DOCUMENT TYPE: USER MANUAL

TITLE:

DEBIE DPU SW
USER MANUAL

	FUNCTION	NAME	DATE	SIGNATURE
PREPARED BY	Design engineer	Ville Sipinen	11.2.2000	
CHECKED BY	PA engineer	Niklas Holsti		
APPROVED BY	Manager, SW	Fredrick von Schoultz		

DOCUMENT IDENTIFICATION	WBS Nr NONE	KEYWORDS DEBIE, DPU

DOCUMENT STATUS SHEET

Issue	Date	Modified sections	Reason for change / Comments
1.0	8.9.1998		Initial issue.
1.1	3.2.1999	See below	Updated for DEBIE CDR
		1.1	This issue applies to the Flight SW
		1.3	Deleted obsolete section (former sections 1.4, 1.5 and 1.6 became now sections 1.3, 1.4 and 1.5)
		2.1	Added description of the SW environment.
		2.2.3	Added Self Test pulse level setting and pulse generation to the list of Sensor Unit interfaces.
		3.1	Added the description of the program RAM memory test and the possible effects of it. Deleted obsolete text about Prototype SW.
		3.3.2	Deleted obsolete text about Prototype SW. Added a bullet explaining TC timing restrictions in case of Self Test TC
		3.3.3	Added note about executing telecommand that aborts TM register telemetry. Updated the description of handling of received telecommands during the Science telemetry.
		3.3.4	Updated the section to correspond to the Flight SW.
		3.3.5	Too frequently sent telecommands are now rejected.
		3.3.6	Deleted obsolete section.
		3.4.1	The HK and Operational Telemetry register non-conformance table is replaced with table of SSF defined TM registers

DOCUMENT STATUS SHEET

Issue	Date	Modified sections	Reason for change / Comments
		3.4.2	Added note that the first two bytes contain the length of the Science Telemetry. The description of the Prototype SW is replaced with description of buffering Hit Events during Science Telemetry.
		3.4.3, 3.5.1, 3.5.2, 3.5.3 3.5.4 and 3.6	Deleted obsolete text about Prototype SW.
		3.5.5	Added new section about Time Update
		3.7	Deleted obsolete text about Prototype SW. Added description of the Sensor Unit Self Test operation, limitations and duration.
		4.1	Deleted obsolete text about Prototype SW.
		5.3	Added information about order of files in the compiler source file list.
		5.4	All source files are compiled with ‘large’ memory model
		5.6	Deleted obsolete text about Prototype SW and incorrect note about data initializations after soft reset.
		6	Deleted the table of Prototype SW Usage Constraints. Added constraints about Sensor Unit Self Test and Sensor Unit states and telecommands.
1.2	31.8.1999	See below. *Note that if some section is added, the following section numbers refer to the section numbers in the old document version.*	Updated for Flight SW delivery
		1.5	No special conventions are used.

DOCUMENT STATUS SHEET

Issue	Date	Modified sections	Reason for change / Comments
		2.2.3	Replaced Plasma1+ with Plasma1e and Plasma1- with Plasma1i. Corrected the type of reset caused by Watchdog counter (software reset replaced by hardware reset). Added description of EventFlag signal.
		3.1	Corrected the text concerning the behaviour after a code RAM memory error is found and added constraint about sending telecommands after power-up reset
		3.3.2	Added new section about immediate TC responses.
		3.3.2	Added constraint about sending StartAcquisition at least 2 seconds after Switch SU# On telecommand.
		3.3.2	Added constraint about sending telecommands after Soft Reset TC
		3.3.4	Added remarks about Write Data Memory and Write Code Memory telecommands
		3.3.5	Maximum telecommand rates defined.
		3.4.1	Deleted table of telemetry data registers defined by SSF (they are now included in [AD2]). Added remark about updating telemetry time registers.
		3.5.1	Delays between reset and first voltage monitorings are defined.
		3.5.2	Delays between reset and voltage and temperature measurements are defined
		3.5.3	Delay between reset and first code checksum check is defined

DOCUMENT STATUS SHEET

Issue	Date	Modified sections	Reason for change / Comments
		3.5.4	Added remark about actions after watchdog alarm.
		3.5.5	The accuracy of the DEBIE clock counter is defined
		3.7	Added restriction about not using Write Code Memory telecommand during SU Self Test sequence. Added remark about Hit Budget. SU Self Test is now delegated to Health Monitoring task.
		3.8	Added section about detector dead-time
		4	Also usage of ‘Write Code Memory’ and ‘Write Data Memory’ is briefly discussed
		4.1	Added the name of the map file of the Keil linker (debie.m51)
		4.2	Added new section about usage of ‘Write Code Memory’ and ‘Write Data Memory’.
		5.1	Delivery Media is defined
		5.3	Added ‘startup.a51’ and ‘rtxconf.a51’ to list of Keil source files used.
		5.5	Added ‘no overlay’ option to the list of linker options
		5.6	Added new section about updating the reference code checksum
		5.6	Added remark that all unassigned program bytes must be set to 255
		6	Added new constraints to the table
1.3	11.2.2000		Updates for Flight SW 1.4
		3.5.5	Clock precision updated for SW modifications (SMR 257, SMR 258) and test results.

DOCUMENT STATUS SHEET

Issue	Date	Modified sections	Reason for change / Comments
		3.3.7 6	Section added on telecommand-specific constraints (SNCR 175, SNCR 207)
		3.5.6	Section “Periodic Activity Diagram” added.

TABLE OF CONTENTS

TABLE OF CONTENTS

1 INTRODUCTION

1.1 Scope

This document describes how to use the DEBIE DPU Software. The purpose of this user manual is to:

- describe the context in which the DEBIE DPU software operates,
- describe the inputs and outputs of the DEBIE DPU software,
- explain how to configure the DEBIE DPU software,
- explain how to compile the DEBIE DPU software and integrate it with other DEBIE software and hardware components to form a working system,
- define the operational usage constraints of the DEBIE DPU software.

The term "user" here means a person tasked with configuring and integrating the DEBIE DPU software as a working part of DEBIE. This manual is not addressed directly to the operators responsible for commanding DEBIE, nor to persons analyzing the scientific data produced by DEBIE. However, some of the information in this manual is relevant to operators and analysts, particularily the operational constraints imposed by DEBIE DPU software design and implementation.

This document is written with the assumption that the DEBIE DPU software conforms to the SW requirements and the various Interface Control Documents. Nonconformances are detailed separately in the nonconformance list accompanying each DEBIE DPU software delivery. Nevertheless, if the nonconformance produces an operational constraint and is intended to remain in the final DEBIE DPU software delivery, then the constraint is listed in this document also.

The present issue of this document applies to the Flight SW version.

1.2 Overview

Chapter 2 gives an overview of the environment for the DEBIE DPU software and the system architecture.

Chapter 3 describes the DEBIE DPU software input and output interfaces: telecommands, telemetry, and control of the DEBIE instrument itself. Most of the detail is elided by referring to other applicable and reference documents.

Chapter 4 suggests ways to monitor and observe the operation of the DEBIE DPU software using the DEBIE DPU software itself, and how this can be used to troubleshoot problems.

Chapter 5 contains the instructions for system building, including distribution media handling, directory and file structure, and DEBIE DPU software configuration and parametrization.

Chapter 6 summarizes the operational constraints and indicates the sections of this document that explain each constraint in detail.

1.3 References

1.3.1 Applicable Documents

[AD1] DEBIE Requirements Specification
DEB-FIN-RS-001

[AD2] DEBIE TM/TC Interface Control Document
DEB-FIN-IC-001

[AD3] DEBIE HW/SW Interface Control Document
DEB-FIN-IC-002

1.3.2 Reference Documents

[RD1] DEBIE DPU SW Software Requirements Document
DEB-SSF-RS-001

1.4 Acronyms and Abbreviations

ADC	Analog to Digital Converter
ANSI	American National Standards Institute
DAC	Digital to Analog Converter
DC	Direct Current
DEBIE	DEBris In-orbit Evaluator
DPU	Data Processing Unit
ICD	Interface Control Document
LSB	Least Significant Byte
MSB	Most Significant Byte
PROM	Programmable Read-Only Memory
SFR	Special Function Register
SRAM	Static Random Access Memory
SSF	Space Systems Finland
SU	Sensor Unit
SW	Software

TBA	To Be Added
TBC	To Be Confirmed
TBD	To Be Defined
TBS	To Be Specified
TC	TeleCommand
TM	TeleMetry

1.5 Conventions

No special conventions are used.

2 SYSTEM OVERVIEW

2.1 Environment

The DEBIE DPU SW runs in 80C32 microcontroller of the data processing unit of the DEBIE instrument.

The data processing unit is connected to the spacecraft and up to 4 Sensor Units via interfaces described in the section 2.2.3.

2.2 System Architecture

The following is an overview of the DEBIE DPU software architecture and design.

2.2.1 Functions and Tasks

The DEBIE DPU software has three basic functions: first, to interface with ground control via the telecommand and telemetry streams, second, to perform particle hit measurements triggered by hit trigger interrupts, and third to perform housekeeping measurement and monitor the health of the instrument.

To fulfil these functions, the DEBIE DPU software is divided into a number of tasks (processing threads).

The tasks or concurrent processing threads in DEBIE DPU Software are

- the *Telecommand Execution Task* for executing ground commands.
- the *Health Monitoring Task* for performing housekeeping measurements and monitoring the health of the instrument.
- the *Hit Trigger ISR Task* for measuring the peak detector outputs of the triggering Sensor Unit. This task is attached directly to the hit trigger interrupt.
- the *Acquisition Task* for performing other data acquisitions for the particle hit event, classification of the event and storing of the event data.

Telemetry gathering is performed by *TM Interrupt Service.*

2.2.2 Data Structures

The main data structures in the DEBIE DPU Software are the following:

- the *“telemetry_data”* that stores housekeeping and operational telemetry data.
- the *“science_data”* that records particle hit events as they occur and holds separate counters of each class of events for each Sensor Unit.

The formats of these data structures are defined in the TM/TC ICD [AD2].

There are obviously many other data structures that are internal and of no concern to the DEBIE DPU Software user.

2.2.3 Interfaces

The interfaces from the DEBIE DPU to the Spacecraft on the one hand, and to the DEBIE Sensor Units on the other hand, are the following. This is an exhaustive list. See [AD2] for details of the spacecraft TM/TC interface and [AD3] for details of the Sensor Unit interface.

Spacecraft interfaces

- DPU power supply. When the power is turned on, the DEBIE sofware executes a hard reset.
- the TC Interface, telecommands sent in serial form by the spacecraft are converted to parallel form by the hardware, and an external interrupt INT0 is generated to the microcontroller when the telecommand is ready in the TC receive register.
- the TM Interface, telemetry words written in the parallel form by the DEBIE DPU Software to the telemetry transmit register are converted to the serial form by the hardware, and an external interrupt INT1 is generated to the microcontroller when the Spacecraft has read the telemetry word.

Sensor Unit interfaces

- Sensor Unit power control. The software can turn each Sensor Unit on or off individually.
- Voltage status signal and register indicates if any (or several) of the supply voltages is in short circuit or otherwise overloading. One signal directly connected to a microcontroller port pin indicates the state of 5V DC/DC converter and a memory mapped register indicates the state of the Sensor Unit high voltages.
- Trigger threshold settings. The software controls a DAC to adjust the triggering sensitivity of each of the three trigger channels in each Sensor Unit.
- DEBIE DPU receives three hit trigger signals from each Sensor Unit. They are OR’ed by hardware to form a single interrupt signal to the microcontroller. The Sensor Unit that is the source of the hit trigger can be determined by two signals connected directly to two microcontroller port pins.
- One signal (EventFlag) connected directly to a microcontroller port pin determines if any Hit Triggers are allowed to cause a particle hit event.
- The A/D-converter Interface. Analog peak detector outputs, Sensor Unit temperatures and secondary supply voltages can be measured with the A/D-converter. The software can select measurement channels, start conversions and see the state of the converter via memory mapped registers.

- The Rise Time and Delay counters. The Rise Time counter measures the rise time of the Plasma1e sensor from the trigger signal to the signal peak. Two delay counters measures the relative timing from Plasma1e and Plasma1i triggers to the Piezo1/2 trigger. The hardware takes care of these measurements and the software reads the results and stores them to the science data.
- Sensor Unit Self Test pulse level setting and pulse generation are needed for performing the Sensor Unit Self Test measurements (see [AD1] and [AD3]).

DPU interfaces

- The watchdog. When the watchdog timer is armed, the software must reset it periodically, otherwise the watchdog triggers a hardware reset.

3 OPERATION

This chapter describes the use and operation of the DEBIE DPU Software as controlled and observed across the interfaces listed in section 2.2.3. The internal functions of the software are generally not described.

The main outputs for observing the operation of the DEBIE DPU Software are the housekeeping, operational and science data telemetry. The reader is assumed to be familiar with the description and content of the telemetry formats as defined in the TM/TC ICD [AD2].

3.1 Initialization

The DEBIE DPU Software is initialized automatically after power-up or any other reset. The cause of the reset has effects on actions that are performed during initializations. See [AD1] and [RD1] for details.

At the start of the initialization when the program is still executed from the PROM memory, the RAM memory intended to be used as program memory is tested. If any failures are found, a corresponding error indicator is set (see [AD2]).

After the program memory test, the software copies itself from the PROM memory to the SRAM memory, and if no failures were found in the memory test, switches the memory mapping to run code from the SRAM. If some failure was found the program execution continues from the PROM.

Although the user could try to localize the memory failure and compensate it by patching the program, it is currently not possible to make the program run from SRAM in this kind of situation, because before the switch is attempted either the whole code RAM area is checked and the error is found again or the previous failed test result is used. In both cases the software refuses to switch program execution to code SRAM.

During initialization, the DEBIE DPU Software performs the DPU Self Test sequence. During this sequence the DEBIE DPU Software is in DPU Self Test mode and after the sequence is completed the Standby mode is entered automatically.

The execution of the complete boot and initialization sequences after a power up reset takes about 3.9 seconds. Before that no telecommands should be sent to the DEBIE.

3.2 Modes and Mode Transitions

There are two active modes where DEBIE DPU Software can be after initialization: Standby and Acquisition. Transitions between these modes are triggered with specific telecommands, “Start Acquisition” and “Stop Acquisition”. The main difference between these two modes is that normal particle hit measurements are performed only in the Acquisition mode. Before entering the Acquisition mode one or more Sensor Units have to be switched to the On state with specific telecommands (see [AD2] for details). Only those Sensor Units which are in the On state can detect particle hits.

3.2.1 Housekeeping and Operational Telemetry

DEBIE DPU Software stores the information about current mode in the “Mode Status” telemetry data register. (By “register” is here meant a program variable which is accessible through telemetry, not a hardware register.) This register is located in the “telemetry_data” data structure described in section 2.2.2 on page 4, and it is one of the Housekeeping and Operational telemetry data registers that can be read with “Send Status Register” telecommand. See [AD2] for details.

3.2.2 Operational Constraints

At least one of the Sensor Units should be in the On state before “Acquisition” mode is entered.

3.3 Telecommand Processing

The received telecommands can be read from the TC receive register which is divided into 8 bit MSB and LSB memory mapped registers. After the TC interrupt signal is received these registers are read and the contents of them checked and actions taken according to the validity and content of the received telecommand.

3.3.1 Telecommand Format

The validity of the telecommands depends on the “TC addresses”, “TC codes” and parity bits as defined in the [AD2]. If any one of the above is invalid, the received telecommand is rejected and it is notified as required in [AD2]. The information about the last received telecommand and its validity are stored in the “telemetry_data” data structure described in section 2.2.2 on page 4.

3.3.2 Immediate Telecommand Responses

The immediate response to a received telecommand is written to TM HW registers within 1 ms from the reception of the telecommand, unless it is received too soon after previous one (see section 3.3.3) or the Science Telemetry is being gathered.

The content of this response follows the specifications in [AD2] with the following exceptions:

- The specifications require that the response to the Clear Error Status TC should show the effect of the telecommand, but currently it can be seen only after the execution of it is completed.

- If StartAcquisition and StopAcquisition telecommands are sent in wrong DEBIE mode, the rejection of those telecommands is shown in the telemetry registers only after the handling of the telecommand is completed.
- If Switch SU On/Off/Self Test telecommands are sent when the state of the SU is not suitable for the telecommand the rejection of the telecommand is shown in the telemetry registers only after the handling of the telecommand is completed.
- If Write Code Memory or Write Data Memory telecommand is sent in wrong DEBIE mode, the rejection of the telecommand is shown in the telemetry registers only after the handling of the telecommand is completed.

In general if the telecommand is rejected for any other reason than invalid format, then the rejection is shown in the telemetry registers only after the handling of the telecommand is completed.

3.3.3 Too Fast Repeated Telecommands

The DEBIE DPU Software will not execute telecommands which are sent less than 10 ms after previous one.

- SU Self Test TC

  These telecommands should be sent at least 12 seconds apart in order to avoid TC Error messages.

- StartAcquisition

  StartAcquisition TC should not be sent less than 2 seconds after sending Switch SU# On TC, because it takes two seconds for the SU to reach the “On” state. If the StartAcquisition is sent earlier, the SU will not be activated for event detection.

- Soft Reset

  The complete execution of the Soft Reset telecommand takes currently about 2.4 seconds. During that time no telecommands should be sent to the DEBIE.

3.3.4 Telecommands during Telemetry

The DEBIE DPU Software will not execute any telecommands during telemetry gathering. If the telemetry gathering is triggered by “Send Status Register” telecommand, the telemetry session will stop when the next telecommand is received and the telecommand is then executed.

If a telecommand sent during Science Data telemetry gathering or ReadData memory, it will be totally ignored: its format will not be checked and it will not be executed.

3.3.5 Telecommands vs Mode

The “Start Acquisition” and “Stop Acquisition” telecommands are mode-sensitive: the former is accepted only in Standby mode, and the latter only in Acquisition mode. If the mode is incorrect, the “TC ERROR” bit in the Error Status register is set.

Also “Write Data Memory” and “Write Code Memory” telecommands are mode sensitive: both are accepted only in Standby mode. The latter has an exceptionally long interrupt blocking period and therefore that telecommand should never be used when some interrupt processing expected.

3.3.6 Command Execution Rate

The maximum telecommand rate which can be maintained without disturbing the timing of the Health Monitoring task is defined with the following scenario:

During one second N telecommands are received and handled. Telecommands are sent 10 ms apart. During those 10 ms periods where a telecommand is sent, seven TM interrupts are received (this limit is caused by interference between TC and TM interrupts and minimum time between two TM interrupts), and during those 10 ms periods where telecommands are not sent, 8 TM interrupts are received.

Two cases are considered concerning the Hit trigger processing. In the one the Science Data is full and in the other it is not. In the first one the Sensor Unit self test cannot be active (see section 3.7) and in the other it is. In both cases one Hit trigger processing per second is considered (in addition to the SU Self Test events in the latter case).

The available time for Health Monitoring task is the one second minus the sum of the execution time of the above actions.

The Margin for the Health Monitoring task is the available time minus the maximum execution time of the Health Monitoring task during one Health Monitoring period (one second).

The timing tests and analysis calculations gives the following result:

- When Science Data is not full and Sensor Unit self test is considered, up to 29 telecommands can be executed per second.
- When Science Data is full and Sensor Unit self test is forbidden, up to 28 telecommands can be executed per second.

3.3.7 Specific Constraints per Telecommand

The following table lists the specific constraints and other noteworthy information not found in [AD2] per telecommand. The order of telecommands follows [AD2].

Table 1: Telecommand Constraints

Telecommand	Constraints	Remarks
Start Acquisition	This TC should not be sent less than 2 seconds after a Switch SU# ON.	See section 3.3.3.
Clear Error Status	The effect of this TC is not yet visible in the Error Status and Mode Status register values returned immediately (within 1 ms) after the TC, but only after the TC is executed (within 10 ms after TC reception).	See section 3.3.2 and SNCR 207, RFW 002.
Soft Reset	User must wait at least 2.4 seconds before sending the next TC.	See section 3.3.3.
Write Program Memory MSB/LSB	These TCs should not be used when fast interrupt response is desired.	See section 3.3.5.
Switch SU# ON/OFF	For ON, the TC should be sent at least 2 seconds before Start Acquisition.	See section 3.3.3.
	For SELF TEST, successive TC's should be spaced by at least 12 seconds.	
	SELF TEST must not be started when the Science Data area is full.	See section 3.7.
Set SU# Plasma 1e to 1i Max Time	The values 129 .. 255 of the parameter (TC Code) are equivalent to the value 128, because the measured delay is saturated to the range -128 .. 127 before it is compared to the limit set by the TC.	

3.4 Telemetry Gathering

3.4.1 Housekeeping and Operational Telemetry

The Housekeeping and Operational Telemetry can be gathered with the "Send Status Register" telecommand. This telecommand defines the first two telemetry data registers from the "telemetry_data" data structure which are to be sent. After these, the following telemetry data registers are sent two bytes at a time when the TM interrupt signal is received, until the next telecommand is received. See [AD2] and [RD1] for details.

The telemetry registers containing the DEBIE time are updated when the Send Register TM telecommand is executed and also when the MSB bytes of those registers are written to the TM HW registers.

3.4.2 Science Telemetry

The Science Data can be gathered with the “Send Science Data File” telecommand. After this telecommand the contents of the whole “science_data” data structure are sent. The first two bytes containing the length of the Science Telemetry block in 16-bit words are sent immediately and then further bytes are sent after each TM interrupt signal is received, until all the science data area is sent.

No telecommands should be sent during science telemetry transmission. See section 3.3.4 on page 9.

When all of the Science Data have been sent, the Science Data structure is erased and all event counters are reset to zero.

If a particle hit occurs during the Science Data telemetry, it is buffered and entered in the Science Data File after the telemetry is completed. The buffer can hold up to 10 event records and any records beyond that are written over the previously last recorded event.

3.4.3 Data Memory Dump

The external data memory can be read with “Read Data Memory MSB” and “Read Data Memory LSB” commands. These commands define the start address of a 32-byte memory area to be read as defined in [AD2]. The memory area is sent two bytes at the time in the “Data Memory Read TM Sequence” defined in [AD2]. The next two bytes are sent after the TM interrupt signal is received until the whole sequence is completed.

No telecommands should be sent during data memory dump transmission. See section 3.3.4 on page 9.

3.5 Periodic Activities

3.5.1 Supply Voltage Monitoring

The voltage status signal and register (see section 2.2.3 on page 5) are monitored once in 10 seconds. If a failure is detected, the failed voltage or voltages are switched off.

The first monitoring action for voltage status signal occurs 8.9 seconds after software reset. Corresponding time for voltage status register monitoring is 9.9 seconds.

3.5.2 Voltage and Temperature Measurements

The supply voltages are measured once in 180 seconds and Sensor Unit temperatures once in 60 seconds. The results are stored in the “telemetry_data” structure described in section 2.2.2 on page 4. If any of the Sensor Unit temperatures is out of the limits, these Sensor Units are switched off.

The first supply voltage measurement occurs 118 seconds after software reset. The first Sensor Unit temperature measurement occurs 24 seconds after software reset.

3.5.3 Memory Checksum

The checksum of the program memory is calculated once in 60 seconds. If a failure is detected, a soft reset is performed. See [AD1] and [RD1] for details.

The first checksum check starts 3.9 seconds after software reset and 5.4 seconds after power-up reset. The difference is caused by initialization of DEBIE data structures which is done only after power-up reset.

3.5.4 Watchdog Reset

The watchdog timer is reset every 10 seconds. If a watchdog alarm occurs, watchdog error counter is increased and the software is reset.

3.5.5 Time Update

Because the DEBIE does not have any HW clock, the DEBIE DPU SW has to implement a software clock. This clock is a 32-bit counter which is updated periodically by the Health Monitoring task. The accuracy of the clock depends on the timing accuracy of the Health Monitoring task which is activated by the RTX real time kernel once in every second.

The RTX uses one of the processor’s internal timers for timing. The accuracy of that timer depends on the accuracy of the processor’s clock signal and its handling in RTX. The processor clock frequency is nominally 11.0592 MHz. Each instruction cycle is 12 clock cycles, thus 9216 instruction cycles are exactly 10 ms which is the desired length of one RTX time slice, the basic timing unit of RTX services. However, RTX does not use the “auto-reload” function of the timer, but spends some time in each slice on reloading and restarting the timer. This delay has been empirically measured as 18 instruction cycles, and so the slice is defined as 9198 cycles.

Under no load (no events or interrupts other than the timer) the DPU software clock is correct to better than 1 s over 24 h. Under load (hit events and TM/TC) the RTX timing will run slow, because the other interrupts will block and delay the reloading and restarting of the timer. However, under moderate load the error is small; for example the error is still less than 1 s / 24 h when there are, per second, one hit event, two TC accesses and two TM accesses.

However because the Health Monitoring task is the lowest priority task of the DEBIE DPU software, the other tasks can have significant effect on the accuracy of the clock counter. Especially if the loads of the other tasks are very high, the Health Monitoring task could be totally blocked. This would however also prevent the resetting of the watchdog counter and eventually trigger the watchdog failure reboot of the DEBIE.

3.5.6 Periodic Activity Diagram

The following two tables illustrate the timing of the periodic activities. Table 2 lists the activities that are done systematically in each 10-second interval by 10 successive activations of the Health Monitoring task. Activities with a period up to 10 seconds are repeated as shown in this table. Activities with a period greater than 10 seconds are done incrementally, with each of the 10 successive activations dedicated to some activities as shown in this table (rightmost column).

Table 2: Periodic Activities for 10 Seconds

Time (s)	Activities repeated at 10 s	Activities with period > 10 s
0	Set Watchdog line LOW.	Temperature monitoring. Code checksumming.
1	Set Watchdog line HIGH.	Code checksumming.
2		SU Self Test monitoring of supply voltage and SU temperature. Code checksumming.
3		SU Self Test triggering and measuring. Code checksumming.
4		Voltage monitoring. Code checksumming.
5	Monitor low-voltage current (V_DOWN).	Code checksumming.
6	Monitor high-voltage current for SU 4.	Code checksumming.
7	Monitor high-voltage current for SU 3.	Code checksumming.
8	Monitor high-voltage current for SU 2.	Code checksumming.
9	Monitor high-voltage current for SU 1. Refresh the hit budget.	Code checksumming and monitoring.

Table 3 details the long-period actvities and is organised into 10 columns and 18+1 rows. Each cell corresponds to one activation of the Health Monitoring task. Thus, each row represents 10 seconds. During these 10 seconds, the DPU SW also repeats all the activities listed in the first table, but these are elided here. Successive rows correspond to successive 10-second periods. The code-checksumming activity is elided from the table for clarity; 1/60th part of the code is checksummed every second. SU Self Test activities are also elided; they are performed in 10-second phases 2 and 3 as soon as possible after the self-test TC.

The “Time” column in these tables shows the time from the first activation of the Health Monitoring task, after a SW reset is completed.

Table 3: Periodic Activities for 180 Seconds

Time (s)	0	1	2	3	4	5	6	7	8	9
0-9										
10-19										
20-29	Temp. mon. SU 4									
30-39	Temp. mon. SU 3									
40-49	Temp. mon. SU 2									
50-59	Temp. mon. SU 1									Code checksum mon.
60-69										
70-79										
80-89	Temp. mon. SU 4									
90-99	Temp. mon. SU 3									
100-109	Temp. mon. SU 2									
110-119	Temp. mon. SU 1				Volt. mon. SU -50 V					Code checksum mon.
120-129					Volt. mon. SU3/4 -5 V					
130-139					Volt. mon. SU1/2 -5 V					
140-149	Temp. mon. SU 4				Volt. mon. DPU +5 V					
150-159	Temp. mon. SU 3				Volt. mon. SU +50 V					
160-169	Temp. mon. SU 2				Volt. mon. SU3/4 +5 V					
170-179	Temp. mon. SU 1				Volt. mon. SU1/2 +5 V					Code checksum mon.
180-...	The above rows are repeated cyclically hereafter, every 180 seconds.									

3.6 Sensor Unit Control

There are several telecommands that can be used to configure the Sensor Units or setting parameters for event classification. With these commands the Sensor Unit states, trigger levels, classification levels, delay time windows and quality formula coefficients can be set. See [AD2] for details.

3.7 Particle Hit Acquisition

When DEBIE is in the Acquisition mode, it responds to “Hit Trigger” interrupts from the Sensor Unit electronics.

The amount of “Hit Triggers” accepted during one Health Monitoring period is limited in order to avoid the blockage of the monitoring actions because of too high “Hit Trigger” load. This is limit is currently 20 triggers per one period which is in average 2 triggers per second - but now all 20 triggers could occur during the first second and triggers would then be rejected during next 9 seconds.

The occurrence of the hit trigger interrupt depends on triggering thresholds that can be adjusted by telecommands.

For each hit trigger, the software determines which Sensor Unit is the source, reads out all the relevant data from the Sensor Unit electronics, and resets the electronics to prepare for the next hit.

If the HW signal “EvenFlag” (see section 2.2.3) allows the recording of the event, each hit event is classified into one of 10 classes by using signal thresholds and signal coincidences. Then, a quality number is computed for the event. Events are stored in the Science Data file according to their quality. If the file is full, a new event is discarded if its quality is lower than the lowest quality in the file, otherwise the new event replaces the oldest event of the lowest quality in the file.

All events are counted by Sensor Unit and class. The counts include also events that are not stored in the Science Data file.

If some Sensor Unit is being Self Tested, all hits are considered to be caused by the Self Test sequence performed for it and the events are classified with highest quality number. This happens even if there came a real hit trigger from some other Sensor Unit. Therefore if there exist any risk for real hit triggers during the Sensor Unit Self Test, the other Sensor Units should be switched Off before starting the Self Test. This will not however prevent the real Hit triggers from the Sensor Unit being Self Tested.

The total duration of the Sensor Unit Self Test depends mainly on the required delays for Peak Detector Reset and Delay Counter Reset signals. In the current implementation the Peak Detector Reset delay takes at least 2 x 10 ms and the Delay Counter Reset delay 10 ms (only 1 ms would be sufficient) giving total delay 30 ms per measured channel. When 5 channels are measured, the total delay from the reset signals is 150 ms. Of course the execution of Hit measurement and Event classification and storage procedures takes also some time, but that should have relatively small effect (TBC) unless the execution of these functions is blocked by other SW functions.

Because the execution of the Sensor Unit Self Test is delegated to the Health Monitoring Task, the DEBIE SW can handle other telecommands during the Sensor Unit Self Test. However some telecommands should not be executed in that time:

- Set Trigger Level: because the Sensor Unit Self Test sequence contains temporary manipulation of the Hit Trigger threshold levels, these should not be modifed for the Sensor Unit being Self Tested.
- Switch Sensor Unit Off: because this telecommand can switch the given Sensor Unit to Off state at any time, it should not be used for the Sensor Unit being Self Tested since that would abort the Self Test sequence.
- Switch Sensor Unit On: because switching a Sensor Unit On can cause some Hit Trigger signals, this telecommand should not be used during the Sensor Unit Self Test even for other Sensor Units.
- Write Code Memory: because this telecommand has exceptionally long interrupt blocking period, it should not be used during SU Self Test sequence.

Sensor Unit self tests must not be executed when the Science Data is full, because then the processing of one Hit trigger event takes 350 ms and during the self test sequence ten Hit triggers are processed and the total time required for the Hit trigger processing would greatly exceed time available for it.

3.8 Detector Dead-Time

The DEBIE software cannot accept new particle hit events before the previous one is processed. This causes the following dead-times for the particle hit detector:

- When the Science Data area is not full the processing of the Hit Trigger causes 9 ms dead-time.
- When the Science Data is full the processing of the Hit Trigger causes 350 ms dead-time.

4 TROUBLESHOOTING

This chapter suggests some ways to observe and monitor the operation of the DEBIE DPU Software, using the DEBIE functions themselves, in particular the "Read Data Memory" telecommand.

Also usage of "Write Code Memory" and "Write Data Memory" telecommands is briefly discussed.

4.1 Read Data Memory

The "Read Data Memory" telecommand can be used to investigate the state of variables and data structures which cannot be read with other kinds of telemetry. In order to use it, the addresses of the interesting variables and data structures have to be known. These addresses can be found from the map file (debie.m51) of the Keil linker.

4.2 Write Code and Data Memory

These telecommands are intended to be used for changing code or data memory, when some code modification is needed, error need to be corrected or searched.

When something should be added to the existing code, one should write the addition to some unsued part of the code memory. The used part of the memory can be found from the file 'debie.m51' which contains the memory map of the DEBIE software. The information can be found from the table called 'link map of module ... (DEBIE)' and the last line of it containing information about code memory. In that line there is described the start address and length of the entity occupying the last used code memory locations. The free code memory space starts after that and ends at 0x7FFF.

The 'Write Data Memory' telecommand can be used to change content of existing variables or setting initial values for some new variables used by new code written with 'Write Code Memory' telecommand. The addresses of the existing variables can be found from the file 'debie.m51' and the free data memory space from the same file in the similar way as the free code memory space. However the last 256 bytes (addresses 0xFF00 ... 0xFFFF) must not be used, because they are reserved for the memory mapped HW registers.

5 SYSTEM GENERATION

5.1 Delivery Media and Structure

The following items are delivered on CD-R disk:

- DEBIE source codes, program binary and hex-file in one zip-file
- Software User Manual in pdf-format
- Design Document in pdf-format
- Test Report in several files in pdf-format

The Software User Manual, Design Document and Test Report is sent also on paper by mail.

The Software Requirements Document, Software Development Plane and Software Verification Plan are not delivered since they are not updated for this deleivery.

5.2 Purchased Components

The DEBIE DPU Software relies on the Keil RTX51 real-time kernel, which is delivered as object code files (input to the linker) and some assembly-language files from which files named startup.a51, rtxconf.a51 and RTXSETUP.DCL are used. These file contains startup code and configuration information and are customised for DEBIE by SSF.

5.3 Source Files

RTX configuration file ‘rtxconf.a51’ and startup file ‘startup.a51’ have to be the last files in the list of the project’s source files.

5.4 Compiling

C-language files are compiled with the Keil C51 compiler with the following options:

- include debug code
- include extended debug information
- enable ANSI promotion rules
- include interrupt vectors in object
- interrupt interval 8
- interrupt offset 0
- optimization level 6: loop rotation
- memory model: “large”

- number of data pointers: 1 - standard 8051

Assembly-language files are compiled with the Keil A51 assembler with the following options:

- include debug information
- define 8051 SFRs
- macro processor: standard

5.5 Linking

The program is linked with Keil BL51 linker with the following options:

- include local symbols
- include public symbols
- include line numbers
- target options: RTX51
- no overlay

The options for the “make” tool are the following:

- run BL51 linker after compile
- automatic dependency checking
- assembler file extension: A51
- c file extension: C
- library file extension: LIB

Environment pathspecs:

- automatically determined

5.6 Updating the Reference Value of Code Checksum

The DEBIE software needs to know the expected checksum of the program memory. Any modifications of the software changes this value, and the following procecedure must be executed every time a modified program is rebuilt, to create a working DEBIE binary. This procedure ensures that the program memory checksum is initially zero. It may become non zero during execution of Write Program Memory telecommands, but the DEBIE software takes care of that itself and no actions concerning new code checksum reference value are required from the user.

- First after all the other code changes are done, change the value of CODE_CHECKSUM constant macro in ‘version.h’-file to zero.
- Rebuild the program and produce a hex file (debie.hex) for it.
- Execute the following command in an Unix shell:
  “dos2unix < debie.hex | hexor 255”
- Change CODE_CHECKSUM to the checksum reported by “hexor”.
- Rebuild the program and produce new hex file.
- Execute the above Unix command again.
- Verify that the result from the ‘hexor’ is zero.

5.7 Loading and Running the DEBIE DPU Software in the Keil dScope debugger

The DEBIE DPU Software executable code shall be loaded into the code memory of the DPU processor at the addresses assigned by the Keil linker. All unassigned program bytes shall be set to 255.

The data memory contents for a hard reset (eg. power-on reset) are arbitrary. All the used data memory is initialized by the DEBIE DPU Software.

6 USAGE CONSTRAINT SUMMARY

This section collects the usage constraints of the DEBIE DPU Software caused by the implementation (but in many cases indirectly enforced by the requirements).

The constraints are listed in the following table, with references to the sections of the present document that describe the constraint in more detail. Refer also to the table of telecommand constraints in section 3.3.7.

Table 4: Usage Constraints

Constraint	Section
Duration of boot and initialization after power up reset	3.1
Delayed responses to rejected telecommands	3.3.2
Minimum delay between SU Self Test telecommands	3.3.3
Minimum delay between Switch SU# On and StartAcquisition	3.3.3
Minimum delay between Soft Reset and any other telecommand	3.3.3
Write Code Memory not to be used when interrupt processing is expected	3.3.5
Maximum telecommand execution rates	3.3.6
Sensor Unit Self Test sequence and Sensor Unit states	3.7
Sensor Unit Self Test sequence and telecommands	3.7
No SU Self Test, when Science Data is full	3.7
Detector dead-time	3.8

DISTRIBUTION LIST

Quantity	Person and Organisation
1	Juha Kuitunen, Patria Finavitec Systems