SPIRE-IFS-PRJ-001029

CARLO GAVAZZI

CARLO GAVAZZI SPACE SpA

HSO/FIRST DPU

DOCUMENT TYPE: Design **DRD :**

TITLE: HSO/FIRST DPU – BSW SOFTWARE ARCHITECTURAL DESIGN DOCUMENT

DOCUMENT No: DPU-AD-CGS-001 **PAGE:** 1 OF 60

ISSUE No: 1 **DATE:** 18 July’ 2001

PREPARED BY: A. BERTOLI

APPROVED BY:

THIS DOCUMENT IS SUBJECT TO THE APPROVAL OF:

	YES	NO		
Product Assurance	X		Approval Signature	Date
Configuration Control	X		Approval Signature	Date
Project Control		X	Approval Signature	Date
Engineering	X		Approval Signature	Date
Managing Director		X	Approval Signature	Date
CGS Program Manager	X		Approval Signature	Date

	YES	NO		
Higher Level Contractor	X		Approval Signature	Date
Customer		X	Approval Signature	Date

Data Management

Signature Date

DOCUMENT CHANGE RECORD

ISSUE	DATE	CHANGE AUTHORITY	PAGES AFFECTED	REMARKS
1	July 18, 2001			

LIST OF VALID PAGES

PAGE	ISSUE N°	PAGE	ISSUE N°	PAGE	ISSUE N°	PAGE	ISSUE N°	PAGE	ISSUE N°
1- 60	1								

TABLE OF CONTENTS

INDEX OF TABLES

INDEX OF FIGURES

1. INTRODUCTION

1.1 SCOPE

The purpose of this document is to specify the Architectural and Detailed Design of the HSO/FIRST DPU on-board Software, named "*DPU Basic S/W*", which has to be developed in the frame of the HSO/FIRST DPU project. The *DPU Basic S/W* is composed of two S/W Items as follow;

- *DPU Boot Software*
- *DPU Board Drivers*.

The *Boot Software* (Power On procedure) is a stand-alone software in charge of uploading on the Program Memory board the Application Software;
The *DPU Board Drivers* are software modules that will be linked with the Application Program developed by the user.
That S/W items will be integral part of the HSO/FIRST DPU board. The picture Figure 1-1 shows the Basic S/W structure.
The software design has been derived from the "HSO/FIRST DPU Software Requirements Document" [AD 11].
The document is prepared and maintained by the CGSPACE-HSO/FIRST DPU software developer team.

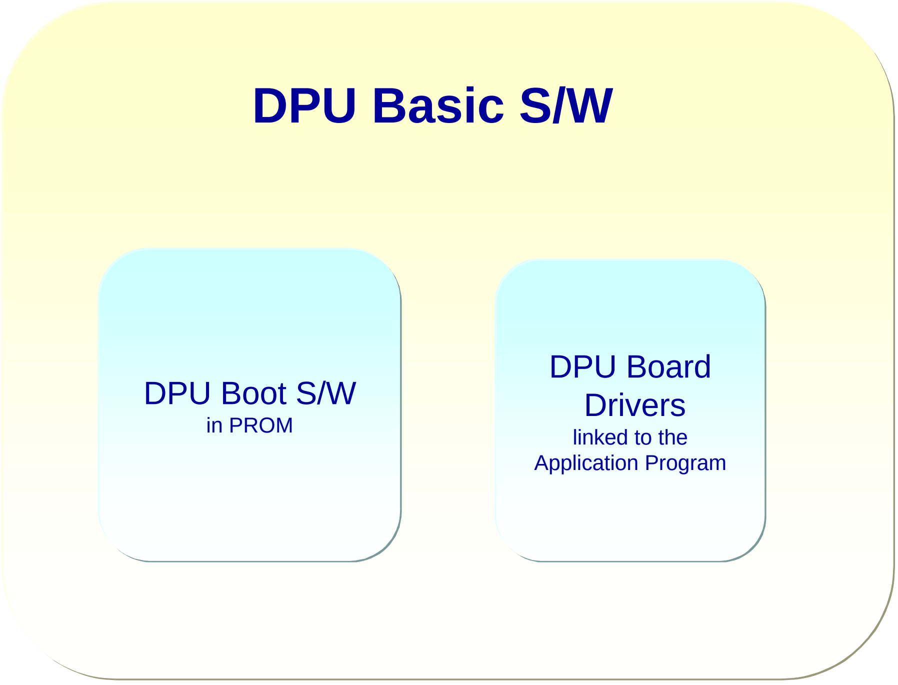

Figure 1-1: DPU Basic S/W structure

1.2 DOCUMENT CONTENT

The DPU Basic S/W includes two distinct S/W Items as defined in the previous paragraph. The DPU Boot S/W is a limited and low complex program; it has to be located in 32Kbyte of PROM memory board. The DPU Board Driver S/W are only simple modules and for them exist a one to one correspondence with the specification and the design. Considering the low complexity of the subject programs, in the SRD have been specified the requirements in natural language only, and the Structured specification are directly inserted in the Architectural Design Document for the DPU Boot S/W only. The DPU Board Driver don't require Structured Specification and Architectural Design Diagram because they are composed of one S/W Components for each type of driver.
The document contains the Diagrams followings:

- DPU Boot Software
  - ✓ Data Flow Diagram,
  - ✓ Control Flow Diagram
  - ✓ Process Specification (PSPEC)
  - ✓ Control Specification (CSPEC),
  - ✓ Architectural Context Diagram (ACD),
  - ✓ Architectural Flow Diagram
  - ✓ Architectural Interconession Diagram,
  - ✓ Matrix Allocation
  - ✓ S/W Modules Design.
  - ✓ Code (TBW)

- DPU Boot Software
  - ✓ S/W Modules description.
    - ✓ IEEE1355 Driver Module Design
    - ✓ MIL-STD-1553 Driver Module Design
    - ✓ EEPROM Driver Module Design
    - ✓ Watchdog Driver Module Design
  - ✓ Code (TBW)

The DFD, CFD, PSPEC, CSPEC, ACD, AFD, AID are compliant to the Tom De-Marco, Derek j. Hatley , Imtiaz A. Pirbhai as defined in [RD-1].
.

1.3 DEFINITIONS ACRONYMS AND ABBREVIATIONS

ADD	Architectural Design Document
ASW	Application Software
CNR	Consiglio Nazionale Ricerche
CPU	Control Processing Unit
CDMS	Computer Data Management System
CGS	Carlo Gavazzi Space
COTS	Commercial Of The Shelf
CSPEC	Control Specification
DDD	Detailed Design Document
DFD	Data Flow Diagram
DM	Data memory
EEPROM	Electrical Erasable Programmable Read Only Memory
ESA	European Space Agency
HK	House-Keeping
HW	Hardware
IFSI	Istituto di Fisica dello Spazio Interplanetario
OBS	On board software
PAT	Process Activation Table
PM	Program Memory
PROM	Programmable Read Only Memory
PSPEC	Processor Specification
RAM	Random Access Memory
S/W	Software
SRD	Software Requirement Document
STD	State Transition Diagram
TBC	To be Confirmed
TBD	To be Defined

2. DOCUMENTS

2.1 APPLICATION DOCUMENT

AD #	Doc Number	Issue/Date		Rev /Date		Title
1	CNR.IFSI.2000TR01	1	Sep 15, 2000			Documento di Specifiche Tecniche per il Contratto delle Data Processing Units del Satellite First dell'ESA" IFSI
2	DPU-SP-CGS-001	2	Oct 30, 2000			FIRST CPU Board Specification
3	DPU-SP-CGS-002	1	Oct 12, 2000			Payload & Spacecraft Interface Board Specification
4			Nov 30, 1999			Technical proposal CGS (Ref. S9-030 November 99)
5						Allegato Tecnico al Contratti ASI
6	CNR.IFSI.2001TR02	1	April 5, 2001			Herschel Space Observatory DPU Applicable Documents Guidelines
7	SCI-PT-ICD-7527	1	Sep 1, 2000	0		Packet Structure - Interface Control Document
8	IFSI-OBS-PL-2000	1	Oct 13, 2000			DPU/ICU OBS PA Plan
9	IFSI-OBS-SP-2000-001	1		3	April 6, 2000	DPU/ICU OBS URD, On Board S/W User Requirement Document
10	CNR.IFSI.2001TR01	Draft3	March 21, 2001			DPU/ICU Switch-ON Procedure
11	DPU-SQ-CGS-001	2	July 12, 2001			HSO/FIRST Software Requirement Document

Table 2-1: Applicable Documents

2.2 REFERENCE DOCUMENT

RD #	Doc Number	Issue/Date		Rev /Date		Title
1	ESA PSS-05-0	2	Feb, 1991			ESA Software Engineering standards
2						Space Level MIL-STD-1553 BC/RT/MT Advanced Communication Engine Terminal BU-61582 Data Book ILC Data Device Corporation 1995
3						MIL-STD-1553 BC/RT/MT Advance Communication Engine (ACE) Data Book ILC Data Device Corporation
4				J-2	June, 1999	ACE/Mini-ACE Series BC/RT/MT Advanced Communication Engine Integrated 1553 Terminal… User's Guide ILC Data Device Corporation, june 1999, REV J-2
5	MIL-STD-1553B Change Notice 4		Jan 15,1996			Interface Standard for Digital Time Division Command/Response Multiplex Data Bus
6						Strategies For Real-Time system Specification
7	DIP-SAPII-DAS-31-06	1	Sept 7, 1998			SMCS 332 User manual

Table 2-2: Reference Documents

3. DPU BASIC SW STRUCTURE

The Structured Methods are used to specify and design the DPU Boot Software. The DPU Board Drivers are directly designed without using structured method because the S/W items are simple to design. In this case all the functionality will be allocated to one physical module as shown in the following Figure 3-1.

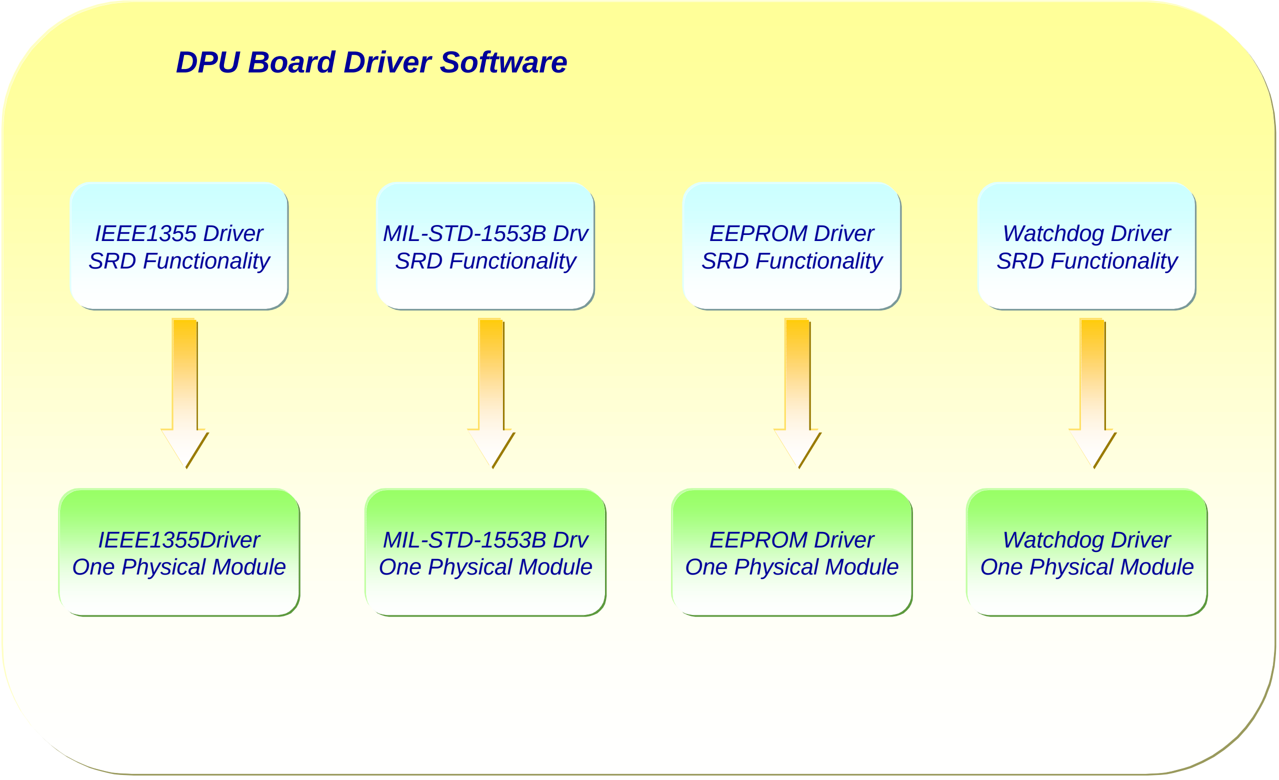
DPU Board Driver Software
IEEE1355 Driver SRD Functionality
MIL-STD-1553B Drv SRD Functionality
EEPROM Driver SRD Functionality
Watchdog Driver SRD Functionality
IEEE1355Driver One Physical Module
MIL-STD-1553B Drv One Physical Module
EEPROM Driver One Physical Module
Watchdog Driver One Physical Module

Figure 3-1: DPU Board Driver Software: SRD Functionality allocation to simple one Physical Module

The DPU Board Driver Software will be composed of four Modules, one for each Driver.

A more deep analysis is required from DPU Boot Software. In this case considering that the SRD gives a simple Requirements definition in natural language, in the ADD is included all the design process using the Data Flow Diagram, Control Flow Diagram, Process Specification, Control Specification, Architectural Context Diagram, Architectural Flow Diagram, Architectural Interconnection Diagram, Allocation Matrix functionality to S/W modules.

3.1 STRUCTURE OF THE DPU BOOT SOFTWARE MODEL

The Model exploit the De Marco, Hatley, Pirphai Structured Analysis Methodology. The Figure 3-2 shows a composite chart of the requirement model and it illustrates how the major components of the requirements model relate and interact to each other. The major part of the model are the DFD, CFD, PSPEC and CSPEC. In Figure 3-2 shows the DFD, CFD, diagrams and PSPEC, CSPEC for the DPU Boot Software.

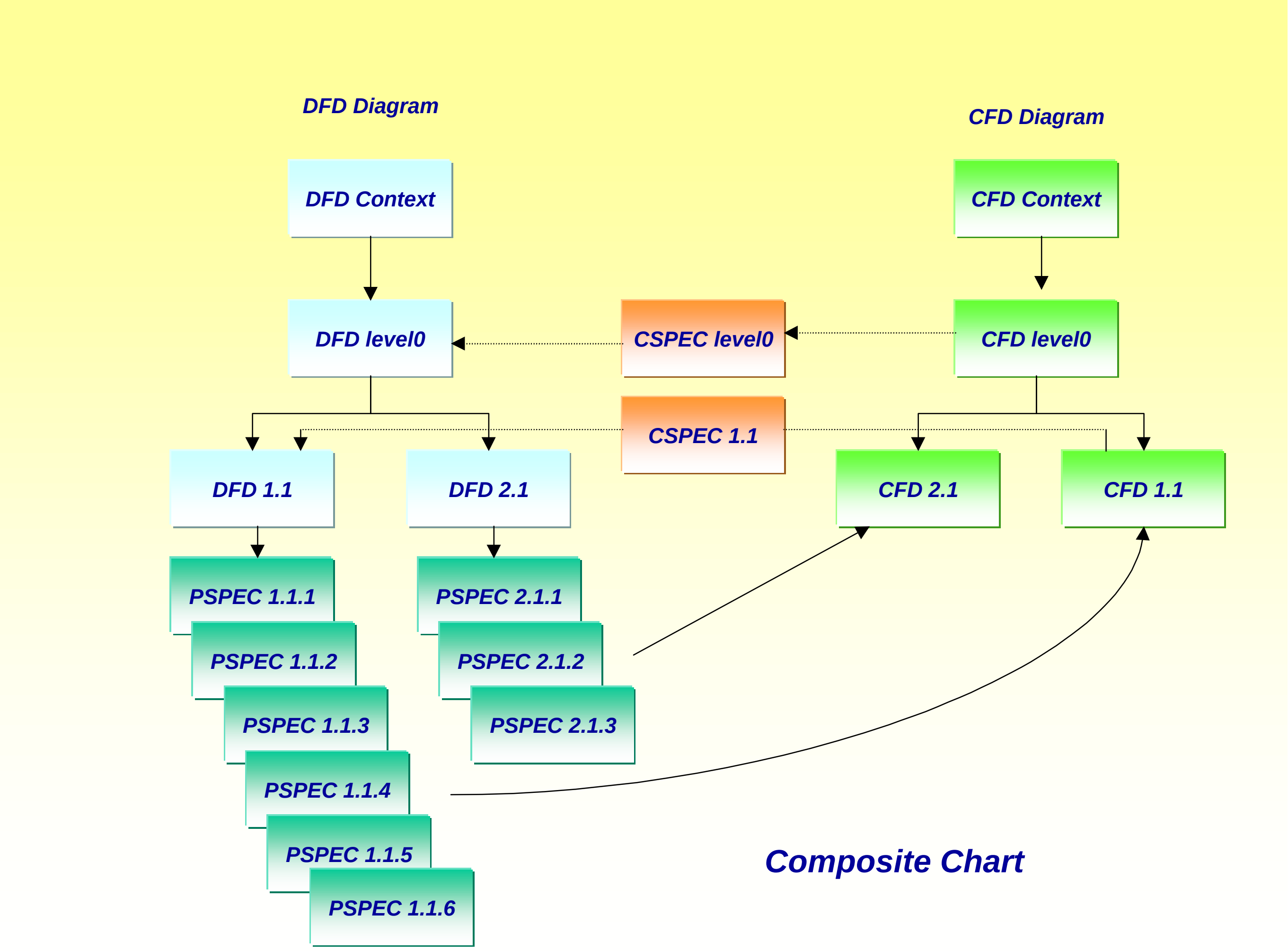

Figure 3-2: Composite Chart of the requirements model

For the DPU Boot Software have been identified nine Process Specification and two control specification. The CSPEC level0 is a State Transition Diagram with Process Activation Table (PAT) while the CSPEC 1.1 is a PAT only. In the Process Specification (PSPEC) will be allocated the SRD requirements in detail.

The following section includes the DFD and CFD diagrams. The DFD and CFD have been included in the same Flow diagram.

3.2 DPU BOOT SOFTWARE DFD & CFD DIAGRAMS

3.2.1 CONTEXT DIAGRAM

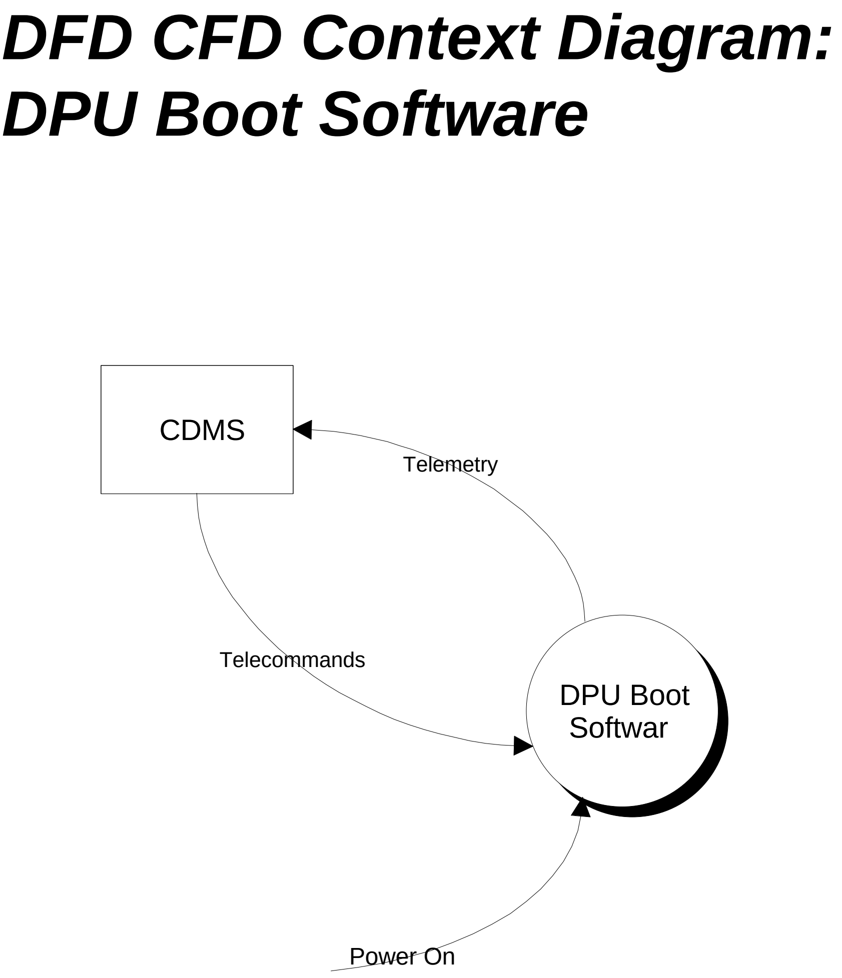

Figure 3-3: Context Diagram

3.2.2 LEVEL 0 DIAGRAM

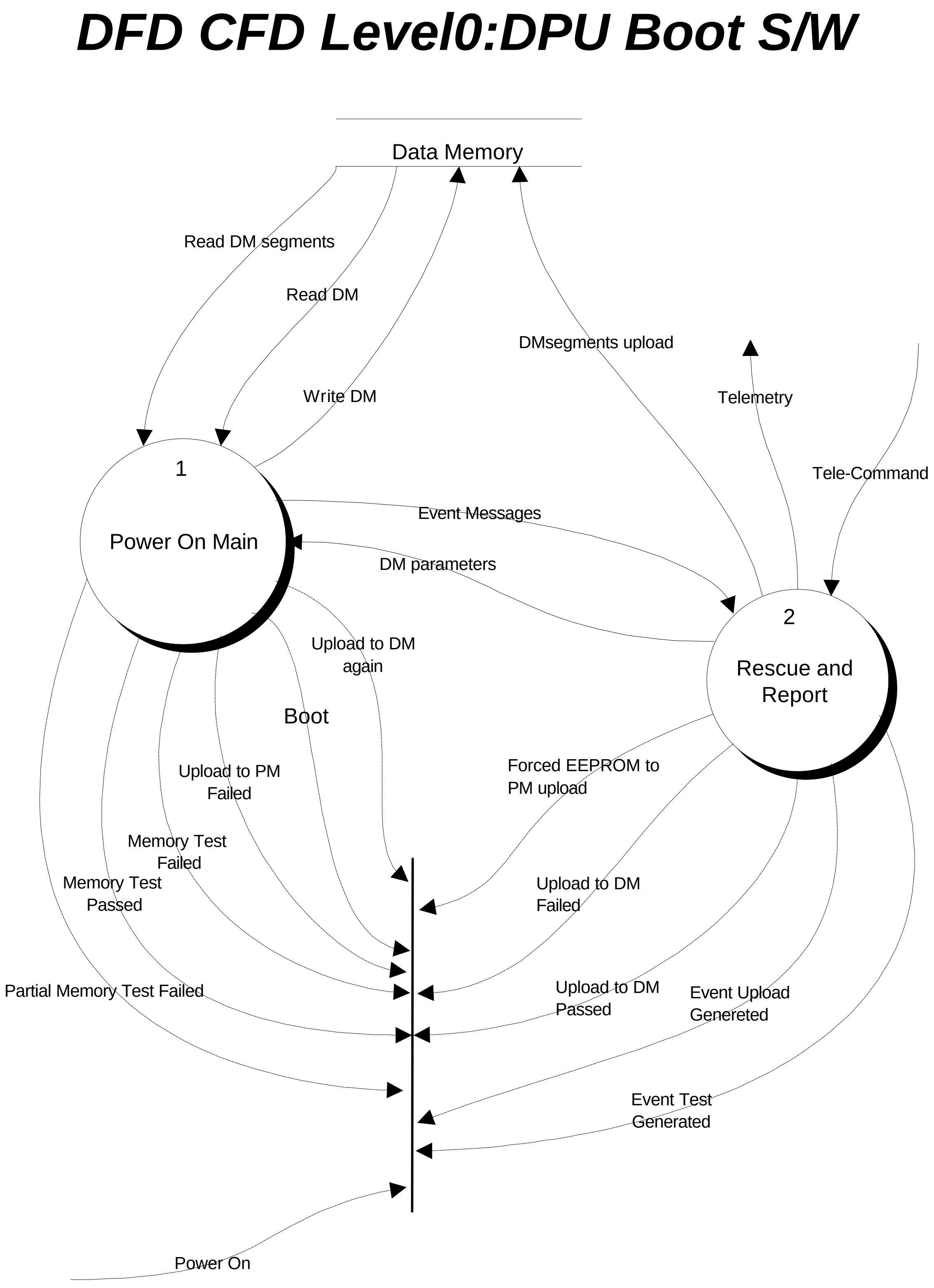

Figure 3-4: level 0 Diagram

3.2.3 DFD 1.1 & CFD 1.1

DFD 1.1 CFD 1.1: Power On Main

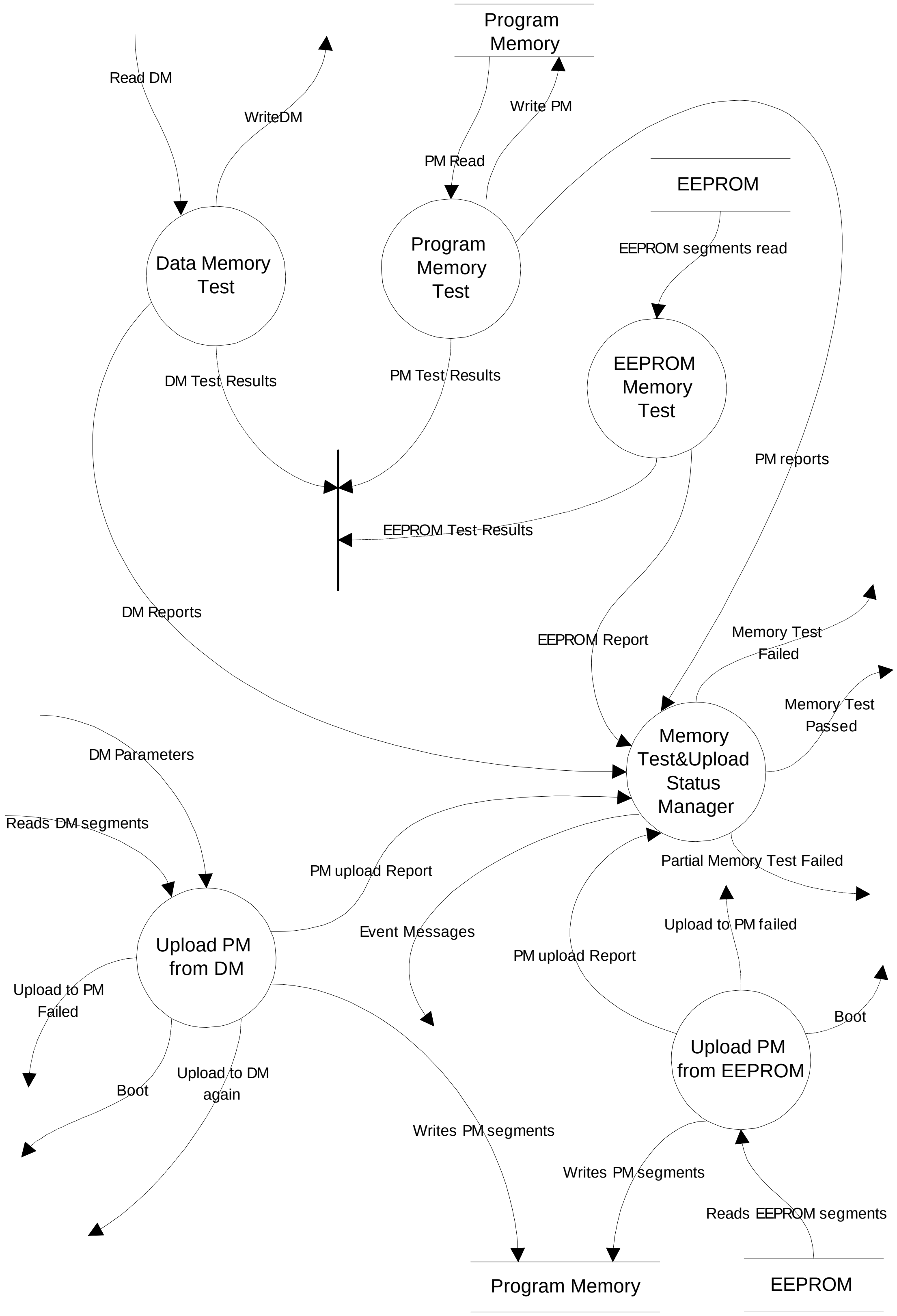

Figure 3-5: DFD 1.1 & CFD 1.1

3.2.4 DFD 2.1 & CFD 2.1

DFD 2.1 CFD 2.1: Rescue & Report

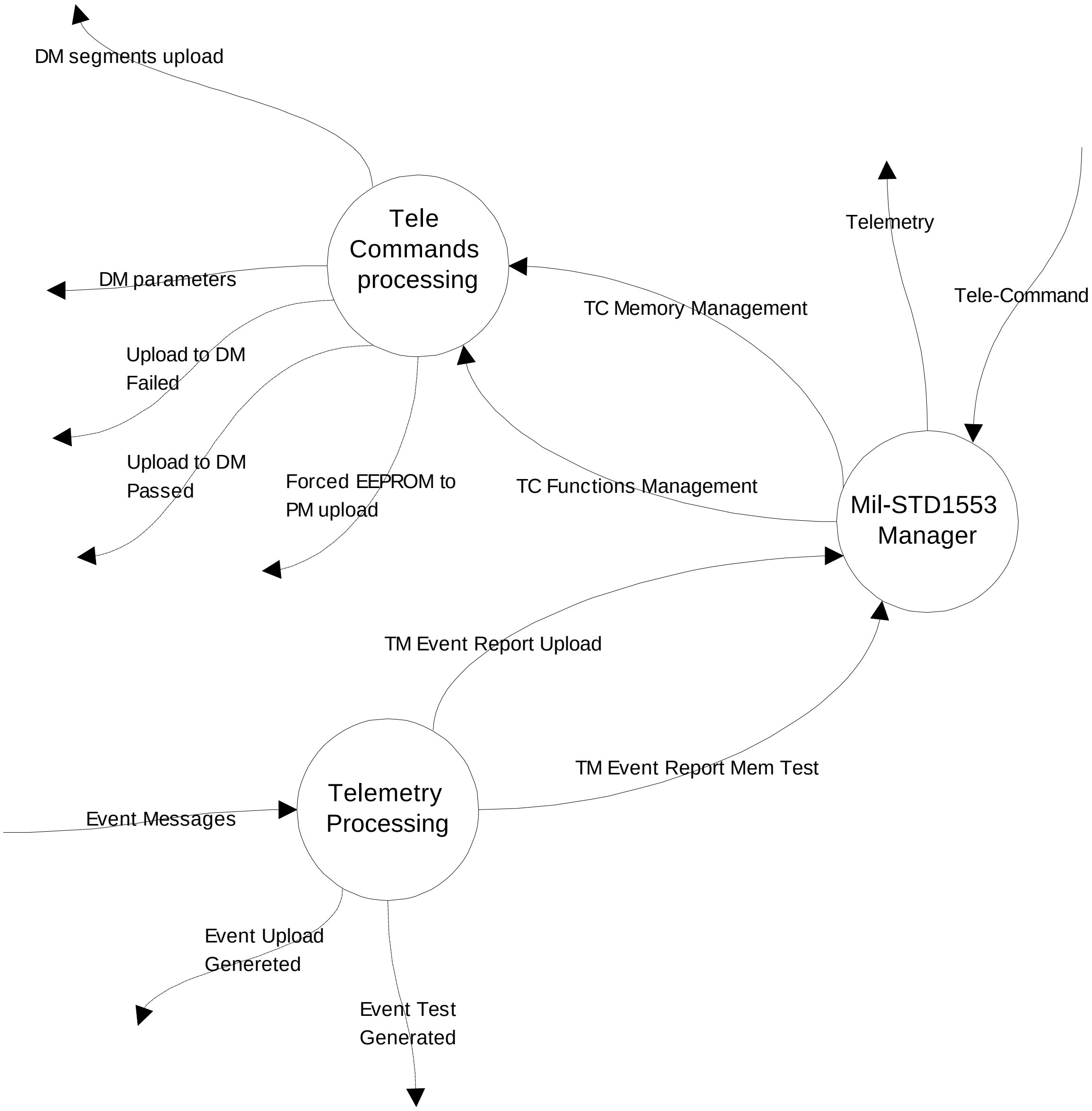

Figure 3-6: DFD 2.1 & DFD 2.1

3.3 PROCESS SPECIFICATION

3.3.1 PSPEC 1.1.1: DATA MEMORY TEST

INPUTS

- *Read DM: reads data memory cell*

OUTPUTS

- *Write DM: writes data memory cell*
- *DM report: transmission to "Memory Test Upload Status Manager" process of Corrupted Data Memory segments addresses*
- *DM Test Result: it is the result of the test. It can assume two value; PASS or FAIL*

PROCESS

The PSPEC has allocated the SRD Requirements as defined in the following table:

SRD requirements allocation table
SRD-3.1.1.0-040
SRD-3.1.1.0-050
SRD-3.1.1.0-330
SRD-3.1.4.0-000
SRD-3.1.4.0-010
SRD-3.1.4.0-020
SRD-3.1.4.0-030
SRD-3.1.4.0-040

Table 3-1: PSPEC 1.1.1 SRD requirements allocation table

3.3.2 PSPEC 1.1.2: PROGRAM MEMORY TEST

INPUTS

- *PM read: PM cell content*

OUTPUTS

- *Write PM: Data to be written in the PM cell*
- *PM report: transmission to "Memory Test Upload Status Manager"t process of corrupted Program Memory segments addresses*
- *PM Test Result: it is the result of the test. It can assume two value; PASS or FAIL*

PROCESS

The PSPEC has allocated the SRD Requirements as defined in the following table:

SRD requirements allocation table
SRD-3.1.1.0-050
SRD-3.1.1.0-330
SRD-3.1.5.0-000
SRD-3.1.5.0-010
SRD-3.1.5.0-020
SRD-3.1.5.0-030
SRD-3.1.5.0-040

Table 3-2: PSPEC 1.1.2 SRD requirements allocation table

3.3.3 PSPEC 1.1.3: EEPROM MEMORY TEST

INPUTS

- *EEPROM segments read: EEPROM segment Data*

OUTPUTS

- *EEPROM report: transmission to "Memory Test Upload Status Manager" process of corrupted EERPOM Memory segments addresses*
- *EEPROM Test Result: it is the result of the test. It can assume two value; PASS or FAIL*

PROCESS

The PSPEC has allocated the SRD Requirements as defined in the following table:

SRD requirements allocation table
SRD-3.1.1.0-050
SRD-3.1.1.0-330
SRD-3.1.1.0-340
SRD-3.1.6.0-000
SRD-3.1.6.0-010
SRD-3.1.6.0-020
SRD-3.1.6.0-030
SRD-3.1.6.0-040
SRD-3.1.6.0-050
SRD-3.1.6.0-060
SRD-3.1.6.0-070

Table 3-3: PSPEC 1.1.3 SRD requirements allocation table

3.3.4 PSPEC 1.1.4: MEMORY TEST & UPLOAD STATUS MANAGER

INPUTS

- *DM Report: acquisition of corrupted DM Memory segments addresses detected after the DM memory test*
- *PM Report: acquisition of corrupted PM Memory segments addresses detected after the PM memory test*
- *EEPROM Report: acquisition of corrupted EEPROM Memory segments addresses detected after the EEPROM Memory test*
- *PM upload Report: acquisition of EEPROM Memory segment address or DM Memory segment address that has failed the PM uploading*

OUTPUTS

- *Memory Test Passed: Control signal indicating the success of the overall memory test*
- *Memory Test Failed: Control signal indicating the not success of one memory test at least*
- *Memory Partial Test Failed: Control signal indicating one memory test failed*
- *Event Messages: Data flow containing the reports deriving from the test or upload*

PROCESS

```
If DM Report | PM report ! EEPROM Report | PM upload Report is true
        Generates and Event Messages = DM Report | PM report ! EEPROM Report | PM upload Report
Otherwise
        Do nothing
```

Set Memory Test Passed: the test have been performed and all the test are “PASSED”
Set Memory Test Failed: the tests have been performed and one test is “FAILED” at least.
Set Memory Partial Failed” after every tests type execution if the performed test is “FAILED”. The Memory Test Partial Failed follows the results of the single test.

3.3.5 PSPEC 1.1.5: UPLOAD PM FROM DM

INPUTS

- *Read DM segment:*
- *DM parameters: information for managing the Data Memory segments.*

OUTPUTS

- *Boot: Control Flow for performing the Jump to start ASW address*
- *Upload to DM Failed: Control Flow informing that the DM uploading is failed*
- *Upload to DM again: Control Flow informing that other DM segments have to be uploaded*
- *PM upload Report: transmission of DM Memory segment address that has failed the PM uploading*
- *Write PM segments:*

PROCESS

The PSPEC has allocated the SRD Requirements as defined in the following table:

SRD requirements allocation table
SRD-3.1.1.0-350
SRD-3.1.1.0-360
SRD-3.1.1.0-370
SRD-3.1.2.0-005
SRD-3.1.2.0-020
SRD-3.1.2.0-030
SRD-3.1.3.0-000
SRD-3.1.8.0-000
SRD-3.1.8.0-010
SRD-3.1.8.0-020
SRD-3.1.8.0-030
SRD-3.1.8.0-040

Table 3-4: PSPEC 1.1.5 SRD requirements allocation table

3.3.6 PSPEC 1.1.6: UPLOAD PM FROM EEPROM

INPUTS

- *Read EEPROM segment: :*

OUTPUTS

- *Boot: Control Flow for performing the Jump to start ASW address*
- *Upload to DM Failed: Control Flow informing that the DM uploading is failed*
- *PM upload Report: transmission of DM Memory segment address that has failed the PM uploading*
- *Write PM segments*

PROCESS

The PSPEC has allocated the SRD Requirements as defined in the following table:

SRD requirements allocation table
SRD-3.1.1.0-340
SRD-3.1.1.0-360
SRD-3.1.1.0-370
SRD-3.1.2.0-000
SRD-3.1.2.0-010
SRD-3.1.2.0-020
SRD-3.1.3.0-000
SRD-3.1.7.0-000
SRD-3.1.7.0-010
SRD-3.1.7.0-020
SRD-3.1.7.0-030
SRD-3.1.7.0-040

Table 3-5: PSPEC 1.1.6 SRD requirements allocation table

3.3.7 PSPEC 2.1.1: TELE-COMMAND PROCESSING

INPUTS

- *Tele-Command Memory Management: Data Flow containing the segment to be uploaded in DM*
- *Tele-Command Function Management: Data Flow containing the command for Forced EEPROM uploading*

OUTPUTS

- *DM segments upload: segments for DM uploading*
- *DM parameter: DM segment information transmission*
- *Upload to DM Failed;*
- *Upload to DM Passed*
- *Forced EEPROM to PM upload*

PROCESS

The PSPEC has allocated the SRD Requirements as defined in the following table:

SRD requirements allocation table
SRD-3.1.1.0-000
SRD-3.1.1.0-010
SRD-3.1.1.0-020
SRD-3.1.1.0-150
SRD-3.1.1.0-160
SRD-3.1.1.0-170
SRD-3.1.1.0-180
SRD-3.1.1.0-190
SRD-3.1.1.0-200
SRD-3.1.1.0-210
SRD-3.1.1.0-220
SRD-3.1.1.0-240
SRD-3.1.1.0-250
SRD-3.1.1.0-260
SRD-3.1.1.0-270
SRD-3.1.1.0-280
SRD-3.1.1.0-290
SRD-3.1.1.0-300
SRD-3.1.1.0-310

Table 3-6: PSPEC 2.1.1 SRD requirements allocation table

3.3.8 PSPEC 2.1.2: TELEMETRY PROCESSING

INPUTS

- *Event Messages: Data flow containing the DM, PM, EEPROM, Upload to PM Reports for sending to CDMS.*

OUTPUTS

- *Event Test Generated: the Control Flow informs when the Event Test message has been transmitted.*
- *Event Upload Generated: the Control Flow informs when the Event Upload to PM Message has been generated.*
- *TM Event report Upload*
- *TM Event Report Memory Test*

PROCESS

The PSPEC has allocated the SRD Requirements as defined in the following table:

SRD requirements allocation table
SRD-3.1.1.0-000
SRD-3.1.1.0-060
SRD-3.1.1.0-070
SRD-3.1.1.0-080
SRD-3.1.1.0-090
SRD-3.1.1.0-100
SRD-3.1.1.0-110
SRD-3.1.1.0-120
SRD-3.1.1.0-130
SRD-3.1.1.0-140
SRD-3.1.1.0-230
SRD-3.1.1.0-320

Table 3-7: PSPEC 2.1.2 SRD requirements allocation table

3.3.9 PSPEC 2.1.3: MIL-STD-1553B MANAGER

INPUTS

- *TeleCommands:*
- *TM Event Report.Memory Test*
- *TM Event Report Upload*

OUTPUTS

- *Telemetry*
- *TC Memory Management*
- *TC Function Management*

PROCESS

The PSPEC has allocated the SRD Requirements as defined in the following table:

SRD requirements allocation table
SRD-3.1.1.0-000
SRD-3.1.1.0-320

Table 3-8: PSPEC 2.1.3 SRD requirements allocation table

3.4 CONTROL SPECIFICATION

3.4.1 CSPEC LEVEL0

3.4.1.1 PROCESS ACTIVATION TABLE

The paragraph reports the PAT considering the STD state Transition diagram reported in the Figure 3-7

Actions	Power On Main (Process)	Rescue & Reort Process
Perform memory test	ACTIVATED	DEACTIVATED
Generate an Event Message	DEACTIVATED	ACTIVATED
Uploading ASW in PM	ACTIVATED	DEACTIVATED
Uploading ASW in DM	DEACTIVATED	ACTIVATED
Exit from Power On procedure	ACTIVATED	DEACTIVATED

Table 3-9: CSPEC level0: Process Activation Table

3.4.1.2 STATE TRANSITION DIAGRAM

CSPEC level0: State Transition Diagram

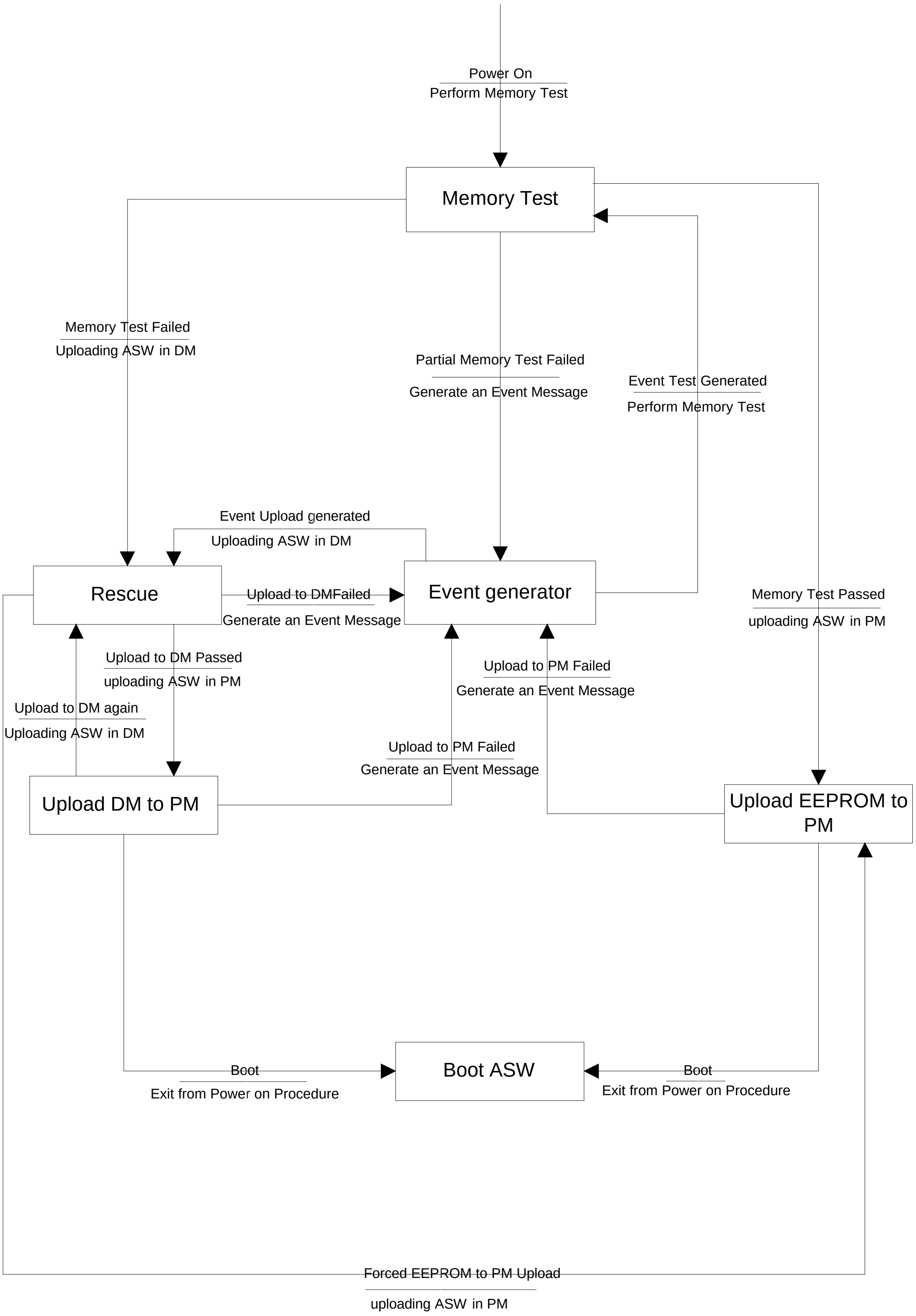

Figure 3-7: State Transition Diagram

3.4.2 CSPEC 1.1: PROCESS ACTIVATION TABLE

PAT defining the Memory Test.

Inputs			Outputs		
DM Test Result	PM Test Result	EEPROM Test Result	DM Test	PM Test	EEPROM Test
Not Tested	Not Tested	Not Tested	DEACTIVATED	ACTIVATED	DEACTIVATED
Not Tested	Tested	Not Tested	ACTIVATED	DEACTIVATED	DEACTIVATED
Tested	Tested	Not Tested	DEACTIVATED	DEACTIVATED	ACTIVATED
Tested	Tested	Tested	DEACTIVATED	DEACTIVATED	DEACTIVATED

Table 3-10: CSPEC 1,1: Process Activation Table

4. ARCHITECTURAL DESIGN

4.1 PHYSICAL MODEL AND DECOMPOSITION FOR DPU BOOT SOFTWARE

4.1.1 ARCHITECTURAL CONTEXT DIAGRAM

The architecture context diagram (ACD) establishes the information boundary between the system being implemented and the environment in which the system has to operate. The ACD is the highet level diagram for any system.

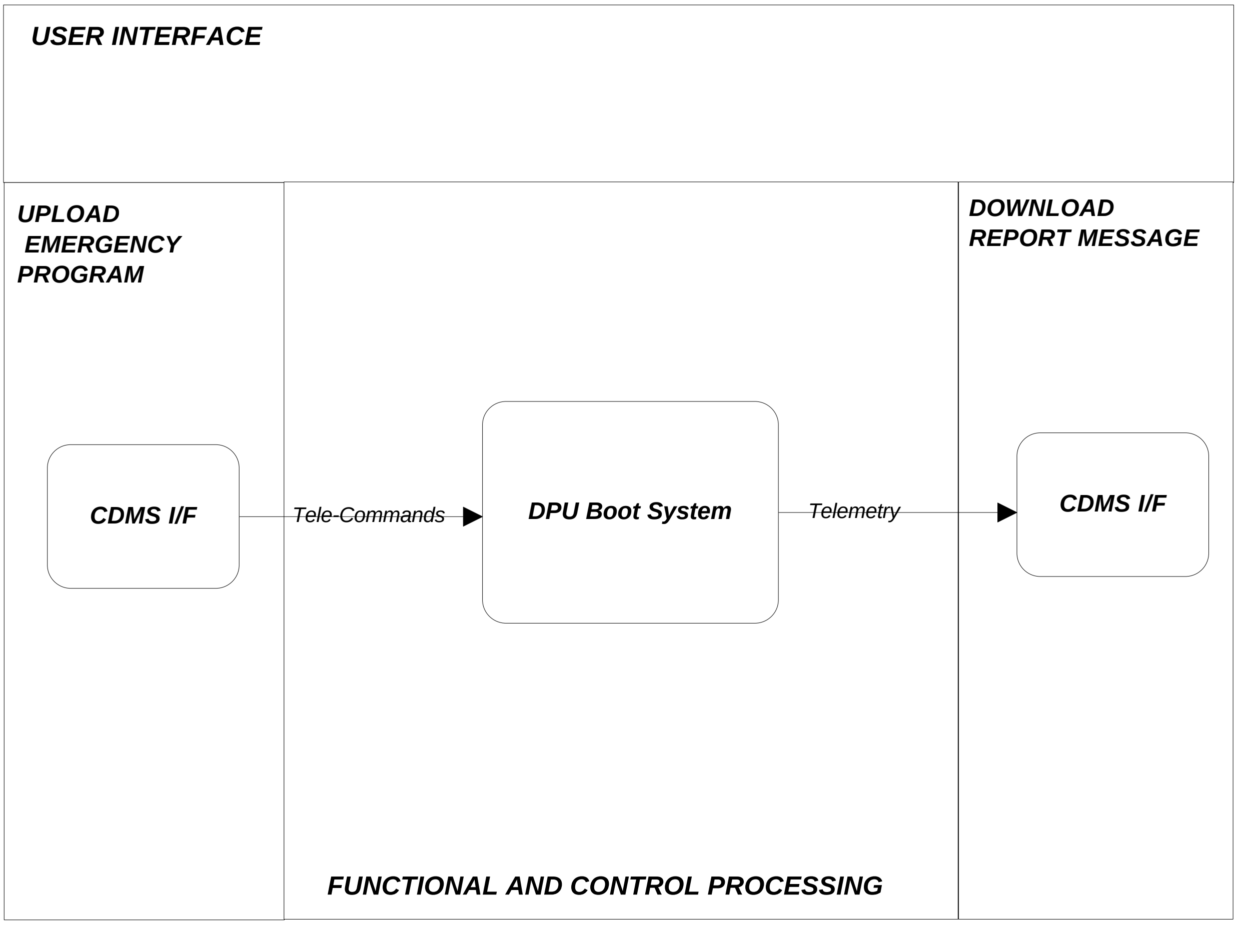

Figure 4-1: Architectural Context Diagram (ACD)

4.1.2 ARCHITECTURE INTERCONNECT DIAGRAM

An architecture interconnect diagram is a representation of the communication channels/bus that exist between the architecture modules. It shows the physical means by which the module communicate.

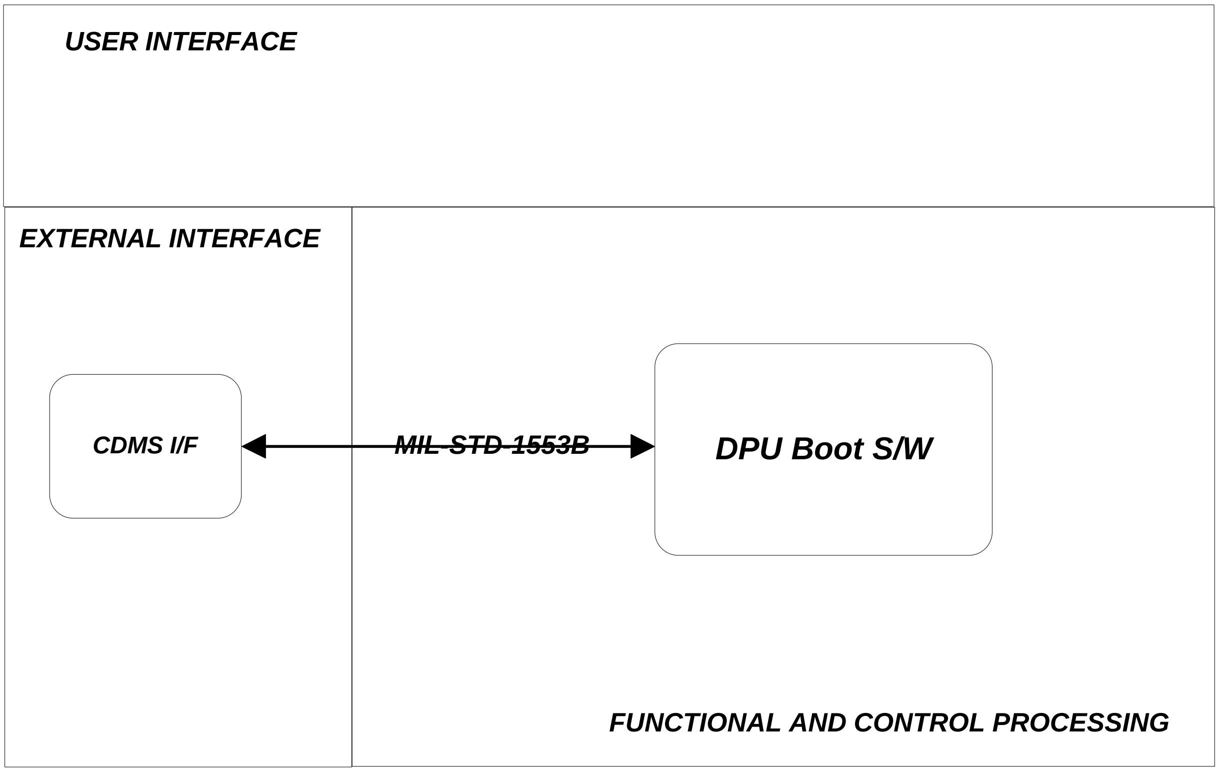

Figure 4-2: Architecture Interconnect Diagram

4.1.3 ARCHITECTURAL FLOW DIAGRAM

An architectural flow diagram is a network representation of a system's physical configuration. It documents the information flow between all architecture module. The AFD also represents the allocation of processes and flow from the data and control flow diagrams into architecture modules

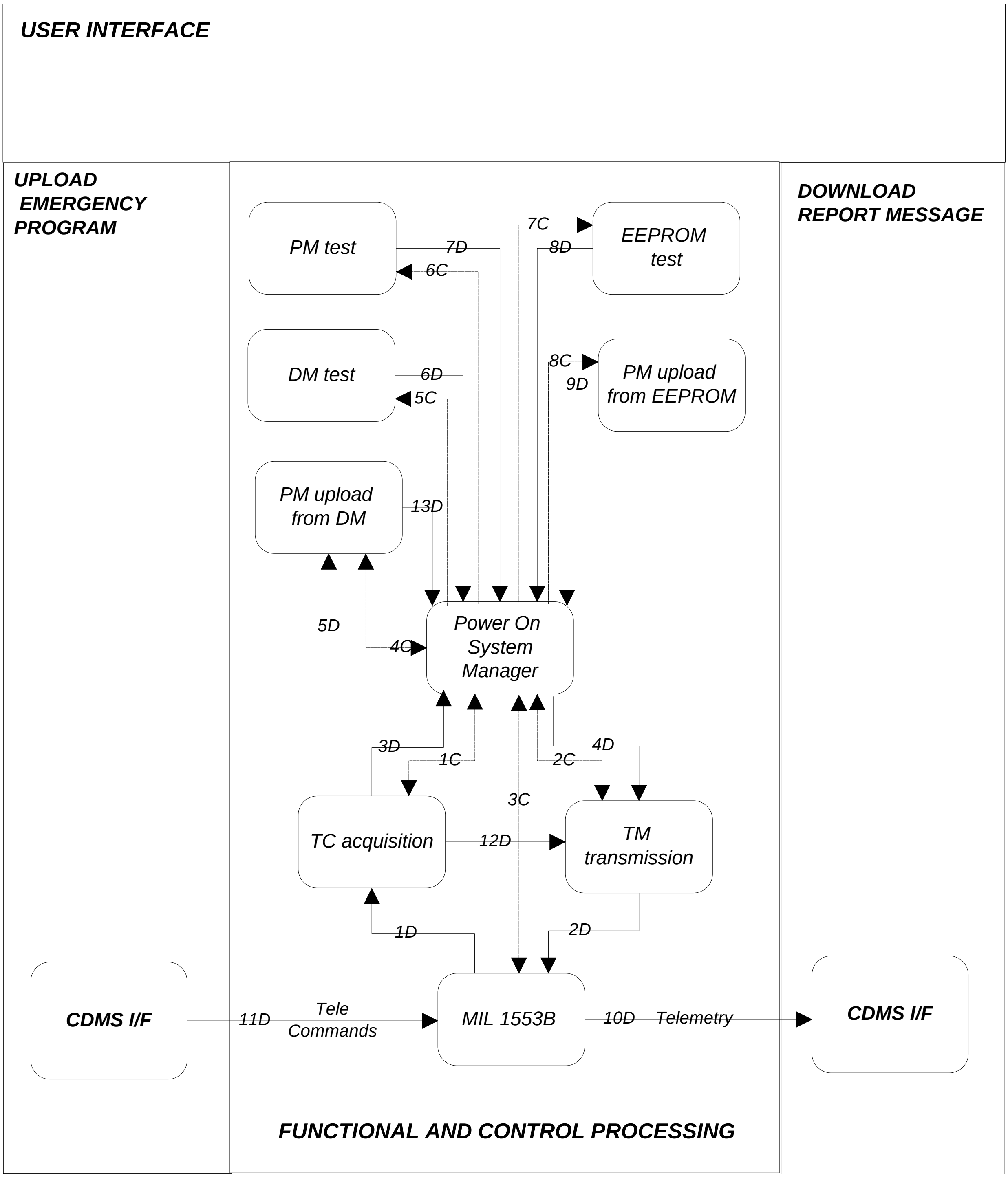

Figure 4-3: Architectural Flow Diagram.(TBC)

Table of the Flow

Number	Name
1D	Telecommand Data packet; Data memory and TC parameter
2D	Telemetry Data packet: Event Report
3D	Data Memory header parameters
4D	Report Messages
5D	Data Memory segments
6D	DM Test Results
7D	PM Test Results
8D	EEPROM Test Results
9D	PM upload from EEPROM Results
10D	Telemetry on MIL1553B
11D	TeleCommand on MIL1553B
12D	Report Message for received TC failed
13D	PM upload from DM Results
1C	Command and Status “TC Acquisition”
2C	Command and Status “TM Transmission”
3C	Command and Status “MIL-1553B”
4C	Command and Status “PM upload from DM”
5C	Command to “DM Test”
6C	Command to “PM Test”
7C	Command to “EEPROM Test”
8C	Command to “PM upload from EEPROM”

Table 4-2: Flows in the Architectural Flow Diagram

4.1.4 ARCHITECTURE TRACEABILITY MATRIX

The traceability Matrix is used to allocate the Process Specification and Control Specification resulting from the Data Flow Diagram and Control Flow Diagram. A table is presented where on the column are inserted the Architectural Modules while in the row are inserted the Process Specification and the Control Specification.

	PM Test	DM Test	EEPROM Test	Power On System Manager	PM upload from DM	PM upload from EEPROM	TC Rx	TM Tx	MIL-STD 1553B
PSPEC 1.1.1		X		X					
PSPEC 1.1.2	X			X					
PSPEC 1.1.3			X	X					
PSCPEC 1.1.4				X					
PSPEC 1.1.5					X				
PSPEC 1.1.6						X			
PSPEC 2.1.1							X		
PSPEC 2.1.2								X	
PSPEC 2.1.3									X
CSPEC level0				X	X	X	X	X	
CSPEC 1,1	X	X	X	X					

4.2 PHYSICAL MODEL AND DECOMPOSITION FOR DPU BOARD DRIVER SOFTWARE

The DPU Board Driver Software will be composed of a number of Software Module according the number of defined drivers. In this case are identified four Driver and therefore four module. In this case the results of the physical decomposition is one Software module for each type of driver.

4.3 S/W MODULES LIST

The Table 4-1 shows the S/W Module identified for the two software items. DPU Boot Software and DPU Board Driver Software. Each module can include one or more file source/header.

N°	Software Item	Module	Description
1.1	*DPU Boot S/W*	*MIL 1553 Module*	*Communication Protocol over MIL1553B*
1.2		*Tele Commands Module*	*TC acquisition and processing*
1.3		*Telemetry*	*Telemetry packetization and sending*
1.4		*Power On System Manager*	*Memory process control*
1.5		*DM test*	*Test of Data Memory*
1.6		*PM test*	*Test of Program Memory*
1.7		*EEPROM test*	*Test of EEPROM memory*
1.8		*PM upload from DM*	*Program Memory uploading from DM*
1.9		*PM upload from EEPROM*	*Program memory uploading from EEPROM*
2.1	*DPU Board Drivers S/W*	*IEEE1355 Module*	*IEEE 1355 Drivers*
2.2		*MIL-STD-1553B Module*	*MIL-STD-1553B Drivers*
2.3		*EEPROM Module*	*EEPROM Drivers*
2.4		*Watchdog Module*	*Watchdog Driver*

Table 4-1: S/W Modules List table.

5. MODULES DEFINITION

5.1 DPU BOOT SOFTWARE

5.1.1 MIL-STD-1553B

5.1.1.1 COMPONENT TYPE

This thread manages the MIL-STD during the PDU Boot Software. This module will included essential MIL-STD-1553B functions, that will be developed in the MIL-STD-1553B driver module, concerning the ACE/MINI ACE chip configuration and new functions for managing the Application Layer services, (Telecommand and Telemetry.services)
In order to limit the memory use (only 32Kbyte of memory availability for the Boot program), the module will only include essential functions to configure and manage the ACE/MIN ACE.

5.1.1.2 FUNCTIONAL REQUIREMENTS

The functional requirement are identified from table following:

Allocated requirements
PSPEC 2.1.3 applicable
The Module shall include the MIL-STD-1553 initialization procedure. It shall be able to configure the ACE chip as Remote Terminal of the bus
The MIL-STD-1553B Module shall be able to acquire TC packets, consecutive sequence of standard message via the MIL-STD-1553B. The Module shall acquire up to four 32Words standard message for each telecommand.
The MIL-STD-1553B Module shall be able to send Telemetry Event Report packets, consecutive sequence of standard message via MIL-STD-1553B. The Module shall send up to two 32words standard message for each Telemetry data packet.
The Module shall only include two Telecommand service type defined in [AD-7]. ▪ TC Memory Management ▪ TC Function Management
The Module shall only include one Telemetry data service type: ▪ Event Report Service
The module shall be able to acquire and transmit TC and TM data packet according to the MILBUS profile defined in [AD-7]-Appendix9, Data Link Layer, Transfer Layer protocol

Table 5-1: MIL-STD-1553: Functional Requirement

5.1.1.3 NON FUNCTIONAL REQUIREMENTS

N/A

5.1.1.4 INTERFACES

The module is interface to the following modules:

Interface
"TC Acquisition" Module
"TM Transmission" Module
"Power on Status Manager" Module

Table 5-2: MIL-STD-1553: Interfaces

5.1.1.5 DATA ELEMENTS REQUIREMENTS

Data Elements requirements
The Module shall manage (write) a data buffer of TBD size for sending the received TC packets to the "TC Acquisition" Module
The Module shall manage (read) a data buffer of TBD size for acquiring the Telemetry Data packet incoming from the "TM transmission" Module.

Table 5-3: MIL-STD-1553: Data Elements Requirements

5.1.1.6 CONTROL REQUIREMENTS

Controls requirements
The Module shall acquire a command flag from the "Power On Status Manager" for starting the MIL-STD Initialization procedure
The Module shall send a status flag to the "Power On Status Manager" containing the actual status of the MILBUS.

Table 5-4: MIL-STD-1553: Control Requirements

5.1.2 TELE-COMMAND ACQUISITION

5.1.2.1 COMPONENT TYPE

This Thread manages the Tele-Command packet received via MIL-STD 1553 Module.

5.1.2.2 FUNCTIONAL REQUIREMENTS

The functional requirement are identified from table following:

Allocated requirements
PSPEC 2.1.1 applicable
CSPEC level0 applicable
The Module shall acquire TC Memory Management from "MIL-STD 1553 Module" (max 128 Words for each TC) and it shall build the Data Memory segment
The Module shall acquire TC Function Management from "MIL-STD 1553
The Module shall verify the TC validity by checking the APID, and the FCS of the received Telecommands
The Module shall check the Data Memory segment FCS (frame check sequence)
The Module shall state valid a DM memory segment when all the TC received are valid and the FCS of the uploaded DM segment is correct.
The Module shall write in DM valid DM segments
The Module shall send a message to the "Power On Status Manager" in case of corrupted DM segment
The Module shall be activated by a "Power On Status Manager" Module when a Application program upload via MIL-STD-1553 is required.
The Module shall inform the "Power On Status Manager" if a EEPROM Boot request TC has been received
The Module shall send a report message to the "Telemetry Transmission" Module when a corrupted DM segment has been detected.
The Module shall be able to reset the DM filling it a TBD value.
The Module shall send to the "Power On Status manager" the Headers of the DM segments.
The Module shall be able to manage the DM header segment.

Table 5-5: TC :Acquisition: Functional Requirements

5.1.2.3 NON FUNCTIONAL REQUIREMENTS

N/A

5.1.2.4 INTERFACES

The module is interface to the following modules:

"PM upload from DM" Module
"MIL-STD 1553" Module
"Power on Status Manager" Module
"TM transmission" Module

Table 5-6: TC Acquisition: Interfaces

5.1.2.5 DATA ELEMENTS REQUIREMENTS

Data Elements requirements
The Module shall manage a data Buffer for storing the DM header segment information of the segments. Uploaded in Data Memory. The buffer shall be erased when all the temporary DM segments shall be uploaded in Program Memory.
The Module shall manage a buffer for memorizing the Data Memory segments
The Module shall manage a buffer to report failure information to the "Telemetry Transmission Module"
The Module shall be manage a buffer for sending the EEPROM Boot request to "Power On status Manager"

Table 5-7: TC Acquisition: Data Elements Requirements

5.1.2.6 CONTROL REQUIREMENTS

Controls requirements
The Module shall acquire a command flag from the "Power On Status Manager" to start /stop an other DM segment uploading
.The Module shall send a status message to "Power On status Manager"

Table 5-8: TC Acquisition: Control Requirements

5.1.3 TELEMETRY TRANSMISSION

5.1.3.1 COMPONENT TYPE

The Module is a thread managing the Event report messages packets transmission via MIL-STD 1553

5.1.3.2 FUNCTIONAL REQUIREMENTS

The functional requirement are identified from table following:

Allocated requirements
PSPEC 2.1.2 applicable
CSPEC level0 applicable
The Module shall be able to generate two type of Event Report Message: ▪ Event report message for reporting of DM,PM,EEPROM corrupted segments ▪ Event report message for reporting PM upload from DM, PM upload from EEPROM fault
The Module shall send the message via MIL when a transmission command flag is received from the "Power On status message"
The Module shall acquire the Event Report Message from "Power On Status Manager" or "TC Acquisition Module

Table 5-9: TM Transmission: Functional Requirements

5.1.3.3 NON FUNCTIONAL REQUIREMENTS

N/A

5.1.3.4 INTERFACES

The module is interface to the following modules:

"TC acquisition " Module
"MIL-STD 1553" Module
"Power on Status Manager" Module

Table 5-10: TM Transmission: Interfaces

5.1.3.5 DATA ELEMENTS REQUIREMENTS

Data Elements requirements
The Module shall manage a data Buffer containing the Event report message ready to be transmitted and messages in progress

Table 5-11: TM Transmission: Data Elements Requirements

5.1.3.6 CONTROL REQUIREMENTS

Controls requirements
The Module shall acquire a command flag from the "Power On Status Manager" to start /stop Event report transmission
.The Module shall send a status message to "Power On status Manager" informing that the Event report Message has been sent

Table 5-12: TM transmission: Control Requirements

5.1.4 PM TEST

5.1.4.1 COMPONENT TYPE

This thread performs the Program memory test

5.1.4.2 FUNCTIONAL REQUIREMENTS

The functional requirement are identified from table following:

Allocated requirements
PSPEC 1.1.2 applicable
CSPEC 1.1 applicable

Table 5-13: PM Test Functional Requirements

5.1.4.3 NON FUNCTIONAL REQUIREMENTS

N/A

5.1.4.4 INTERFACES

The module is interface to the following modules:

Interfaces
"Power on Status Manager" Module

Table 5-14: PM Test Interfaces

5.1.4.5 DATA ELEMENTS REQUIREMENTS

Data Elements requirements
The Module shall manage a data Buffer containing the Number of Corrupted segments and the Addresses of the Corrupted segments
The Module shall transmit the PM test result data

Table 5-15: PM Test: Data Elements Requirements

5.1.4.6 CONTROL REQUIREMENTS

Controls requirements
The Module shall acquire a command flag from the "Power On Status Manager" to start Program Memory Test
The Module shall send a status flag to the "Power On Status Manager" containing the test status: (NOT PERFORMED, IN PROGRESS, PERFORMED)

Table 5-16: PM Test: Control Requirements

5.1.5 DM TEST

This thread performs the Data Memory test

5.1.5.1 FUNCTIONAL REQUIREMENTS

The functional requirement are identified from table following:

Allocated requirements
PSPEC 1.1.1 applicable
CSPEC 1.1 applicable

Table 5-17: DM Test: Functional Requirements

5.1.5.2 NON FUNCTIONAL REQUIREMENTS

N/A

5.1.5.3 INTERFACES

The module is interface to the following modules:

Interfaces
“Power on Status Manager” Module

Table 5-18: DM Test Interfaces

5.1.5.4 DATA ELEMENTS REQUIREMENTS

Data Elements requirements
The Module shall manage a data Buffer containing the Number of Corrupted segments and the Addresses of the Corrupted segments
The Module shall transmit the DM test result data

Table 5-19: DM Test: Data Elements Requirements

5.1.5.5 CONTROL REQUIREMENTS

Controls requirements
The Module shall acquire a command flag from the “Power On Status Manager” to start Data Memory Test
The Module shall send a status flag to the “Power On Status Manager” containing the test status: (NOT PERFORMED, IN PROGRESS, PERFORMED)

Table 5-20: DM Test Control Requirements

5.1.6 EEPROM TEST

This thread performs the EEPROM memory test

5.1.6.1 FUNCTIONAL REQUIREMENTS

The functional requirement are identified from table following:

Allocated requirements
PSPEC 1.1.3 applicable
CSPEC 1.1 applicable

Table 5-21: EEPROM Test: Functional Requirements

5.1.6.2 NON FUNCTIONAL REQUIREMENTS

N/A

5.1.6.3 INTERFACES

The module is interface to the following modules:

Interfaces
"Power on Status Manager" Module

Table 5-22: EEPROM Test: Interfaces

5.1.6.4 DATA ELEMENTS REQUIREMENTS

Data Elements requirements
The Module shall manage a data Buffer containing the Number of Corrupted segments and the Addresses of the Corrupted segments
The Module shall transmit the EEPROM test result data

Table 5-23: EEPROM Test: Data Element Requirements

5.1.6.5 CONTROL REQUIREMENTS

Controls requirements
The Module shall acquire a command flag from the "Power On Status Manager" to start EEPROM Memory Test
The Module shall send a status flag to the "Power On Status Manager" containing the test status: (NOT PERFORMED, IN PROGRESS, PERFORMED)

Table 5-24: EEPROM Test; Control Requirements

5.1.7 POWER ON SYSTEM MANAGER

This thread shall control the DPU Boot S/W program activities.

5.1.7.1 FUNCTIONAL REQUIREMENTS

The functional requirement are identified from table following:

Allocated requirements
PSPEC 1.1.4 applicable
PSPEC 1.1.1 applicable
PSPEC 1.1.2.applicable
CSPEC level0
CSPEC 1.1 applicable

Table 5-25: Power On System Mng: Functional Requirements

5.1.7.2 NON FUNCTIONAL REQUIREMENTS

N/A

5.1.7.3 INTERFACES

The module is interface to the following modules:

Interfaces
“TC Acquisition” Module
“TM Transmission” Module
“MIL-STD 1553” Module
“PM upload from DM” Module
“PM upload from EEPROM” Module
“PM Test” Module
“DM Test” Module
“EEPROM Test” Module

Table 5-26: : Power On System Mng: Interfaces

5.1.7.4 DATA ELEMENTS REQUIREMENTS

Data Elements requirements
The Module shall manage input/Output data Buffer containing the Number of Corrupted segments and the Addresses of the Corrupted segments
The Module shall manage input/Output data Buffer containing the Address of the segment in fault during the Program Memory upload and the fault cause.
The Module shall manage the EEPROM test result data
The Module shall manage the PM test result data
The Module shall manage the DM test result data
The Module shall manage data Buffer containing the DM header of the uploaded segments.
The Module shall manage a data Buffer containing the Command for EEPROM boot request

Table 5-27: : Power On System Mng: Data Elements Requirements

5.1.7.5 CONTROL REQUIREMENTS

Controls requirements
The Module shall send a command flag to the "EEPROM Test " to start EEPROM Memory Test
The Module shall send a command flag to the "PM Test " to start PM Memory Test
The Module shall send a command flag to the "PM Test " to start DM Memory Test
The Module shall send a command flag to the "PM Upload from EEPROM" to start the PM uplòading
The Module shall send a command flag to the "PM Upload from DM" to start/stop the PM uplòading
The Module shall send a command flag to the "MIL-STD-1553B" to start the MIL-STD initialization
The Module shall send a command flag to the "TM transmission" to start the Event message transmission
The Module shall send a command flag to the "TC Acquisition" to start /stop an other DM segment uploading
The Module shall acquire a status flag from the "Mil-STD-1553B" containing the actual status of the MILBUS
.The Module shall acquire a status message from "TC Acquisition "
.The Module shall acquire a status message from "TM transmission " informing that the Event report Message has been sent
The Module shall acquire a status flag from the "PM Test" containing the test status: (NOT PERFORMED, IN PROGRESS, PERFORMED)
The Module shall acquire a status flag from the "DM Test" containing the test status: (NOT PERFORMED, IN PROGRESS, PERFORMED)
The Module shall acquire a status flag from the "EEPROM Test" containing the test status: (NOT PERFORMED, IN PROGRESS, PERFORMED)
The Module shall acquire a status flag from the "PM upload from DM " containing the Uploading status: (READY TO START, IN PROGRESS, PERFORMED, EEPROM BOOT REQUESTED)
The Module shall acquire a status flag from the "PM Upload from EEPROM Test" containing the test status: (NOT PERFORMED, IN PROGRESS, PERFORMED)

Table 5-28: : Power On System Mng: Control Requirements

5.1.8 PM UPLOAD FROM DM

This thread performs the Application Program uploading from Data Memory to Program Memory

5.1.8.1 FUNCTIONAL REQUIREMENTS

The functional requirement are identified from table following:

Allocated requirements
PSPEC 1.1.5 applicable
CSPEC level0 applicable
The Module shall be able to stop the uploading if an EEPROM boot request is requested by Tele-Command. In this case the Module shall reset the DM and PM memory before of starting the EEPROM uploading.
The Module shall verify the correctness of the uploaded PM segment by mean of the FCS
The Module shall verify the correctness of the uploaded application program computing and comparing the FCS on the Application Program. The Address of the Application Program FCS is TBD.
The Module shall be able to generate report messages to the "Power On Status Manager" containing the Address of the PM segment in fault.

Table 5-29: PM Upload from DM: Functional Requirements

5.1.8.2 NON FUNCTIONAL REQUIREMENTS

N/A

5.1.8.3 INTERFACES

The module is interface to the following modules:

Interfaces
"Power on Status Manager" Module
"TC Acquisition

Table 5-30: PM Upload from DM: Interfaces

5.1.8.4 DATA ELEMENTS REQUIREMENTS

Data Elements requirements
The Module shall manage a buffer containing the parameter of the DM segment ready to upload
The Module shall manage a buffer containing the Address of segment in fault and the fault cause.
The Module shall manage the Uploading Result (PASSED or FAILED)

Table 5-31: PM Upload from DM: Data Elements Requirements

5.1.8.5 CONTROL REQUIREMENTS

Controls requirements
The Module shall acquire a command flag from the "Power On Status Manager" to start the Program Memory Uploading
The Module shall send a status flag to the "Power On Status Manager" containing the Uploading status: (READY TO START, IN PROGRESS, PERFORMED, EEPROM BOOT REQUESTED)

Table 5-32: PM Upload from DM: Control Requirements

5.1.9 PM UPLOAD FROM EEPROM

This thread performs the Application Program uploading from EEPROM to Program Memory

5.1.9.1 FUNCTIONAL REQUIREMENTS

The functional requirement are identified from table following:

Allocated requirements
PSPEC 1.1.6 applicable
CSPEC level0 applicable
The Module shall verify the correctness of the uploaded PM segment by mean of the FCS
The Module shall verify the correctness of the uploaded application program computing and comparing the FCS on the Application Program. The Address of the Application Program FCS is TBD.
The Module shall be able to generate report messages to the "Power On Status Manager" containing the Address of the PM segment in fault.

Table 5-33: PM Upload from EEPROM: Functional Requirements

5.1.9.2 NON FUNCTIONAL REQUIREMENTS

N/A

5.1.9.3 INTERFACES

The module is interface to the following modules:

Interfaces
"Power on Status Manager" Module

Table 5-34: PM Upload from EEPROM: Interfaces

5.1.9.4 DATA ELEMENTS REQUIREMENTS

Data Elements requirements
The Module shall manage a buffer containing the Address of segment in fault and the fault cause.
The Module shall manage the Uploading Result (PASSED or FAILED)

Table 5-35: PM Upload from EEPROM: Data Elements Requirements

5.1.9.5 CONTROL REQUIREMENTS

Controls requirements
The Module shall acquire a command flag from the "Power On Status Manager" to start the Program Memory Uploading
The Module shall send a status flag to the "Power On Status Manager" containing the Uploading status: (READY TO START, IN PROGRESS, PERFORMED)

Table 5-36: PM Upload from EEPROM: Control Requirements

5.2 DPU BOARD DRIVER SOFTWARE

5.2.1 IEEE1355 MODULE

5.2.1.1 COMPONENT TYPE

Sources and headers file driving the IEEE1355. The functions shall allow to configure the IEEE1355 for receiving and transmitting data.

The module is composed of two sources and three headers files as follow:

- Ieee1355Config.c, Ieee1355Config.h
- Ieee1355TxRx.c, Ieee1355TxRx.h
- Ieee1355API.h

The module shall be compiled with relocable option. The User shall link in the Ieee1355Lib.a to the Application program and shall include the Ieee1355API.h header file.

5.2.1.2 FUNCTIONAL REQUIREMENTS

The module allocates the functional Software Requirement followings

Allocated SRD requirements
SRD-3.2.0.0-000
SRD-3.2.0.0-010
SRD-3.2.0.0-020
SRD-3.2.0.0-030
SRD-3.2.0.0-040
SRD-3.2.0.0-050
SRD-3.2.0.0-060
SRD-3.2.0.0-070
SRD-3.2.0.0-080
SRD-3.2.0.0-090
SRD-3.2.0.0-100
SRD-3.2.0.0-110
SRD-3.2.0.0-120
SRD-3.2.0.0-130
SRD-3.2.0.0-140
SRD-3.2.0.0-150
SRD-3.2.0.0-160
SRD-3.2.0.0-170
SRD-3.2.0.0-180
SRD-3.2.0.0-190
SRD-3.2.0.0-200
SRD-3.2.0.0-210

Table 5-37: Allocated SRD IEEE1355 Functional requirements

5.2.1.3 NON FUNCTIONAL REQUIREMENTS

The module allocates the Non functional Software Requirement followings

Allocated SRD requirements
SRD-4.1.1.0-000

Table 5-38: Allocated SRD IEEE1355 Non Functional Requirements

5.2.1.4 INTERFACES

The API interface functions are the followings, already specified in SRD

FUNCTION	DESCRIPTION
Smcs_ResetLink(smcs_ID,nLink)	Performs a complete reset of the link <nLink> of the <smcs_id> SMCS device.
Smcs_SetTimeOut (smcs_ID, timeout)	Set the time-out value for the SMCS; *Time out is only used if a read or write data is set to wait until completion.*
Smcs_OpenLink(smcs_ID,nLink)	Opens one of the three link of the SMCS
Smcs_CloseLink(smcs_ID,nLink)	Closes one of the three link of the SMCS
Smcs_StartLinkMaster(smcs_ID,nLink,speed)	Starts a Link as a Master at a specified Link <speed> : Links sends out NULLs tokens. A link cannot be started unless it is currently open,possible speed values are: Link_Speed_10000_Kbit 10 Mbit/sec Link_Speed_2500_Kbit 2,5 Mbit/sec Link_Speed_1250_Kbit 1,25 Mbit/sec
Smcs_StartLinkSlave(smcs_ID,nLink,speed)	Starts a Link as a Slave at a specified Link speed: This function first wait until it receives a NULL token from the corresponding Master Link. If the NULL is not received within the timeout period then it returns FALSE. Possible Link Speed values are: Link_Speed_10000_Kbit 10 Mbit/sec Link_Speed_2500_Kbit 2,5 Mbit/sec Link_Speed_1250_Kbit 1,25 Mbit/sec If the NULL is received within the time-out period, then the link is started.
Smcs_StopLink(smcs_ID,nLink)	Stops a currently running Link, This command stops a link transmitting which also causes a disconnect error in the other end of the link,
GetLinkStatus(smcs_ID, nLink)	Gets the status register for that Link (CHx_DSM_STAR)
Smcs_WriteLink(smcs_ID,nLink, pBuffer,ByteSize,bWait)	Transmits <ByteSize> fixed number of Bytes starting at address <pBuffer> over the link <nLink> of the <smcs_ID> device . If bWait is true the functions returns immediately, other way it returns when the transfer is completed or an error is detected. <ByteSize> determines the length of the packet transmitted ; it must be <= 16Kbytes / 6
Smcs_Read Link (smcs_ID,nLink, pBuffer,ByteSize,bWait)	Receives , over the link <nLink> of the <smcs_ID> device , <ByteSize> fixed number of Bytes and writes them in a user buffer starting at address <pBuffer>. If bWait is true the functions returns immediately, other way it returns when the transfer is completed or an error is detected. <ByteSize> determines the length of the packet received ; it must be <= 16Kbytes / 6
Smcs_GetReadStatus(smcs_ID,nLink,)	Returns the current transmit status for the specified link <nLink> Possible values: Transfer_not-started, Transfer_started, Transfer_Done, Transefr_error_diconnect, Transfer_error Parity, Transfer_error_timeout, Transfer_error_link_not _started, etc.)
Smcs_GetWriteStatus(smcs_ID,nLink,)	Returns the current receive status for the specified link <nLink> Possible values: Transfer_not-started, Transfer_started, Transfer_Done, Transfer_error_diconnect, Transfer_error Parity,, Transfer_error_timeout, Transfer_error_link_not _started, etcc.)

Smcs_ReadPackets(smcs_ID, nLink, pBuffer, ByteSize, NumPackets, bWait)	Reads a defined <NumPackets> over the link <nLink> of the <smcs_ID> device, and assigns the number of byte <ByteSize>. The data are stored in a buffer pointed by pBuffer. If <bwait> is true the functions return immediately, other way it returns when the transfer is completed or an error is detected.
Smcs_GetLastReadSize(smcs_ID, nLink)	Returns the number of received data over the link <nLink> of the <smcs_ID> device.
Smcs_GetLastWriteSize(smcs_ID,nLink)	Returns the number of transmitted data over the Link <nLink> of the <smcs_ID> device.
Smcs_GetLastPacketsNum(smcs_ID, nLink)	Returns the number of received packet over the link <nLink> of the <smcs_ID> device.
Smcs_WriteToBoardMemory(smcs_ID, startaddress, bytesize, pbuffer)	Writes <ByteSize> data, from the buffer pointed by <pbuffer>, in the <smcs_ID> DPRAM starting from <startaddress> of the <smcs_ID> device
Smcs_ReadFromBoardMemory(smcs_ID, startaddress, bytesize, pbuffer)	Reads <bytesize> data from <smcs_ID> DPRAM starting from <startaddress> and writes them in the buffer pointed by <pbuffer>
Smcs_WriteRegister(smcs_ID, offset, value)	Writes <value> in the <smcs_ID> with displacement <offset>, in a generic register
Smcs_ReadRegister(smcs_ID, offset)	Reads and returns the value contained in a register of the <smcs_ID> device with displacement <offset>

Table 5-39: IEEE1355 Application Interface functions

5.2.1.5 DATA ELEMENTS REQUIREMENTS

N/A

5.2.1.6 CONTROL REQUIREMENTS

N/A

5.2.2 MIL-STD-1553B MODULE

5.2.2.1 COMPONENT TYPE

Sources and headers file driving the MIL-STD-1553B. The functions shall allow to configure the MIL-STD-1553B as Remote Terminal only.

The module is composed of four sources and four headers files as follow:

- MilConfig.c, MilConfig.h
- MilRT.c, MilRT.h
- MilInit.c, MilInit.h
- MilAPI.c, MilAPI.h

The module shall be compiled with relocable option. The User shall link in the MilLib.a to the Application program and shall include the MilAPI.h header file.

5.2.2.2 FUNCTIONAL REQUIREMENTS

The module allocates the functional Software Requirement followings

Allocated SRD requirements
SRD-3.3.2.0-000
SRD-3.3.2.0-010
SRD-3.3.3.1-000
SRD-3.3.3.1-010
SRD-3.3.3.1-020
SRD-3.3.3.1-030
SRD-3.3.3.1-040
SRD-3.3.3.1-050
SRD-3.3.3.1-060
SRD-3.3.3.2-000
SRD-3.3.3.2-010
SRD-3.3.3.2-020
SRD-3.3.3.2-030
SRD-3.3.3.2-040
SRD-3.3.3.2-050
SRD-3.3.3.2-060
SRD-3.3.3.3-000
SRD-3.3.3.3-010
SRD-3.3.3.3-020
SRD-3.3.3.3-030
SRD-3.3.3.3-040
SRD-3.3.3.3-050
SRD-3.3.3.3-060
SRD-3.3.3.3-070
SRD-3.3.3.3-080
SRD-3.3.3.3-090
SRD-3.3.3.3-100
SRD-3.3.3.3-110
SRD-3.3.3.3-120
SRD-3.3.3.3-130
SRD-3.3.3.3-140
SRD-3.3.3.3-150
SRD-3.3.3.3-160

Table 5-40: Allocated SRD MIL1355 Functional requirements

5.2.2.3 NON FUNCTIONAL REQUIREMENTS

The module allocates the Non functional Software Requirement followings

Allocated SRD requirements
SRD-4.1.1.0-000
SRD-4.1.2.0-000
SRD-4.1.2.0-010
SRD-4.1.2.0-020
SRD-4.1.2.0-030

Table 5-41: Allocated SRD MIL1553 NF requirements

5.2.2.4 INTERFACES

The API interface functions are the followings, already specified in SRD.

FUNCTION	***DESCRIPTION***
RTOpen(Conf)	This sets up all global structures (Conf) used by the software driver and presets them to default values. The designated configuration structure is then set active and the routine returns a pointer to Conf structure.
RTClose(Conf)	Called at end of ACE operation. This routine clears up all global structures used by the Drivers. It deactivates the Conf "context".
RTAddressRead(Conf, Rtaddr)	It reads the RT address for the ACE Chip and returns it into Rtaddr.
RTsacw2word(Conf, Sacw)	It converts the information contained into subaddress control word (Sacw) data structure fields to a single 16bit word.
RTword2sacw (Conf, word)	It converts the information contained into a single 16bit word to a subaddress control word data structure fields.
RTDefSA(Conf, SubAddr, Sacw)	This routine is used to configure the memory management and interrupt schemes for the respective subaddress for transmit, receive and broadcast commands. Valid memory management schemes are SINGLE_MESSAGE and CIRCULAR_128. End of message interrupts may be generated by enabling the selected message type (ex: Sacw->RxEoMInt=YES). Circular buffer interrupts may be generated by enabling the selected message type (ex: Sacw->TxCircBuffInt=YES) provided.
RTConfigureMemory(Conf)	This functions will configure the DPRAM memory interfacing the ACE chip as defined by the user.
RTRun(Conf)	This function will clear the descriptor stack, it will reset the stack pointer and it will put the ACE device in running.
RTDefMsgLegal(Conf, MessType, subaddr, wc)	Sets the message legality to legal, based on the message subaddress, word count, and transmit/receive bit
RTDefMsgIllegal(Conf, MessType, subaddr, wc)	Sets the message legality to illegal, based on the message subaddress, word count and transmit/receive bit
RTIrqMsgSaEnable(Conf, sa, t_r, selection)	This routine will enable selected interrupts (EndOfMessage, Circular Buffer RollOver, SubAddress)
RTIrqMsgSaDisable(Conf, sa, t_r, selection)	This routine will disable selected Interrupt (EndOfMessage, Circular Buffer RollOver, SubAddress)
RTModeIrqEnable(Conf, broadcast, t_r, data, map);	This routine will enable selected mode code interrupts.
RTStop(Conf)	It stops the Remote Terminal
RTSelfTest(Conf)	This routine will perform a self Test of the RT internal register Stack Area and Lookup tables.
RTModeIrqDisable(Conf, broadcast, t_r, data, map)	This routine will disable selected mode code interrupts
Reset(Conf)	It performs a Software Reset of the ACE chip. It resets all the ACE register and the DPRAM area.
RTReadDescMsg(Conf, MessageNum, Message)	This routine will return a structure which contains the four word descriptor stack entry (block status word, time tag, data block

	pointer, and command word).
BuRTWriteEnhMCData(Conf, addr, data)	The ACE has separate data locations for mode codes. These data locations (from 110h-13fh) can be written by this routine.
RTReadEnhMCData(Conf, addr)	The ACE has separate data locations for mode codes. These data locations (from 110h-13fh) can be written by this routine
RTCreateFrame(Conf, FrameID)	Create a Frame pointer to manage the Frame of standard message packets.
RTAddMessageToFrame(Conf, FrameID, sa, t_r, word_count)	It adds Standard Message to the selected Frame. This function allows to create homogeneous frame (for Instance TM frame, or TC frame). NOTE: The functions will allow to add to the same frame, standard messages with different subaddress only. The functions will not allow to insert in the frame MODE CODE messages.
RTFrameRead(Conf, FrameID, buffer)	It reads the received messages of the frame and it writes them into the buffer
RTFrameWrite(Conf, FrameID, buffer)	It write the buffer data into the messages of the Frame.
RTReadSingleMessage(Conf, buffer, sa, WordCount)	It reads the single standard message and it puts the read data in the buffer.
RTWriteSingleMessage(Conf, buffer, sa, WordCount)	It write the data in the buffer into the single standard message.

Table 5-42: MIL1355 API Interface functions

5.2.2.5 DATA ELEMENTS REQUIREMENTS

N/A

5.2.2.6 CONTROL REQUIREMENTS

N/A

5.2.3 EEPROM MODULE

5.2.3.1 COMPONENT TYPE

Sources and headers files writing the EEPROM . The functions shall allow to write single cell and segments in EEPROM

The module is composed of one source and one header file as follow:

- EEPROM.c, EEPROM.h

The module shall be compiled with relocable option. The User shall link the EEPROMLib.a to the Application program and shall include the EEPROM.h header file.

5.2.3.2 FUNCTIONAL REQUIREMENTS

The module allocates the functional Software Requirement followings

Allocated SRD requirements
SRD-3.5.0.0-000
SRD-3.5.0.0-010
SRD-3.5.0.0-020
SRD-3.5.0.0-030
SRD-3.5.0.0-040

Table 5-43: Allocated SRD EEPROM Functional requirements

5.2.3.3 NON FUNCTIONAL REQUIREMENTS

N/A

5.2.3.4 INTERFACES

The API interface functions are the followings, already specified in SRD.

FUNCTION	***DESCRIPTION***
EEPROMWriteCell(Address, Data)	It writes a single EEPROM memory cell. It reads the same data to verify if the written data matches with the read data. it returns an error code if a mismatch is detected or 0 if the data is correct.
EEPROMClearCell(Address)	It clears a single EEPROM memory cell by writing 0x00000000. It returns an error code if the memory cell is not cleared.
EEPROMReadCell(Address)	It reads a single EEPROM memory cell and it returns the read value
EEPROMWriteSegment(NumberOfSegment, Header(structure), Data(buffer))	It writes an EEPROM segment including header and Data. The Users has to specify the Number of Segment, the Header parameters and the Data. The function writes the header and Data in the EEPROM and it computes and verifies the specified EEPROM FCS in the Header parameter with the computed FCS. In case of failure the function returns an error message.
EEPROMDeleteSegment(NumberOfSegment)	It deletes the EEPROM segment at NumberOfSegment address.

Table 5-44: EEPROM module API Interface functions

5.2.3.5 DATA ELEMENTS REQUIREMENTS

N/A

5.2.3.6 CONTROL REQUIREMENTS

N/A

5.2.4 WATCHDOG MODULE

5.2.4.1 COMPONENT TYPE

Sources and headers files setting and refreshing the board watchdog. The functions shall allow to set and refresh the watchdog

The module is composed of one source and one header file as follow:

- WDog.c, WDog.h

The module shall be compiled with relocable option. The User shall link the WDog.a to the Application program and shall include the WDog.h header file.

5.2.4.2 FUNCTIONAL REQUIREMENTS

The module allocates the functional Software Requirement followings

Allocated SRD requirements
SRD-3.6.0.0-000
SRD-3.6.0.0-010
SRD-3.6.0.0-020

Table 5-45: Allocated SRD Watchdog Functional requirements

5.2.4.3 NON FUNCTIONAL REQUIREMENTS

N/A

5.2.4.4 INTERFACES

The API interface functions are the followings, already specified in SRD.

FUNCTION	*DESCRIPTION*
WDSet(value)	It sets the delay time between two Watchdog triggers. The possible value are defined in [AD2]
WDRefresh(void)	It refresh the watchdog

Table 5-46: Watchdog API interface functions

5.2.4.5 DATA ELEMENTS REQUIREMENTS

N/A

5.2.4.6 CONTROL REQUIREMENTS

N7A