3. ON-BOARD COMPUTER

3.1 OVERVIEW

The On-Board Computer system of SRMSAT-2: Mission to the Moon is responsible for platform data processing, communication handling, payload data handling, propulsion maneuvering, interfacing with various components and attitude control operations that are executed throughout the mission life. The software architecture provides a functional overview of the entire On-board software system. The On-board data handling and management forms a crucial part of the satellite operations which includes interfacing with most of the components. The microcontroller plays a critical role in computation; scheduling and carrying out various tasks; collecting and organizing telemetry data. Thus, the software should be developed around a microcontroller that has desired characteristics and features. Efficient communication between space and ground segments is achieved by the usage of recommended standards like CCSDS, which increase inter-operability and efficiency. Error detection and correction schemes are to be implemented for correcting bit flips that might occur as a result of exposure to harsh environments.

3.2 MICROCONTROLLER SELECTION

Power constraints

Essentially, microcontrollers should exhibit very low power consumption. Allocated power budget should account for the presence of two microcontrollers (primary and flight dynamics) along with other digital components of the On Board Computer such as multiplexers and external memory while having a margin of about 30%.

Processing Power (MIPS)

Processor speed is denoted by the number of instruction cycles that can be completed within a second. Performance is a major issue when it comes to managing all the satellite operations. The faster the processing capability of the microcontroller, the better is the response. This is directly linked to the clock speed and power consumption of the system.

Interface Hardware support

The on board computer should be able to interface with components from different subsystems using I^2C, SPI, and GPIO. This assists simultaneous communication between systems and prevents address conflicts between devices.

Temperature and Radiation Handling

The microcontroller operating temperature must lie within acceptable temperature ranges. Also, adequate radiation protection will be provided by the satellite's radiation shield.

Development Environment

Having a conducive development environment is of paramount importance during the development phase. A good development environment is essential for enhancing software development and debugging processes of microcontrollers.

PARAMERER	ATSAMA5D35-CM
Maximum Operating Frequency	532
Max I/O Pins and Ext Interrupts	160
Maximum Power Consumption(@)	1.144
Temperature Range	-40° to 85°C
No. of SPI and I^2C interfaces	6 and 3
ADC Channels	12
ADC Resolution(bits)	12
ROM(Kbytes)	160
SRAM (Kbytes)	128

Table 3.1. Selected Microcontrollers

The ATSAMA5D35-CM was selected as the microcontroller to be used for the mission

Subsystem Functionality and Interfacing

The decision for using two microcontrollers of the same model has been made due to various reasons brought out in the following paragraphs.

The use of two microcontrollers ensures redundancy and a fail-safe system while also reducing the computational load on the microprocessors. The primary or Master microcontroller is used for the TTC, Thermal, Payload and Power operations and the secondary microcontroller to perform ADCS and propulsion operations. In the unlikely event for one of the microcontrollers to fail, the other takes over its operations. This is realized by setting up common connections and a common data pool.

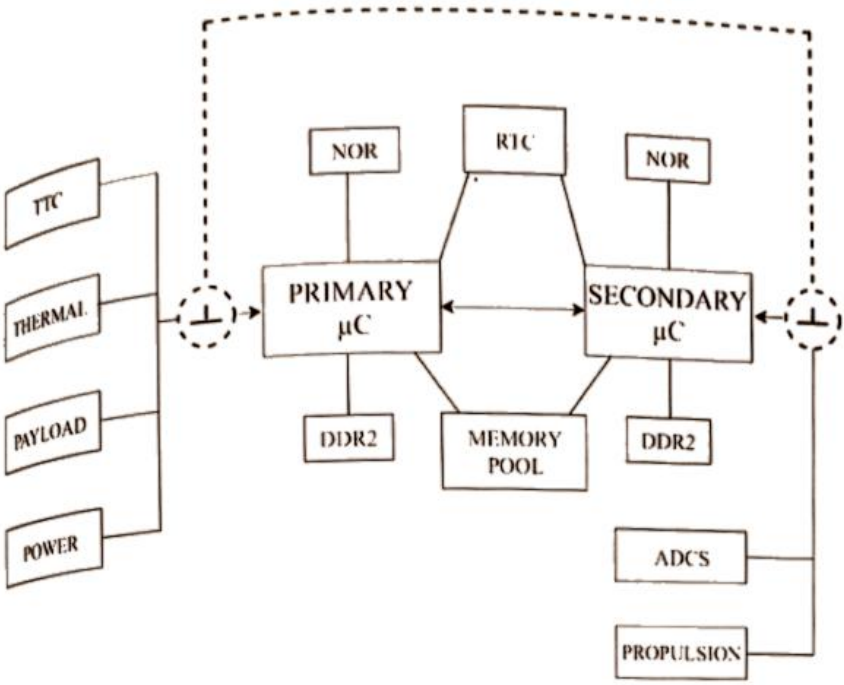

Figure 3.1: Interfacing between the microcontrollers

The OBC is responsible for interfacing with all the different subsystems, establishing communication, performing computations and making decisions. The interfacing details between the various components and the OBC have been aggregated with respect to each subsystem.

Symbols	
◄——►	I2C
◄- - -►	RS422
◄·····►	SPI
- - - -►	GPIO
%	Data*
#	Command**

Figure 3.2: Legend for Subsystem Interface Diagrams

* % represents data transfer between the component and the microcontroller.

** #represents Status Commands from the microcontroller.

Status command stands for a digital 1/0 command to the components. Status commands like Sun Sensor status, MPPTU status, etc. send request from µC to components for data retrieval or control configurations.

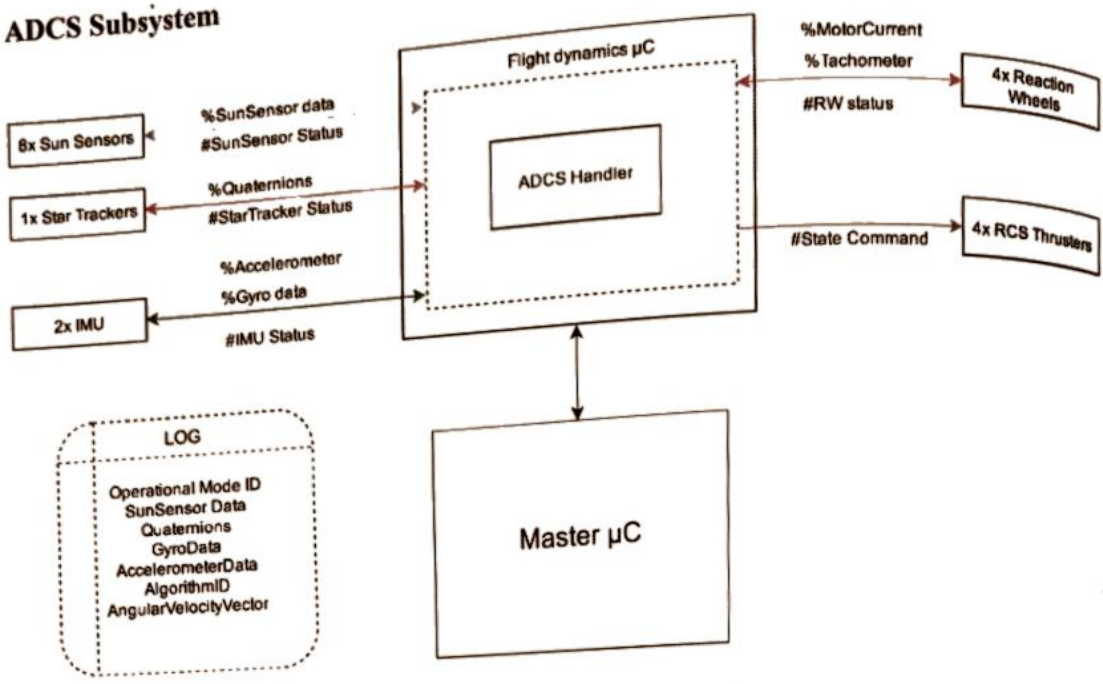

Figure 3.3: ADCS Subsystem interface

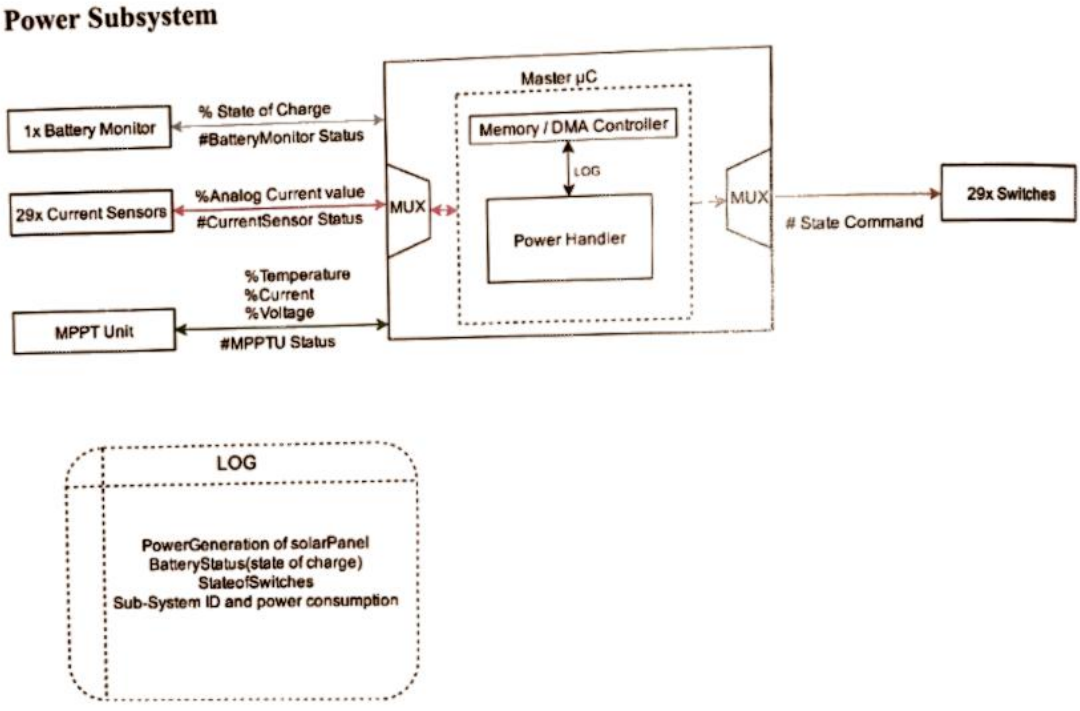

Figure 3.4: Power Subsystem Interface

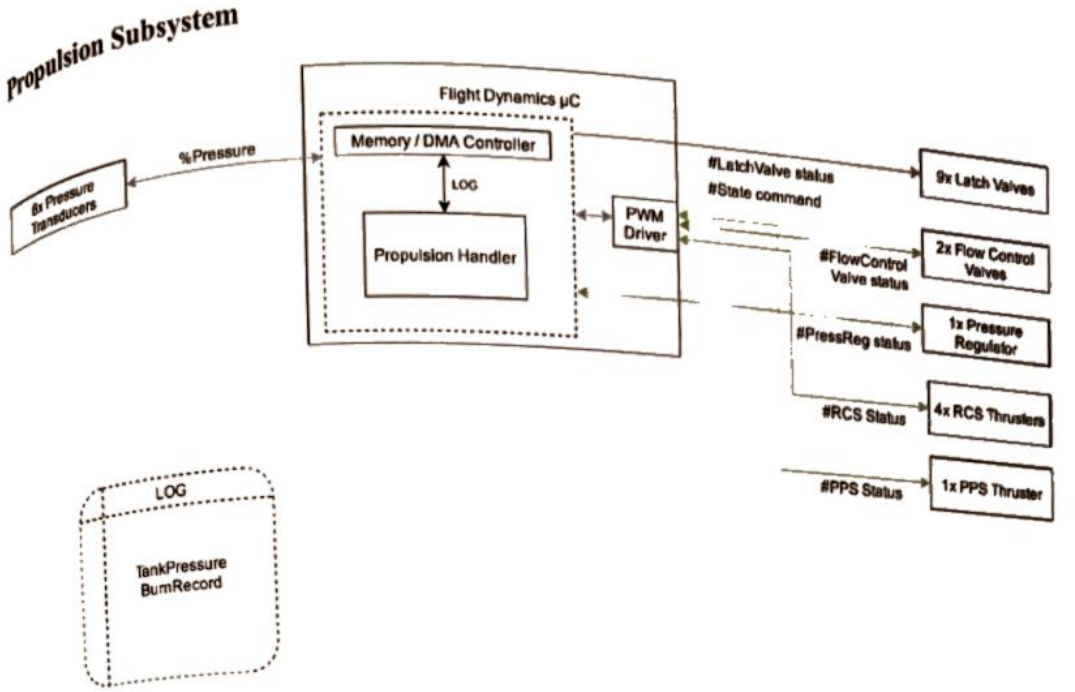

Figure 3.5: Propulsion Subsystem Interface

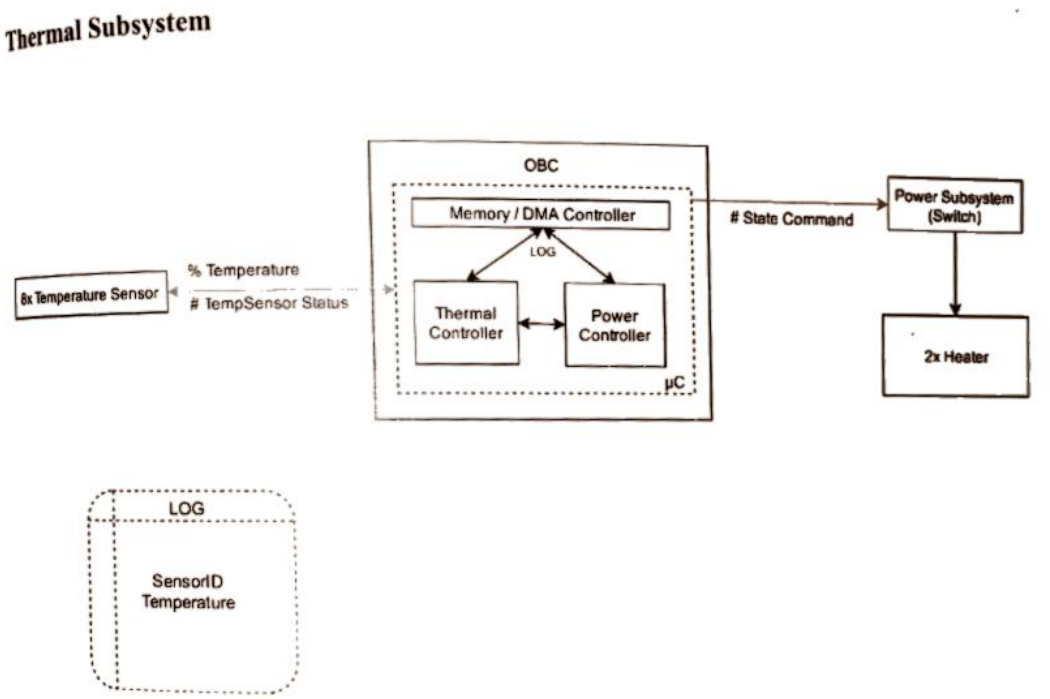

Figure 3.6: Thermal Subsystem Interface

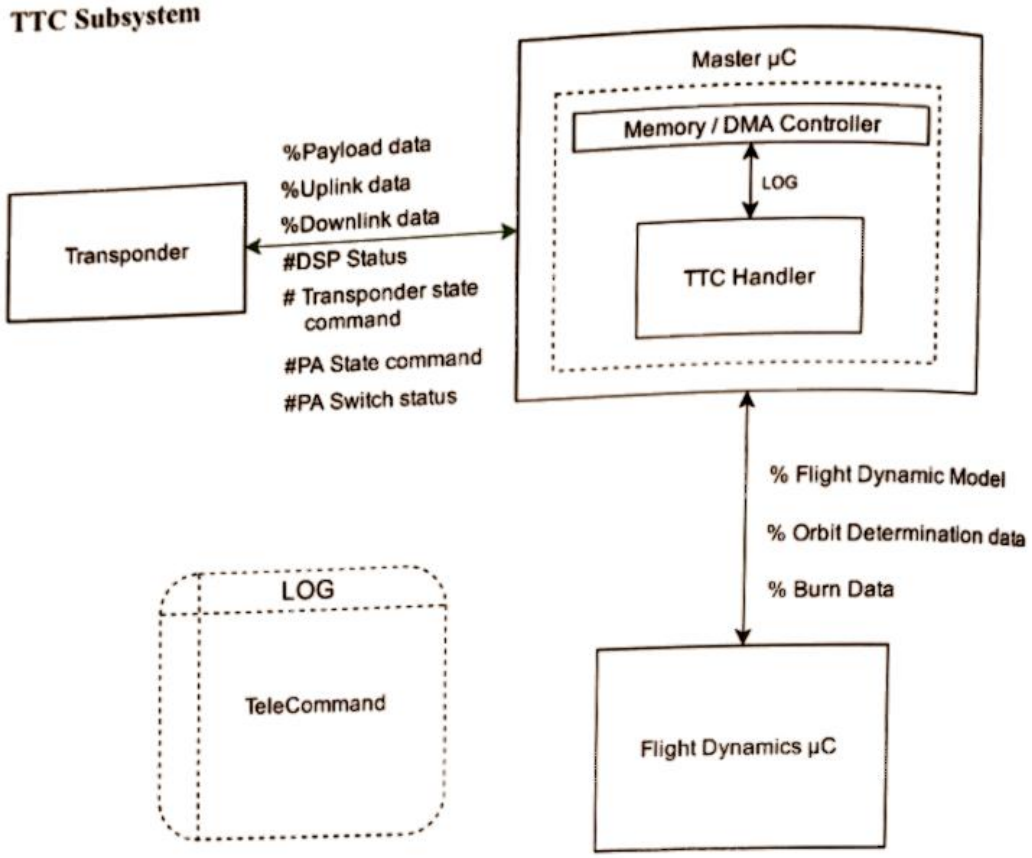

Figure 3.7: TTC Subsystem Interface

3.2 MEMORY BUDGET

Requirements

The following are the requirements based on the type, size; and read and write access speeds of the different memories*.

*Larson, J. W., Wertz, R. J., "*Space Mission Analysis and Design*", Third Edition, Microcosm Press and Kluwer Academic Publishers, Tables. 16-13, 16-15.

	Type	Size	Speed
Program Memory	Non-volatile	~4 Mbyte	Fast read
Data Memory	Volatile	~3.5 Mbyte	Fast read and write
Bootable Memory	Non-volatile	64 Kbyte	Fast read
Redundant Memory	Non-volatile	~16 Mbyte	Fast read
Log Memory	Non-volatile	TBD	Fast write
Payload Memory	Non-volatile	TBD	Fast read and write

Table 3.3: Requirements for different memory types in the SRMSAT-2

		Size (Kwords)		Complexity		Size (Kwords) rev.	
	Functions	code size	data size	code complexity	data complexity	code size	data size
Communications	command processing	1	4	4	2	4	8
	telemetry processing	1	2.5	4	2	4	5
Attitude sensor processing	rate gyro	0.8	0.5	2	2	1.6	1
	sun sensor	0.5	0.1	2	2	1	0.2
	star tracker	2	15	2	2	4	30
Attitude determination & Control	kinematic integration	2	0.2	2	2	4	0.4
	error determination	1	0.1	2	2	2	0.2
	precession control	3.3	1.5	2	2	6.6	3
	thruster control	0.6	0.4	2	2	1.2	0.8
	reaction wheel ctrl	1	0.3	2	2	2	0.6
	ephemens propagation	2	0.3	2	2	4	0.6
	orbit propagation	13	4	4	4	52	16
Autonomy	simple autonomy	2	1	2	2	4	2
Other functions	power management	1.2	0.5	3	2	3.6	1
	thermal control	0.8	1.5	2	2	1.6	3
Fault detection	monitors	4	1	2	2	8	2
	fault correction	2	10	4	2	8	20
Payload	payload sub	8	40	2	2	16	80
					TOTAL	127.6	173.8
Operating System	executive functions	3.5	2	4	2	14	4
	Run-time kernel	8	4	6	4	48	16
	Equipment Handlers	28	9.8	2	2	56	19.6
	test diagnostics	0.7	0.4	2	2	1.4	0.8
	math utilities	1.2	0.2	2	2	2.4	0.4
					TOTAL	121.8	40.8

PARAMETERS	
d (number of equipment handlers)	14

TOTAL (in Kwords)	249.4	214.6
TOTAL (in Kbyte)	997.6	858.4

Table 3.4: Program & Data Memory Estimate

Memory Budget Calculation

The various memory options provided by the proposed microcontroller CPU module, ATSAMA5D35-CM, have been analyzed for the required memory types. The memory budget is estimated based on literature study[1]

	Memory Device	Memory Size
Program Memory	NOR Flash - External *	16 Mbyte
Data Memory	DDR2 SDRAM **	512 Mbyte
Bootable Memory	ROM - Internal ***	160 Kbyte
Redundant Memory	NAND Flash (with ECC)	256 Mbyte
Log Memory	- External	
Payload Memory		

Table 3.5: SAM

* As per our estimates, the 160 Kbyte internal ROM is not sufficient to hold the program memory. Hence, we will store the program memory in the 16 Mbyte NOR flash. The program in the NOR flash can be executed by either using eXecuteIn Place (XIP) or by copying into RAM before execution.

** Since the 128 Kbyte SRAM is not sufficient for the proper functioning of the program as per estimates, the DDR2 RAM interfaced with the microcontroller is added. The size of the data memory is more than the required because the SAMA5D35-CM is available only in these specifications.

*** The microcontroller boots by default into this memory. This will contain the bootstrapping code for loading and transferring the flow of control to the program memory.

[1] Larson, J. W., Wertz, R. J., "*Space Mission Analysis and Design*", Third Edition, Microcosm Press and Kluwer Academic Publishers, Tables. 16-13, 16-15.

3.3 SOFTWARE ARCHITECTURE

An On Board Software(OBSW) of a spacecraft platform is the program which controls the OBC and implements the following functions:

- Telecommand processing for spacecraft command from ground
- Generation of data for telemetry
- Spacecraft Control comprising of:
  - Attitude and orbit control
  - Power control
  - Thermal control
  - Payload instruments operation
- System status monitoring
- Fault detection, isolation, and recovery

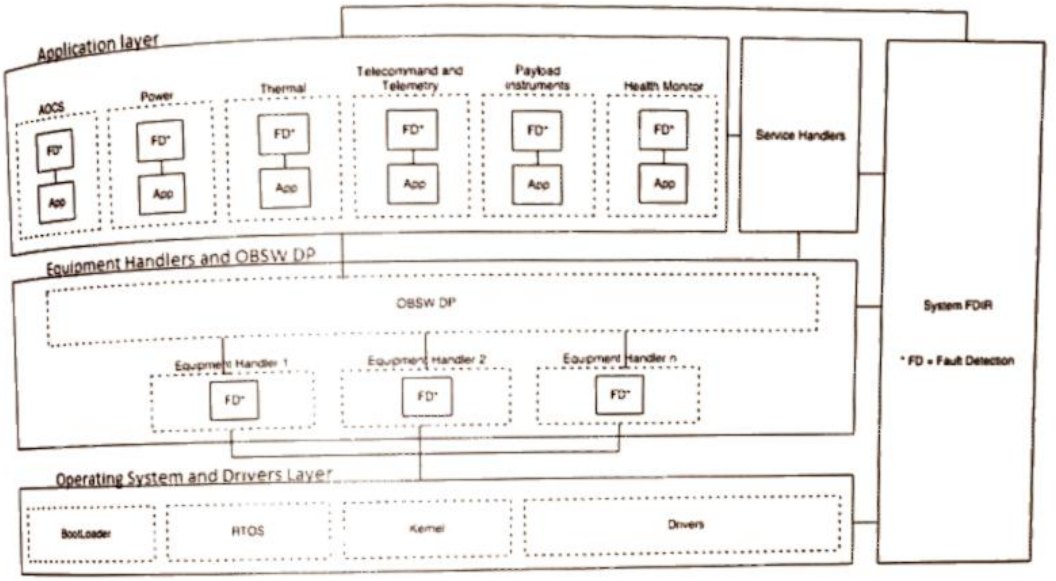

Figure 3.8: Software Architecture

The representation of static architecture with the functional modes tied to various OBSW hierarchy levels provide an adequate description of the various functional modules and their interfaces.

The OBSW static architecture consists of the following main components:

- Operating system and drivers layer
- Equipment Handlers and OBSW Data Pool (OBSW DP)
- Application Layer
- Service Handlers
- Fault detection, isolation, and recovery

Operating System and Device Drivers Layer

This layer comprises of the Bootloader, the RTOS, the Kernel and the drivers. The bootloader programmed in the boot memory and will initialize the microcontroller to be able to boot the Real-Time Operating System (RTOS) and the OBSW layers built on top, from the program memory. The RTOS is booted first, and the non-operating system part of the OBSW is started after RTOS.

Drivers for I/O channels are interfaced between the OBSW and hardware. These include I^2C, SPI, serial, etc. drivers. These are interface drivers.

Equipment Handlers and OBSW Data Pool (OBSW-DP)

Above the RTOS, kernel and the drivers, reside the equipment handlers. They perform command writing from the OBSW via drivers to the connected devices like actuators, switches, etc. Furthermore, the equipment handlers will perform cyclic or on request onboard telemetry acquisition.

The OBSW data pool (DP) is a vector containing all operational variables which are processed inside the OBSW such as system time, sensor data, actuator data, temperatures, current and voltages, payload data, etc. The payload data storage is isolated and fixed and not shared by other telemetry data. It is mandatory that the variable name in both the OBSW DP and the OBSW code should be same. The same name should be used in telecommand and telemetry packets too. A flag corresponds to outdated data in the DP when handlers are unable to access an updated data from the connected device.

Application Layer

The application layer (above the OBSW DP) controls:

- AOCS
- Power subsystem
- Thermal control
- Telecommand and Telemetry
- Payload instruments
- Health monitor

The applications read input data from the OBSW-DP and place computed output back into other variables of the OBSW DP. Each application has access to its own defined OBSW DP variable. Some variables are shared with all the applications like system clock on board time. Each application runs in different threads with different update cycles which depend on the request. Each application is defined to perform according to the commanded operational mode.

Health monitor primarily focusses on health check-ups of OBSW. It stores the following information as pair of name and value:

- Preselected set of OBSW variables from each domain of OBSW DP
- Memory pointer, timing parameters, RTOS flags and other OBSW internal variables
- Task scheduling parameters for selected applications
- Data bus access flags, etc.

The objective of the health monitor is to:

- Check for proper functioning of the threads
- Ensure Boundary limitations on the initialized parameters
- Direct logging of the parameters

Service Handlers

The service handlers reside beside the application layer. It is dedicated to extract and process the services mentioned in the telecommands, and service tag ID with the telemetry for sending a report of respective services to the ground station. Services refer to on-board software commands defined uniformly and distinctly both in the ground station and the spacecraft. Here, PUS (Packet Utilization Standard) defined by ECSS standards are considered as the service handlers. It reserves numbers 0-127 to represent pre-defined services and numbers 128-255 are free for mission specific use. Each of the services has a dedicated handler. For example, as per PUS, Service 1 represents telecommand verification service. So, whenever the service 1 is called, telecommand verification is handled by the specific handler assigned to it. Such service handlers need a proper mechanism to route telecommand/telemetry between different applications to perform the required service. Unique application IDs are defined for routing of services from the service handlers to other components in the OBSW.

Fault Detection, Isolation and Recovery (FDIR)

FDIR functionality is implemented on several levels inside the OBSW. Fault detection is performed in all applications and handlers. Based on fault detection, fault isolation and recovery involves:

- Compliance with Electrical Power System to handle electrical faults and failures
- Triggering a reconfiguration to the redundant counterpart of the failed equipment
- Switching to Safe Mode in case of low power or during long duration of eclipse

Requirements for FDIR design include:

- A definition of clear hierarchy, identification of failure type and management on the particular FDIR level.
- Ability of the Ground Station to perform a detailed status analysis and failure event history analysis for unique failure identification.
- Ability of the Ground Station to alter operational limits to avoid future Safe Modes – e.g. in cases of failures triggered by equipment degradation.
- Provision for OBSW patch and dumps
- Accountability for all the envisioned failures during spacecraft operations based on their symptom sets

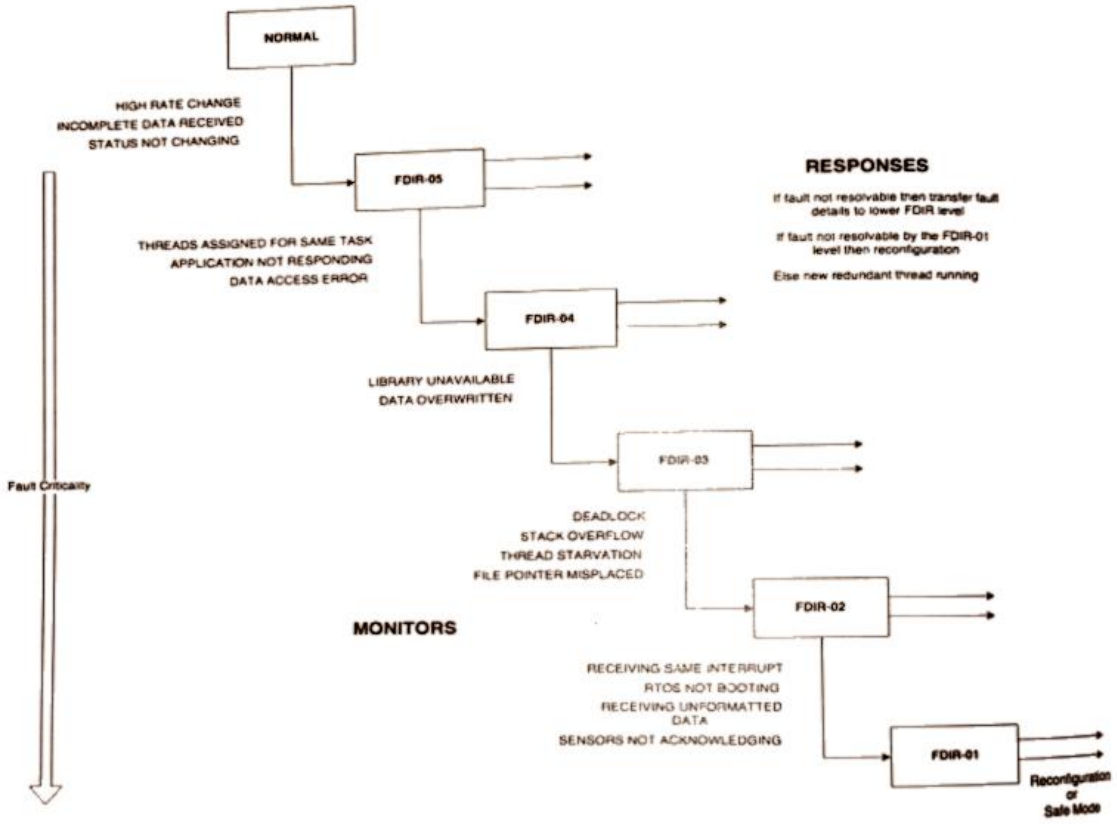

Figure 3.9: FDIR Hierarchy Level

Figure 3.9 depicts the identification and propagation of faults through different levels and the recovery process that follows. If recovery fails even after level 1, a system reconfiguration is done, or the operational mode switches to safe mode.

Triple Modular Redundancy will be used for Error Detection and Correction.