**RESTRICTED ACCESS**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **DISTRIBUTION :** | **Firm** | **To** | **Copies** | e-mail |
|  | AEE-C |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

**SW MODULE/GLOBAL INTEGRATION TEST PLAN**

OBJECT: This document gathers the tests for the integration of Belt function Execution algorithm

SUMMARY: Those integration tests are done on *BFE* SW-C for PP Platform. These tests have been designed for High Power platform project with Autosar environment.

CONCLUSION:

**EVOLUTION OF THE DOCUMENT**

|  |  |  |  |
| --- | --- | --- | --- |
| Issue | Date | Author | Motive and nature of the modifications |
| 1.1  1.2 | 04.10.12  18.10.12 | CSA  FLD | First release.  Add High-Power Tests |
| 1.1 | 10.03.15 | CRE | Rename “BFE – SW Integration Test Plan.docx” with “BFE – Unit Tests Document.docx” |
| 1.2 | 03.10.15 | SFL | Tests on Temperature adaptivity |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**CONTENTS**

1. scope of INTEGRation 4

2. List of applicable documents 4

3. list of reference documents 4

4. Terminology 4

5. Integration Test Strategy 5

6. LIST OF TESTS 6

6.1 Tests for BFE\_AC\_BeltFunctionExecution\_BFE\_runScheduleStep 8

6.1.1 INT\_BFE\_01001: Periodicity and scheduling 8

6.1.2 INT\_BFE\_01002 : Step Identifier 9

6.1.3 INT\_BFE\_01003: Power degradation factor computation 10

6.1.4 INT\_BFE\_01004: Cycle execution counter update 11

6.2 Tests for BFE\_ProvideStepConfig 12

6.2.1 INT\_BFE\_02001: Steps parameters extraction 12

6.3 Tests for BFE\_runExecuteHighPowerStep 13

6.3.1 INT\_BFE\_03001: Runnable scheduling 13

6.3.2 INT\_BFE\_03002: test of the current regulation algorithm 13

6.3.3 INT\_BFE\_03003: test of the boost interrupt mechanism in case of a too low motor current 17

6.3.4 INT\_BFE\_03004: test of the boost interrupt mechanism at the end of a High-Power step 18

6.3.5 INT\_BFE\_03005: test of a cycle with a High-Power step 19

6.4 Tests for ManageStepEvents function 22

6.4.1 INT\_BFE\_04001: Step with ‘Trigger Off’ option 22

6.4.2 INT\_BFE\_04002: Step with ‘Current Interruption point’ option 23

6.4.3 INT\_BFE\_04003: Step with ‘Motor Blocked’ option 25

6.4.4 INT\_BFE\_04004: Step with ‘Belt Blocked’ option 26

6.4.5 INT\_BFE\_04005: Step with ‘Belt Movement Detection’ option 27

6.5 Tests for the InitConsigns function 28

6.5.1 INT\_BFE\_05001: Step duration management 28

6.5.2 INT\_BFE\_05002: Internal data exchanged between the 10ms and the 2ms main functions 30

6.5.3 INT\_BFE\_05003: Direction reverse protection management 31

6.6 Temperature adaptivity 32

6.6.1 INT\_BFE\_06001 : Default tensioning temperature correction factors 32

6.6.2 INT\_BFE\_06002 : Nominal tensioning temperature correction with current/motor/PWM steps 33

6.6.3 INT\_BFE\_06003 : No temperature adaptation with absent or invalid step configuration 37

6.6.4 INT\_BFE\_06004 : No PRESAFE temperature adaptation with invalid temperature 40

6.6.5 INT\_BFE\_06005 : PRESAFE temperature adaptation with valid temperature outside the table 42

6.7 Power degradation – NA since PN14 not yet available 45

6.7.1 INT\_BFE\_07001 : Start-up profile is not power degradabled 45

6.7.2 INT\_BFE\_07002 : Power degradation on current controlled steps 45

6.7.3 INT\_BFE\_07003 : Power degradation on voltage controlled steps 47

6.7.4 INT\_BFE\_07004 : No Power degradation on not power degradabled steps 49

6.7.5 INT\_BFE\_07005 : Cycle with power degradabled and not power degradabled steps 51

6.7.6 INT\_BFE\_07006 : No Power degradation abortion 53

6.7.7 INT\_BFE\_07007 : Inhibition by power degradation 54

7. Annex 56

7.1 Adaptation of Belt Function Selection module 56

7.2 Cycles & steps parameters for integration tests 56

7.3 Power degradation look up table 57

7.4 Current interruption point look up tables 57

# scope of INTEGRation

The purpose of the tests is to check that the BFE component (in charge of the Power stage PWM duty cycle computation) is well integrated with all other applicative layers.

# List of applicable documents

|  |  |  |  |
| --- | --- | --- | --- |
| **Nb** | **DOCUMENT** | **REFERENCE** | **Company** |
|  | SW work Product Follow-up (SPF) | Exxxxxx | **AEFC** |
|  | BFE – SW Design Document | MKS reference | **AEFC** |
|  |  |  |  |

Note: The documents releases are specified in the SPF.

# list of reference documents

|  |  |  |  |
| --- | --- | --- | --- |
| **Nb** | **DOCUMENT** | **REFERENCE** | **Company** |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Note: The documents releases are specified in the SPF.

# Terminology

ASM: Analog Signals Management

BFE: Belt Function Execution

BFS: Belt Function Selection

NVM: Non Volatile Memory

PAL: Power Abstraction Layer (Actuator / Sensor)

RTE: Real Time Environment

TBC: To Be Confirmed

TBD: To Be Defined

VSA: Vehicle Signal Abstraction

# Integration Test Strategy

The Integration Test strategy adopted is detailed in the SW Global Integration Test Follow-up. It specifies the test environment (software components, hardware means, compilation options used, EEPROM file, performance and timing …)

* Tests definition

The integration tests are divided in 4 types: nominal, robustness, endurance and qualification tests.

Nominal Test: A nominal test consists to apply entry values sequences to the software that should cause all the behaviours expected in the design documents.

Robustness Test: A robustness test consists to inject faults and limits values to the software in order to check that the behaviour is coherent to the designs documents.

Endurance Test: An endurance test is done in order to check the repetitiveness and the repeatability of the results.

Qualification Test: A qualification test aims at precisely checking (and reporting) a behaviour or a metric (for instance to measure a timing or a memory consumption).

# LIST OF TESTS

The classification to use for the test type is:

- Nominal tests (N),

- Robustness tests (R),

- Endurance tests (E).

- Qualification tests (Q)

|  |  |  |
| --- | --- | --- |
| **Test**  **Id** | **Type**  **Classification** | **Purpose** |
| **BFE\_AC\_BeltFunctionExecution\_BFE\_runScheduleStep** | | |
| INT\_BFE\_01001 | N | Periodicity and scheduling |
| INT\_BFE\_01002 | N | Step Identifier |
| INT\_BFE\_01003 | N | Power degradation factor computation |
| INT\_BFE\_01004 | N | Cycle execution counter update |
| **BFE\_ProvideStepConfig** | | |
| INT\_BFE\_02001 | N | Steps parameters extraction |
| **BFE\_runExecuteHighPowerStep** | | |
| INT\_BFE\_03001 | Q | Runnable scheduling |
| INT\_BFE\_03002 | N | Test of the current regulation algorithm |
| INT\_BFE\_03003 | N | Test of the boost interrupt mechanism in case of a too low motor current |
| INT\_BFE\_03004 | N | Test of the boost interrupt mechanism at the end of a High-Power step |
| INT\_BFE\_03005 | N | Test of a cycle with a High-Power step |
| **ManageStepEvents** | | |
| INT\_BFE\_04001 | N | Step with ‘Trigger Off’ option |
| INT\_BFE\_04002 | N | Step with ‘Current Interruption point’ option |
| INT\_BFE\_04003 | N | Step with ‘Motor Blocked’ option |
| INT\_BFE\_04004 | N | Step with ‘Belt Blocked’ option |
| INT\_BFE\_04005 | N | Step with ‘Belt Movement Detection’ option |
| **InitConsigns** | | |
| INT\_BFE\_05001 | N | Step duration management |
| INT\_BFE\_05002 | N | Internal data exchanged between the 10ms and the 2ms main functions |
| INT\_BFE\_05003 | N |  |
|  |  |  |
|  |  |  |

## Tests for BFE\_AC\_BeltFunctionExecution\_BFE\_runScheduleStep

### INT\_BFE\_01001: Periodicity and scheduling

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  | |  |  |  | |  |  | |  |  |
|  | **Type of the test: N** | | | | | | | | | | |  |
|  |  |  | |  |  |  | |  |  | |  |  |
|  | **Purpose of the test:**  The goal of this test is to check that the function in charge of step scheduling is called with the right period and that the first call is performed after the initialization phase. | | | | | | | | | | |  |
|  |  |
|  |  |
|  |  |  |  | |  | |  |  | |  |  |  |
|  | **Environment:**  CAN environment to stay alive.  ECU or mock-up flashed with an instrumented code.  Debugger plugged with 2 breakpoints :   * At the closing brace of RTE\_Init function * At the beginning of BFE step scheduling main function   A toggle pin at the entry of BFE main function shall be added and a scope probe shall be used at the debug pin to check the periodicity | | | | | | | | | | |  |
|  |  |
|  |  |
|  |  |  |  | |  | |  |  | |  |  |  |
|  | **INITIAL STATE:**  ECU flashed and not running | | | | | | | | | | |  |
|  |  |
|  |  |
|  |  |
|  |  |  |  | |  | |  |  | |  |  |  |
|  | **ACTION:**   1. Perform an ‘In Target reset’ of the application. 2. Run the application and wait the stop at the first breakpoint point 3. Press ‘Go’ button to continue until the next breakpoint 4. Remove the 2nd breakpoint & press ‘Go’ button | | | | | | | | | | |  |
|  |  |
|  |  |
|  |  |
|  |  |  |  | |  | |  |  | |  |  |  |
|  | **EXPECTED RESULT:**   1. Nothing expected 2. Check that the program halts at the end of the RTE\_Init function at first 3. Check that the program halts at the entry of the BFS main function. 4. Check on the scope that the signal on the debug pin toggles every 10 ms   [COVERS: DSG\_BFS\_00004] | | | | | | | | | | |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |  |  | |  | |  |  | |  |  |  |
|  |  |  |  | |  | |  |  | |  |  |  |

### INT\_BFE\_01002 : Step Identifier

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  | |  |  |  | |  |  | |  |  |
|  | **Type of the test: N** | | | | | | | | | | |  |
|  |  |  | |  |  |  | |  |  | |  |  |
|  | **Purpose of the test:**  The goal of this test is to check that the function in charge of step scheduling updates correctly the current executed step Identifier that will be used to extract steps parameters. | | | | | | | | | | |  |
|  |  |
|  |  |
|  |  |  |  | |  | |  |  | |  |  |  |
|  | **Environment:**  CAN environment to stay alive.  ECU or mock-up flashed with an instrumented code.  On the Watch Window of the debugger :   * Access to the selected cycle Id * Access to u8IdxExecutedStep & u8StepToExecute | | | | | | | | | | |  |
|  |  |
|  |  |
|  |  |  |  | |  | |  |  | |  |  |  |
|  | **INITIAL STATE:**  ECU flashed and not running  Cycle parameters for Cycle 1 : 4 steps configured  Cycle parameters for Cycle 2 : 8 steps configured (different from the 4 steps already defined for Cycle 1)  Software instrumented as described on the Annex chapter. | | | | | | | | | | |  |
|  |  |
|  |  |
|  |  |
|  |  |  |  | |  | |  |  | |  |  |  |
|  | **ACTION:**   1. Perform an ‘In Target reset’ of the application. 2. Run the application 3. Launch the Cycle 1 (u8SeletedCycle\_test = 0x00 and u8OverWriteBFSOutput = 0xAA) 4. Remove the flag to overwrite the BFS output before the end of Cycle 1 execution (u8OverWriteBFSOutput = 0x55) 5. Launch the Cycle 2 (u8SeletedCycle\_test = 0x01 and u8OverWriteBFSOutput = 0xAA) 6. Remove the flag to overwrite the BFS output before the end of Cycle 2 execution (u8OverWriteBFSOutput = 0x55) | | | | | | | | | | |  |
|  |  |
|  |  |
|  |  |
|  |  |  |  | |  | |  |  | |  |  |  |
|  | **EXPECTED RESULT:**   1. Nothing expected 2. Check that no cycle is executed. 3. Check that the first cycle starts. 4. Check that the executed cycle continues until his end and during the belt function, u8IdxExecutedStep takes the values {0,1,2,3,255} and u8StepToExecute takes the values defined in the cycle parameters array. These data shall change only when BFE\_bIntStepFlag = 1 5. Check that the second cycle starts. 6. Check that the executed cycle continues until his end and during the belt function, u8IdxExecutedStep takes the values {0,1,2,3,4,5,6,7,255} and u8StepToExecute takes the values defined in the cycle parameters array. These data shall change only when BFE\_bIntStepFlag = 1 | | | | | | | | | | |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |  |  | |  | |  |  | |  |  |  |

### INT\_BFE\_01003: Power degradation factor computation

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  | |  |  |  | |  |  | |  |  |
|  | **Type of the test: N** | | | | | | | | | | |  |
|  |  |  | |  |  |  | |  |  | |  |  |
|  | **Purpose of the test:**  The goal of this test is to check that the function in charge of step scheduling compute correctly the power degradation factor when a new cycle begins. | | | | | | | | | | |  |
|  |  |
|  |  |
|  |  |  |  | |  | |  |  | |  |  |  |
|  | **Environment:**  CAN environment to stay alive.  ECU or mock-up flashed with an instrumented code.  Debugger plugged  On the Watch Window of the debugger :   * Read Access to u8DegradationPowerFactor   This test is only applicable if the Battery voltage is available on the CAN bus. | | | | | | | | | | |  |
|  |  |
|  |  |
|  |  |  |  | |  | |  |  | |  |  |  |
|  | **INITIAL STATE:**  ECU flashed and not running  Belt functions 01/02 configured with at least one power degradable step.  Belt functions 03 configured without any power degradable steps.  To trig belt functions, the instrumentation described in 7.1 can be used at the BFS algo level. | | | | | | | | | | |  |
|  |  |
|  |  |
|  |  |
|  |  |  |  | |  | |  |  | |  |  |  |
|  | **ACTION:**   1. Perform an ‘In Target reset’ of the application. 2. Run the application 3. Set the CAN battery voltage to a value upper than 13V then trig the cycle 01 4. During the execution of cycle 01 : changes the battery voltage on the CAN and set the signal to a value between ] 9.0 ; 10.8 [ V 5. Wait the end of cycle 01 then launch the cycle 02 6. Wait the end of cycle 02 then launch the cycle 03 | | | | | | | | | | |  |
|  |  |
|  |  |
|  |  |
|  |  |  |  | |  | |  |  | |  |  |  |
|  | **EXPECTED RESULT:**   1. Nothing expected 2. Check that no cycle is executed (excepted the Anti-patina belt function). 3. Checks that the belt function starts and that the degradation factor is equal to 128 4. Checks that the degradation factor does not change. 5. Checks that the degradation factor is updated with the expected value at the beginning of the cycle 02 (for the expected value see the 7.3) 6. Checks that the degradation factor is updated with the value 128 at the beginning of cycle 03. | | | | | | | | | | |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |  |  | |  | |  |  | |  |  |  |
|  |  |  |  | |  | |  |  | |  |  |  |

### INT\_BFE\_01004: Cycle execution counter update

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  | |  |  |  | |  |  | |  |  |
|  | **Type of the test: N** | | | | | | | | | | |  |
|  |  |  | |  |  |  | |  |  | |  |  |
|  | **Purpose of the test:**  The goal of this test is to check that the function in charge of step scheduling updates correctly the executed cycles counters values. And that this array is correctly recorded and restored at shutdown/startup. | | | | | | | | | | |  |
|  |  |
|  |  |
|  |  |  |  | |  | |  |  | |  |  |  |
|  | **Environment:**  CAN environment to stay alive.  ECU or mock-up flashed with an instrumented code.  Debugger plugged  On the Watch Window of the debugger :   * Read Access to NVP\_au32ExecutedCycleCounters | | | | | | | | | | |  |
|  |  |
|  |  |
|  |  |  |  | |  | |  |  | |  |  |  |
|  | **INITIAL STATE:**  ECU flashed and not running  To trig belt functions, the instrumentation described in 7.1 can be used at the BFS algorithm level. | | | | | | | | | | |  |
|  |  |
|  |  |
|  |  |
|  |  |  |  | |  | |  |  | |  |  |  |
|  | **ACTION:**   1. Perform an ‘In Target reset’ of the application. 2. Run the application 3. Start the cycle 01 4. Maintain the triggering request until the restart of the cycle 01 | | | | | | | | | | |  |
|  |  |
|  |  |
|  |  |
|  |  |  |  | |  | |  |  | |  |  |  |
|  | **EXPECTED RESULT:**   1. Nothing expected 2. Check that no cycle is executed (excepted the Anti-patina belt function). 3. Checks that counter for the cycle 01 is incremented by 1 4. Checks that counter for the cycle 01 is incremented by 1 again | | | | | | | | | | |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |  |  | |  | |  |  | |  |  |  |
|  |  |  |  | |  | |  |  | |  |  |  |

## Tests for BFE\_ProvideStepConfig

### INT\_BFE\_02001: Steps parameters extraction

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  | |  |  |  | |  |  | |  |  |
|  | **Type of the test: N** | | | | | | | | | | |  |
|  |  |  | |  |  |  | |  |  | |  |  |
|  | **Purpose of the test:**  The goal of this test is to check that the function in charge of extracting steps parameters read correctly the steps parameters. | | | | | | | | | | |  |
|  |  |
|  |  |
|  |  |  |  | |  | |  |  | |  |  |  |
|  | **Environment:**  CAN environment to stay alive.  ECU or mock-up flashed with an instrumented code.  Debugger plugged  On the Watch Window of the debugger read Access to :   * stStepOrderOptions * stStepConstraints * u16StepDuration * bIsRunningStep * bStartNewStep * u8StepToExecute | | | | | | | | | | |  |
|  |  |
|  |  |
|  |  |  |  | |  | |  |  | |  |  |  |
|  | **INITIAL STATE:**  ECU flashed and not running  To trig belt functions, the instrumentation described in 7.1 can be used at the BFS algorithm level. | | | | | | | | | | |  |
|  |  |
|  |  |
|  |  |
|  |  |  |  | |  | |  |  | |  |  |  |
|  | **ACTION:**   1. Perform an ‘In Target reset’ of the application. 2. Run the application 3. Start the cycle 01 4. Maintain the triggering request until the restart of the cycle 01 | | | | | | | | | | |  |
|  |  |
|  |  |
|  |  |
|  |  |  |  | |  | |  |  | |  |  |  |
|  | **EXPECTED RESULT:**   1. Nothing expected 2. Check that no cycle is executed (excepted the Anti-patina belt function). 3. Checks that counter for the cycle 01 is incremented by 1 4. Checks that counter for the cycle 01 is incremented by 1 again | | | | | | | | | | |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |  |  | |  | |  |  | |  |  |  |
|  |  |  |  | |  | |  |  | |  |  |  |

## Tests for BFE\_runExecuteHighPowerStep

### INT\_BFE\_03001: Runnable scheduling

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  | |  |  |  | |  |  | |  |  |
|  | **Type of the test: Q**N, R, E or Q | | | | | | | | | | |  |
|  |  |  | |  |  |  | |  |  | |  |  |
|  | **Purpose of the test: This test aims at checking the periodicity of the runnable BFE\_AC\_BeltFunctionExecution\_BFE\_runExecuteHighPowerStep**  Briefly sum-up the purpose of the test. | | | | | | | | | | |  |
|  |  |
|  |  |
|  |  |  |  | |  | |  |  | |  |  |  |
|  | **Environment:**  ECU mockup (with a high-power hardware) flashed  BFE configuration expected : BFE\_CFG\_OPT\_HIGH\_POWER defined  Instrument a S/W with:  a toggle pin at the beginning of the function *BFE\_AC\_BeltFunctionExecution\_BFE\_runExecuteHighPowerStep*  Debugger with trace32  CAN environment  Capture the following signals with the oscilloscope:   * The debug pin Describe the tools needed for the test (use of emulator / debugger, CAN simulation, oscilloscope), debug pins, instrumented code... | | | | | | | | | | |  |
|  |  |
|  |  |
|  |  |  |  | |  | |  |  | |  |  |  |
|  | **INITIAL STATE:**  ECU flashed and not running. | | | | | | | | | | |  |
|  |  |
|  |  |
|  |  |
|  |  |  |  | |  | |  |  | |  |  |  |
|  | **ACTION:**   1. Run the application | | | | | | | | | | |  |
|  |  |
|  |  |
|  |  |
|  |  |  |  | |  | |  |  | |  |  |  |
|  | **EXPECTED RESULT:**   1. Check that the runnable is called every 400 µs | | | | | | | | | | |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |  |  | |  | |  |  | |  |  |  |
|  |  |  |  | |  | |  |  | |  |  |  |

### INT\_BFE\_03002: test of the current regulation algorithm

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  | |  |  |  | |  |  | |  |  |
|  | **Type of the test: N**N, R, E or Q | | | | | | | | | | |  |
|  |  |  | |  |  |  | |  |  | |  |  |
|  | **Purpose of the test: This test aims at checking step by step the behavior of the algorithm during a High-Power step and the values of the parameters sent to the PAL Actuator (Motor Power order and boost duty cycle), in function of the motor current and the battery voltage**  Briefly sum-up the purpose of the test. | | | | | | | | | | |  |
|  |  |
|  |  |
|  |  |  |  | |  | |  |  | |  |  |  |
|  | **Environment:**  This test needs the following modifications in the sources :  I- Modify the cycles configuration, in the file *NVP\_Const.h* :  Configure the cycle 1 with just one high-power step (40 A, 200 ms)   * Configuration of the step parameters   **#define** KU16\_STEP\_01\_TIME (0x0014)  **#define** KU8\_STEP\_01\_ORDER\_VAL (0x50)  **#define** KU8\_STEP\_01\_ORDER\_TYP (0x01)  **#define** KU16\_STEP\_01\_OPTIONS (0x0001)   * Definition of the cycle 1   **#define** KU8\_CYCLE\_1\_NEXT\_CYCLE (0xFF)  **#define** KAU8\_CYCLE\_1\_STEP\_1\_ID (0)  **#define** KAU8\_CYCLE\_1\_STEP\_2\_ID (255)  **#define** KAU8\_CYCLE\_1\_STEP\_3\_ID (255)  **#define** KAU8\_CYCLE\_1\_STEP\_4\_ID (255)  **#define** KAU8\_CYCLE\_1\_STEP\_5\_ID (255)  **#define** KAU8\_CYCLE\_1\_STEP\_6\_ID (255)  **#define** KAU8\_CYCLE\_1\_STEP\_7\_ID (255)  **#define** KAU8\_CYCLE\_1\_STEP\_8\_ID (255)  II- Modify the file PAL\_SensorsManagement.c (in order to control the value of the motor current received by the BFE function)  Declare a global variable  sint32 s32\_MotorCurentInmA\_Test = 0;  Modify the two functions which send the value of the current to the BFE :  **void** **PAL\_AC\_SensorsManagement\_PAL\_runReadMotorCurrentInmA**(s32MotorCurrentInmAType \*  s32MotorCurrentInmA)  {  \*s32MotorCurrentInmA = s32\_MotorCurentInmA\_Test;  }  **void** **PAL\_AC\_SensorsManagement\_PAL\_runReadSignedMotorCurrentInA**(s8MotorCurrentInAType \*  s8MotorCurrentInA)  {  \*s8MotorCurrentInA = s32\_MotorCurentInmA\_Test/1000;  }  III- In order to avoid hardware or mechanical issues during breakpoint, modify the PAL\_ActuatorsManagement.c source (so that no PWM order is sent to the half bridges or to the boost driver) :    Declare global variables  sint16 s16MotorPowerOrder\_Test = 0;  uint16 u16BoostDutyCycle\_Test = 0;  In the function **PAL\_AC\_ActuatorsManagement\_PAL\_runApplyPowerOrder()**, just after the line  **if** (KU8\_TRUE == bSystemContextStatus) {  Replace all the code between the brackets by these two lines :  s16MotorPowerOrder\_Test = s16MotorPowerOrder;  u16BoostDutyCycle\_Test = u16BoostDutyCycle;  }  **else** {  .  .  .  *(keep the rest of the function)*  IV- Modify the function **BFE\_AC\_BeltFunctionExecution\_BFE\_runExecuteHighPowerStep(),** in order to control the value of the V+ voltage  Declare a global variable :  uint16 u16\_MotorVp\_Test = 12000;  In the BFE function, just after the line  Rte\_Call\_pclMotorAbstraction\_GetMotorVp(&u16MotorVp\_a);  add the following line :  u16MotorVp\_a = u16\_MotorVp\_Test;  BFE configuration expected : BFE\_CFG\_OPT\_HIGH\_POWER defined  ECU mockup (with a high-power hardware) flashed  Debugger with trace32  CAN environment | | | | | | | | | | |  |
|  |  |
|  |  |
|  |  |  |  | |  | |  |  | |  |  |  |
|  | **INITIAL STATE:**  ECU flashed and not running.  Add a breakpoint in the function *BFE\_AC\_BeltFunctionExecution\_BFE\_runExecuteHighPowerStep()*, just after the line :  Rte\_Call\_pclMotorPowerOrder\_SetPowerOrder(s16PWMOrder, KU8\_ONE, (uint16) (KU16\_PWM\_MAX\_BOOST - u16IntegerTemp));  Use the debugger to control the following variables :   * *s16MotorPowerOrder\_Test* * *u16BoostDutyCycle\_Test* * *s32\_MotorCurentInmA\_Test* * *u16\_MotorVp\_Test* | | | | | | | | | | |  |
|  |  |
|  |  |
|  |  |
|  |  |  |  | |  | |  |  | |  |  |  |
|  | **ACTION:**   1. Run the application 2. Launch the pre-crash cycle 1 3. Change the value of *s32\_MotorCurentInmA\_Test* : 20000   Run the application until *s16MotorPowerOrder\_Test* reaches the value of 6400   1. Run the application until the value of *u16BoostDutyCycle\_Test* stops changing 2. Change the value of *s32\_MotorCurentInmA\_Test* : 15000   Run the application until the value of *u16BoostDutyCycle\_Test* stops changing   1. Change the value of *s32\_MotorCurentInmA\_Test* : 44000   Run the application until *u16BoostDutyCycle\_Test* reaches the value of 0   1. Run the application until *s16MotorPowerOrder\_Test* stops changing 2. Change the value of *u16\_MotorVp\_Test* : 11000   Run the application (click the “Go” button several times)   1. Change the value of *u16\_MotorVp\_Test* : 13000   Run the application (click the “Go” button several times) | | | | | | | | | | |  |
|  |  |
|  |  |
|  |  |
|  |  |  |  | |  | |  |  | |  |  |  |
|  | **EXPECTED RESULT:**   1. The breakpoint is not reached 2. The program stops at the breakpoint 3. Each time the program stops at the breakpoint, check that the value of *s16MotorPowerOrder\_Test* is increasing, and the value of *u16BoostDutyCycle\_Test* is null, as long as *s16MotorPowerOrder\_Test* is under 6400, and starts to increase as soon as *s16MotorPowerOrder\_Test* reaches the value of 6400 4. Check that the value of *s16MotorPowerOrder\_Test* is 6400, and *u16BoostDutyCycle\_Test* is increasing until 256   (Isupply = Imot × 1/(1-PWMBoost/512) with Imot = 20 A and I*supply* = 40 A 🡺 PWM\_Boost = 50%)   1. Check that the value of *s16MotorPowerOrder\_Test* is 6400, and *u16BoostDutyCycle\_Test* is increasing until 320   (Isupply = Imot × 1/(1-PWMBoost/512) with Imot = 15 A and I*supply* = 40 A 🡺 PWM\_Boost = 62,5%)   1. Each time the program stops at the breakpoint, check that the value of *s16MotorPowerOrder\_Test* is 6400, and *u16BoostDutyCycle\_Test* is decreasing 2. Each time the program stops at the breakpoint, check that the value of *u16BoostDutyCycle\_Test* is null, and the value of *s16MotorPowerOrder\_Test* is decreasing until a value around 5824   (Isupply = Imot × PWMHB/6400, with Imot = 44 A and Isupply = 40 A 🡺 PWMHB = 91%)   1. Each time the program stops at the breakpoint, check that the value of *u16BoostDutyCycle\_Test* is increasing (to maintain the motor voltage to 12 V) 2. Each time the program stops at the breakpoint, check that the value of *u16BoostDutyCycle\_Test* is decreasing | | | | | | | | | | |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |  |  | |  | |  |  | |  |  |  |
|  |  |  |  | |  | |  |  | |  |  |  |

### INT\_BFE\_03003: test of the boost interrupt mechanism in case of a too low motor current

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  | |  |  |  | |  |  | |  |  |
|  | **Type of the test: N**N, R, E or Q | | | | | | | | | | |  |
|  |  |  | |  |  |  | |  |  | |  |  |
|  | **Purpose of the test: This test aims at checking that the boost is interrupted if the measured current in the motor is not increasing while the boost duty cycle is at its maximum value.**  Briefly sum-up the purpose of the test. | | | | | | | | | | |  |
|  |  |
|  |  |
|  |  |  |  | |  | |  |  | |  |  |  |
|  | **Environment:**  This test needs modifications in the sources : see previous test  ECU mockup (with a high-power hardware) flashed  Debugger with trace32  CAN environment | | | | | | | | | | |  |
|  |  |
|  |  |  |  | |  | |  |  | |  |  |  |
|  | **INITIAL STATE:**  ECU flashed and not running.  Add a breakpoint in the function *BFE\_AC\_BeltFunctionExecution\_BFE\_runExecuteHighPowerStep()*,  just after the line :  Rte\_Call\_pclMotorPowerOrder\_SetPowerOrder(s16PWMOrder, KU8\_ONE, (uint16) (KU16\_PWM\_MAX\_BOOST - u16IntegerTemp));  Use the debugger to control the following variables :   * *s16MotorPowerOrder\_Test* * *u16BoostDutyCycle\_Test* * *s32\_MotorCurentInmA\_Test* | | | | | | | | | | |  |
|  |  |
|  |  |
|  |  |
|  |  |  |  | |  | |  |  | |  |  |  |
|  | **ACTION:**   1. Run the application 2. Launch the pre-crash cycle 1 3. Click the “go” button several times (the value of *s32\_MotorCurentInmA\_Test* must keep a value of 0) | | | | | | | | | | |  |
|  |  |
|  |  |
|  |  |
|  |  |  |  | |  | |  |  | |  |  |  |
|  | **EXPECTED RESULT:**   1. The breakpoint is not reached 2. The program stops at the breakpoint 3. Check that the value of *u16BoostDutyCycle\_Test* is null as long as *s16MotorPowerOrder\_Test* is under 6400, and starts to increase as soon as *s16MotorPowerOrder\_Test* reaches the value of 6400, until its maximum value, and then starts to decrease until 0. | | | | | | | | | | |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |  |  | |  | |  |  | |  |  |  |
|  |  |  |  | |  | |  |  | |  |  |  |

### INT\_BFE\_03004: test of the boost interrupt mechanism at the end of a High-Power step

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  | |  |  |  | |  |  | |  |  |
|  | **Type of the test: N**N, R, E or Q | | | | | | | | | | |  |
|  |  |  | |  |  |  | |  |  | |  |  |
|  | **Purpose of the test: This test aims at checking that the boost is interrupted 40 ms before the end of the step, even if the current has not reached the order**Briefly sum-up the purpose of the test. | | | | | | | | | | |  |
|  |  |
|  |  |
|  |  |  |  | |  | |  |  | |  |  |  |
|  | **Environment:**  This test needs modifications in the sources : see previous test  To make the test easier, the high power step will be shorten (50 ms) : modify the step configuration in the file NVP\_Const.h :  **#define** KU16\_STEP\_01\_TIME (0x0005)  **#define** KU8\_STEP\_01\_ORDER\_VAL (0x50)  **#define** KU8\_STEP\_01\_ORDER\_TYP (0x01)  **#define** KU16\_STEP\_01\_OPTIONS (0x0001)  ECU mockup (with a high-power hardware) flashed  Debugger with trace32  CAN environment | | | | | | | | | | |  |
|  |  |
|  |  |  |  | |  | |  |  | |  |  |  |
|  | **INITIAL STATE:**  ECU flashed and not running.  Add a breakpoint in the function *BFE\_AC\_BeltFunctionExecution\_BFE\_runExecuteHighPowerStep()*,  just after the line :  Rte\_Call\_pclMotorPowerOrder\_SetPowerOrder(s16PWMOrder, KU8\_ONE, (uint16) (KU16\_PWM\_MAX\_BOOST - u16IntegerTemp));  Use the debugger to control the following variables :   * *s16MotorPowerOrder\_Test* * *u16BoostDutyCycle\_Test* * *s32\_MotorCurentInmA\_Test* | | | | | | | | | | |  |
|  |  |
|  |  |
|  |  |
|  |  |  |  | |  | |  |  | |  |  |  |
|  | **ACTION:**   1. Run the application 2. Launch the pre-crash cycle 1 3. Change the value of *s32\_MotorCurentInmA\_Test* : 1000   Run the application   1. Click the “go” button several times | | | | | | | | | | |  |
|  |  |
|  |  |
|  |  |
|  |  |  |  | |  | |  |  | |  |  |  |
|  | **EXPECTED RESULT:**   1. The breakpoint is not reached 2. The program stops at the breakpoint 3. The program stops at the breakpoint : check that the value of *s16MotorPowerOrder\_Test* has increased, and the value of *u16BoostDutyCycle\_Test* is null 4. Check that the value of *u16BoostDutyCycle\_Test* is null until *s16MotorPowerOrder\_Test* is under 6400, and starts to increase as soon as *s16MotorPowerOrder\_Test* reaches the value of 6400   After 10ms (25 steps of 400 µs), check that the boost duty cycle starts to decrease until 0 (the boost is interrupted). | | | | | | | | | | |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |  |  | |  | |  |  | |  |  |  |
|  |  |  |  | |  | |  |  | |  |  |  |

### INT\_BFE\_03005: test of a cycle with a High-Power step

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  | |  |  |  | |  |  | |  |  |
|  | **Type of the test: N**N, R, E or Q | | | | | | | | | | |  |
|  |  |  | |  |  |  | |  |  | |  |  |
|  | **Purpose of the test: This test aims at checking the correct behavior of a high power step in “real time” (the duration of the step and the current order).**  Briefly sum-up the purpose of the test. | | | | | | | | | | |  |
|  |  |
|  |  |
|  |  |  |  | |  | |  |  | |  |  |  |
|  | **Environment:**  (**Beware** : do not keep the soft modifications from the previous tests)  Modify the cycles configuration, in the file *NVP\_Const.h* :   * Configuration of the step parameters   **#define** KU16\_STEP\_01\_TIME (0x0014)  **#define** KU8\_STEP\_01\_ORDER\_VAL (0x04)  **#define** KU8\_STEP\_01\_ORDER\_TYP (0x01)  **#define** KU16\_STEP\_01\_OPTIONS (0x0001)  **#define** KU16\_STEP\_02\_TIME (0x0014)  **#define** KU8\_STEP\_02\_ORDER\_VAL (0x04)  **#define** KU8\_STEP\_02\_ORDER\_TYP (0x01)  **#define** KU16\_STEP\_02\_OPTIONS (0x0000)  **#define** KU16\_STEP\_03\_TIME (0x000A)  **#define** KU8\_STEP\_03\_ORDER\_VAL (0x00)  **#define** KU8\_STEP\_03\_ORDER\_TYP (0x81)  **#define** KU16\_STEP\_03\_OPTIONS (0x0000)   * Definition of the cycle 1   **#define** KU8\_CYCLE\_1\_NEXT\_CYCLE (0x03)  **#define** KAU8\_CYCLE\_1\_STEP\_1\_ID (0)  **#define** KAU8\_CYCLE\_1\_STEP\_2\_ID (1)  **#define** KAU8\_CYCLE\_1\_STEP\_3\_ID (2)  **#define** KAU8\_CYCLE\_1\_STEP\_4\_ID (255)  **#define** KAU8\_CYCLE\_1\_STEP\_5\_ID (255)  **#define** KAU8\_CYCLE\_1\_STEP\_6\_ID (255)  **#define** KAU8\_CYCLE\_1\_STEP\_7\_ID (255)  **#define** KAU8\_CYCLE\_1\_STEP\_8\_ID (255)  In the file *BFE\_BeltFunctionExecution.c* :   * add a toggle pin (DEBUG\_04) in the function *BFE\_AC\_BeltFunctionExecution\_BFE\_runExecuteHighPowerStep*   just before the call of the function *Rte\_Call\_pclMotorPowerOrder\_SetPowerOrder*   * add a toggle pin (DEBUG\_05) in the function *BFE\_AC\_BeltFunctionExecution\_BFE\_runExecuteSteps*   just before the call of the function *Rte\_Call\_pclMotorPowerOrder\_SetPowerOrder*  ECU mockup (with a high-power hardware) flashed (for more accurate results, it is better to use the right half-bridges calibration values related to the mock-up used)  Debugger with trace32  CAN environment  Resistive load  Capture the following signals with the oscilloscope:   * The debug pin * Current supply (with a current probe) | | | | | | | | | | |  |
|  |  |
|  |  |
|  |  |  |  | |  | |  |  | |  |  |  |
|  | **INITIAL STATE:**  ECU flashed and not running. | | | | | | | | | | |  |
|  |  |
|  |  |
|  |  |
|  |  |  |  | |  | |  |  | |  |  |  |
|  | **ACTION:**   1. Run the application 2. Configure the oscilloscope to trig on a rising edge of the toggle pin *DEBUG\_04*   Launch the pre-crash cycle 1   1. Change the value of the KU8\_STEP\_01\_ORDER\_VAL for higher values (20A, 30A, 40A) | | | | | | | | | | |  |
|  |  |
|  |  |
|  |  |
|  |  |  |  | |  | |  |  | |  |  |  |
|  | **EXPECTED RESULT:**   1. Check that the first debug pin *DEBUG\_04* is not toggled, and that the second debug pin *DEBUG\_05* is toggled. 2. Check that the first debug pin *DEBUG\_04* is toggled during the first step (the High-Power step) and not toggled during the second step (the standard step).   Check that the second debug pin *DEBUG\_05* is not toggled during the first step (the High-Power step) and toggled during the second step (the standard step).  Check that the duration of the High-Power step corresponds to the duration defined in the file NVP\_Const.h  Check that the supply current measured with the probe corresponds to the order defined in the file NVP\_Const.h   1. Check that the supply current measured with the probe corresponds to the order defined in the file NVP\_Const.h | | | | | | | | | | |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |  |  | |  | |  |  | |  |  |  |
|  |  |  |  | |  | |  |  | |  |  |  |

## Tests for ManageStepEvents function

### INT\_BFE\_04001: Step with ‘Trigger Off’ option

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  | |  |  |  | |  |  | |  |  |
|  | **Type of the test: N** | | | | | | | | | | |  |
|  |  |  | |  |  |  | |  |  | |  |  |
|  | **Purpose of the test:**  The goal of this test is to check that the ‘trigger off’ option will interrupt correctly the running step. | | | | | | | | | | |  |
|  |  |
|  |  |
|  |  |  |  | |  | |  |  | |  |  |  |
|  | **Environment:**  CAN environment to stay alive.  ECU or mock-up flashed with an instrumented code and BFE\_CFG\_OPT\_TRIGG\_OFF defined.  Debugger plugged  On the Watch Window of the debugger read Access to :   * BFE\_bIntStepFlag * u8StepToExecute   The Trigger Off option implementation is specific to each project. So the code instrumentation needed can’t be described : just adapt the SW component in charge of Rte\_Write\_psrVehAbstract\_b8IsCarAccelLow or Rte\_Write\_psrCarInformation\_b8IsCarAccelLow to control the Boolean value | | | | | | | | | | |  |
|  |  |
|  |  |
|  |  |  |  | |  | |  |  | |  |  |  |
|  | **INITIAL STATE:**  ECU flashed and not running  To trig belt functions, the instrumentation described in 7.1 can be used at the BFS algorithm level.  Cycle 01 configured with 2 steps :  1st step = 35 sec / -10% PWM / trigger off option  2nd step = 100 ms / 0% PWM / ramp | | | | | | | | | | |  |
|  |  |
|  |  |
|  |  |
|  |  |  |  | |  | |  |  | |  |  |  |
|  | **ACTION:**   1. Perform an ‘In Target reset’ of the application. 2. Run the application 3. Set the b8IsCarAccelLow data to FALSE then start the cycle 01 4. Set the b8IsCarAccelLow data to TRUE 5. Start again the cycle 01 | | | | | | | | | | |  |
|  |  |
|  |  |
|  |  |
|  |  |  |  | |  | |  |  | |  |  |  |
|  | **EXPECTED RESULT:**   1. Nothing expected 2. Checks that no cycle is executed (excepted the Anti-patina belt function). 3. Checks that the cycle 01 starts and executes the first step 4. Checks that the first step is interrupted in less than 20 ms and the 2nd step starts 5. Checks that the 1st step will be executed for 10ms then the 2nd step begins. | | | | | | | | | | |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |  |  | |  | |  |  | |  |  |  |
|  |  |  |  | |  | |  |  | |  |  |  |

### INT\_BFE\_04002: Step with ‘Current Interruption point’ option

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  | |  |  |  | |  |  | |  |  |
|  | **Type of the test: N** | | | | | | | | | | |  |
|  |  |  | |  |  |  | |  |  | |  |  |
|  | **Purpose of the test:**  The goal of this test is to check that the ‘Current Interruption point’ option will interrupt correctly the running step. | | | | | | | | | | |  |
|  |  |
|  |  |
|  |  |  |  | |  | |  |  | |  |  |  |
|  | **Environment:**  CAN environment to stay alive.  ECU or mock-up flashed with an instrumented code and BFE\_CFG\_OPT\_ADAPT\_CURRENT defined.  Debugger plugged  On the Watch Window of the debugger read Access to :   * BFE\_bIntStepFlag * u8StepToExecute   For this test, it shall be possible to change the temperature value provided by SBC module, and it shall be also possible to change the motor current in mA provided by the PAL sensor component.  For this test, the excel sheet in the 7.4 can be used to get the current threshold that shall be reached to interrupt the executed step. | | | | | | | | | | |  |
|  |  |
|  |  |
|  |  |  |  | |  | |  |  | |  |  |  |
|  | **INITIAL STATE:**  ECU flashed and not running  To trig belt functions, the instrumentation described in 7.1 can be used at the BFS algorithm level.  Cycle 01 configured with 2 steps :  1st step = 35 sec / -10% PWM / ‘current interruption point’ option with the value 20  2nd step = 100 ms / 0% PWM / ramp  Cycle 02 configured with 2 steps :  1st step = 35 sec / -10% PWM / ‘current interruption point’ option with the value 40  2nd step = 100 ms / 0% PWM / ramp | | | | | | | | | | |  |
|  |  |
|  |  |
|  |  |
|  |  |  |  | |  | |  |  | |  |  |  |
|  | **ACTION:**   1. Perform an ‘In Target reset’ of the application. 2. Run the application 3. Set the motor current to 0mA and set temperature to 200 lsb then start the cycle 01 4. Set the motor current to a value lower than the threshold provided by 7.4 5. Set the motor current to a value upper than the threshold provided by 7.4 6. Set the motor current to 0mA and set temperature to 400 lsb then restart the cycle 01 7. Set the motor current to a value lower than the threshold provided by 7.4 8. Set the motor current to a value upper than the threshold provided by 7.4 9. Repeat actions 3 to 8 with the cycle 02 | | | | | | | | | | |  |
|  |  |
|  |  |
|  |  |
|  |  |  |  | |  | |  |  | |  |  |  |
|  | **EXPECTED RESULT:**   1. Nothing expected 2. Checks that no cycle is executed (excepted the Anti-patina belt function). 3. Checks that the cycle 01 starts and executes the first step 4. Checks that the first step continues 5. Checks that the first step is interrupted in less than 20 ms and the 2nd step starts 6. Same as -3- 7. Same as -4- 8. Same as -5- 9. Same results than -3- to -8- but threshold are now based on the value 40 for the current interruption option | | | | | | | | | | |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |  |  | |  | |  |  | |  |  |  |
|  |  |  |  | |  | |  |  | |  |  |  |

### INT\_BFE\_04003: Step with ‘Motor Blocked’ option

**TODO**

### INT\_BFE\_04004: Step with ‘Belt Blocked’ option

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  | |  |  |  | |  |  | |  |  |
|  | **Type of the test: N** | | | | | | | | | | |  |
|  |  |  | |  |  |  | |  |  | |  |  |
|  | **Purpose of the test:**  The goal of this test is to check that the ‘Belt Blocked’ option will interrupt correctly the running step. | | | | | | | | | | |  |
|  |  |
|  |  |
|  |  |  |  | |  | |  |  | |  |  |  |
|  | **Environment:**  CAN environment to stay alive.  ECU or mock-up flashed with an instrumented code and BFE\_CFG\_OPT\_BELT\_BLOCKED\_HES defined.  Debugger plugged  On the Watch Window of the debugger read Access to :   * BFE\_bIntStepFlag * u8StepToExecute   Mockup instrumented with a specific motor to control the magnetic wheel OR  BDS service in charge belt speed retrieving shall be adapted to be able to set the belt speed with a global data | | | | | | | | | | |  |
|  |  |
|  |  |
|  |  |  |  | |  | |  |  | |  |  |  |
|  | **INITIAL STATE:**  ECU flashed and not running  To trig belt functions, the instrumentation described in 7.1 can be used at the BFS algorithm level.  Cycle 01 configured with 2 steps :  1st step = 35 sec / -10% PWM / Belt blocked option  2nd step = 100 ms / 0% PWM / ramp | | | | | | | | | | |  |
|  |  |
|  |  |
|  |  |
|  |  |  |  | |  | |  |  | |  |  |  |
|  | **ACTION:**   1. Perform an ‘In Target reset’ of the application. 2. Run the application 3. Set the belt speed to a value ≠ 0 or supply the motor of the magnetic wheel then start the cycle 01 4. Set the belt speed to 0 or cut the supply of the motor 5. Start again the cycle 01 | | | | | | | | | | |  |
|  |  |
|  |  |
|  |  |
|  |  |  |  | |  | |  |  | |  |  |  |
|  | **EXPECTED RESULT:**   1. Nothing expected 2. Checks that no cycle is executed (excepted the Anti-patina belt function). 3. Checks that the cycle 01 starts and executes the first step 4. Checks that the first step is interrupted in less than 20 ms and the 2nd step starts 5. Checks that the 1st step will be executed for 10ms then the 2nd step begins. | | | | | | | | | | |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |  |  | |  | |  |  | |  |  |  |
|  |  |  |  | |  | |  |  | |  |  |  |

### INT\_BFE\_04005: Step with ‘Belt Movement Detection’ option

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  | |  |  |  | |  |  | |  |  |
|  | **Type of the test: N** | | | | | | | | | | |  |
|  |  |  | |  |  |  | |  |  | |  |  |
|  | **Purpose of the test:**  The goal of this test is to check that the ‘Belt Movement Detection’ option will interrupt correctly the running step. | | | | | | | | | | |  |
|  |  |
|  |  |
|  |  |  |  | |  | |  |  | |  |  |  |
|  | **Environment:**  CAN environment to stay alive.  ECU or mock-up flashed with an instrumented code and BFE\_CFG\_OPT\_BELT\_MVT\_DETECT defined.  Debugger plugged  On the Watch Window of the debugger read Access to :   * BFE\_bIntStepFlag * u8StepToExecute   Mockup instrumented with a specific motor to control the magnetic wheel OR  BDS service in charge belt speed retrieving shall be adapted to be able to set the belt speed with a global data | | | | | | | | | | |  |
|  |  |
|  |  |
|  |  |  |  | |  | |  |  | |  |  |  |
|  | **INITIAL STATE:**  ECU flashed and not running  To trig belt functions, the instrumentation described in 7.1 can be used at the BFS algorithm level.  Cycle 01 configured with 2 steps :  1st step = 35 sec / -10% PWM / Belt Movement detection option  2nd step = 100 ms / 0% PWM / ramp | | | | | | | | | | |  |
|  |  |
|  |  |
|  |  |
|  |  |  |  | |  | |  |  | |  |  |  |
|  | **ACTION:**   1. Perform an ‘In Target reset’ of the application. 2. Run the application 3. Set the belt speed to 0 or cut the supply of the motor then start the cycle 01 4. Set the belt speed to a value ≠ 0 or supply the motor of the magnetic wheel 5. Start again the cycle 01 | | | | | | | | | | |  |
|  |  |
|  |  |
|  |  |
|  |  |  |  | |  | |  |  | |  |  |  |
|  | **EXPECTED RESULT:**   1. Nothing expected 2. Checks that no cycle is executed (excepted the Anti-patina belt function). 3. Checks that the cycle 01 starts and executes the first step 4. Checks that the first step is interrupted in less than 20 ms and the 2nd step starts 5. Checks that the 1st step will be executed for 10ms then the 2nd step begins. | | | | | | | | | | |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |  |  | |  | |  |  | |  |  |  |
|  |  |  |  | |  | |  |  | |  |  |  |

## Tests for the InitConsigns function

### INT\_BFE\_05001: Step duration management

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  | |  |  |  | |  |  | |  |  |
|  | **Type of the test: N** | | | | | | | | | | |  |
|  |  |  | |  |  |  | |  |  | |  |  |
|  | **Purpose of the test:**  The goal of this test is to check that the ‘Belt Movement Detection’ option will interrupt correctly the running step. | | | | | | | | | | |  |
|  |  |
|  |  |
|  |  |  |  | |  | |  |  | |  |  |  |
|  | **Environment:**  CAN environment to stay alive.  ECU or mock-up flashed with an instrumented code and BFE\_CFG\_OPT\_BELT\_MVT\_DETECT defined.  Debugger plugged  On the Watch Window of the debugger read Access to :   * BFE\_bIntStepFlag * u8StepToExecute   One debug pin shall be used to output the state of BFE\_bIntStepFlag  The 3 others debugs pin shall be associated to the step 1 / 2 / 3 of the cycle 01 | | | | | | | | | | |  |
|  |  |
|  |  |
|  |  |  |  | |  | |  |  | |  |  |  |
|  | **INITIAL STATE:**  ECU flashed and not running  To trig belt functions, the instrumentation described in 7.1 can be used at the BFS algorithm level.  Cycle 01 configured with 3 steps :  1st step = 200 ms / -10% PWM / no option  2nd step = 40 ms / -10% PWM / no option  3rd step = 3 sec / -10% PWM / no option | | | | | | | | | | |  |
|  |  |
|  |  |
|  |  |
|  |  |  |  | |  | |  |  | |  |  |  |
|  | **ACTION:**   1. Perform an ‘In Target reset’ of the application. 2. Run the application 3. Start cycle 01 | | | | | | | | | | |  |
|  |  |
|  |  |
|  |  |
|  |  |  |  | |  | |  |  | |  |  |  |
|  | **EXPECTED RESULT:**   1. Nothing expected 2. Checks that no cycle is executed (excepted the Anti-patina belt function). 3. Checks that the cycle 01 starts and :  * During 200 ms the debug pin of the step 1 shall be at +5V, the 2 others shall be to 0V * After 190 ms and during 10 ms the BFE\_bIntStepFlag shall be at +5V * Then for the 40 next ms the debug pin of the step 2 shall be at +5V, the 2 others shall be to 0V * After 30 ms and during 10 ms the BFE\_bIntStepFlag shall be at +5V * Then for the 3 next sec the debug pin of the step 3 shall be at +5V, the 2 others shall be to 0V | | | | | | | | | | |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |  |  | |  | |  |  | |  |  |  |
|  |  |  |  | |  | |  |  | |  |  |  |

### INT\_BFE\_05002: Internal data exchanged between the 10ms and the 2ms main functions

### INT\_BFE\_05003: Direction reverse protection management

## Temperature adaptivity

### INT\_BFE\_06001 : Default tensioning temperature correction factors

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  | |  |  |  | |  |  | |  |  |
|  | **Type of the test: N** N, R, E or Q | | | | | | | | | | |  |
|  |  |  | |  |  |  | |  |  | |  |  |
|  | **Purpose of the test:**  The aim of this test is to check the default tensioning temperature correction factors. | | | | | | | | | | |  |
|  |  |
|  |  |
|  |  |  |  | |  | |  |  | |  |  |  |
|  | **Environment:**  A PP4G board flashed.  Power supply.  CANalyzer simulation. | | | | | | | | | | |  |
|  |  |
|  |  |
|  |  |  |  | |  | |  |  | |  |  |  |
|  | **INITIAL STATE:**  Describe here if a special configuration is needed here (EEP parameters with some particular values…). | | | | | | | | | | |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |  |  | |  | |  |  | |  |  |  |
|  | **ACTION:**  …  Action N : ...Action  Action 1: Run the software and read tensioning temperature correction factors.  0x22 0xFD 0x36; | | | | | | | | | | |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |  |  | |  | |  |  | |  |  |  |
|  | **EXPECTED RESULT:**  Result 1: Check the default tensioning temperature correction factors are the specified ones.  [COVERS: DES\_TF\_E\_1738]  **It is important for traceability to know the SW Module Design and/or SW Achitecture requirement covered by this test:**  **[COVERS : DSG\_MOD\_XXXXX/ARCH\_XXXXX]**  **MOD: SW component trigram**  **ARCH: SW Architecture prefix**  **XXXXX: requirement number**  Result 1 : ... [COVERS : DSG\_MOD\_xxxxx]  Result 2 : ... [COVERS : ARCH\_xxxxx]  …  Result N : ... [COVERS : DSG\_MOD\_xxxxx or ARCH\_xxxxx] | | | | | | | | | | |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |  |  | |  | |  |  | |  |  |  |
|  |  |  |  | |  | |  |  | |  |  |  |

### INT\_BFE\_06002 : Nominal tensioning temperature correction with current/motor/PWM steps

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  | |  |  |  | |  |  | |  |  |
|  | **Type of the test: N** N, R, E or Q | | | | | | | | | | |  |
|  |  |  | |  |  |  | |  |  | |  |  |
|  | **Purpose of the test:**  The aim of this test is to check that the tensioning temperature correction is applied to configured current/motor/PWM steps and, in comparison, that it is not applied to not configured steps. | | | | | | | | | | |  |
|  |  |
|  |  |
|  |  |  |  | |  | |  |  | |  |  |  |
|  | **Environment:**  A PP4G board flashed.  Power supply.  CANalyzer simulation.  BDM is mandatory to check the applied order. | | | | | | | | | | |  |
|  |  |
|  |  |
|  |  |  |  | |  | |  |  | |  |  |  |
|  | **INITIAL STATE:**  Describe here if a special configuration is needed here (EEP parameters with some particular values…).  Add some instrumented code into PMP component to allow setting internal temperature with Trace32.  **static uint8 b8Test;**  **static uint16 u16TestTemperature;**  void **PMP\_runGetFilteredTemperature** ( … )  {  …    **if(b8Test == 1)**  **{**  **\*pu16FilteredTemperature = = u16TestTemperature;**  **}**  }  Start CAN simulation and load application. | | | | | | | | | | |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |  |  | |  | |  |  | |  |  |  |
|  | **ACTION:**  …  Action N : ...Action  Action 1: Run the software and write new Tensioning and Comfort temperature correction parameters.  0x2E 0xFD 0x36 0x00 0x38 0x00 0xC6 0x01 0x31 0x01 0x87 0x01 0xC8 0x02 0x1E 0x02 0x52 0xC0 0xC0 0xC0 0x80 0x60 0x40 0x51;  0x2E 0xFD 0x37 0x00 0x38 0x00 0xC6 0x01 0x31 0x01 0x87 0x01 0xC8 0x02 0x1E 0x02 0x52 0x80  0x46 0x42 0x40 0x4A 0x50 0x51;  Temperature adaptation for a motor voltage step:  Action 2: Write a new mono step cycle (PRE level 1 for instance) with a motor voltage order of 6V during 3s, where the W weighted configuration bit is set.  For instance cycle PRE 1 should be: 0x04 0x11 0x07 0x0B 0x01 0x2C 0x18 0x04 0x00 0x06 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0x26 0x5F;  Action 3: Write a new mono step cycle (PRE level 4 for instance) with a motor voltage order of 6V during 3s, where neither W weighted configuration bit is not set.  For instance cycle PRE 4 should be: 0x04 0x11 0x07 0x0B 0x01 0x2C 0x18 0x04 0x00 0x04 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0x26 0x5D;  Action 4: Set the temperature to a value where the Tensioning and Comfort temperature correction factors are different from 1, for instance at -40°C, BSR factor = 2 and PRE factor = 3 (set **u16TestTemperature** =0x38 and b8Test=1 with Trace32).  Action 5: Watch *s32FirstOrderValue* from *runScheduleStep* BFE runnable, with Trace32. Launch PRE level 1.  Action 6: Launch PRE level 4.  Action 7: Set the temperature to another value where the PRE and BSR adaptation factors are different, for instance at 38°C, BSR factor = 1 and PRE factor = 2 (set **u16TestTemperature** =0x187 and b8Test=1 with Trace32).  Action 8: Launch PRE level 1.  Action 9: Launch PRE level 4.  Action 10: Set the temperature to a value between 2 temperature consecutive points to check the interpolation, for instance at 30°C, PRE factor = 2 (set **u16TestTemperature** =0x164=29.77°C and b8Test=1 with Trace32).  Action 11: Launch PRE level 1.  Action 12: Launch PRE level 4.  Temperature adaptation for a motor current step:  Action 13: Write a new mono step cycle (PRE level 2 for instance) with a motor current order of 8A during 3s, where the W weighted configuration bit is set.  For instance cycle PRE 2 should be: 0x04 0x11 0x07 0x0B 0x01 0x2C 0x10 0x01 0x00 0x06 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0x1E 0x5C;  Action 14: Write a new mono step cycle (PRE level 5 for instance) with a motor current order of 8A during 3s, where W weighted configuration bit is not set.  For instance cycle PRE 5 should be: 0x04 0x11 0x07 0x0B 0x01 0x2C 0x10 0x01 0x00 0x04 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0x1E 0x5A;  Action 15: Set the temperature for instance at -40°C, where BSR factor = 2 and PRE factor = 3 (set **u16TestTemperature** =0x38 and b8Test=1 with Trace32).  Action 16: Launch PRE level 2.  Action 17: Launch PRE level 5.  Action 18: Set the temperature for instance at 38°C, where BSR factor = 1 and PRE factor = 2 (set u16TestAdcMux=0x187 and b8Test=1 with Trace32).  Action 19: Launch PRE level 2.  Action 20: Launch PRE level 5.  Temperature adaptation for a PWM step:  Action 21: Write a new mono step cycle (PRE level 3 for instance) with a PWM order of 50% during 3s, where the W weighted configuration bit is set.  For instance cycle PRE 3 should be: 0x04 0x11 0x07 0x0B 0x01 0x2C 0x32 0x00 0x00 0x06 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0x40 0x5B;  Action 22: Write a new mono step cycle (PRE level 6 for instance) with a PWM order of 50% during 3s, where W weighted configuration bit is not set.  For instance cycle PRE 6 should be: 0x04 0x11 0x07 0x0B 0x01 0x2C 0x32 0x00 0x00 0x04 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0x40 0x59;  Action 23: Set the temperature for instance at -40°C, where BSR factor = 2 and PRE factor = 3 (set **u16TestTemperature** =0x38 and b8Test=1 with Trace32).  Action 24: Launch PRE level 3.  Action 25: Launch PRE level 6.  Action 26: Set the temperature for instance at 38°C, where BSR factor = 1 and PRE factor = 2 (set **u16TestTemperature** =0x187 and b8Test=1 with Trace32).  Action 27: Launch PRE level 3.  Action 28: Launch PRE level 6. | | | | | | | | | | |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |  |  | |  | |  |  | |  |  |  |
|  |  |  |  | |  | |  |  | |  |  |  |
|  | **EXPECTED RESULT:**  **It is important for traceability to know the SW Module Design and/or SW Achitecture requirement covered by this test:**  **[COVERS : DSG\_MOD\_XXXXX/ARCH\_XXXXX]**  **MOD: SW component trigram**  **ARCH: SW Architecture prefix**  **XXXXX: requirement number**  Result 1 : ... [COVERS : DSG\_MOD\_xxxxx]  Result 2 : ... [COVERS : ARCH\_xxxxx]  …  Result N : ... [COVERS : DSG\_MOD\_xxxxx or ARCH\_xxxxx]  Result 1: Check that start-up profile is triggered. Read PRE and BSR tensioning weight parameters to check them (0x22 0xFD 0x36 and 0x22 0xFD 0x37) or check them with Trace32 into Ram mirror of Nvm block.  Temperature adaptation for a motor voltage step:  Result 3: Read the level 1 and 4 PRE cycles parameters (0x22 0xFD 0x10 and 0x22 0xFD 0x13) or check them withTrace32 into Ram mirror of Nvm block.  Result 5: Check the PRE level 1 cycle is triggered and the order is adapted with a weighted factor 3:  *s32FirstOrderValue =* 3 x voltage order x 4 x 1024 = 73728.  Result 6: Check the PRE level 4 cycle is triggered and the order is not adapted:  *s32FirstOrderValue =* 1 x voltage order x 4 x 1024 = 24576.  Result 8: Check the PRE level 1 cycle is triggered and the order is adapted with a weighted factor 2:  *s32FirstOrderValue =* 2 x voltage order x 4 x 1024 = 49152.  Result 9: Check the PRE level 4 cycle is triggered and the order is not adapted:  *s32FirstOrderValue =* 1 x voltage order x 4 x 1024 = 24576.  Result 11: Check the PRE level 1 cycle is triggered and the order is adapted with a weighted factor 2.40625:  *s32FirstOrderValue =* 2.40625 x voltage order x 4 x 1024 = 59136.  Result 12: Check the PRE level 4 cycle is triggered and the order is not adapted:  *s32FirstOrderValue =* 1 x voltage order x 4 x 1024 = 24576.  Temperature adaptation for a motor current step:  Result 14: Read the level 2 and 5 PRE cycles parameters (0x22 0xFD 0x11 and 0x22 0xFD 0x14) or check them withTrace32 into Ram mirror of Nvm block.  Result 16: Check the PRE level 2 cycle is triggered and the order is adapted with a weighted factor 3:  *s32FirstOrderValue =* 3 x current order x 2 x 1024 = 49152.  Result 17: Check the PRE level 5 cycle is triggered and the order is not adapted:  *s32FirstOrderValue =* 1 x current order x 2 x 1024 = 16384.  Result 19: Check the PRE level 2 cycle is triggered and the order is adapted with a weighted factor 2:  *s32FirstOrderValue =* 2 x current order x 2 x 1024 = 32768.  Result 20: Check the PRE level 5 cycle is triggered and the order is not adapted:  *s32FirstOrderValue =* 1 x current order x 2 x 1024 = 16384.  Temperature adaptation for a PWM step:  Result 22: Read the level 3 and 6 PRE cycles parameters (0x22 0xFD 0x12 and 0x22 0xFD 0x15) or check them withTrace32 into Ram mirror of Nvm block.  Result 24: Check the PRE level 3 cycle is triggered and the order is adapted with a weighted factor 3:  *s32FirstOrderValue =* 3 x PWM order x 1024 = 153 600.  Result 25: Check the PRE level 6 cycle is triggered and the order is not adapted:  *s32FirstOrderValue =* 1 x PWM order x 1024 = 51 200.  Result 27: Check the PRE level 3 cycle is triggered and the order is adapted with a weighted factor 2:  *s32FirstOrderValue =* 2 x PWM order x 1024 = 102 400.  Result 28: Check the PRE level 6 cycle is triggered and the order is not adapted:  *s32FirstOrderValue =* 1 x PWM order x 1024 = 51 200.  [COVERS: DES\_TF\_G\_1458, DES\_TF\_G\_1499, DES\_TF\_G\_1500]  [COVERS: DES\_TF\_G\_1496] | | | | | | | | | | |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  | | | | | | | | | | |  |
|  |  |  |  | |  | |  |  | |  |  |  |
|  |  |  |  | |  | |  |  | |  |  |  |

### INT\_BFE\_06003 : No temperature adaptation with absent or invalid step configuration

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  | |  |  |  | |  |  | |  |  |
|  | **Type of the test: N** N, R, E or Q | | | | | | | | | | |  |
|  |  |  | |  |  |  | |  |  | |  |  |
|  | **Purpose of the test:**  The aim of this test is to check that the temperature adaptation is not applied to the steps whom the weighted configuration is absent or invalid. | | | | | | | | | | |  |
|  |  |
|  |  |
|  |  |  |  | |  | |  |  | |  |  |  |
|  | **Environment:**  A PP4G board flashed.  Power supply.  CANalyzer simulation.  BDM is mandatory to check the applied order. | | | | | | | | | | |  |
|  |  |
|  |  |
|  |  |  |  | |  | |  |  | |  |  |  |
|  | **INITIAL STATE:**  Describe here if a special configuration is needed here (EEP parameters with some particular values…).  Add some instrumented code into PMP component to allow setting internal temperature with Trace32.  **static uint8 b8Test;**  **static uint16 u16TestTemperature;**  void **PMP\_runGetFilteredTemperature** ( … )  {  …    **if(b8Test == 1)**  **{**  **\*pu16FilteredTemperature = = u16TestTemperature;**  **}**  }  Start CAN simulation and load application. | | | | | | | | | | |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |  |  | |  | |  |  | |  |  |  |
|  | **ACTION:**  …  Action N : ...Action  No temperature adaptation configured for a motor voltage step:  Action 1: Run the software and write new PRE and BSR tensioning weight parameters.  0x2E 0xFD 0x36 0x00 0x38 0x00 0xC6 0x01 0x31 0x01 0x87 0x01 0xC8 0x02 0x1E 0x02 0x52 0xC0 0xC0 0xC0 0x80 0x60 0x40 0x51;  0x2E 0xFD 0x37 0x00 0x38 0x00 0xC6 0x01 0x31 0x01 0x87 0x01 0xC8 0x02 0x1E 0x02 0x52 0x80  0x46 0x42 0x40 0x4A 0x50 0x51;  Action 2 : Write a new mono step cycle (PRE level 4 for instance) with a motor voltage order of 6V during 3s, where W weighted configuration bit is not set.  For instance cycle PRE 4 should be: 0x04 0x11 0x07 0x0B 0x01 0x2C 0x18 0x04 0x00 0x04 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0x26 0x5D;    Action 3: Set the temperature to a value where the PRE and BSR adaptation factors are different from 1, for instance at -40°C, BSR factor = 2 and PRE factor = 3 (set **u16TestTemperature** =0x38 and b8Test=1 with Trace32).  Action 4: Watch *s32FirstOrderValue* from *runScheduleStep* BFE runnable, with Trace32. Launch PRE level 4.  No temperature adaptation configured for a motor current step:  Action 5: Write a new mono step cycle (PRE level 5 for instance) with a motor current order of 8A during 3s, where W weighted configuration bit is not set.  For instance cycle PRE 5 should be: 0x04 0x11 0x07 0x0B 0x01 0x2C 0x10 0x01 0x00 0x04 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0x1E 0x5A;  Action 6: Watch *s32FirstOrderValue* from *runScheduleStep* BFE runnable, with Trace32. Launch PRE level 5.  No temperature adaptation configured for a PWM step:  Action 7: Write a new mono step cycle (PRE level 6 for instance) with a PWM order of 50% during 3s, where W weighted configuration bit is not set.  For instance cycle PRE 6 should be: 0x04 0x11 0x07 0x0B 0x01 0x2C 0x32 0x00 0x00 0x04 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0x40 0x59;  Action 8: Watch *s32FirstOrderValue* from *runScheduleStep* BFE runnable, with Trace32. Launch PRE level 6.  Invalid temperature adaptation configured for a voltage step:  Action 9: Write a new mono step cycle (PRE level 7 for instance) with a motor voltage order of 6V during 3s, where W weighted configuration bit is not set.  For instance cycle PRE 7 should be: 0x04 0x11 0x07 0x0B 0x01 0x2C 0x18 0x04 0x00 0x16 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0x26 0x6F;  Action 10: Watch *s32FirstOrderValue* from *runScheduleStep* BFE runnable, with Trace32. Launch PRE level 7. | | | | | | | | | | |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |  |  | |  | |  |  | |  |  |  |
|  | **EXPECTED RESULT:**  No temperature adaptation configured for a motor voltage step:  **It is important for traceability to know the SW Module Design and/or SW Achitecture requirement covered by this test:**  **[COVERS : DSG\_MOD\_XXXXX/ARCH\_XXXXX]**  **MOD: SW component trigram**  **ARCH: SW Architecture prefix**  **XXXXX: requirement number**  Result 1 : ... [COVERS : DSG\_MOD\_xxxxx]  Result 2 : ... [COVERS : ARCH\_xxxxx]  …  Result N : ... [COVERS : DSG\_MOD\_xxxxx or ARCH\_xxxxx]  Result 1: Check that start-up profile is triggered. Read PRE tensioning weight parameters to check them (0x22 0xFD 0x36) or check them with Trace32 into Ram mirror of Nvm block.  Result 2: Read PRE cycle parameters (0x22 0xFD 0x10) or check them withTrace32 into Ram mirror of Nvm block.  Result 4: Check the cycle is triggered and the order is not adapted (*s32FirstOrderValue =* voltage order x 4 x 1024 = 24576).  No temperature adaptation configured for a motor current step:  Result 5: Read PRE cycle parameters or check them withTrace32 into Ram mirror of Nvm block.  Result 6: Check the cycle is triggered and the order is not adapted (*s32FirstOrderValue =* current order x 2 x 1024 = 16384).  No temperature adaptation configured for a motor current step:  Result 7: Read PRE cycle parameters or check them withTrace32 into Ram mirror of Nvm block.  Result 8: Check the cycle is triggered and the order is not adapted (*s32FirstOrderValue =* PWM order x 1024 = 51200).  Invalid temperature adaptation configured for a voltage step:  Result 9: Read PRE cycle parameters or check them withTrace32 into Ram mirror of Nvm block.  Result 10: Check the cycle is triggered and the order is not adapted (*s32FirstOrderValue =* voltage order x 4 x 1024 = 24576).  [COVERS: DES\_TF\_G\_1496] | | | | | | | | | | |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |  |  | |  | |  |  | |  |  |  |
|  |  |  |  | |  | |  |  | |  |  |  |

### INT\_BFE\_06004 : No PRESAFE temperature adaptation with invalid temperature

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  | |  |  |  | |  |  | |  |  |
|  | **Type of the test: N** N, R, E or Q | | | | | | | | | | |  |
|  |  |  | |  |  |  | |  |  | |  |  |
|  | **Purpose of the test:**  The aim of this test is to check that no PRESAFE temperature adaptation is applied when temperature is invalid. | | | | | | | | | | |  |
|  |  |
|  |  |
|  |  |  |  | |  | |  |  | |  |  |  |
|  | **Environment:**  A ReMA board flashed.  Power supply.  CANalyzer simulation.  BDM is mandatory to check the applied order. | | | | | | | | | | |  |
|  |  |
|  |  |
|  |  |  |  | |  | |  |  | |  |  |  |
|  | **INITIAL STATE:**  Describe here if a special configuration is needed here (EEP parameters with some particular values…).  Add some instrumented code into SBC component to allow setting internal temperature with Trace32.  **static uint8 b8Test;**  **static uint16 u16TestTemperature;**  void **PMP\_runGetFilteredTemperature** ( … )  {  …    **if(b8Test == 1)**  **{**  **\*pu16FilteredTemperature = = u16TestTemperature;**  **}**  }  Start CAN simulation and load application. | | | | | | | | | | |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |  |  | |  | |  |  | |  |  |  |
|  | **ACTION:**  …  Action N : ...Action  Action 1: Run the software and read default PRESAFE tensioning weight parameters.  0x22 0xFD 0x36;  Action 2: Write a new mono step cycle (PRE level 1 for instance) with a motor voltage order of 6V during 3s, where the W weighted configuration bit is set.  For instance cycle PRE 1 should be: 0x04 0x11 0x07 0x0B 0x01 0x2C 0x18 0x04 0x00 0x06 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0x26 0x5F;  Action 3: Set the valid temperature of -39.98°C, where PRESAFE adaptation factor shall be 2 (set **u16TestTemperature** =0x0038 and b8Test=1 with Trace32).  Action 4: Launch PRE level 1.  Action 5: Set the valid temperature of 25°C, where PRESAFE adaptation factor shall be 1 (set **u16TestTemperature** =0x14F and b8Test=1 with Trace32).  Action 6: Launch PRE level 1.  Action 7: Set an invalid temperature below minimum, for instance at -51°C (set **u16TestTemperature** =0x0008 and b8Test=1 with Trace32).  Action 8: Launch PRE level 1.  Action 9: Set an invalid temperature above maximum, for instance at 126°C (set **u16TestTemperature** =0x0301 and b8Test=1 with Trace32).  Action 10: Launch PRE level 1. | | | | | | | | | | |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |  |  | |  | |  |  | |  |  |  |
|  | **EXPECTED RESULT:**  Result 1: Check that start-up profile is triggered. Read PRESAFE tensioning weight parameters to check them (0x22 0xFD 0x36) or check them with Trace32 into Ram mirror of Nvm block.  Result 2: Read PRE cycle parameters or check them withTrace32 into Ram mirror of Nvm block.  Result 4: Check the cycle is triggered and the order is adapted with a factor 2:  *s32FirstOrderValue =* 2 x voltage order x 4 x 1024 = 49152.  Result 6: Check the cycle is triggered and the order is adapted with a factor 1:  *s32FirstOrderValue =* 1 x voltage order x 4 x 1024 = 24576.  Result 7 and 10: Check the cycle is triggered and the order is adapted with a factor 1:  *s32FirstOrderValue =* 1 x voltage order x 4 x 1024 = 24576.  [COVERS: DES\_TF\_G\_86, DES\_TF\_G\_1484]  **It is important for traceability to know the SW Module Design and/or SW Achitecture requirement covered by this test:**  **[COVERS : DSG\_MOD\_XXXXX/ARCH\_XXXXX]**  **MOD: SW component trigram**  **ARCH: SW Architecture prefix**  **XXXXX: requirement number**  Result 1 : ... [COVERS : DSG\_MOD\_xxxxx]  Result 2 : ... [COVERS : ARCH\_xxxxx]  …  Result N : ... [COVERS : DSG\_MOD\_xxxxx or ARCH\_xxxxx] | | | | | | | | | | |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |  |  | |  | |  |  | |  |  |  |
|  |  |  |  | |  | |  |  | |  |  |  |

### INT\_BFE\_06005 : PRESAFE temperature adaptation with valid temperature outside the table

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  | |  |  |  | |  |  | |  |  |
|  | **Type of the test: N** N, R, E or Q | | | | | | | | | | |  |
|  |  |  | |  |  |  | |  |  | |  |  |
|  | **Purpose of the test:**  The aim of this test is to check the PRESAFE temperature adaptation is applied when temperature is valid but above/below the highest/lowest temperature into the table of weighted parameters. | | | | | | | | | | |  |
|  |  |
|  |  |
|  |  |  |  | |  | |  |  | |  |  |  |
|  | **Environment:**  A ReMA board flashed.  Power supply.  CANalyzer simulation.  BDM is mandatory to check the applied order. | | | | | | | | | | |  |
|  |  |
|  |  |
|  |  |  |  | |  | |  |  | |  |  |  |
|  | **INITIAL STATE:**  Describe here if a special configuration is needed here (EEP parameters with some particular values…).  Add some instrumented code into SBC component to allow setting internal temperature with Trace32.  **static uint8 b8Test;**  **static uint16 u16TestTemperature;**  void **PMP\_runGetFilteredTemperature** ( … )  {  …    **if(b8Test == 1)**  **{**  **\*pu16FilteredTemperature = = u16TestTemperature;**  **}**  }  Start CAN simulation and load application. | | | | | | | | | | |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |  |  | |  | |  |  | |  |  |  |
|  | **ACTION:**  …  Action N : ...Action  Action 1: Run the software and read default PRESAFE tensioning weight parameters.  0x22 0xFD 0x36;  Action 2: Write a new mono step cycle (PRE level 1 for instance) with a motor voltage order of 6V during 3s, where the W weighted configuration bit is set.  For instance cycle PRE 1 should be: 0x04 0x11 0x07 0x0B 0x01 0x2C 0x18 0x04 0x00 0x06 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0x26 0x5F;  Action 3: Set a valid temperature below minimum temperature into weighted NVM table, for instance at -45°C (set **u16TestTemperature** =0x0022 and b8Test=1 with Trace32).  Action 4: Launch PRE level 1.  Action 5: Set an invalid temperature above maximum temperature into weighted NVM table, for instance at 90°C (set **u16TestTemperature** =0x0267 and b8Test=1 with Trace32).  Action 6: Launch PRE level 1. | | | | | | | | | | |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |  |  | |  | |  |  | |  |  |  |
|  | **EXPECTED RESULT:**  Result 1: Check that start-up profile is triggered. Read PRESAFE tensioning weight parameters to check them (0x22 0xFD 0x36) or check them with Trace32 into Ram mirror of Nvm block.  Result 2: Read PRE cycle parameters or check them withTrace32 into Ram mirror of Nvm block.  Result 4: Check the cycle is triggered and the order is adapted with a factor 2:  *s32FirstOrderValue =* 2 x voltage order x 4 x 1024 = 49152.  Result 6: Check the cycle is triggered and the order is adapted with a factor 1.265625:  *s32FirstOrderValue =* 1.265625 x voltage order x 4 x 1024 = 31104.  [COVERS: DES\_TF\_G\_1499]  **It is important for traceability to know the SW Module Design and/or SW Achitecture requirement covered by this test:**  **[COVERS : DSG\_MOD\_XXXXX/ARCH\_XXXXX]**  **MOD: SW component trigram**  **ARCH: SW Architecture prefix**  **XXXXX: requirement number**  Result 1 : ... [COVERS : DSG\_MOD\_xxxxx]  Result 2 : ... [COVERS : ARCH\_xxxxx]  …  Result N : ... [COVERS : DSG\_MOD\_xxxxx or ARCH\_xxxxx] | | | | | | | | | | |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |  |  | |  | |  |  | |  |  |  |
|  |  |  |  | |  | |  |  | |  |  |  |

## Power degradation – NA since PN14 not yet available

### INT\_BFE\_07001 : Start-up profile is not power degradabled

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  | |  |  |  | |  |  | |  |  |
|  | **Type of the test: N** N, R, E or Q | | | | | | | | | | |  |
|  |  |  | |  |  |  | |  |  | |  |  |
|  | **Purpose of the test:**  The aim of this test is to check that power degradation has no influence on start-up profile. | | | | | | | | | | |  |
|  |  |
|  |  |
|  |  |  |  | |  | |  |  | |  |  |  |
|  | **Environment:**  A flashed board.  Power supply.  CANalyzer simulation. | | | | | | | | | | |  |
|  |  |
|  |  |
|  |  |  |  | |  | |  |  | |  |  |  |
|  | **INITIAL STATE:**  Describe here if a special configuration is needed here (EEP parameters with some particular values…).  By default, PN14 is 13V. | | | | | | | | | | |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |  |  | |  | |  |  | |  |  |  |
|  | **ACTION:**  …  Action N : ...Action  Action 1: Run the software.  Action 2 : Set PN14 strictly under 9V. Ask for a reset by diagnostic service (0x11 0x01).  Action 3: Stop CAN simulation and wait for ECU sleep.  Action 4: Start CAN simulation. | | | | | | | | | | |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |  |  | |  | |  |  | |  |  |  |
|  | **EXPECTED RESULT:**  **It is important for traceability to know the SW Module Design and/or SW Achitecture requirement covered by this test:**  **[COVERS : DSG\_MOD\_XXXXX/ARCH\_XXXXX]**  **MOD: SW component trigram**  **ARCH: SW Architecture prefix**  **XXXXX: requirement number**  Result 1 : ... [COVERS : DSG\_MOD\_xxxxx]  Result 2 : ... [COVERS : ARCH\_xxxxx]  …  Result N : ... [COVERS : DSG\_MOD\_xxxxx or ARCH\_xxxxx]  Result 1 : Check that start-up profile is triggered.  Result 2 : Check that start-up profile is normally triggered.  Result 3: ECU sleeps (check battery consumption).  Result 4 : Check that start-up profile is normally triggered. | | | | | | | | | | |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |  |  | |  | |  |  | |  |  |  |
|  |  |  |  | |  | |  |  | |  |  |  |

### INT\_BFE\_07002 : Power degradation on current controlled steps

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  | |  |  |  | |  |  | |  |  |
|  | **Type of the test: N** N, R, E or Q | | | | | | | | | | |  |
|  |  |  | |  |  |  | |  |  | |  |  |
|  | **Purpose of the test:**  The aim of this test is to check that current controlled steps are correctly power degraded. | | | | | | | | | | |  |
|  |  |
|  |  |
|  |  |  |  | |  | |  |  | |  |  |  |
|  | **Environment:**  A flashed board.  Power supply.  CANalyzer simulation.  BDM is needed. | | | | | | | | | | |  |
|  |  |
|  |  |
|  |  |  |  | |  | |  |  | |  |  |  |
|  | **INITIAL STATE:**  Describe here if a special configuration is needed here (EEP parameters with some particular values…).  Into NVM\_Const.h, modify the definition of a cycle in order to program a single step cycle, with a current order, such as:  /\* Definition of cycle 1 parameters \*/  /\* TEST POWER DEGRADATION: CYCLE WITH SINGLE STEP - CURRENT ORDER - POWER DEG ENABLED \*/  #define KU8\_CYCLE\_1\_WEEK (0x04)  #define KU8\_CYCLE\_1\_YEAR (0x11)  #define KU8\_CYCLE\_1\_OPTION (0x07)  #define KU8\_CYCLE\_1\_NEXT\_CYCLE (0x0B) /\* Standard Hard release \*/  #define KAU8\_CYCLE\_1\_STEP\_1\_ID (3)  #define KAU8\_CYCLE\_1\_STEP\_2\_ID (255)  #define KAU8\_CYCLE\_1\_STEP\_3\_ID (255)  #define KAU8\_CYCLE\_1\_STEP\_4\_ID (255)  #define KAU8\_CYCLE\_1\_STEP\_5\_ID (255)  #define KAU8\_CYCLE\_1\_STEP\_6\_ID (255)  #define KAU8\_CYCLE\_1\_STEP\_7\_ID (255)  #define KAU8\_CYCLE\_1\_STEP\_8\_ID (255)  /\* Definition of cycle 1 CRC \*/  #define KU16\_CRC\_CYCLE\_1\_PARAM 0x1D95  /\* 8A current regulated step for 260ms \*/  #define KU16\_STEP\_04\_TIME (0x0064) /\* 260ms \*/  #define KU8\_STEP\_04\_ORDER\_VAL (0x10) /\* 8A \*/  #define KU8\_STEP\_04\_ORDER\_TYP (0x01) /\* Motor current \*/  #define KU16\_STEP\_04\_OPTIONS (0x0007)  Watch s32FirstOrderValue with Trace32.  By default, PN14 is 13V. | | | | | | | | | | |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |  |  | |  | |  |  | |  |  |  |
|  | **ACTION:**  …  Action N : ...Action  Action 1: Run the software. Trig the programmed cycle by diagnostic request (0x31 0x01 0xF7 0x03 0x00;).  Action 2 : Set PN14 at 10.8V. Trig the cycle.  Action 3: Set PN14 at 10.2V. Trig the cycle.  Action 4: Set PN14 at 9.6V. Trig the cycle.  Action 5: Set PN14 at 9.0V. Trig the cycle.  Action 6: Set PN14 at 8.9V. Try to trig the cycle. | | | | | | | | | | |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |  |  | |  | |  |  | |  |  |  |
|  | **EXPECTED RESULT:**  **It is important for traceability to know the SW Module Design and/or SW Achitecture requirement covered by this test:**  **[COVERS : DSG\_MOD\_XXXXX/ARCH\_XXXXX]**  **MOD: SW component trigram**  **ARCH: SW Architecture prefix**  **XXXXX: requirement number**  Result 1 : ... [COVERS : DSG\_MOD\_xxxxx]  Result 2 : ... [COVERS : ARCH\_xxxxx]  …  Result N : ... [COVERS : DSG\_MOD\_xxxxx or ARCH\_xxxxx]  Result 1 to 5: Check the cycle is triggered and the s32FirstOrderValue is:   |  |  |  | | --- | --- | --- | | PN14 (V) | power  degradation | Expected First order | | 13 | 100% | 16384 | | 10,8 | 100% | 16384 | | 10,2 | 90% | 15488 | | 9,6 | 80% | 14592 | | 9 | 70% | 13696 | | 8,95 | 0% | 0 |   Result 6 : Check the cycle is not triggered | | | | | | | | | | |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |  |  | |  | |  |  | |  |  |  |
|  |  |  |  | |  | |  |  | |  |  |  |

### INT\_BFE\_07003 : Power degradation on voltage controlled steps

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  | |  |  |  | |  |  | |  |  |
|  | **Type of the test: N** N, R, E or Q | | | | | | | | | | |  |
|  |  |  | |  |  |  | |  |  | |  |  |
|  | **Purpose of the test:**  The aim of this test is to check that voltage controlled steps are correctly power degraded. | | | | | | | | | | |  |
|  |  |
|  |  |
|  |  |  |  | |  | |  |  | |  |  |  |
|  | **Environment:**  A flashed board.  Power supply.  CANalyzer simulation.  BDM is needed. | | | | | | | | | | |  |
|  |  |
|  |  |
|  |  |  |  | |  | |  |  | |  |  |  |
|  | **INITIAL STATE:**  Describe here if a special configuration is needed here (EEP parameters with some particular values…).  Into NVM\_Const.h, modify the definition of a cycle in order to program a single step cycle, with a voltage order, such as:  /\* Definition of cycle 2 parameters \*/  /\* TEST POWER DEGRADATION: CYCLE WITH SINGLE STEP - VOLTAGE ORDER - POWER DEG ENABLED \*/  #define KU8\_CYCLE\_2\_WEEK (0x04)  #define KU8\_CYCLE\_2\_YEAR (0x11)  #define KU8\_CYCLE\_2\_OPTION (0x07)  #define KU8\_CYCLE\_2\_NEXT\_CYCLE (0x0B) /\* Standard Hard release \*/  #define KAU8\_CYCLE\_2\_STEP\_1\_ID (2)  #define KAU8\_CYCLE\_2\_STEP\_2\_ID (255)  #define KAU8\_CYCLE\_2\_STEP\_3\_ID (255)  #define KAU8\_CYCLE\_2\_STEP\_4\_ID (255)  #define KAU8\_CYCLE\_2\_STEP\_5\_ID (255)  #define KAU8\_CYCLE\_2\_STEP\_6\_ID (255)  #define KAU8\_CYCLE\_2\_STEP\_7\_ID (255)  #define KAU8\_CYCLE\_2\_STEP\_8\_ID (255)  /\* Definition of cycle 1 CRC \*/  #define KU16\_CRC\_CYCLE\_2\_PARAM 0x2598  /\* Fixed voltage of 6V for 1000ms \*/  #define KU16\_STEP\_03\_TIME (0x0064) /\* 1000ms \*/  #define KU8\_STEP\_03\_ORDER\_VAL (0x18) /\* 6V \*/  #define KU8\_STEP\_03\_ORDER\_TYP (0x04) /\* Motor voltage \*/  #define KU16\_STEP\_03\_OPTIONS (0x0007)  Watch s32FirstOrderValue with Trace32.  By default, PN14 is 13V. | | | | | | | | | | |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |  |  | |  | |  |  | |  |  |  |
|  | **ACTION:**  …  Action N : ...Action  Action 1: Run the software. Trig the programmed cycle by diagnostic request (0x31 0x01 0xF7 0x03 0x01;).  Action 2 : Set PN14 at 10.8V. Trig the cycle.  Action 3: Set PN14 at 10.2V. Trig the cycle.  Action 4: Set PN14 at 9.6V. Trig the cycle.  Action 5: Set PN14 at 9.0V. Trig the cycle.  Action 6: Set PN14 at 8.9V. Try to trig the cycle. | | | | | | | | | | |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |  |  | |  | |  |  | |  |  |  |
|  | **EXPECTED RESULT:**  **It is important for traceability to know the SW Module Design and/or SW Achitecture requirement covered by this test:**  **[COVERS : DSG\_MOD\_XXXXX/ARCH\_XXXXX]**  **MOD: SW component trigram**  **ARCH: SW Architecture prefix**  **XXXXX: requirement number**  Result 1 : ... [COVERS : DSG\_MOD\_xxxxx]  Result 2 : ... [COVERS : ARCH\_xxxxx]  …  Result N : ... [COVERS : DSG\_MOD\_xxxxx or ARCH\_xxxxx]  Result 1 to 5: Check the cycle is triggered and the s32FirstOrderValue is:   |  |  |  | | --- | --- | --- | | PN14 (V) | power  degradation | Expected First order | | 13 | 100% | 24576 | | 10,8 | 100% | 24576 | | 10,2 | 90% | 23232 | | 9,6 | 80% | 21888 | | 9 | 70% | 20544 | | 8,95 | 0% | 0 |   Result 6 : Check the cycle is not triggered | | | | | | | | | | |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |  |  | |  | |  |  | |  |  |  |
|  |  |  |  | |  | |  |  | |  |  |  |

### INT\_BFE\_07004 : No Power degradation on not power degradabled steps

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  | |  |  |  | |  |  | |  |  |
|  | **Type of the test: N** N, R, E or Q | | | | | | | | | | |  |
|  |  |  | |  |  |  | |  |  | |  |  |
|  | **Purpose of the test:**  The aim of this test is to check that not power degradabled steps, either current or voltage controlled steps, are not power degraded. | | | | | | | | | | |  |
|  |  |
|  |  |
|  |  |  |  | |  | |  |  | |  |  |  |
|  | **Environment:**  A flashed board.  Power supply.  CANalyzer simulation.  BDM is needed. | | | | | | | | | | |  |
|  |  |
|  |  |
|  |  |  |  | |  | |  |  | |  |  |  |
|  | **INITIAL STATE:**  Describe here if a special configuration is needed here (EEP parameters with some particular values…).  Into NVM\_Const.h, modify the definition of 2 cycles in order to program single step cycles, with no power degradation option, such as:  /\* Definition of cycle 2 parameters \*/  /\* TEST POWER DEGRADATION: CYCLE WITH SINGLE STEP - CURRENT ORDER - POWER DEG DISABLED \*/  #define KU8\_CYCLE\_3\_WEEK (0x04)  #define KU8\_CYCLE\_3\_YEAR (0x11)  #define KU8\_CYCLE\_3\_OPTION (0x07)  #define KU8\_CYCLE\_3\_NEXT\_CYCLE (0x0B) /\* Standard Hard release \*/  #define KAU8\_CYCLE\_3\_STEP\_1\_ID (4)  #define KAU8\_CYCLE\_3\_STEP\_2\_ID (255)  #define KAU8\_CYCLE\_3\_STEP\_3\_ID (255)  #define KAU8\_CYCLE\_3\_STEP\_4\_ID (255)  #define KAU8\_CYCLE\_3\_STEP\_5\_ID (255)  #define KAU8\_CYCLE\_3\_STEP\_6\_ID (255)  #define KAU8\_CYCLE\_3\_STEP\_7\_ID (255)  #define KAU8\_CYCLE\_3\_STEP\_8\_ID (255)  /\* Definition of cycle 3 CRC \*/  #define KU16\_CRC\_CYCLE\_3\_PARAM 0x1D91  #define KU16\_STEP\_05\_TIME (0x0064) /\* 1000ms \*/  #define KU8\_STEP\_05\_ORDER\_VAL (0x10) /\* 8A \*/  #define KU8\_STEP\_05\_ORDER\_TYP (0x01)  #define KU16\_STEP\_05\_OPTIONS (0x0003) /\* power deg disabled \*/  /\* Definition of cycle 4 parameters \*/  /\* TEST POWER DEGRADATION: CYCLE WITH SINGLE STEP - VOLTAGE ORDER - POWER DEG DISABLED \*/  #define KU8\_CYCLE\_4\_WEEK (0x04)  #define KU8\_CYCLE\_4\_YEAR (0x11)  #define KU8\_CYCLE\_4\_OPTION (0x07)  #define KU8\_CYCLE\_4\_NEXT\_CYCLE (0x0B) /\* Standard Hard release \*/  #define KAU8\_CYCLE\_4\_STEP\_1\_ID (1)  #define KAU8\_CYCLE\_4\_STEP\_2\_ID (255)  #define KAU8\_CYCLE\_4\_STEP\_3\_ID (255)  #define KAU8\_CYCLE\_4\_STEP\_4\_ID (255)  #define KAU8\_CYCLE\_4\_STEP\_5\_ID (255)  #define KAU8\_CYCLE\_4\_STEP\_6\_ID (255)  #define KAU8\_CYCLE\_4\_STEP\_7\_ID (255)  #define KAU8\_CYCLE\_4\_STEP\_8\_ID (255)  /\* Definition of cycle 4 CRC \*/  #define KU16\_CRC\_CYCLE\_4\_PARAM 0x2594  /\* Fixed voltage of 7.5V for 20ms \*/  #define KU16\_STEP\_02\_TIME (0x0064) /\* 1000ms \*/  #define KU8\_STEP\_02\_ORDER\_VAL (0x18) /\* 6V \*/  #define KU8\_STEP\_02\_ORDER\_TYP (0x04)  #define KU16\_STEP\_02\_OPTIONS (0x0003) /\* not degraded \*/  Watch s32FirstOrderValue with Trace32.  By default, PN14 is 13V. | | | | | | | | | | |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |  |  | |  | |  |  | |  |  |  |
|  | **ACTION:**  …  Action N : ...Action  Action 1: Run the software. Trig the current controlled cycle by diagnostic request (0x31 0x01 0xF7 0x03 0x02;). Trig the voltage controlled cycle by diagnostic request (0x31 0x01 0xF7 0x03 0x03;).  Action 2 : Set PN14 at 10.8V. Trig the 2 cycles.  Action 3: Set PN14 at 10.2V. Trig the 2 cycles.  Action 4: Set PN14 at 9.6V. Trig the 2 cycles.  Action 5: Set PN14 at 9.0V. Trig the 2 cycles.  Action 6: Set PN14 at 8.9V. Trig the 2 cycles. | | | | | | | | | | |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |  |  | |  | |  |  | |  |  |  |
|  | **EXPECTED RESULT:**  **It is important for traceability to know the SW Module Design and/or SW Achitecture requirement covered by this test:**  **[COVERS : DSG\_MOD\_XXXXX/ARCH\_XXXXX]**  **MOD: SW component trigram**  **ARCH: SW Architecture prefix**  **XXXXX: requirement number**  Result 1 : ... [COVERS : DSG\_MOD\_xxxxx]  Result 2 : ... [COVERS : ARCH\_xxxxx]  …  Result N : ... [COVERS : DSG\_MOD\_xxxxx or ARCH\_xxxxx]  Result 1 to 6:  Check the current controlled cycle is always triggered and the s32FirstOrderValue is always 16384.  Check the voltage controlled cycle is always triggered and the s32FirstOrderValue is always 24576. | | | | | | | | | | |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |  |  | |  | |  |  | |  |  |  |
|  |  |  |  | |  | |  |  | |  |  |  |

### INT\_BFE\_07005 : Cycle with power degradabled and not power degradabled steps

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  | |  |  |  | |  |  | |  |  |
|  | **Type of the test: N** N, R, E or Q | | | | | | | | | | |  |
|  |  |  | |  |  |  | |  |  | |  |  |
|  | **Purpose of the test:**  The aim of this test is to check a cycle which contains different configuration of steps.  Step 1: current controlled - power degradabled  Step 2: current controlled - not power degradabled  Step 3: voltage controlled - power degradabled  Step 4: voltage controlled - not power degradabled  Step 5: PWM controlled - never power degradabled | | | | | | | | | | |  |
|  |  |
|  |  |
|  |  |  |  | |  | |  |  | |  |  |  |
|  | **Environment:**  A flashed board.  Power supply.  CANalyzer simulation.  BDM is needed. | | | | | | | | | | |  |
|  |  |
|  |  |
|  |  |  |  | |  | |  |  | |  |  |  |
|  | **INITIAL STATE:**  Describe here if a special configuration is needed here (EEP parameters with some particular values…).  Into NVM\_Const.h, modify the definition of a cycle, such as:  /\* Definition of cycle 7 parameters \*/  #define KU8\_CYCLE\_7\_WEEK (0x26)  #define KU8\_CYCLE\_7\_YEAR (0x11)  #define KU8\_CYCLE\_7\_OPTION (0x07)  #define KU8\_CYCLE\_7\_NEXT\_CYCLE (0x0B) /\* Standard Hard release \*/  #define KAU8\_CYCLE\_7\_STEP\_1\_ID (3)  #define KAU8\_CYCLE\_7\_STEP\_2\_ID (4)  #define KAU8\_CYCLE\_7\_STEP\_3\_ID (2)  #define KAU8\_CYCLE\_7\_STEP\_4\_ID (1)  #define KAU8\_CYCLE\_7\_STEP\_5\_ID (23)  #define KAU8\_CYCLE\_7\_STEP\_6\_ID (255)  #define KAU8\_CYCLE\_7\_STEP\_7\_ID (255)  #define KAU8\_CYCLE\_7\_STEP\_8\_ID (255)  /\* 8A current regulated step for 260ms \*/  #define KU16\_STEP\_04\_TIME (0x0064) /\* 260ms \*/  #define KU8\_STEP\_04\_ORDER\_VAL (0x10) /\* 8A \*/  #define KU8\_STEP\_04\_ORDER\_TYP (0x01) /\* Motor current \*/  #define KU16\_STEP\_04\_OPTIONS (0x0007)  #define KU16\_STEP\_05\_TIME (0x0064) /\* 1000ms \*/  #define KU8\_STEP\_05\_ORDER\_VAL (0x10) /\* 8A \*/  #define KU8\_STEP\_05\_ORDER\_TYP (0x01)  #define KU16\_STEP\_05\_OPTIONS (0x0003) /\* power deg disabled \*/  #define KU16\_STEP\_03\_TIME (0x0064) /\* 1000ms \*/  #define KU8\_STEP\_03\_ORDER\_VAL (0x18) /\* 6V \*/  #define KU8\_STEP\_03\_ORDER\_TYP (0x04) /\* Motor voltage \*/  #define KU16\_STEP\_03\_OPTIONS (0x0007)  #define KU16\_STEP\_02\_TIME (0x0064) /\* 1000ms \*/  #define KU8\_STEP\_02\_ORDER\_VAL (0x18) /\* 6V \*/  #define KU8\_STEP\_02\_ORDER\_TYP (0x04)  #define KU16\_STEP\_02\_OPTIONS (0x0003) /\* not degraded \*/  #define KU16\_STEP\_24\_TIME (0x0064)  #define KU8\_STEP\_24\_ORDER\_VAL (0xCE)  #define KU8\_STEP\_24\_ORDER\_TYP (0x00)  #define KU16\_STEP\_24\_OPTIONS (0x0003)  /\* Definition of cycle 7 CRC \*/  #define KU16\_CRC\_CYCLE\_7\_PARAM 0x4F4A  Watch s32FirstOrderValue with Trace32.  By default, PN14 is 13V. | | | | | | | | | | |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |  |  | |  | |  |  | |  |  |  |
|  | **ACTION:**  …  Action N : ...Action  Action 1: Run the software. Trig the cycle by diagnostic request (0x31 0x01 0xF7 0x03 0x06;).  Action 2 : Set PN14 at 10.8V. Trig the cycle.  Action 3: Set PN14 at 10.2V. Trig the cycle.  Action 4: Set PN14 at 9.6V. Trig the cycle.  Action 5: Set PN14 at 9.0V. Trig the cycle.  Action 6: Set PN14 at 8.9V. Try to trig the cycle. | | | | | | | | | | |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |  |  | |  | |  |  | |  |  |  |
|  | **EXPECTED RESULT:**  **It is important for traceability to know the SW Module Design and/or SW Achitecture requirement covered by this test:**  **[COVERS : DSG\_MOD\_XXXXX/ARCH\_XXXXX]**  **MOD: SW component trigram**  **ARCH: SW Architecture prefix**  **XXXXX: requirement number**  Result 1 : ... [COVERS : DSG\_MOD\_xxxxx]  Result 2 : ... [COVERS : ARCH\_xxxxx]  …  Result N : ... [COVERS : DSG\_MOD\_xxxxx or ARCH\_xxxxx]  Result 1 to 5:  Check the s32FirstOrderValue is:   |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | PN14 (V) | power  degradation | Expected order 1 | Expected order 2 | Expected order 3 | Expected order 4 | Expected order 5 | | 13 | 100% | 16384 | 16384 | 24576 | 24576 | -51200 | | 10,8 | 100% | 16384 | 16384 | 24576 | 24576 | -51200 | | 10,2 | 90% | 15488 | 16384 | 23232 | 24576 | -51200 | | 9,6 | 80% | 14592 | 16384 | 21888 | 24576 | -51200 | | 9 | 70% | 13696 | 16384 | 20544 | 24576 | -51200 |   Result 6: Check the cycle is not triggered. | | | | | | | | | | |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |  |  | |  | |  |  | |  |  |  |
|  |  |  |  | |  | |  |  | |  |  |  |

### INT\_BFE\_07006 : No Power degradation abortion

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  | |  |  |  | |  |  | |  |  |
|  | **Type of the test: N** N, R, E or Q | | | | | | | | | | |  |
|  |  |  | |  |  |  | |  |  | |  |  |
|  | **Purpose of the test:**  The aim of this test is to check that a cycle can not be aborted by power degradation. | | | | | | | | | | |  |
|  |  |
|  |  |
|  |  |  |  | |  | |  |  | |  |  |  |
|  | **Environment:**  A flashed board.  Power supply.  CANalyzer simulation. | | | | | | | | | | |  |
|  |  |
|  |  |
|  |  |  |  | |  | |  |  | |  |  |  |
|  | **INITIAL STATE:**  Choose a cycle which contains at least one power degradabled step.  By default, PN14 is 9.5V. | | | | | | | | | | |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |  |  | |  | |  |  | |  |  |  |
|  | **ACTION:**  …  Action N : ...Action  Action 1: Run the software. Trig the cycle by diagnostic request (0x31 0x01 0xF7 0x03 0x0n;).  Action 2 : Trig the cycle and set PN14 to 8.9V during the tensioning phase of the cycle. | | | | | | | | | | |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |  |  | |  | |  |  | |  |  |  |
|  | **EXPECTED RESULT:**  **It is important for traceability to know the SW Module Design and/or SW Achitecture requirement covered by this test:**  **[COVERS : DSG\_MOD\_XXXXX/ARCH\_XXXXX]**  **MOD: SW component trigram**  **ARCH: SW Architecture prefix**  **XXXXX: requirement number**  Result 1 : ... [COVERS : DSG\_MOD\_xxxxx]  Result 2 : ... [COVERS : ARCH\_xxxxx]  …  Result N : ... [COVERS : DSG\_MOD\_xxxxx or ARCH\_xxxxx]  Result 1: Check the cycle is triggered.  Result 2: Check the cycle is triggered and not aborted. | | | | | | | | | | |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |  |  | |  | |  |  | |  |  |  |
|  |  |  |  | |  | |  |  | |  |  |  |

### INT\_BFE\_07007 : Inhibition by power degradation

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  | |  |  |  | |  |  | |  |  |
|  | **Type of the test: N** N, R, E or Q | | | | | | | | | | |  |
|  |  |  | |  |  |  | |  |  | |  |  |
|  | **Purpose of the test:**  The aim of this test is to check that a power degradabled cycle is inhibited by power degradation, if correction factor is null. It also checks the cycle is triggered again when the correction factor becomes not null, if the presafe situation is still present. | | | | | | | | | | |  |
|  |  |
|  |  |
|  |  |  |  | |  | |  |  | |  |  |  |
|  | **Environment:**  A flashed board.  Power supply.  CANalyzer simulation. | | | | | | | | | | |  |
|  |  |
|  |  |
|  |  |  |  | |  | |  |  | |  |  |  |
|  | **INITIAL STATE:**  Choose a cycle which contains at least one power degradabled step.  By default, PN14 is 9.5V. | | | | | | | | | | |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |  |  | |  | |  |  | |  |  |  |
|  | **ACTION:**  …  Action N : ...Action  Action 1: Run the software. Trig the cycle by presafe situation (‘:’ to select level, ‘!’ to activate it). Stop the presafe situation (activate no level).  Action 2: Set PN14 to 8.9V and try to trig the cycle by setting a new presafe situation.  Action 3: Stop the presafe situation (activate no level).  Action 4 : Set a new presafe situation.  Action 5: Set PN14 at 9.0V. Then stop the presafe situation (activate no level). | | | | | | | | | | |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |  |  | |  | |  |  | |  |  |  |
|  | **EXPECTED RESULT:**  **It is important for traceability to know the SW Module Design and/or SW Achitecture requirement covered by this test:**  **[COVERS : DSG\_MOD\_XXXXX/ARCH\_XXXXX]**  **MOD: SW component trigram**  **ARCH: SW Architecture prefix**  **XXXXX: requirement number**  Result 1 : ... [COVERS : DSG\_MOD\_xxxxx]  Result 2 : ... [COVERS : ARCH\_xxxxx]  …  Result N : ... [COVERS : DSG\_MOD\_xxxxx or ARCH\_xxxxx]  Result 1: Check the cycle is triggered, with its tensioning and releasing phases.  Result 2: Check the cycle is not triggered.  Result 3: Check the releasing phase is not triggered.  Result 4: Check there is no motor activation, as for result 2.  Result 5: Check the tensioning phase is triggered as soon as the PN14 is set at 9.0V. Check the releasing phase is triggered then. | | | | | | | | | | |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |  |  | |  | |  |  | |  |  |  |
|  |  |  |  | |  | |  |  | |  |  |  |

# Annex

## Adaptation of Belt Function Selection module

The BFS algorithm shall be modified in order to drive with a debugger the Selected Cycle Identifier that is one of the main inputs for the BFE.

Snippet for BFS :

|  |
| --- |
| /\* Declaration of local data for BFE tests execution : \*/  uint8 u8SeletedCycle\_test = 0xFF;  uint8 u8OverWriteBFSOutput = 0x55;  /\* Modification of RTE services call in BFS algo \*/  if ( 0xAA == u8OverWriteBFSOutput)  {  Rte\_Write\_psrSelectedCycle\_u8CycleNumber(u8SeletedCycle\_test);  }  else  {  Rte\_Write\_psrSelectedCycle\_u8CycleNumber(u8SelectedCycleBFS);  } |

## Cycles & steps parameters for integration tests

For integration tests, it would be better to change the belt function parameters to avoid DTC qualification, or HW/Mechanical issues. It’s better to put the 3 first cycles in releasing direction.

|  |  |  |
| --- | --- | --- |
| **Byte :** | **Raw** | **Unit/Comment** |
| 1 | 0x04 | Calendar Week |
| 2 | 0x11 | Year |
| 3 | 0x07 |  |
| 4 | 0xFF | No following cycle |
| 5 | 0x00 | Step Id #0 |
| 6 | 0x01 | Step Id #1 |
| 7 | 0xFF | No Step |
| 8 | 0xFF | No Step |
| 9 | 0xFF | No Step |
| 10 | 0xFF | No Step |
| 11 | 0xFF | No Step |
| 12 | 0xFF | No Step |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Steps parameters : | | | | | | |
|  | **Byte :** | **Raw** | **Value** | **Unit/Comment** | | |
| **Step 1** | 1 | 03 |  |  |  |  |
| 2 | E8 | 10000 | ms |  |  |
| 3 | F6 | -10 | % PWM |  |  |
| 4 | 00 |  | S - Straight - Motor Power in % PWM | | |
| 5 | 00 | 0 |  |  |  |
| 6 | 00 |  |  |  |  |
| **Step 2** | 1 | 00 |  |  |  |  |
| 2 | 0A | 100 | ms |  |  |
| 3 | 00 | 0 | % PWM |  |  |
| 4 | 80 |  | A - Ramp-step - Motor Power in % PWM | | |
| 5 | 00 | 0 |  |  |  |
| 6 | 00 |  |  |  |  |

With this configuration the tests can be done with a complete retractor or a ‘free’ motor without DTCs.

This cycle is like a ‘long’ anti patina cycle.

## Power degradation look up table

Use the following excel sheet to compute the expected values for INT\_BFE\_01003:



## Current interruption point look up tables

Use the following excel sheet to compute the expected values for INT\_BFE\_04002:

