**Appendix**

**Data-driven Mutation Testing:**

**LuxSpace Case Study**

This Appendix describe the procedures adopted to execute data-driven mutation testing on the LuxSpace case study system. Section 1 provide a detailed overview of the case study and the function targeted by data-driven mutation testing. Section 2 describes the fault models defined for the case study. Section 3 describes the integration of mutation probes into ADCS\_IF\_SW.

# 1. Overview of the case study

In the case of LXS, data-driven mutation testing is applied to assess the quality of the test cases that exercise the ADCS software interface of the ESAIL system (hereafter, ADCS\_IF\_SW). In ESAIL, the ADCS\_IF\_SW is used to manage and collect data from hardware devices (e.g., sensors). Detailed specifications for the ADCS interface appear in the document *ESAIL-LXS-ICD-P-0184* *ADCS IF SW External ICD.*

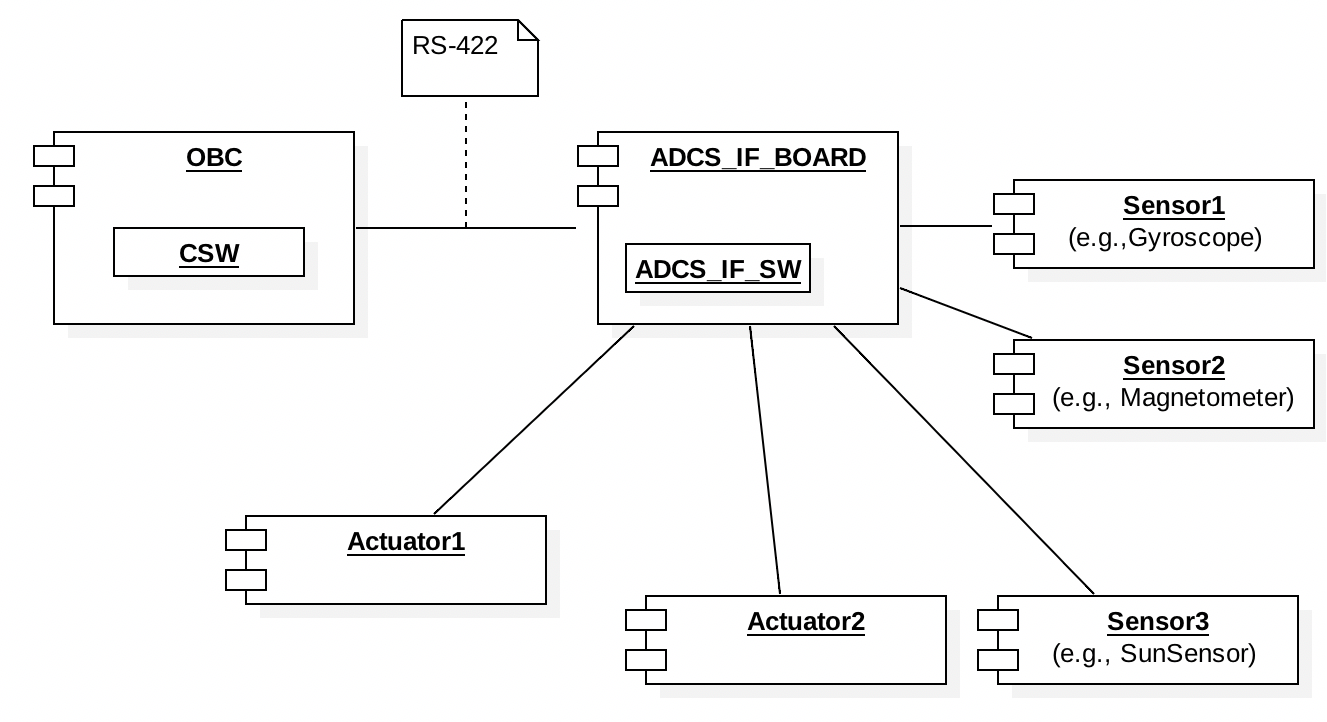


Figure 1: OBC-ADCS integration in ESAIL

Figure 1 provides an overview of the integration between ESAIL OBC and the ADCS board. ESAIL CSW (central software) runs on an onboard controller (OBC) with a Leon 3 microprocessor. The OBC is connected to ADCS interface boards (ADCS\_IF\_BOARD) through RS-422. The ADCS\_IF\_BOARD runs its own controller (ADCS\_IF\_SW). Each board processes data received from sensors and controls actuators. The ADCS\_IF\_SW is the target of data-driven mutation testing and is the software layer where mutation probes are installed.

The ADCS\_IF\_SW implements functions used by the OBC to send data to devices (i.e., set their configuration) and functions that send devices data to the OBC.

The function of the ADCS\_IF\_SW that manages the communication between the ADCS and the OBC, i.e., *ObcRecvBlockCb*. The function is implemented in file *AdcsIf.c*.

The SVF simulator used for testing runs the OBC software but it simulates the behaviour of the ADCS\_IF\_SW. The ADCS\_IF\_SW is not executed inside the SVF but only simulated.

The ESAIL system test suite contains test cases that exercise the integration between OBC and the ADCS\_IF\_SW but the ADCS\_IF\_SW is not actually run. The test suite that exercises the ADCS\_IF\_SW is one that should execute with hardware in the loop.

Since the functions that send data to the devices are tested with hardware in the loop, in the context of FAQAS, we will apply data-driven mutation testing only to verify the functions used by the ADCS to send data to the OBC.

Although in principle also messages from the OBC to the ADCS\_IF\_SW could be tested, the current test suite, which does not run the ADCS\_IF\_SW prevents it. Indeed, the simulator used in the current test suite makes assumptions about the messages received thus it would be very easy to break it by altering its input messages. To alter the messages sent from OBC to ADCS\_IF\_SW it would be necessary to (1) use a simulator that actually runs the ADCS\_IF\_SW or (2) target the test cases that include hardware in the loop.

Case (2) above, i.e., testing with hardware in the loop, is technically feasible because it is just a matter of deploying on the hardware a modified software that performs the mutation. However mutated packets may break some of the assumptions made when developing the software and thus break the hardware (e.g., altering the voltage of the board). For every mutation to be performed it might be necessary to ensure that the hardware is not going to be damaged. Such type of testing might thus be out of the budget for the current project and may require a dedicated project by itself.

The implementation of function *ObcRecvBlockCb* is shown in Section 1.1. It mainly consists of a switch command (line 138) that generates a response for the OBC after invoking a *data generation method* selected according to the request received on the data link. For example, Line 146 invokes method *GetIfStatus*, which prepares a response packet containing the information about the ADCS status.

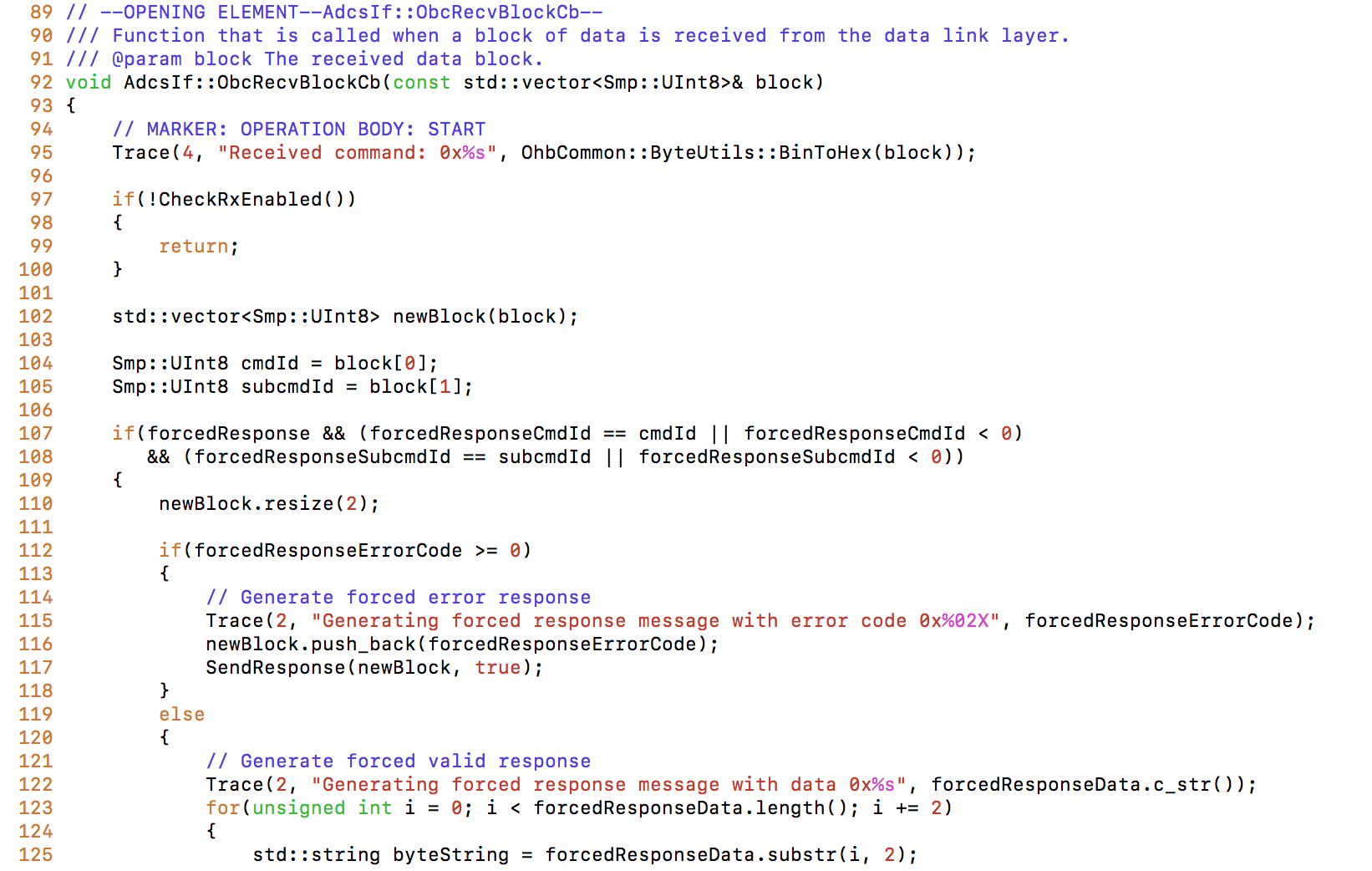
Each *data generation method* receive as input an object of type *std::vector* (i.e., the object *newBlock)* that will be used to store the data to be sent to the OBC. The vector *newBlock* acts as a buffer; it contains elements of type *UInt8*, the length of the vector matches the size of the response message indicated in *ESAIL-LXS-ICD-P-0184 (one element per byte).* Table 1 reports, for each feature targeted by data-driven mutation testing, the page in *ESAIL-LXS-ICD-P-0184 that describes the data format, the* ADCS\_IF\_SW function that fill the content of message, and the size of the response message (i.e., the length of *std::vector*).

Table 2:Features targeted by data-driven mutation testing and message size

|  |  |  |  |
| --- | --- | --- | --- |
| **ADCS Feature** | **Page** | **ADCS\_IF\_SW function** | **Message size (bytes)** |
| ADCS IF Status | 19 | GetIfStatus | 6 |
| ADCS IF HK | 22 | GetIfHk | 37 |
| GYTM - Gyroscope TM | 34 | GetGyroTm | 21 |
| MMTX - Magnetometer TX | 41 | GetMgtmTm | 2 |
| Sun Sensor TM | 42 | GetSsTm | 48 |
| SSTP - Sun Sensor Temperature | 45 | GetSsTemp | 12 |
| Reaction Wheel TX | 50 | GetRwTm | 2 |
| SpaceCraft HK | 60 | GetIfScHk | 18 |
| Magnetorquer Set PWM RSP | 57 | GetMgtqTm | 39 |

Each invocation of a *data generation method* generates a response (i.e., the vector *newBlock*) that may either contain the desired result or an error code. The response generated in the first case is referred to as *nominal response message*, the response generated in the second case is an *error response message*. The reponse message is sent to the OBC through the invocation of function SendResponse (Lines 298 and 312). When an error code is generated, the data generation method returns *CR\_Failure*. The response code is read by function *ObcRecvBlockCb* to determine if it is necessary to trim the buffer before sending back to OBC; this behaviour is handled by the parameter *true* passed to SendResponse (Line 312).

## 1.1 Function ObcRecvBlockCb











# 2. Fault Model

In the case of ADCS\_IF\_SW we have defined a total of 18 fault models, two for each feature listed in Table 1. For each feature, one fault model captures the fault that might affect a nominal response message, one fault model captures the faults that might affect an error response message.

In the following sections we describe the fault models by providing for each byte of the response message (column *Byte*), the relevant bits (column *Bit*), a description of the information that is supposed to be transmitted by the byte (column *Description*), the type of data written on the byte (column *Type*), the fault classes that might affect the byte (column *Fault class*). We do not report the span of the item since it can be deducted from the table; indeed, descriptions that span over multiple rows correspond to data types that, to be loaded, require the readin of multiple data items. Concerning data types, the type DOUBLE is used for data items that internally to ESAIL are represented using the type Smp::Float64. On the channel, Smp:Float64 is transmitted as *<****PTC****=3,* ***PCF****=6> Unsigned Integer 10bits, which in the code is represented with Smp::Int16.*

For each fault class, we indicate the value of the parameters required to configure the corresponding mutation operator (see Table 2.1 of D2). We use the keyword @MIB to

indicate that the parameter value should be derived from the MIB database for ESAIL, more precisely from the file OCP.dat. In the database, the min and max range value for the nominal cases are reported. For example, Figure 2 shows a portion of the OBC.dat from which we can determine that MIN and the MAX values for AIFN031U are 3 and 3.6, respectively. The delta (i.e., parameter D) is coincides with the lowest positive number that can be represented with the number of decimals appearing in the rage (e.g., 0.1 for AIFN031U and 0.01for AIFN031U). for For some of the data items in the table we report also the corresponding identifier in OBC.dat. Missing identifiers will be reported in the coming months while refining the approach; indeed, decisions on the data items to be addressed by the approach may change after the first preliminary tests.

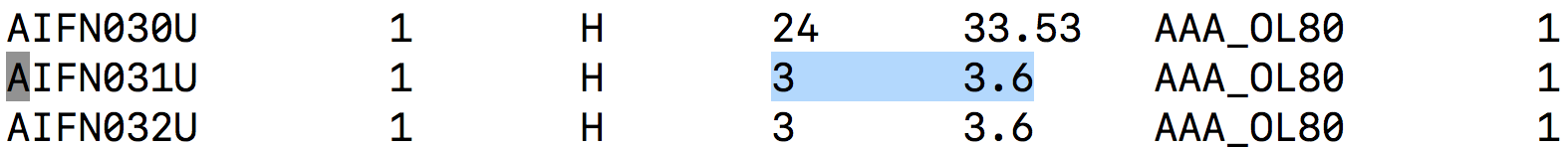


Figure 2: Portion of OBC.dat

In column Fault class, the label NONE indicates that we are not interested into performing data-driven mutation testing for that specific byte. In general, we do not target with data-drivem mutation those data items that do not concern features covered by the test suite. These are typically data items that do not cause a crash of the on board software or data items used only for self-testing of the board.

Columns Byte, Bit, and Description match the columns of corresponding tables in *ESAIL-LXS-ICD-P-0184.*

## 2.1 ADCS IF Status

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Byte** | **Bit** | **Description** | **Type** | **Fault class** |
| 1 | 2..0 | Reset Source  Provides information about last reset. The bit is cleared after the first read of the status  0 = No reset  1 = Power-on Reset  2 = External Reset (released by JTAG adapter)  3 = Watchdog Reset  4 = Brown-out Reset  5 = JTAG AVR Reset (logic reset by JTAG)  6 = Not used  7 = Not used | BIN | BF(MIN=3;MAX=3)  BF(MIN=4;MAX=4)  BF(MIN=5;MAX=7) |
| 3 | ADCS IF ready  This bit is set when ADCS is ready to read/write to units. In the boot of the ADCS IF shall be a time to initialize all modules and units. After initialization of the ADCS IF, modules and units, shall go to a ready state.  While ADCS IF is not ready, the available commands are:   * ASST * ASHK * ASCT |
| 4 | OBC communication error  This bit is set if a communication error between OBC and ADCS IF occurred in the last command. The bit is cleared after the first reading of the status  0 = No error  1 = Communication error |
| 7..5 | Unit communication error  This bit is set if a communication error between ADCS IF and ADCS unit occurred. The bit is cleared after the first read of the status  0 = No error  1 = Communication error |
| 2 | 7..0 | Unit in error  Provides a list of units in error.  0 = No error  1 = Unit error  Each bit is assigned to one unit:  Bit 0 = Gyroscope unit  Bit 1 = Reaction Wheel  Bit 2 = Magnetorquer  Bit 3 = Magnetometer  Bit 4 = Sun Sensor | BIN | BF(MIN=0;MAX=4) |
| 3 | 7..0 | Watchdog Reset Counter  Watchdog Reset counter value.  Increment in every watchdog reset.  Value is stored in non-volatile memory  To clear watchdog reset counter, shall be used the ASCF command. | INT | None: ESAIL OBC does not deal with anomalous values of reset counters. Thus we do not expect ESAIL test suite to fail in case of a high reset counter.. |
| 4 | 7..0 | Overall Reset Counter  Overall reset counter value.  Increment in every device reset.  Value is stored in non-volatile memory  To clear overall reset counter, shall be used the ASCF command. | INT | None: same as above. |
| 5 | 1..0 | Gyroscope enable  Enable/Disable status of nominal or redundant bus transceiver.  0 = Disabled both transceivers  1 = Enabled nominal transceiver only  2 = Enabled redundant transceiver only  3 = not existing (reserved for future needs) | BIN | BF(MIN=0;MAX=2)  BF(MIN=2;MAX=4)  BF(MIN=5;MAX=7) |
| 4..2 | Reaction Wheel enable  Enabled/Disabled status of bus transceiver.  0 = Disabled transceiver  1 = Enabled transceiver  7..2 = not existing (reserved for future needs) |
| 7..5 | 3 axis Magnetorquer enable  General Enable/Disable status of the Magnetorquer Driver for all three axis.  0 = Disabled  1 = Enabled  Bit assignement:  Bit 0 = Enabled/Disabled Driver  Bit 1 = 0 not used (reserved for future needs)  Bit 2 = 0 not used (reserved for future needs) |
| 6 | 1..0 | Magnetometer enable  Enable/Disable status of nominal or redundant bus transceiver.  0 = Disabled both transceivers  1 = Enabled nominal transceiver only  2 = Enabled redundant transceiver only  3 = not existing (reserved for future needs) | BIN | BF(MIN=0;MAX=1)  BF(MIN=2;MAX=7) |
| 7..2 | Sun Sensor board ADC enable  Enabled/Disabled Sun Sensor board ADC, see Note 3)  0 = Disabled  1 = Enabled  Each bit is assigned to one ADC:  Bit 0 = Enabled/Disabled ADC2  Bit 1 = Enabled/Disabled ADC3  Bit 2 = Enabled/Disabled ADC4  Bit 3 = Enabled/Disabled ADC5  Bit 4 = Enabled/Disabled ADC6  Bit 5 = Enabled/Disabled ADC7 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Byte** | **Bit** | **Description** | **Type** | **Fault class** |
| 1 | 7..0 | Error type  . | HEX | IV(VALUE=0x51)  IV(VALUE=0x52)  IV(VALUE=0x53)  IV(VALUE=0x54)  IV(VALUE=0x55)  IV(VALUE=0x57)  IV(VALUE=0x58)  IV(VALUE=0x59)  IV(VALUE=0x5A)  IV(VALUE=0x5B)  IV(VALUE=0x5C) |

## 2.2 ASHK - ADCS IF HK

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Byte** | **Bit** | **Description** | **Type** | **Fault class** |
| 1 | 7..0 | VCC1N  OBC Nominal transceiver circuit voltage |  | NONE |
| 2 | 7..0 |  |  |
| 3 | 7..0 | VCC1R  OBC Redundant transceiver circuit voltage |  | NONE |
| 4 | 7..0 |  | NONE |
| 5 | 7..0 | VCC2  Gyroscope transceiver/UART circuit voltage |  | NONE |
| 6 | 7..0 |  | NONE |
| 7 | 7..0 | VCC3  Magnetometer transceiver/UART circuit voltage |  | NONE |
| 8 | 7..0 |  | NONE |
| 9 | 7..0 | VCC4  Reaction Wheel transceiver/UART circuit voltage |  | NONE |
| 10 | 7..0 |  | NONE |
| 11 | 7..0 | VCCa  Internal power supply (5.5V), measured with ADC0 |  | NONE |
| 12 | 7..0 |  | NONE |
| 13 | 7..0 | VCCb  Internal power supply (5.5V), measured with ADC1 |  |  |
| 14 | 7..0 | DOUBLE | VAT(T=@MIB;D=@MIB)  ID: AIFN031U |
| 15 | 7..0 | VBUS  Unit input bus voltage | DOUBLE | VAT(T=@MIB;D=@MIB) |
| 16 | 7..0 |  |  |
| 17 | 7..0 | VCC5  Supply voltage for ADC2, ADC3, ADC4 and VCCB1.  Sun-sensor PCB |  | NONE |
| 18 | 7..0 |  | NONE |
| 19 | 7..0 | VCC6  Supply voltage for ADC5, ADC6, ADC7 and VCCB2.  Sun-sensor PCB |  | NONE |
| 20 | 7..0 |  | NONE |
| 21 | 7..0 | VCC5\_IN  LDO input voltage for ADC2, ADC3, ADC4 and VCCB1.  Sun-sensor PCB |  | NONE |
| 22 | 7..0 |  | NONE |
| 23 | 7..0 | VCC6\_IN  LDO input voltage for ADC5, ADC6, ADC7 and VCCB2.  Sun-sensor PCB |  | NONE |
| 24 | 7..0 |  | NONE |
| 25 | 7..0 | VCC\_SW1  SSB internal switched power supply, measured by ADC3  Remark: the voltage VCC\_SW is measured 2 times with two different ADC. This allows to compare the results and conclude for a drift in the ADC’s. |  | VAT(T=@MIB;D=@MIB) |
| 26 | 7..0 |  |  |
| 27 | 7..0 | VCC\_SW2  SSB internal switched power supply, measured by ADC6  Remark: the voltage VCC\_SW is measured 2 times with two different ADC. This allows to compare the results and conclude for a drift in the ADC’s. |  | NONE |
| 28 | 7..0 |  | NONE |
| 29 | 7..0 | T\_PCB\_TEMP1  Main Board PCB Temperature, sensor 1  Temperature of VCC DC/DC regulator.  Remark: 1/2 is measured on the same place, it’s to compare the values to discover a measurement failure | DOUBLE | VOR(MIN=@MIB; MAX=@MIB;D=@MIB) |
| 30 | 7..0 |  |  |
| 31 | 7..0 | T\_PCB\_TEMP2  Main Board PCB Temperature, sensor 2  Temperature of VCC DC/DC regulator.  Remark: 1/2 is measured on the same place, it’s to compare the values to discover a measurement failure |  |  |
| 32 | 7..0 | DOUBLE | VOR(MIN=@MIB; MAX=@MIB;D=@MIB) |
| 33 | 7..0 | T\_PCB\_TEMP3a  Sun Sensor Board PCB Temperature, sensor 3a.  Temperature of VCC5 LDO regulator.  Remark: 3a/b is measured on the same place, it’s to compare the values to discover a measurement failure | DOUBLE | VOR(MIN=@MIB; MAX=@MIB;D=@MIB) |
| 34 | 7..0 | DOUBLE |  |
| 35 | 7..0 | T\_PCB\_TEMP3b  Sun Sensor Board PCB Temperature, sensor 3b.  Temperature of VCC5 LDO regulator.  Remark: 3a/b is measured on the same place, it’s to compare the values to discover a measurement failure | DOUBLE | VOR(MIN=@MIB; MAX=@MIB;D=@MIB) |
| 36 | 7..0 | DOUBLE |  |
| 37 | 7..0 | T\_PCB\_TEMP4  Sun Sensor Board PCB Temperature, sensor 4. Temperature of VCC6 LDO regulator. | DOUBLE | VOR(MIN=@MIB; MAX=@MIB;D=@MIB) |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Byte** | **Bit** | **Description** | **Type** | **Fault class** |
| 1 | 7..0 | Error type  . | HEX | IV(VALUE=0x51)  IV(VALUE=0x52)  IV(VALUE=0x53)  IV(VALUE=0x54)  IV(VALUE=0x55)  IV(VALUE=0x57)  IV(VALUE=0x58)  IV(VALUE=0x59)  IV(VALUE=0x5A)  IV(VALUE=0x5B)  IV(VALUE=0x5C) |

## 2.3 GYTM - Gyroscope TM

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Byte** | **Bit** | **Description** | **Type** | **Fault class** |
| 1 | 7..0 | Unit identifier  Identification of the unit that addresses the message  0 = Nominal  1 = Redundant | INT | BF(MIN=0,MAX=0) |
| 21..2 | 7..0 | Gyroscope Telemetry  All telemetry data from Gyroscope.  Message is the same sent from Gyroscope unit without adding/removing data | HEX | NONE: the type of data transmitted appear to bee too much complicate to be mutated in such a way of triggering a test failure. Could be trageted in the future. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Byte | Bit | Description |  |  |
| 1 | 7..0 | Error type  . | HEX | IV(VALUE=0x51)  IV(VALUE=0x52)  IV(VALUE=0x53)  IV(VALUE=0x54)  IV(VALUE=0x56) |

## 2.4 MMTX - Magnetometer TX

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Byte** | **Bit** | **Description** | **Type** | **Fault class** |
| 1 | 7..0 | Unit identifier  Identification of the unit that addresses the message  0 = Nominal  1 = Redundant | BIN | BF(MIN=0;MAX=0) |
| 2 | 7..0 | Self Test Result  0 = no error  1 = error detected during TX self test  2-255 = reserved | BIN | BF(MIN=0;MAX=0;STATE=0) |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Byte** | **Bit** | **Description** | **Type** | **Fault class** |
| 1 | 7..0 | Error type  . | HEX | IV(VALUE=0x51)  IV(VALUE=0x52)  IV(VALUE=0x53)  IV(VALUE=0x54)  IV(VALUE=0x56)  IV(VALUE=0x5D) |

## 2.5 Sun Sensor TM

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Byte** | **Bit** | **Description** | **Type** | **Fault class** |
| 1 | 7..0 | Photodiode Q1 current ADC #3 | DOUBLE | VAT(T=@MIB;D=@MIB) |
| 2 | 7..0 |  |
| 3 | 7..0 | Photodiode Q2 current ADC #3 | DOUBLE | VAT(T=@MIB;D=@MIB) |
| 4 | 7..0 |  |  |
| 5 | 7..0 | Photodiode Q3 current ADC #3 | DOUBLE | VAT(T=@MIB;D=@MIB) |
| 6 | 7..0 |  |  |
| 7 | 7..0 | Photodiode Q4 current ADC #3 | DOUBLE | VAT(T=@MIB;D=@MIB) |
| 8 | 7..0 |  |  |
| 9 | 7..0 | Photodiode Q1 current ADC #2 | DOUBLE | VAT(T=@MIB;D=@MIB) |
| 10 | 7..0 |  |  |
| 11 | 7..0 | Photodiode Q2 current ADC #2 | DOUBLE | VAT(T=@MIB;D=@MIB) |
| 12 | 7..0 |  |  |
| 13 | 7..0 | Photodiode Q3 current ADC #2 | DOUBLE | VAT(T=@MIB;D=@MIB) |
| 14 | 7..0 |  |  |
| 15 | 7..0 | Photodiode Q4 current ADC #2 | DOUBLE | VAT(T=@MIB;D=@MIB) |
| 16 | 7..0 |  |  |
| 17 | 7..0 | Photodiode Q1 current ADC #6 | DOUBLE | VAT(T=@MIB;D=@MIB) |
| 18 | 7..0 |  |  |
| 19 | 7..0 | Photodiode Q2 current ADC #6 | DOUBLE | VAT(T=@MIB;D=@MIB) |
| 20 | 7..0 |  |  |
| 21 | 7..0 | Photodiode Q3 current ADC #6 | DOUBLE | VAT(T=@MIB;D=@MIB) |
| 22 | 7..0 |  |  |
| 23 | 7..0 | Photodiode Q4 current ADC #6 | DOUBLE | VAT(T=@MIB;D=@MIB) |
| 24 | 7..0 |  |  |
| 25 | 7..0 | Photodiode Q1 current ADC #5 | DOUBLE | VAT(T=@MIB;D=@MIB) |
| 26 | 7..0 |  |  |
| 27 | 7..0 | Photodiode Q2 current ADC #5 | DOUBLE | VAT(T=@MIB;D=@MIB) |
| 28 | 7..0 |  |  |
| 29 | 7..0 | Photodiode Q3 current ADC #5 | DOUBLE | VAT(T=@MIB;D=@MIB) |
| 30 | 7..0 |  |  |
| 31 | 7..0 | Photodiode Q4 current ADC #5 | DOUBLE | VAT(T=@MIB;D=@MIB) |
| 32 | 7..0 |  |  |
| 33 | 7..0 | Photodiode Q1 current ADC #4 | DOUBLE | VAT(T=@MIB;D=@MIB) |
| 34 | 7..0 |  |  |
| 35 | 7..0 | Photodiode Q2 current ADC #4 | DOUBLE | VAT(T=@MIB;D=@MIB) |
| 36 | 7..0 |  |  |
| 37 | 7..0 | Photodiode Q3 current ADC #4 | DOUBLE | VAT(T=@MIB;D=@MIB) |
| 38 | 7..0 |  |  |
| 39 | 7..0 | Photodiode Q4 current ADC #4 | DOUBLE | VAT(T=@MIB;D=@MIB) |
| 40 | 7..0 |  |  |
| 41 | 7..0 | Photodiode Q1 current ADC #7 | DOUBLE | VAT(T=@MIB;D=@MIB) |
| 42 | 7..0 |  |  |
| 43 | 7..0 | Photodiode Q2 current ADC #7 | DOUBLE | VAT(T=@MIB;D=@MIB) |
| 44 | 7..0 |  |  |
| 45 | 7..0 | Photodiode Q3 current ADC #7 | DOUBLE | VAT(T=@MIB;D=@MIB) |
| 46 | 7..0 |  |  |
| 47 | 7..0 | Photodiode Q4 current ADC #7 | DOUBLE | VAT(T=@MIB;D=@MIB) |
| 48 | 7..0 |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Byte** | **Bit** | **Description** | **Type** | **Fault class** |
| 1 | 7..0 | Error type  . | HEX | IV(VALUE=0x51)  IV(VALUE=0x54)  IV(VALUE=0x56) |

## 2.6 SSTP - Sun Sensor Temperature

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Byte** | **Bit** | **Description** | **Type** | **Fault class** |
| 1 | 7..0 | Temperature reading from ADC #3 | DOUBLE | VOR(MIN=@MIB;MAX=@MIB;D=@MIB) |
| 2 | 7..0 |
| 3 | 7..0 | Temperature reading from ADC #2 | DOUBLE | VOR(MIN=@MIB;MAX=@MIB;D=@MIB) |
| 4 | 7..0 |  |  |
| 5 | 7..0 | Temperature reading from ADC #6 | DOUBLE | VOR(MIN=@MIB;MAX=@MIB;D=@MIB) |
| 6 | 7..0 |  |  |
| 7 | 7..0 | Temperature reading from ADC #5 | DOUBLE | VOR(MIN=@MIB;MAX=@MIB;D=@MIB) |
| 8 | 7..0 |  |  |
| 9 | 7..0 | Temperature reading from ADC #4 | DOUBLE | VOR(MIN=@MIB;MAX=@MIB;D=@MIB) |
| 10 | 7..0 |  |  |
| 11 | 7..0 | Temperature reading from ADC #7 | DOUBLE | VOR(MIN=@MIB;MAX=@MIB;D=@MIB) |
| 12 | 7..0 |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Byte** | **Bit** | **Description** | **Type** | **Fault class** |
| 1 | 7..0 | Error type  . | HEX | IV(VALUE=0x51)  IV(VALUE=0x54)  IV(VALUE=0x56) |

## 2.7 Reaction Wheel TX

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Byte** | **Bit** | **Description** | **Type** | **Fault class** |
| 1 | 7..0 | Unit identifier  Identification of the unit that addresses the message  0 = Nominal  1 = Redundant | BIN | BF(MIN=0,MAX=0) |
| 2 | 7..0 | Self Test Result  0 = no error  1 = error detected during TX self test  2-255 = reserved | BIN | BF(MIN=0,MAX=0) |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Byte** | **Bit** | **Description** | **Type** | **Fault class** |
| 1 | 7..0 | Error type  . | Hex | IV(VALUE=0x51)  IV(VALUE=0x52)  IV(VALUE=0x53)  IV(VALUE=0x54)  IV(VALUE=0x56)  IV(VALUE=0x5D) |

## 2.8 SpaceCraft HK

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Byte** | **Bit** | **Description** | **Type** | **Fault class** |
| 1 | 7..0 | TMTC\_SW1  Identifies the switching position of the TMTC switch 1: the voltage is ~1.1V for position A and 2.2V for position B. 0V or 3.3V will indicate a short or an interruption. | DOUBLE | VAT(T=3.3;D=0)  VBT(T=0;D=0)  ID: AIFN086X |
| 2 | 7..0 |  |  |
| 3 | 7..0 | TMTC\_SW2  Identifies the switching position of the TMTC switch 2: the voltage is ~1.1V for position A and 2.2V for position B. 0V or 3.3V will indicate a short or an interruption. | DOUBLE | VOR(MIN=@MIB;MAX=@MIB;D=@MIB) |
| 4 | 7..0 |  | ID: AIFN087X |
| 5 | 7..0 | SC\_TEMP1  Temperature SC-TEMP1 of a sensor in the S/C structure | DOUBLE | VOR(MIN=@MIB;MAX=@MIB;D=@MIB) |
| 6 | 7..0 |  |  |
| 7 | 7..0 | SC\_TEMP2  Temperature SC-TEMP2 of a sensor in the S/C structure | DOUBLE | VOR(MIN=@MIB;MAX=@MIB;D=@MIB) |
| 8 | 7..0 |  |  |
| 9 | 7..0 | SC\_TEMP3  Temperature SC-TEMP3 of a sensor in the S/C structure | DOUBLE | VOR(MIN=@MIB;MAX=@MIB;D=@MIB) |
| 10 | 7..0 |  |  |
| 11 | 7..0 | SC\_TEMP4  Temperature SC-TEMP4 of a sensor in the S/C structure | DOUBLE | VOR(MIN=@MIB;MAX=@MIB;D=@MIB) |
| 12 | 7..0 |  |  |
| 13 | 7..0 | SC\_TEMP5  Temperature SC-TEMP5 of a sensor in the S/C structure | DOUBLE | VOR(MIN=@MIB;MAX=@MIB;D=@MIB) |
| 14 | 7..0 |  |  |
| 15 | 7..0 | SC\_TEMP6  Temperature SC-TEMP6 of a sensor in the S/C structure | DOUBLE | VOR(MIN=@MIB;MAX=@MIB;D=@MIB) |
| 16 | 7..0 |  |  |
| 17 | 7..0 | SC\_TEMP7  Temperature SC-TEMP7 of a sensor in the S/C structure | DOUBLE | VOR(MIN=@MIB;MAX=@MIB;D=@MIB) |
| 18 | 7..0 |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Byte | Bit | Description |  |  |
| 1 | 7..0 | Error type  . | HEX | IV(VALUE=0x56) |

## 2.9 Magnetorquer Set PWM RSP

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Byte** | **Bit** | **Description** | **Type** | **Fault class** |
| 1 | 7..0 | Unit identifier Magnetometer  Identification of the Magnetometer unit that addresses the message  0 = Nominal  1 = Redundant | BIN | BF(MIN=0;MAX=0) |
| 2 | 7..0 | Magnetometer Data request reply Byte1  Sync(LSB) (Note 1) |  | NONE: Not to address with the approach because the effect of a mutation is not predictable (the trasferred data is complex, e.g., signal). |
| 3 | 7..0 | Magnetometer Data request reply Byte2  Sync(MSB) (Note 1) |  | NONE: same as above. |
| 4 | 7..0 | Magnetometer Data request reply Byte3  RAdr (Note 1) |  | NONE: same as above. |
| 5 | 7..0 | Magnetometer Data request reply Byte4  Sadr (Note 1) |  | NONE: same as above. |
| 6 | 7..0 | Magnetometer Data request reply Byte5  ReplyMsg (Note 1) |  | NONE: same as above. |
| 7 | 7..0 | Magnetometer Data request reply Byte6  Bx Low (Note 1) |  | NONE: same as above. |
| 8 | 7..0 | Magnetometer Data request reply Byte7  Bx Middle |  | NONE: same as above. |
| 9 | 7..0 | Magnetometer Data request reply Byte8  CS error + Average + pos Clip X + neg Clip X + BX High (Note 1) |  | NONE: same as above. |
| 10 | 7..0 | Magnetometer Data request reply Byte9  By Low (Note 1) |  | NONE: same as above. |
| 11 | 7..0 | Magnetometer Data request reply Byte10  By Middle (Note 1) |  | NONE: same as above. |
| 12 | 7..0 | Magnetometer Data request reply Byte11  spare + pos Clip Y + neg Clip Y + BY High (Note 1) |  | NONE: same as above. |
| 13 | 7..0 | Magnetometer Data request reply Byte12  Bz Low (Note 1) |  | NONE: same as above. |
| 14 | 7..0 | Magnetometer Data request reply Byte13  Bz Middle (Note 1) |  | NONE: same as above. |
| 15 | 7..0 | Magnetometer Data request reply Byte14  spare + pos Clip Z + neg Clip Z + BZ High (Note 1) |  | NONE: same as above. |
| 16 | 7..0 | Magnetometer Data request reply Byte15  CS (Note 1) |  | NONE: same as above. |
| 17 | 7..0 | Magnetorquer nX Current - on  Current MTXA\_N when powered | DOUBLE | VOR(MIN=@MIB,MAX= @MIB,D=@MIB) |
| 18 | 7..0 |  |  |
| 19 | 7..0 | Magnetorquer nX Current - off  Current MTXA\_N when unpowered | DOUBLE | VOR(MIN=0.14,MAX= 0.21,D=0.01) |
| 20 | 7..0 |  | ID: AIFR074I |
| 21 | 7..0 | Magnetorquer pX Current - on  Current MTXA\_P when powered | DOUBLE | VOR(MIN=@MIB,MAX= @MIB,D=@MIB) |
| 22 | 7..0 |  |  |
| 23 | 7..0 | Magnetorquer pX Current - off  Current MTXA\_P when unpowered | DOUBLE | VOR(MIN=@MIB,MAX= @MIB,D=@MIB) |
| 24 | 7..0 |  | ID: AIFR075I |
| 25 | 7..0 | Magnetorquer nY Current - on  Current MTYA\_N when powered | DOUBLE | VOR(MIN=@MIB,MAX= @MIB,D=@MIB) |
| 26 | 7..0 |  | ID: AIFR076I |
| 27 | 7..0 | Magnetorquer nY Current - off  Current MTYA\_N when unpowered | DOUBLE | VOR(MIN=@MIB,MAX= @MIB,D=@MIB) |
| 28 | 7..0 |  | ID:AIFR077I |
| 29 | 7..0 | Magnetorquer pY Current - on  Current MTYA\_P when powered |  | VOR(MIN=@MIB,MAX= @MIB,D=@MIB) |
| 30 | 7..0 | DOUBLE | ID: AIFR078I |
| 31 | 7..0 | Magnetorquer pY Current - off  Current MTYA\_P when unpowered | DOUBLE | VOR(MIN=@MIB,MAX= @MIB,D=@MIB) |
| 32 | 7..0 |  | ID: AIFR079I |
| 33 | 7..0 | Magnetorquer nZ Current - on  Current MTZA\_N when powered | DOUBLE | VOR(MIN=@MIB,MAX= @MIB,D=@MIB) |
| 34 | 7..0 |  | ID:AIFR080I |
| 35 | 7..0 | Magnetorquer nZ Current - off  Current MTZA\_N when unpowered | DOUBLE | VOR(MIN=@MIB,MAX= @MIB,D=@MIB) |
| 36 | 7..0 |  |  |
| 37 | 7..0 | Magnetorquer pZ Current - on  Current MTZA\_P when powered | DOUBLE | VOR(MIN=@MIB,MAX= @MIB,D=@MIB) |
| 38 | 7..0 |  |  |
| 39 | 7..0 | Magnetorquer pZ Current - off  Current MTZA\_P when unpowered | DOUBLE | VOR(MIN=@MIB,MAX= @MIB,D=@MIB) |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Byte** | **Bit** | **Description** | **Type** | **Fault class** |
| 1 | 7..0 | Error type  . | Hex | IV(VALUE=0x51)  IV(VALUE=0x52)  IV(VALUE=0x53)  IV(VALUE=0x54)  IV(VALUE=0x56)  IV(VALUE=0x5D)  IV(VALUE=0x5E) |

# 3. Mutation Probes

Mutation probes are manually integrated into the source code of function *ObcRecvBlockCb*. Figure 3 shows an example of how we integrate mutation probes. All the probes are integrated following the same pattern; more precisely, for each data generation function we manually insert two invocations to the FAQAS mutation probe API, one to perform mutation of the nominal response message (Line 154, in Figure 3) the other one to mutate an error response message (Line 149). The choice of the data model to pass to the FAQAS mutation probe API is based on the value of *cr*, the variable that captures the return status of the specific data generation function invoked (function *GetIfHk* in Figure 3).

The function \_FAQAS\_mutate takes as input the fault model to be used to drive the mutation. Fault models are automatically generated from template files matching to the tables reported in Section 2.

|  |
| --- |
| 144:            case 0:  145:            {  146:               cr = GetIfStatus(newBlock);  147:    if ( cr == CR\_Failure ){  148: FaultModel \*dm = \_FAQAS\_GetIfStatus\_FM\_Error ()  149: \_FAQAS\_mutate( newBlock, dm );  150: \_FAQAS\_delete\_DM( dm )  151: } else {  152: FaultModel \*dm = \_FAQAS\_GetIfStatus\_FM ()  153: \_FAQAS\_mutate( newBlock, dm);  154: \_FAQAS\_delete\_DM( dm ) 155: }  156:            } |

Figure 3: Mutation probe for GetIfStatus