigger can preempt it.        */

#endif
```

8.2 dpu_ctrl.h

```
/*-----------------------------------------------------------------------------
/*                                                                           --
/*    Copyright (C) 1998 : Space Systems Finland Ltd.                        --
/*                                                                           --
/*-----------------------------------------------------------------------------
/*
/*    System Name:   DEBIE DPU SW
/*
/*      $Workfile:   dpu_ctrl.h  $
/*        $Author:   holsti  $
/*          $Date:   Fri May 28 14:59:30 1999  $
/*      $Revision:   1.23  $
/*           $Log:   /home/share/fileserver/projects/debie/design/code/DHI/PVCS/
dpu_ctrl.h,p
 *
 *    Rev 1.23   Fri May 28 14:59:30 1999   holsti
 * SMR_209: Memory Patch TC corrections.
 * SNCR_194:
 *
 *    Rev 1.22   Wed May 12 14:03:56 1999   holsti
 * SMR_201: Fast memory test and other reset updates.
 * SNCR_181:
 * SNCR_188:
 * SNCR_214:
 * SNCR_215:
 *
 *    Rev 1.21   Wed Apr 28 17:36:06 1999   holsti
 * SMR_193: Repair block telemetry.
 * SNCR_221:
 * SNCR_222:
 *
 *    Rev 1.20   Thu Mar 18 09:27:26 1999   paloheim
 * SMR_155: Corrected code patching
 * SNCR_161:
 * SNCR_162:
 *
 *    Rev 1.19   Thu Mar 11 10:48:48 1999   paloheim
 * SMR_147: Event storage condition signal check added
 * SNCR_120:
 *
 *    Rev 1.18   Tue Mar 09 12:36:06 1999   paloheim
 * SMR_146: TM alignment constraints are taken into account.
 * SNCR_149:
 * SNCR_150:
 *
 *    Rev 1.17   Thu Feb 18 15:15:48 1999   paloheim
 * SMR_138: Edited reentrant function declarations and line lengths
 * SNCR_139:
 *
 *    Rev 1.16   Thu Feb 11 15:34:16 1999   paloheim
 * SMR_136: Use of error and mode status bits corrected
 *  SNCR_134:
 *  SNCR_136:
```

```
* SNCR_137:
*
*    Rev 1.15   Mon Feb 01 16:00:02 1999   paloheim
* SMR_131: Line lengths checked and edited
* SNCR: none
*
*    Rev 1.14   Mon Jan 25 14:25:00 1999   paloheim
* SMR_128: CalculateChecksum edited
* SNCR_115
* SNCR_117
*
*    Rev 1.13   Tue Jan 19 09:54:50 1999   paloheim
* SMR_116: Code copying enabled
* SNCR_087
* SNCR_111
*
*    Rev 1.12   Mon Jan 18 11:47:42 1999   paloheim
* SMR_114: Problems due to multiple inclusion prevented
* SNCR_118
*
*    Rev 1.11   Tue Jan 05 16:10:18 1999   paloheim
* SMR_110: Health monitoring updated
*
*    Rev 1.10   Wed Dec 30 14:25:08 1998   ville
* SMR_103: Write Memory telecommands implemented.
*
*    Rev 1.9   Tue Dec 29 13:37:34 1998   ville
* SMR_102: Science Data protected.
*
*    Rev 1.8   Mon Sep 21 14:58:46 1998   ville
* SMR_063: Warm reset made possible
* SNCR_065
*
*    Rev 1.7   Mon Sep 07 12:26:46 1998   ville
* SMR_053: GetResultClass changed to non-reentrant
* SNCR_055
*
*    Rev 1.6   Fri Sep 04 17:30:16 1998   ville
* SMR_050: Program not copied in Prototype SW
* SNCR_051
*
*    Rev 1.5   Thu Aug 13 11:23:42 1998   ville
* SMR_024: Macros changed to portable
* SNCR_024
*
*    Rev 1.4   Wed Aug 12 13:55:32 1998   ville
* SMR_021: CopyProgramCode function added
* SNCR_015
*
*    Rev 1.3   Tue Aug 04 13:53:18 1998   ville
* SMR_006: Typos in include file check corrected
* SNCR_007
*
*    Rev 1.2   Mon Aug 03 11:28:44 1998   ville
```

```
 * SMR_003: Enumeration redefinitions in dpu_ctrl.h
 * SNCR_004
 *
 *    Rev 1.1   Tue Jul 28 16:51:30 1998   ville
 * Type definitions modified: typedef used
 * Local variables eliminated from reentrant functions
/*                     dpu_ctrl.h,p  $
 *
 *    Rev 1.0   Thu Jul 02 10:30:42 1998   ville
 * Initial revision.
/*
/*- * ------------------------------------------------------------------------*/

/* Part of : DHI */

#ifndef DPU_CTRL_H
#define DPU_CTRL_H

#ifndef REG52_DEFINED
   #include
   #define REG52_DEFINED
#endif

#ifndef EXTERNAL
   #include "keyword.h"
#endif

#define SAME 1
#define NOT_SAME 0

#define MEMORY_PATCHED 1
#define MEMORY_NOT_PATCHED 0

#ifndef HIGH
   #define HIGH 1
#endif

#ifndef LOW
   #define LOW 0
#endif

#define SELECTED 1
#define NOT_SELECTED 0
#define RESET_OK 1
#define RESET_NOT_OK 0

#define ACCEPT_EVENT 1
#define REJECT_EVENT 0

sbit    EVENT_FLAG   = P1^4;
/* Event storage condition signal. */

/* memory addresses for program copy                                    */
#define PROGRAM_COPY_START 0x1000
```

```
#define PROGRAM_COPY_END   0x8000
/* this can be replaced with real end address (+1) of used program code      */
/* given in the linker map file                                              */

/* Comment or delete following definition, if program should be executed     */
/* from RAM                                                                  */
/* #define USE_ALWAYS_PROM */

/* memory addresses for patching                                             */
#define BEGIN_SRAM1       0x1000
#define END_SRAM1         0x7FFF
#define BEGIN_SRAM3       0x8000
#define END_SRAM3         0xFEFF
#define BEGIN_DATA_RAM    0x0000


#define SCIENCE_DATA_START_ADDRESS 0x0000
/* First free absolute data address. */

#define   INITIAL_CHECKSUM_VALUE   0
/* A value given to 'reference_checksum' variable at 'Boot()'. */
/* It is zero, since one code byte is dedicated to a constant  */
/* that ensures that the checksum of the PROM is zero.         */

#define SET_DATA_BYTE(ADDR,VALUE) XBYTE[(ADDR)] = (VALUE)
#define GET_DATA_BYTE(ADDR)       XBYTE[(ADDR)]
#define GET_CODE_BYTE(ADDR)       CBYTE[(ADDR)]
#define DATA_POINTER(ADDR)        ((unsigned char *)(ADDR))


/* some register addresses */

sbit    WD_STATUS   = P1^3;
sbit    WD_RESET    = P1^7;
sbit    MEM_CONF    = P1^5;

/* macros to handle above registers */

#define SET_WD_RESET_HIGH WD_RESET = HIGH
#define SET_WD_RESET_LOW  WD_RESET = LOW
#define SET_MEM_CONF_PROM MEM_CONF = HIGH
#define SET_MEM_CONF_SRAM MEM_CONF = LOW

/* macro used in healthmonitoring */
#define CHECK_CURRENT(BIT_NUMBERS)      (XBYTE[HV_STATUS] &<HardSpace
                                       (BIT_NUMBERS))
/* Checks whether given bit in the HV Status Register is HIGH or LOW. */



/*type definitions*/

typedef unsigned char DEBIE_mode_t;
/* Debie mode index. Valid values:  */
```

```
/* 00 DPU self test                */
/* 01 Stand by                     */
/* 10 Acquisition                  */


typedef enum {
   power_up_reset_e = 0, /* Don't change value ! */
   watchdog_reset_e = 1, /* Don't change value ! */
   soft_reset_e,
   warm_reset_e,
   error_e,
   checksum_reset_e
}  reset_class_t;

extern reset_class_t EXTERNAL s_w_reset;

typedef enum  {
   PROM_e, SRAM_e
} memory_configuration_t;


#define MAX_TIME 0xFFFFFFFF
/* Maximum value for DEBIE time. */

typedef unsigned long int dpu_time_t;
/* Data type for DEBIE time. */

/*-------------------------------------------------------*/
/* Multi-byte TM types without alignment constraints.    */
/* The Keil/8051 system does not constrain alignment for */
/* multi-byte types such as long int, but the Unix test  */
/* systems do, and we must declare the TM structures to  */
/* avoid constraints.                                    */
/* Note that we still assume that the two systems have   */
/* the same endianness.                                  */
/* Note also that the native type is used in the TM when */
/* the component happens to have the correct alignment.  */
/* The special types are used only when the alignment is */
/* otherwise incorrect.                                  */
/* Data is moved between the native type and the special */
/* TM type only using the COPY or VALUE_OF macros. Thus, */
/* on the 8051 the two types can in fact be identical,   */
/* with COPY defined as direct assignment ("=").         */

typedef dpu_time_t tm_dpu_time_t;
/* The TM type corresponding to dpu_time_t (32-bit int). */

typedef unsigned short int tm_ushort_t;
/* The TM type corresponding to unsigned short int.      */
/*                                                       */
/*-------------------------------------------------------*/


typedef struct {
```

```
   unsigned char *source;
   unsigned int   destination;
   unsigned char  data_amount;
   unsigned char  execution_command;
} memory_patch_variables_t;
/* Holds parameters for PatchCode function:                          */
/*    source             source address of the patch                */
/*                       (should be between 0x8000 and 0xFF00 -     */
/*                        data_amount)                              */
/*    destination        destination address of the patch           */
/*                       (should be between 0x1000 and 0x8000 -     */
/*                        data_amount)                              */
/*    data_amount        amount of bytes to be patched (max 255)    */
/*    execution_command  action executed after patch                */
/*                          0x00 - continue normally                */
/*                          0x09 - execute soft reset               */
/*                          0x37 - execute warm reset               */
/*                          0x5A - jump to start address of patched */
/*                                 memory area                      */

#define CALL_PATCH(FUNCTION) FUNCTION()
/* Jump to the patched memory. */

extern unsigned char EXTERNAL code_not_patched;
/* Initial value is 1, set at Boot(). Value is 1 when code   */
/* checksum value is valid, cleared to 0 when code memory is */
/* patched, set to 1 when next checksum calculation          */
/* period is started.                                         */

extern unsigned char EXTERNAL reference_checksum;
/* Expected value for code checksum. Will be changed when   */
/* code memory is patched.                                  */


/* Function prototypes: */

extern void Init_DPU (reset_class_t reset_class);

extern reset_class_t GetResetClass(void);

extern void SignalMemoryErrors (void);

extern void SetMemoryConfiguration (memory_configuration_t memory);

extern memory_configuration_t GetMemoryConfiguration(void) compact reentrant;

extern void PatchCode(memory_patch_variables_t xdata *patch_variables);

extern void Reboot(reset_class_t boot_type);


/* Assembly-language function prototypes (asmfuncs.a51): */

extern unsigned char TestMemBits (unsigned short address);
```

```
extern unsigned char TestMemData (
                unsigned short start,
                unsigned char  bytes);

extern unsigned char TestMemSeq (
                unsigned short start,
                unsigned char  bytes);

#endif
```

8.3 isr_ctrl.h

```
/*-----------------------------------------------------------------------------
/*                                                                           --
/*    Copyright (C) 1998 : Space Systems Finland Ltd.                        --
/*                                                                           --
/*-----------------------------------------------------------------------------
/*
/*    System Name:   DEBIE DPU SW
/*
/*      $Workfile:   isr_ctrl.h  $
/*        $Author:   holsti  $
/*          $Date:   Sun Jul 25 15:02:08 1999  $
/*      $Revision:   1.16  $
/*           $Log:   /home/share/fileserver/projects/debie/design/code/DHI/PVCS/
isr_ctrl.h,p  $
 *
 *    Rev 1.16   Sun Jul 25 15:02:08 1999   holsti
 * SMR_220: RTX-51 Timer Interrupt priority is high.
 * SNCR_250:
 *
 *    Rev 1.15   Mon May 17 22:50:20 1999   holsti
 * SMR_205: Corrected RTX-51 usage mistages.
 * SNCR_232:
 *
 *    Rev 1.14   Thu May 13 16:08:28 1999   paloheim
 * SMR_203: RTX-error indication implemented
 * SNCR_227:
 * SNCR_225:
 * SNCR_186:
 *
 *    Rev 1.13   Thu Feb 18 15:16:14 1999   paloheim
 * SMR_138: Edited reentrant function declarations and line lengths
 * SNCR_139:
 *
 *    Rev 1.12   Mon Feb 15 14:08:10 1999   paloheim
 * SMR_137: SU Self Test Implemented
 * SNCR: none
 *
 *    Rev 1.11   Mon Feb 08 15:42:20 1999   paloheim
 * SMR_132: Hits during one Monitoring period limited
 * SNCR_109:
 *
 *    Rev 1.10   Mon Feb 01 16:00:26 1999   paloheim
 * SMR_131: Line lengths checked and edited
 * SNCR: none
 *
 *    Rev 1.9   Mon Jan 18 11:53:48 1999   paloheim
 * SMR_114: Problems due to multiple inclusion prevented
 * SNCR_118
 *
 *    Rev 1.8   Mon Dec 28 12:17:10 1998   ville
 * SMR_101: Interrupt priorities set
 * SNCR_096
 *
```

```
 *    Rev 1.7   Wed Dec 02 12:03:20 1998   ville
 * SMR_098: Minimum TC interval implemented
 * SNCR_103
 *
 *    Rev 1.6   Fri Oct 02 17:11:18 1998   ville
 * SMR_078: Reading of ADC channel register deleted etc.
 * SNCR_079
 * SNCR_080
 * SNCR_083
 *
 *    Rev 1.5   Mon Sep 07 16:37:26 1998   ville
 * SMR_055: Correct parameter for ResetInterruptMask used
 * SNCR_057
 *
 *    Rev 1.4   Tue Sep 01 15:35:04 1998   ville
 * SMR_044: Hit trigger handling corrected
 * SNCR_045
 *
 *    Rev 1.3   Fri Aug 21 14:52:08 1998   ville
 * SMR_034: WaitInterrupt made portable
 * SNCR_035
 *
 *    Rev 1.2   Tue Jul 28 16:56:10 1998   ville
 * Local variables eliminated from reentrant functions
 *
 *    Rev 1.1   Fri Jul 24 14:58:26 1998   ville
 * Updated during DHI coding
 *
 *    Rev 1.0   Thu Jul 02 10:33:30 1998   ville
 * Initial revision.
/*
/*- * ------------------------------------------------------------------------*/

/* Part of : DHI */


/*Type definitions*/

#ifndef ISR_CTRL_H
#define ISR_CTRL_H

#ifndef RTX51_DEFINED
   #include
   #define RTX51_DEFINED
#endif

#ifndef REG52_DEFINED
   #include
   #define REG52_DEFINED
#endif

#define SET_HIT_TRIGGER_ISR_FLAG EXF2=1

#define DISABLE_HIT_TRIGGER      EXEN2=0
```

```
#define HIT_TRIGGER_FLAG         EXEN2

#define CLEAR_HIT_TRIGGER_ISR_FLAG EXF2=0
#define ENABLE_HIT_TRIGGER         EXEN2=1

#define SET_INTERRUPT_PRIORITIES IP=0x2D
/* Sets Timer 1, Timer 2 and External interrupt 0 and 1 */
/* priorities high, others to low.                      */

#define DISABLE_INTERRUPT_MASTER EA=0
#define ENABLE_INTERRUPT_MASTER  EA=1
/* Clear and set interrupt master enable bit */

#define DISABLE_TC_TIMER_ISR  ET0 = 0
/* Macro for clearing TC timer interrupt enable flag */

#define TC_TIMER_OVERFLOW_FLAG  TF0
/* TC timer overflow flag */

#define CLEAR_TC_TIMER_OVERFLOW_FLAG  TF0 = 0
#define SET_TC_TIMER_OVERFLOW_FLAG    TF0 = 1
/* TC timer overflow flag manipulation macros. */



/* Function prototypes */

extern void AttachInterrupt(unsigned char ISR_VectorNumber);

extern void EnableInterrupt(unsigned char ISR_VectorNumber);

extern void DisableInterrupt(unsigned char ISR_VectorNumber);

extern signed char SetInterruptMask(unsigned char ISR_MaskNumber);

extern signed char ResetInterruptMask(unsigned char ISR_MaskNumber);

extern void WaitInterrupt (unsigned char ISR_VectorNumber, unsigned char timer);

#endif
```

8.4 keyword.h

```
/*-----------------------------------------------------------------------------
/*                                                                           --
/*    Copyright (C) 1998 : Space Systems Finland Ltd.                        --
/*                                                                           --
/*-----------------------------------------------------------------------------
/*
/*    System Name:   DEBIE DPU SW
/*
/*      $Workfile:   keyword.h  $
/*        $Author:   paloheim  $
/*          $Date:   Tue Mar 09 12:37:20 1999  $
/*      $Revision:   1.9  $
/*           $Log:   /home/share/fileserver/projects/debie/design/code/DHI/PVCS/
keyword.h,p  $
 *
 *    Rev 1.9   Tue Mar 09 12:37:20 1999   paloheim
 * SMR_146: TM alignment constraints are taken into account.
 * SNCR_149:
 * SNCR_150:
 *
 *    Rev 1.8   Mon Jan 18 11:51:34 1999   paloheim
 * SMR_114: Problems due to multiple inclusion prevented
 * SNCR_118
 *
 *    Rev 1.7   Fri Sep 25 10:35:48 1998   ville
 * SMR_065: LOCATION macro changed to portable
 * SNCR_067
 *
 *    Rev 1.6   Thu Sep 24 11:03:20 1998   ville
 * SMR_064: Macros for two Keil special keywords added
 * SNCR_066
 *
 *    Rev 1.5   Tue Aug 25 15:19:24 1998   ville
 * SMR_040: Task function prototypes replaced etc.
 * SNCR_041
 *
 *    Rev 1.4   Tue Aug 04 11:07:04 1998   ville
 * SMR_004: Invalid Keil keyword macros corrected
 * SNCR_006
 *
 *    Rev 1.3   Tue Jun 02 15:42:08 1998   ville
 * Tried to fix PVCS header
/*
/*    Rev 1.2   Tue Jun 02 15:30:14 1998   ville
/* Improved PVCS header
/*
/*    Rev 1.1   Tue Jun 02 15:18:16 1998   ville
/* Added PVCS header
/*
/*- * ----------------------------------------------------------------------*/

/* Macro definitions for Keil specific keywords to be used */
/* in portable parts of the DEBIE DPU software             */
```

```
/* Part of : DHI */

/* Some macros for task and interrupt management */
#ifndef KEYWORD_H
#define KEYWORD_H


#ifndef REG52_DEFINED
   #include
   #define REG52_DEFINED
#endif


#define COPY(DEST,SOURCE)  DEST=(SOURCE)
/* Copies the value of SOURCE to the location DEST. */

#define VALUE_OF(SOURCE)   (SOURCE)
/* Returns the (integer) value of SOURCE. */


#define TASK(TASK_NUMBER)    _task_ TASK_NUMBER
#define PRIORITY(LEVEL)      _priority_ LEVEL
#define INTERRUPT(SOURCE)    interrupt SOURCE
#define USED_REG_BANK(BANK)  using BANK


/* Macro for declaring re-entrant function */

#define REENTRANT_FUNC    reentrant

/* Memory model handling macros */

#define SMALL_DATA        small
#define COMPACT_DATA      compact
#define LARGE_DATA        large

#define PROGRAM           code

#define EXTERNAL          xdata
#define DIRECT_INTERNAL   data
#define INDIRECT_INTERNAL idata

#define LOCATION(ADDRESS) _at_ ADDRESS

#endif
```

8.5 msg_ctrl.h

```
/*-----------------------------------------------------------------------------
/*                                                                           --
/*    Copyright (C) 1998 : Space Systems Finland Ltd.                        --
/*                                                                           --
/*-----------------------------------------------------------------------------
/*
/*    System Name:   DEBIE DPU SW
/*
/*      $Workfile:   msg_ctrl.h  $
/*        $Author:   holsti  $
/*          $Date:   Mon May 17 22:50:44 1999  $
/*      $Revision:   1.11  $
/*           $Log:   /home/share/fileserver/projects/debie/design/code/DHI/PVCS/
msg_ctrl.h,p  $
 *
 *    Rev 1.11   Mon May 17 22:50:44 1999   holsti
 * SMR_205: Corrected RTX-51 usage mistages.
 * SNCR_232:
 *
 *    Rev 1.10   Thu May 13 16:09:14 1999   paloheim
 * SMR_203: RTX-error indication implemented
 * SNCR_227:
 * SNCR_225:
 * SNCR_186:
 *
 *    Rev 1.9   Tue May 04 11:46:50 1999   paloheim
 * SMR_198: Corrected critical errors
 * SNCR_227
 * SNCR_219
 *
 *    Rev 1.8   Tue Mar 16 13:37:58 1999   paloheim
 * SMR_149: Corrected embedded control-H in line
 * SNCR_148:
 *
 *    Rev 1.7   Thu Feb 18 15:16:42 1999   paloheim
 * SMR_138: Edited reentrant function declarations and line lengths
 * SNCR_139:
 *
 *    Rev 1.6   Mon Feb 01 16:00:42 1999   paloheim
 * SMR_131: Line lengths checked and edited
 * SNCR: none
 *
 *    Rev 1.5   Mon Jan 18 11:49:22 1999   paloheim
 * SMR_114: Problems due to multiple inclusion prevented
 * SNCR_118
 *
 *    Rev 1.4   Sun Sep 06 20:00:26 1998   ville
 * SMR_052: Correct memory models used
 * SNCR_054
 *
 *    Rev 1.3   Thu Aug 06 15:44:54 1998   ville
 * SMR_011: Macros renamed
 * SNCR_012
```

```
 *
 *    Rev 1.2   Tue Jul 28 17:00:06 1998   ville
 * Type definitions modified: typedef used for structs
 * Local variables eliminated from reentrant functions
 *
 *    Rev 1.1   Fri Jul 24 15:02:08 1998   ville
 * Updated during DHI coding
 *
 *    Rev 1.0   Thu Jul 02 10:35:08 1998   ville
 * Initial revision.
/*
/*- * ------------------------------------------------------------------------*/

/* Part of : DHI */

/* Macros */

#ifndef MSG_CTRL_H
#define MSR_CTRL_H

#ifndef RTX51_DEFINED
   #include
   #define RTX51_DEFINED
#endif

#ifndef REG52_DEFINED
   #include
   #define REG52_DEFINED
#endif



#define SEND_MESSAGE(BOX, MESSAGE)    isr_send_message((BOX),(MESSAGE))
#define SEND_TASK_MESSAGE(BOX, MESSAGE, TIMEOUT)    os_send_message((BOX),(MESSAGE),(TIM
EOUT))

/* Type definitions */

#define MSG_RECEIVED 1
/* The value of execution_result in incoming_mail_t that */
/* signifies that a mail message has been received.      */
/* Must be different from NOT_OK as defined in RTX51.h.  */

#define TIMEOUT_OCCURRED 4
/* The value of execution_result in incoming_mail_t that */
/* signifies that the wait for mail has timed out.       */
/* Must be different from NOT_OK as defined in RTX51.h.  */


typedef struct {
   unsigned char  mailbox_number;
   unsigned int   message;
   unsigned char  timeout;
   signed char execution_result;
```

```
   /*This variable is used to indicate execution results.                    */
}outgoing_mail_t ;

typedef struct {
   unsigned char        mailbox_number;
   unsigned char        timeout;
   unsigned int xdata   *message;
   signed char execution_result;  /* This variable is used to indicate       */
                                  /* execution results.                      */

   signed char wait_result;       /* Result from a RTX operation.            */

   unsigned char event_selector;  /* The value of this variable defines the  */
                                  /* execution of the wait-task.             */
}incoming_mail_t ;

typedef struct {
   signed char execution_result;  /* This variable is used to indicate       */
                                  /* execution results.                      */

   signed char wait_result;       /* Result from a RTX operation.            */

   unsigned char event_selector;  /* The value of this variable defines the  */
                                  /* execution of the wait-task.             */

   unsigned char semaphore;       /* semaphore number                        */

   unsigned char timeout;         /* time-out definition                     */

}  wait_token_variables_t;
extern  wait_token_variables_t token_variables;

/* Function prototypes */

extern void WaitMail(incoming_mail_t xdata *message) compact reentrant;

#endif
```

8.6 su_ctrl.h

```
/*-----------------------------------------------------------------------------
/*                                                                           --
/*    Copyright (C) 1998 : Space Systems Finland Ltd.                        --
/*                                                                           --
/*-----------------------------------------------------------------------------
/*
/*    System Name:   DEBIE DPU SW
/*
/*      $Workfile:   su_ctrl.h  $
/*        $Author:   paloheim  $
/*          $Date:   Mon Aug 16 17:30:10 1999  $
/*      $Revision:   1.27  $
/*           $Log:   /home/share/fileserver/projects/debie/design/code/DHI/PVCS/
su_ctrl.h,p  $
 *
 *    Rev 1.27   Mon Aug 16 17:30:10 1999   paloheim
 * SMR_227: New SU Self Test sequence
 * SNCR_249:
 * SNCR_256:
 *
 *    Rev 1.26   Tue Jun 01 12:36:34 1999   paloheim
 * SMR_213: Commented reentrancy, edited comments and updated SW
 * SNCR_198:
 * SNCR_171:
 * SNCR_186:
 * SNCR_238: no scp
 * SNCR_154: no scp
 *
 *    Rev 1.25   Thu May 13 13:20:42 1999   paloheim
 * SMR_202: DPU and SU self test sequences implemented
 * SNCR: none
 *
 *    Rev 1.24   Wed May 12 13:07:12 1999   paloheim
 * SMR_201: Fast memory test and other reset updates
 * SNCR_181:
 * SNCR_214:
 * SNCR_215:
 *
 *    Rev 1.23   Sat Apr 17 02:34:12 1999   holsti
 * SMR_190: TestPulseLevel renamed to SetTestPulseLevel.
 * SNCR_211
 *
 *    Rev 1.22   Sat Apr 17 00:26:26 1999   holsti
 * SMR_189: Additional ADC channel selection delay.
 * SNCR_210:
 *
 *    Rev 1.21   Thu Apr 08 18:42:00 1999   paloheim
 * SMR_183: Initialised classification parameters after power-up
 * SNCR_201:
 *
 *    Rev 1.20   Tue Mar 16 14:16:22 1999   paloheim
 * SMR_151: Adjusted ReadDelayCounters dependence on sizeof(int)
 * SNCR_153:
```

```
*
*    Rev 1.19   Tue Mar 16 13:48:10 1999   paloheim
* SMR_150: Corrected size of data for COPY
* SNCR_152:
*
*    Rev 1.18   Mon Mar 01 11:13:18 1999   paloheim
* SMR_145: SU supply status modified
* SNCR_147:
*
*    Rev 1.17   Mon Feb 22 15:43:34 1999   paloheim
* SMR_142: Set default trigger thresholds and fixed failed_address parameter
* SNCR_143:
*
*    Rev 1.16   Thu Feb 18 15:17:00 1999   paloheim
* SMR_138: Edited reentrant function declarations and line lengths
* SNCR_139:
*
*    Rev 1.15   Mon Feb 15 14:11:26 1999   paloheim
* SMR_137: SU Self Test Implemented
* SNCR: none
*
*    Rev 1.14   Mon Feb 01 16:01:02 1999   paloheim
* SMR_131: Line lengths checked and edited
* SNCR: none
*
*    Rev 1.13   Mon Jan 18 11:52:22 1999   paloheim
* SMR_114: Problems due to multiple inclusion prevented
* SNCR_118
*
*    Rev 1.12   Mon Jan 11 16:51:20 1999   paloheim
* SMR_111: Update SU state transitions
* SNCR_107
*
*    Rev 1.11   Mon Jan 04 13:55:32 1999   paloheim
* SMR_109: ADC Channel selection delay updated
* SNCR_104
*
*    Rev 1.10   Fri Nov 13 13:26:44 1998   ville
* SMR_090: New Hit trigger sequence implemented.
* SNCR_095
*
*    Rev 1.9   Tue Sep 29 10:57:50 1998   ville
* SMR_068: Reading of write-only registers deleted
* SNCR_068
*
*    Rev 1.8   Mon Aug 31 13:02:12 1998   ville
* SMR_043: Reset Peak Detector pulse generated
* SNCR_044
*
*    Rev 1.7   Thu Aug 13 11:22:40 1998   ville
* SMR_024: Macros changed to portable
* SNCR_024
*
*    Rev 1.6   Tue Aug 11 16:24:10 1998   ville
```

```
 * SMR_015: Variable type redefined
 * SNCR_018
 *
 *    Rev 1.5   Wed Aug 05 15:40:50 1998   ville
 * SMR_009: ResetPeakDetector function added
 * SNCR_011
 *
 *    Rev 1.4   Wed Aug 05 14:36:54 1998   ville
 * SMR_008: ReadDelayCounters parameter redefined
 * SNCR_009
 *
 *    Rev 1.3   Mon Aug 03 10:58:50 1998   ville
 * SMR_002: SetTriggerLevel function prototype redefined
 * SNCR_003
 *
 *    Rev 1.2   Tue Jul 28 17:02:46 1998   ville
 * Type definitions modified: typedef used for structs
 * Local variables eliminated from reentrant functions
 *
 *    Rev 1.1   Fri Jul 24 15:04:16 1998   ville
 * Updated during DHI coding
 *
 *    Rev 1.0   Thu Jul 02 10:36:14 1998   ville
 * Initial revision.
/*
/*- * ----------------------------------------------------------------------------*/

/* Macros, functions, constants and types for controlling Sensor Units */

/* Part of : DHI */

#ifndef SU_CTRL_H
#define SU_CTRL_H

#ifndef REG52_DEFINED
   #include
   #define REG52_DEFINED
#endif

#ifndef RTX51_DEFINED
   #include
   #define RTX51_DEFINED
#endif

#ifndef ABS_ADD_DEFINED
   #include
   #define ABS_ACC_DEFINED
#endif

/* Sensor Channels */

#define  PLASMA_1_PLUS  0
#define  PLASMA_1_MINUS 1
#define  PZT_1          2
```

```
#define  PZT_2          3
#define  PLASMA_2_PLUS  4
#define  PZT_1_2        5


#define SU_1 1
#define SU_2 2
#define SU_3 3
#define SU_4 4

#define SU_1_ON 1
#define SU_2_ON 2
#define SU_3_ON 3
#define SU_4_ON 4

#define SU_1_OFF 1
#define SU_2_OFF 2
#define SU_3_OFF 3
#define SU_4_OFF 4

#define LOW_PLASMA_SELF_TEST_THRESHOLD         0x15
#define LOW_PIEZO_SELF_TEST_THRESHOLD          0x0D
#define HIGH_PLASMA_1_PLUS_SELF_TEST_THRESHOLD 0xAB
#define HIGH_PLASMA_SELF_TEST_THRESHOLD        0x80
#define HIGH_PIEZO_SELF_TEST_THRESHOLD         0x2B
#define MAX_PLASMA_SELF_TEST_THRESHOLD         0xFF
#define MAX_PIEZO_SELF_TEST_THRESHOLD          0xFF
/* Self test threshold levels. */

#define PLASMA_1_PLUS_LOW          0x12
#define PLASMA_1_MINUS_LOW         0x07
#define PLASMA_2_PLUS_LOW          0x0d
#define PZT_1_LOW                  0x2B
#define PZT_2_LOW                  0x26
/* Low level test pulses. */

#define PLASMA_1_PLUS_HIGH         0x50
#define PLASMA_1_MINUS_HIGH        0x24
#define PLASMA_2_PLUS_HIGH         0x40
#define PZT_1_HIGH                 0xB5
#define PZT_2_HIGH                 0xA1
/* High level test pulses. */

#define SU_NOT_ACTIVATED 0
#define SU_NOT_DEACTIVATED 0

#define CHANNEL_NOT_SELECTED 5
#define SU_NOT_SELECTED 6

#define TRIGGER_SET_OK 1

#define DEFAULT_THRESHOLD 0x0D
/* Default Trigger threshold is mid-scale value. */
```

```
#define DEFAULT_TEST_PULSE_LEVEL 0x00

#define DEFAULT_CLASSIFICATION_LEVEL 0
#define DEFAULT_MAX_TIME             255
#define DEFAULT_MIN_TIME             0
/* These default levels are only temporary */

#define SU_ONOFF_MASK 3
/* Bit mask for SU Status register manipulation when SU is */
/* switched ON or OFF.                                     */

#define SU_STATE_TRANSITION_OK     1
#define SU_STATE_TRANSITION_FAILED 0

#define NO_SU 0

/* Trigger level register base addresses */

#define  SU_1_TRIGGER_BASE    0xFFB0
#define  SU_2_TRIGGER_BASE    0xFFB3
#define  SU_3_TRIGGER_BASE    0xFFC0
#define  SU_4_TRIGGER_BASE    0xFFC3

#define  DELAY_1_2_MSB 0xFF40
#define  DELAY_1_LSB 0xFF30
#define  DELAY_2_LSB 0xFF50

#define SU_1_THRESHOLD1 0xFFB0
#define SU_1_THRESHOLD2 0xFFB1
#define SU_1_THRESHOLD3 0xFFB2

#define SU_2_THRESHOLD1 0xFFB3
#define SU_2_THRESHOLD2 0xFFB4
#define SU_2_THRESHOLD3 0xFFB5

#define SU_3_THRESHOLD1 0xFFC0
#define SU_3_THRESHOLD2 0xFFC1
#define SU_3_THRESHOLD3 0xFFC2

#define SU_4_THRESHOLD1 0xFFC3
#define SU_4_THRESHOLD2 0xFFC4
#define SU_4_THRESHOLD3 0xFFC5

/* Some other register addresses */

#define  TEST_PULSE_LEVEL             0xFFB6
#define  SU_SELF_TEST_CH              0xFFE0

#define GET_MSB_COUNTER               XBYTE[DELAY_1_2_MSB]
#define GET_LSB1_COUNTER              XBYTE[DELAY_1_LSB]
#define GET_LSB2_COUNTER              XBYTE[DELAY_2_LSB]
#define RISE_TIME_COUNTER             0xFF60

#define SU_CONTROL 0xFFD0
```

```
#ifndef LOW
   #define LOW 0
#endif

/* Macros */


#define SU_1_MINUS_50   1
#define SU_1_PLUS_50    2
#define SU_2_MINUS_50   4
#define SU_2_PLUS_50    8
#define SU_3_MINUS_50   16
#define SU_3_PLUS_50    32
#define SU_4_MINUS_50   64
#define SU_4_PLUS_50    128

#define  TRIGGER_SOURCE T0 + 2 * T1

#define  HV_STATUS       0xFF70
#define  SET_COUNTER_RESET(LEVEL) COUNTER_RESET = LEVEL

sbit TRIGGER_SOURCE_0    = P3^4;
sbit TRIGGER_SOURCE_1    = P3^5;
sbit     V_DOWN          = P1^0;
sbit     COUNTER_RESET   = P1^6;

/* Type definitions */

typedef unsigned char sensor_number_t;
/* Sensor Unit number. Valid values SU_1, SU_2, SU_3 and SU_4 */
/* which must be successive integers.                         */
/* As a special case for some variables, the value NO_SU      */
/* indicates "no Sensor Unit". This possibility is always     */
/* shown by a comment on that variable, otherwise only real   */
/* SU numbers are allowed.                                    */


typedef unsigned char sensor_index_t;
/* Sensor Unit index. Valid values 0, 1, 2 and 3.  */
/* Index 0 corresponds to Sensor Unit number SU_1. */


typedef struct {
   unsigned short int FromPlasma1Plus;
   unsigned short int FromPlasma1Minus;
}delays_t ;

typedef struct {
   unsigned char sensor_unit;
   unsigned char channel;
   unsigned char level;
   unsigned char execution_result;
```

```
   unsigned int  base;
}trigger_set_t ;

typedef struct {
   unsigned char V_down_bit;
   unsigned char HV_status;
}voltage_status_t ;


extern unsigned char xdata SU_ctrl_register;
/* This variable stores values of write-only registers */

/* Function prototypes */

/* Sensor Unit status */


/* Delay and rise time counters */
extern void ReadDelayCounters (delays_t EXTERNAL *delay);
extern unsigned char ReadRiseTimeCounter(void) compact reentrant;
extern void ResetDelayCounters(void) compact reentrant;
extern void ResetPeakDetector(unsigned char unit);
extern void SignalPeakDetectorReset(
   unsigned char low_reset_value,
   unsigned char high_reset_value);


/* Trigger levels */
extern void SetTriggerLevel(trigger_set_t xdata *setting) compact reentrant;

/* Test pulse level */
extern void SetTestPulseLevel(unsigned char level) compact reentrant;

extern void GetVoltageStatus(voltage_status_t xdata *v_status)
               compact reentrant;


/* Sensor Unit power control */
extern void Switch_SU_On  (unsigned char SU_Number, unsigned char xdata
               *execution_result) compact reentrant;

extern void Switch_SU_Off (unsigned char SU_Number, unsigned char xdata
               *execution_result) compact reentrant;

/* Sensor Unit calibration */

extern void EnableAnalogSwitch(sensor_index_t self_test_SU_index);
extern void DisableAnalogSwitch(sensor_index_t self_test_SU_index);
extern void SelectSelfTestChannel(unsigned char channel);
extern void SelectTriggerSwitchLevel(
           unsigned char test_channel,
           sensor_index_t self_test_SU_index);
extern void SelectStartSwitchLevel(
           unsigned char test_channel,
```

```
            sensor_index_t self_test_SU_index);


#endif
```

8.7 taskctrl.h

```
/*-----------------------------------------------------------------------------
/*                                                                           --
/*    Copyright (C) 1998 : Space Systems Finland Ltd.                        --
/*                                                                           --
/*-----------------------------------------------------------------------------
/*
/*    System Name:   DEBIE DPU SW
/*
/*      $Workfile:   taskctrl.h  $
/*        $Author:   holsti  $
/*          $Date:   Mon May 17 22:51:12 1999  $
/*      $Revision:   1.11  $
/*           $Log:   /home/share/fileserver/projects/debie/design/code/DHI/PVCS/
taskctrl.h,p  $
 *
 *    Rev 1.11   Mon May 17 22:51:12 1999   holsti
 * SMR_205: Corrected RTX-51 usage mistages.
 * SNCR_232:
 *
 *    Rev 1.10   Sun May 16 09:19:26 1999   holsti
 * SMR_204: RTX interface portable and no calls from interrupts.
 * SNCR_230:
 *
 *    Rev 1.9   Thu May 13 16:09:50 1999   paloheim
 * SMR_203: RTX-error indication implemented
 * SNCR_227:
 * SNCR_225:
 * SNCR_186:
 *
 *    Rev 1.8   Sat Apr 17 00:27:28 1999   holsti
 * SMR_189: Additional ADC channel selection delay.
 * SNCR_210:
 *
 *    Rev 1.7   Thu Feb 18 15:17:16 1999   paloheim
 * SMR_138: Edited reentrant function declarations and line lengths
 * SNCR_139:
 *
 *    Rev 1.6   Mon Feb 01 16:01:24 1999   paloheim
 * SMR_131: Line lengths checked and edited
 * SNCR: none
 *
 *    Rev 1.5   Mon Jan 18 11:54:42 1999   paloheim
 * SMR_114: Problems due to multiple inclusion prevented
 * SNCR_118
 *
 *    Rev 1.4   Mon Jan 04 13:53:30 1999   paloheim
 * SMR_109: ADC Channel selection delay updated
 * SNCR_104
 *
 *    Rev 1.3   Fri Nov 13 13:30:08 1998   ville
 *  SMR_090: New Hit trigger sequence implemented.
 * SNCR_095
 *
```

```
 *    Rev 1.2   Tue Jul 28 17:04:48 1998   ville
 * Type definitions modified: typedef used for structs
 * Local variables eliminated from reentrant functions
 *
 *    Rev 1.1   Fri Jul 24 14:59:40 1998   ville
 * Updated during DHI coding
 *
 *    Rev 1.0   Thu Jul 02 10:37:44 1998   ville
 * Initial revision.
/*
/*- * ----------------------------------------------------------------------*/

/* Part of : DHI */

/* Type definitions */

#ifndef TASKCTRL_H
#define TASKCTRL_H

#ifndef RTX51_DEFINED
   #include
   #define RTX51_DEFINED
#endif

#define MACHINE_CYCLE     1.085
/* The machine (processor) cycle time, in microseconds. */

#define DELAY_LIMIT(TIME) (unsigned short)((((TIME) / MACHINE_CYCLE) - 4) / 2)
/* Computes the number of ShortDelay() argument-units that corresponds */
/* to a certain delay TIME in microseconds. Note that this formula can */
/* yield values larger than ShortDelay() can implement in one call.    */
/* This formula is mainly intended for use with compile-time constant  */
/* values for TIME.                                                    */

#define MAX_SHORT_DELAY 255
/* The largest possible argument for ShortDelay(). */


typedef struct {
   unsigned char  rtx_task_number;
   void           (code *task_main_function)(void);
} task_info_t;

/* Function prototypes */

extern void ShortDelay (unsigned char delay_loops);

extern void CreateTask(task_info_t xdata *new_task);

extern void WaitInterval(unsigned char time);

extern void WaitTimeout(unsigned char time) compact reentrant;

extern void SetTimeSlice(unsigned int time_slice);
```

```
extern void StartSystem(unsigned char task_number);

extern void SendTaskMail (
   unsigned char mailbox,
   unsigned char message,
   unsigned char timeout);


#define Send_ISR_Mail(mailbox,message)                <HardReturn
   if (isr_send_message (mailbox, message) == NOT_OK)  <HardReturn
   {                                                   <HardReturn
      telemetry_data.isr_send_message_error = mailbox; <HardReturn
   }
/* Send_ISR_Mail is to be used from C51 interrupt routines to send    */
/* mail messages to tasks. If RTX-51 reports an error, the mailbox    */
/* number is set in telemetry. The reason is probably the following:  */
/* -Specified mailbox does not exist(wrong mailbox parameter).        */
/* Send_ISR_Mail is made a macro instead of a function to avoid using */
/* reentrant functions from interrupt routines.                       */
/* Users of Send_ISR_Mail must have access to telemetry_data.         */

#endif
```

8.8 ttc_ctrl.h

```
/*-----------------------------------------------------------------------------
/*                                                                           --
/*    Copyright (C) 1998 : Space Systems Finland Ltd.                        --
/*                                                                           --
/*-----------------------------------------------------------------------------
/*
/*    System Name:   DEBIE DPU SW
/*
/*      $Workfile:   ttc_ctrl.h  $
/*        $Author:   holsti  $
/*          $Date:   Sun May 16 09:20:10 1999  $
/*      $Revision:   1.11  $
/*           $Log:   /home/share/fileserver/projects/debie/design/code/DHI/PVCS/
ttc_ctrl.h,p  $
 *
 *    Rev 1.11   Sun May 16 09:20:10 1999   holsti
 * SMR_204: RTX interface portable and no calls from interrupts.
 * SNCR_231:
 *
 *    Rev 1.10   Wed May 12 13:37:04 1999   paloheim
 * SMR: none, line length edited
 * SNCR: none
 *
 *    Rev 1.9   Mon Jan 18 11:50:30 1999   paloheim
 * SMR_114: Problems due to multiple inclusion prevented
 * SNCR_118
 *
 *    Rev 1.8   Wed Dec 02 12:02:26 1998   ville
 * SMR_098: Minimum TC interval implemented
 * SNCR_103
 *
 *    Rev 1.7   Mon Sep 07 16:37:00 1998   ville
 * SMR_055: Correct parameter for ResetInterruptMask used
 * SNCR_057
 *
 *    Rev 1.6   Sun Sep 06 18:58:50 1998   ville
 * SMR_051: TM interrupt enabled and disabled in TC ISR
 * SNCR_053
 *
 *    Rev 1.5   Fri Aug 21 15:36:16 1998   ville
 * SMR_035: Correct interrupt types set
 * SNCR_035
 *
 *    Rev 1.4   Thu Aug 20 15:38:56 1998   ville
 * SMR_033: Interrupt flags cleared in ISRs
 * SNCR_034
 *
 *    Rev 1.3   Thu Aug 06 16:25:36 1998   ville
 * SMR_013: Macros redefined
 * SNCR_014
 *
 *    Rev 1.2   Thu Aug 06 10:12:54 1998   ville
 * SMR_010: TM interrupt flag cleared
```

```
 * SNCR_010
 *
 *    Rev 1.1   Tue Aug 04 11:26:58 1998   ville
 * SMR_005: Overlapping definitions removed
 * SNCR_005
 *
 *    Rev 1.0   Thu Jul 02 10:38:20 1998   ville
 * Initial revision.
/*
/*- * ------------------------------------------------------------------------*/

/* Macros and function prototypes for handling Telecommand and Telemetry     */
/* interface.                                                                */

/* Part of : DHI */

#ifndef TTC_CTRL_H
#define TTC_CTRL_H

#ifndef RTX51_DEFINED
   #include "RTX51.h"
   #defined RTX51_DEFINED
#endif

#ifndef REG52_DEFINED
   #include
   #define REG52_DEFINED
#endif

#ifndef ABS_ADD_DEFINED
   #include
   #define ABS_ACC_DEFINED
#endif

/* TC and TM register handling */

#define READ_TC_MSB XBYTE[0xFF10]
#define READ_TC_LSB XBYTE[0xFF00]

#define WRITE_TM_LSB(TM_LSB) XBYTE[0xFF80] = (TM_LSB)
#define WRITE_TM_MSB(TM_MSB) XBYTE[0xFF90] = (TM_MSB)

/* TM Interrupt flag */

#define CLEAR_TM_INTERRUPT_FLAG IE1 = 0

/* TC Interrupt flag*/
#define CLEAR_TC_INTERRUPT_FLAG IE0 = 0

/*TM and TC interrupt controls*/

#define SET_INT_TYPE1_EDGE IT1 = 1
#define SET_INT_TYPE0_EDGE IT0 = 1
```

```
/*TM interrupt service handling*/
#define TM_ISR_MASK  0x04

/* Error Status register bits concerning TM/TC interface */

#define PARITY_ERROR 2
#define TC_ERROR     1

#define TC_OR_PARITY_ERROR (TC_ERROR + PARITY_ERROR)

#define TC_INT_MIN_INTERVAL (10000000 / 1085)
/* Minimum interval between two consecutive telecommands */
/* in machine cycles : 10000000 ns  / (1085 ns / cycle)  */

#define TC_TIMER_LSB  ((0xFFFF - TC_INT_MIN_INTERVAL) & 0xFF)
#define TC_TIMER_MSB  ((0xFFFF - TC_INT_MIN_INTERVAL)  8)
/* MSB and LSB of the initial value for TC interval timer */

#define SET_TC_TIMER_MODE  TMOD = (TMOD & 0xF0) | 0x01
/* Set TC timer (0) mode : Mode 1, counter operation, SW control */

#define INIT_TC_TIMER_MSB  TH0 = TC_TIMER_MSB
#define INIT_TC_TIMER_LSB  TL0 = TC_TIMER_LSB
/* TC timer initialization macros */

#define START_TC_TIMER     TR0 = 1
#define STOP_TC_TIMER      TR0 = 0
/* TC timer run control macros    */

#endif
```

DISTRIBUTION LIST

Quantity	Person and Organisation
1	Juha Kuitunen, Patria Finavitec Systems