**Appendix**

**Data-driven Mutation Testing:**

**LuxSpace**

PHDU-PDO FAULT MODEL

This document describes the procedures to execute data-driven mutation testing on the LuxSpace Remote Commands over PDO (Process Data Object)) case study system.

In the Section 1 we will provide a brief overview of the case study. Then, in Section 2, we will describe the commands implemented in the command/response protocol and the data-driven mutation operators we plan on applying to the buffer that contains them.

# Overview of the case study

The PDOs (TPDO1 and RPDO1) handle communication between the OnBoard Computer (OBC) and the PDHU (Payload Data Handling Unit).

The OBC initiates a remote access by sending a command (Cmd) on TPDO1 and the PDHU processes the command and send back its answer (Ans) on RPDO1.

![Diagram

Description automatically generated]()

The layouts of the commands and answers are described in the table below. Both are composed of a signal ID that must match between command and answer and a payload with size depending on the ID.

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Position | Size | Description |
| CmdID | Byte 0 | UNSIGNED8 | ID that defines the command |
| CmDData | Byte 1.7 | 7 Bytes | The command payload. The size and the meaning of the payload depend on the command ID |
| AnsID | Byte 0 | UNSIGNED8 | ID that matches with the CmdID |
| AnsData | Byte 1.7 | Up to 7 Bytes | The Answer payload. The size and the meaning of the payload depend on the command ID |

The first byte of the answer payload is always a status that reflect the command handling status and the 6 remaining answer payload bytes are meaningful only when the status is different from 00h (Command OK)

|  |  |
| --- | --- |
| Status Code | Description |
| 00h | Command OK |
| 01h | Unknown command ID |
| 02h | Bad command payload size: the received command payload size does not match with the expected one according to the command ID |
| 03h | Bad parameter value |
| 04h | Not allowed: the command is not allowed in the context |
| 05h | Command aborted |
| 06h | Command pending |

# Commands description

The CmID or AnsID occupies the first byte of the signal array. It is expressed as a Hexadecimal number of size UNSIGNED8. The highlighted ones are implemented in the SVF.

|  |  |  |
| --- | --- | --- |
| CmdID | Command Name | Command Description |
| 0x01 | STO\_SND\_FRM | Read from storage and send a range of frames to the PDD |
| 0x02 | STO\_ACK\_FRM | Acknowledge a range of frames |
| 0x03 | STO\_RST\_FRM | Reset all frames |
| 0x04 | STO\_GET\_HEAD | Retrieve the 5-bytes storage Head pointer |
| 0x05 | STO\_GET\_TAIL | Retrieve the 5-bytes storage tail pointer |
| 0x10 | SM\_TST\_CHIP | Storage Maintenance: Test Chip |
| 0x11 | SM\_TST\_BLOCK | Storage Maintenance: Test Block |
| 0x12 | SM\_RESET | Storage Maintenance: Storage Reset |
| 0x13 | SM\_SET\_CB0\_SIZE | Storage Maintenance: Set the frontier between both storage |
| 0x1A | PDHU\_RESET | Request a PDHU reset |
| 0x1B | DBG\_SELFTST | Debug/Monitoring: Perform a self-test |
| 0x1C | CRC\_COMPUTE | Compute the CCIT CRC |

The payload varies according to the command.

Not all of these commands are being used by the OBSW. The implemented ones are STO\_SND\_FRM, STO\_GET\_HEAD, STO\_GET\_TAIL and CRC\_COMPUTE.

Below we report the operators (Fault Class) to be used for each data item; however, red color is used to indicate data items that shall not be mutated.

## **RMTCMD(O1,01) STO\_SND\_FRM**

Read from storage and send a range of frames to the PDD. The command is only available in ACTIVE mode.

### Ans Payload

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Byte | Name | Type | Description | Fault Class |
| BYTE [0] ID | AnsID=CmdID | HEX (UNSIGNED8) | ID that defines the command: 0x01 | IV(Value=0x02)  IV(Value=0x03)  IV(Value=0x04) IV(Value=0x05) IV(Value=0x10) IV(Value=0x11) IV(Value=0x12) IV(Value=0x13)  IV(Value=0x1A)  IV(Value=0x1B)  IV(Value=0x01C) |
| BYTE [1] | Command Status | UINT8 | Command status code | IV(Value=05h) to make it seem like the command was aborted  IV(value=01h)  IV(value=02h)  IV(value=03h)  IV(value=04h)  IV(value=06h)  IV(value=00h) |
| BYTE [2...5] | PendingRequest | UINT32 | number of pending requests in the Storage Request FIFO | HV(Value=?)  To show the same number of pending requests even when it changes  SS(Delta=?)  To add or subtract a fixed number to the number of pending requests |
| BYTE [6...7] | UNUSED | - | - |  |

### **RMTCMD(O1,04) STO\_GET\_HEAD**

Retrieve the 5-bytes storage Head pointers.

### Ans Payload

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Byte | Name | Type | Description | Fault Class |
| BYTE [0] ID | AnsID=CmdID | HEX (UNSIGNED8) | ID that defines the command: 0x04 | IV(Value=0x01)  IV(Value=0x02)  IV(Value=0x03)  IV(Value=0x05) IV(Value=0x010) IV(Value=0x011) IV(Value=0x012) IV(Value=0x013)  IV(Value=0x01A)  IV(Value=0x01B)  IV(Value=0x01C) |
| BYTE [1] Payload | Command Status | UINT8 | Command status code | IV(Value=05h) to make it seem like the command was aborted.  IV(value=01h)  IV(value=02h)  IV(value=03h)  IV(value=04h)  IV(value=06h)  IV(value=00h)  VAT(T=06h, D=1)  FVAT(T=06h, D=1) |
| BYTE [2...6] Payload | SequenceID | UINT40 | Sequence ID corresponding to the head of the storage  Byte #1 is the LSByte  Byte #5 is the MSByte | BF(LSBit) for each byte? |
| BYTE [7] Payload | UNUSED | - | - |  |

### **RMTCMD(O1,05) STO\_GET\_TAIL**

Retrieve the 5-bytes storage Tail pointer.

### Ans Payload

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Byte | Name | Type | Description | Fault Class |
| BYTE [0] ID | AnsID=CmdID | HEX (UNSIGNED8) | ID that defines the command: 0x05 | IV(Value=0x01)  IV(Value=0x02)  IV(Value=0x03)  IV(Value=0x04) IV(Value=0x010) IV(Value=0x011) IV(Value=0x012) IV(Value=0x013)  IV(Value=0x01A)  IV(Value=0x01B)  IV(Value=0x01C) |
| BYTE [1] Payload | Command Status | UINT8 | Command status code | IV(Value=05h) to make it seem like the command was aborted.  IV(value=01h)  IV(value=02h)  IV(value=03h)  IV(value=04h)  IV(value=06h)  IV(value=00h)  VAT(T=06h, D=1)  FVAT(T=06h, D=1) |
| BYTE [2...6] Payload | SequenceID | UINT40 | Sequence ID corresponding to the tail of the storage  Byte #1 is the LSByte  Byte #5 is the MSByte |  |
| BYTE [7] Payload | UNUSED | - | - |  |

***RMTCMD(O1,1C) CRC\_COMPUTE***

Compute the CRC CCITT of MRAM data.

The command is only available when the PDHU is in PASSIVE mode.

### Ans Payload

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Byte | Name | Type | Description | Fault Class |
| BYTE [0] ID | AnsID=CmdID | HEX (UNSIGNED8) | ID that defines the command: 0x01C | IV(Value=0x01)  IV(Value=0x02)  IV(Value=0x03)  IV(Value=0x04)  IV(Value=0x05)  IV(Value=0x010)  IV(Value=0x011)  IV(Value=0x012)  IV(Value=0x013)  IV(Value=0x01A)  IV(Value=0x01B) |
| BYTE [1] Payload | Command Status | UINT8 | Command status code | IV(Value=05h) to make it seem like the command was aborted.  IV(Value=00h) to activate the syndrome part even if the command status was not 00h  IV(Value=04h) to simulate a different PHDU mode  IV(Value=05h) to make it seem like the command was aborted.  IV(value=01h)  IV(value=02h)  IV(value=03h)  IV(value=06h) |
| BYTE [2] Payload | Syndrom |  | MS Byte processed syndrom, only valid when returned status code is 00h | BF |
| BYTE [3] Payload | Syndrom |  | MS Byte processed syndrom, only valid when returned status code is 00h | BF |
| BYTE [4.7] Payload | UNUSED | - | - |  |

|  |  |
| --- | --- |
| Status Code | Description |
| 00h | Command OK |
| 03h | if the CRC parameters coherency is bad |
| 04h | if the PDHU mode is ACTIVE |

# Probe Insertion

The probes were inserted in the method *PdhuPdoService::IndicationReceived,* which handles the RMTCMD.

The method is defined in the file *PdhuPdoService.cpp,* contained in the folder

Svf/Models/CAN/src/Pdhu/.

Each message type is handled by a different switch case and was targeted by a different fault modelas shown below.

1. **case** 0x04: //RMTCMD(O1,04) STO\_GET\_HEAD
2. {
3. ::Smp::UInt64 sequenceId;
4. auto statusCode = Pdhu->StorageReadBack->GetHeadPointer(data[1], sequenceId);
6. **if**(statusCode == RCSC\_CmdOk)
7. {
8. Generic::Utils::SerializeLe(sequenceId, newData.begin() + 2, 5);
10. // MANUALLY INSERTED PROBE
11. mutate\_FM\_STO\_GET\_HEAD( &newData );
12. // END OF THE PROBE
14. }
15. **else**
16. {
17. newData[1] = statusCode;
18. }
19. }