**RESTRICTED ACCESS**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **DISTRIBUTION :** | **Firm** | **To** | **Ref** | **Copies** | **1st page** | **e-mail** |
|  | FCE | Xavier Christmann, Claude Redon, Adel Bassaid | 8 |  |  |  |
|  | FCE | Hamza Zetti, Lory Médas, M. Pastor, Aki Saito, Sabrine Bouazizi, Wail Amri, Kérima Adjadi | 10 |  |  |  |
|  | RBE | M. Ianos, D.Andris | 2 |  |  |  |
|  |  |  |  |  |  |  |
|  | FCE | Secretary ship | 1 |  |  |  |

**SW Architecture Design & Interface Description :**

**PAL sw UNIT**

OBJECT: This document is the description of the design & interfaces for *PAL* SW unit.

SUMMARY: This document provides a high-level view of the *PAL* SW unit. The inputs of this document are provided by the software requirement. It is linked to the DAI\_MMA\_SWarchitectureDesignInterfaceDescription document.

CONCLUSION: Applicable from R01.0 SW release

**THIS DOCUMENT CONTAINS HIDDEN TEXT**

EVOLUTION OF THE DOCUMENT

|  |  |  |  |
| --- | --- | --- | --- |
| **Issue** | **Date** | **Author** | **Motive and nature of the modifications** |
| A00 | 31/08/2016 | C. Redon | First release (extract from the full PP4G architecture document) |
| A01 | 23/09/2016 | J. Lacroix | Update of requirement reference |
| A02 | 26/09/2016 | J. Lacroix | Removed of obsolete traceability |
| A03 | 28/10/2016 | P-O Pilot | Rename all auto-tests. Add mainFunction runnable definition. Add missing called function in some runnable |
| A04 | 18/11/2016 | C. Redon | Correction of traceability issues |
| Start extended description based on mainstream document | | | |
| 1.1.1.2 | 12/07/2019 | A. Vaché | Update traceability to match PP4G extended platform requirements IDs |
| 1.1.1.3 | 19/07/2019 | A. Vaché | Remove traceability to no more existing requirements |
| 1.1.1.4 | 08/08/2019 | A. Vaché | Solve some traceability issues highlighted by reqtify |
| 1.1.1.5 | 19/11/2019 | W. AMRI | Add missing traceability, Issue correction: 801071. |
| 1.1.1.6 | 26/11/2019 | W. AMRI | Update template information |
| Start DAI MMA description based on extended document | | | |
| 1.2.2.1 | 06/01/2022 | A. Negrea | First revision |
| 1.2.2.2 | 06/01/2022 | A. Negrea | Duplicate revision |
| 1.2.2.3 | 21/02/2022 | A. Negrea | Add fix after review |
| 1.2.2.4 | 17/03/2022 | A. Negrea | Update according to SRM |
| 1.2.2.5 | 23/03/2022 | A. Negrea | Update according to SRM2 |
| 1.2.2.6 | 30/03/2022 | A. Negrea | Update according to SRM3 |
| 1.2.2.7 | 18/08/2022 | M. Obada | Update for R3.0 release |
| 1.2.2.8 | 19/08/2022 | M. Obada | Update traceability after SRM |
| 1.2.2.9 | 23/08/2022 | M. Obada | Fix findings from review |
| 1.2.2.10 | 23/08/2022 | M. Obada | Update runnables and number of pages |
| 1.2.2.11 | 23/08/2022 | M. Obada | Fix findings from review |
| 1.2.2.12 | 23/08/2022 | M. Obada | Ad called functions to runnables |
| 1.2.2.13 | 09/11/2022 | M. Obada | Update for R4.0 release |
| 1.2.2.14 | 10/11/2022 | M. Obada | Update after SRM |
| 1.2.2.15 | 19.12.2022 | M. Obada | Update for R5.0 |
| 1.2.2.16 | 15/02/2023 | A. Paval | Updates for R5.0 |
| 1.2.2.17 | 03.05.2023 | M. Obada | Update for R6.0 |
| 1.2.2.18 | 09.05.2023 | M. Obada | Update after review |

This document contains **40** pages.

**CONTENTS**

1. Documentation 7

1.1. Upper Level Relevant Documents 7

1.2. Design interface description Documents 8

1.3. Design Specification Documents 10

1.4. Freescale Documents 10

1.5. Tier2 Documents 11

1.6. HW Datasheet 11

1.7. Other Documents 11

1.8. Glossary And Definition 12

2. Description 13

3. Technical functions - Overview 13

4. Technical functions - Actuator 17

4.1. Motor power order scaling 17

4.2. To control the motor WITHOUT booster 18

4.2.1. To control the motor in tensioning direction 19

4.2.2. To control the motor in releasing direction 20

5. Technical functions - Sensor 21

5.1. To provide the motor current 21

6. Technical functions – Auto-tests 24

6.1. To check the HW self-protection state 24

6.2. Runnables - Main 26

6.3. PAL\_Init 26

6.3.1. Definition 26

6.3.1. Data flow / Parameters 26

6.3.2. Called functions 26

6.4. PAL\_runMainFunction 26

6.4.1. Definition 26

6.4.1. Data flow / Parameters 26

6.4.2. Called functions 27

7. Runnables - Actuator 27

7.1. PAL\_runSetPowerOrder 27

7.1.1. Definition 27

7.1.1. Data flow / Parameters 27

7.1.2. Called functions 28

7.2. PAL\_DisablePowerStage 28

7.2.1. Definition 28

7.2.2. Called functions 28

7.3. PAL\_ StartFreeWheelingState 29

7.3.1. Definition 29

7.3.2. Called functions 29

7.4. PAL\_Cfg\_Init 29

7.4.1. Definition 29

7.4.2. Called functions 30

7.5. PAL\_AT\_Init 30

7.5.1. Definition 30

8. Runnables - Sensor 30

8.1. PAL\_runReadMotorCurrentInmA 30

8.1.1. Definition 30

8.1.2. Called functions 31

8.2. PAL\_runReadMotorCurrentInA 31

8.2.1. Definition 31

8.2.2. Called functions 31

9. Runnables – To check power-bridge 32

9.1. PAL\_Autotest\_CheckHighSideSwRegulation 32

9.1.1. Definition 32

9.2. PAL\_Autotest\_CheckHWSelfProtection 32

9.2.1. Definition 32

9.2.2. Called functions 32

9.3. PAL\_Autotest\_CheckCommandConsistency 33

9.3.1. Definition 33

9.4. PAL\_Autotest\_CheckMosfetLowSC 33

9.4.1. Definition 33

9.4.1. Data flow 33

9.4.2. Called functions 33

9.5. PAL\_Autotest\_CheckMosfetHighSC 34

9.5.1. Definition 34

9.5.1. Data flow 34

9.5.2. Called functions 34

9.6. PAL\_Autotest\_CheckMosfetOCAT 34

9.6.1. Definition 34

9.6.1. Data flow 35

9.6.2. Called functions 35

10. Runnables – To check motor 35

10.1. PAL\_Autotest\_CheckMotorConnection 35

10.1.1. Definition 35

10.1.2. Called functions 36

10.2. PAL\_Autotest\_CheckMotorDisengagement 36

10.2.1. Definition 36

10.2.2. Data flow 36

10.2.3. Called functions 36

10.3. PAL\_Autotest\_CheckMotorCurrent 37

10.3.1. Definition 37

10.3.2. Data flow 37

10.3.3. Called functions 37

10.4. PAL\_Autotest\_CheckMotorThermalProctection 37

10.4.1. Definition 37

10.4.2. Called functions 38

10.1. PAL\_Autotest\_CheckMotorPowerOrder 38

10.1.1. Definition 38

10.1.2. Data flow 38

10.1.3. Called functions 38

10.2. PAL\_Autotest\_CheckMotorSC 39

10.2.1. Definition 39

10.2.2. Data flow 39

10.2.3. Called functions 39

11. MCU resources 40

**TABLE OF FIGURES**

[**Figure 1: PAL - Static description** 16](#_Toc127364556)

[**Figure 2: Pal: Motor power order scaling** 17](#_Toc127364557)

[**Figure 3: Pal - To control the Motor power (without booster)** 18](#_Toc127364558)

[**Figure 4: Pal - To drive the motor in TENSIONING direction without booster** 19](#_Toc127364559)

[**Figure 5: Pal - To drive the motor in RELEASING direction without booster** 20](#_Toc127364560)

[**Figure 9: Pal - Motor current acquisition chain** 21](#_Toc127364561)

[**Figure 10: Pal - Motor current selection** 22](#_Toc127364562)

[**Figure 11: Pal - Motor current computation** 23](#_Toc127364563)

[**Figure 12: Current / HW self-protection area** 24](#_Toc127364564)

[**Figure 13: Pal- to monitor the HW self-protection** 25](#_Toc127364565)

# Documentation

## Upper Level Relevant Documents

This section presents all the documents needed to write the software architecture design document.

|  |  |  |  |
| --- | --- | --- | --- |
| Nb | **Document** | **Reference** | **Company** |
|  | TF-A: To Manage the power supply | /RevAS/30\_DES\_Requirements/Technical Functions/  DES\_TF\_A\_To\_Manage\_The\_Power\_Supply | RBE/FCE |
|  | TF-B: To Manage the communication | /RevAS/30\_DES\_Requirements/Technical Functions/  DES\_TF\_B\_To\_Manage\_The\_Communication | RBE/FCE |
|  | TF-C: To Secure PP ECU functioning using Pictus MCU | /RevAS/30\_DES\_Requirements/Technical Functions/  DES\_TF\_C\_To\_Secure\_PP\_ECU\_Functioning\_Pictus | RBE/FCE |
|  | TF-D: To Program MCU | /RevAS/30\_DES\_Requirements/Technical Functions/DES\_TF\_D\_To\_Program\_MCU | RBE/FCE |
|  | TF-E: To Manage Diagnostic Requests | /RevAS/30\_DES\_Requirements/Technical Functions/DES\_TF\_E\_To\_Manage\_Diagnostic\_Requests | RBE/FCE |
|  | TF-F: To Perform Measurements | /RevAS/30\_DES\_Requirements/Technical Functions/DES\_TF\_F\_To\_Perform\_Measurements | RBE/FCE |
|  | TF-G: To Drive the Motor | /RevAS/30\_DES\_Requirements/Technical Functions/DES\_TF\_G\_To\_Drive\_the\_Motor | RBE/FCE |
|  | TF-H: To Perform Autotests | /RevAS/30\_DES\_Requirements/Technical Functions/DES\_TF\_H\_To\_Perform\_Autotests | RBE/FCE |
|  | TF-I: To Manage the Failure | /RevAS/30\_DES\_Requirements/Technical Functions/DES\_TF\_I\_To\_Manage\_The\_Failure | RBE/FCE |
|  | TF-J: To Manage NVM - NVP (Non Volatile Parameters) | /RevAS/30\_DES\_Requirements/Technical Functions/DES\_TF\_J\_To\_Manage\_NVM | RBE/FCE |
|  | TF-K: To Ensure ECU Protection and Integration | /RevAS/30\_DES\_Requirements/Technical Functions/DES\_TF\_K\_To\_Ensure\_ECU\_Protection\_And\_Integration | RBE/FCE |
|  | TF-L: To Ensure ECU Integration in Environment EMC ESD | /RevAS/30\_DES\_Requirements/Technical Functions/DES\_TF\_L\_To\_Ensure\_ECU\_Integration\_In\_Environment\_EMC\_ESD | RBE/FCE |
|  | TF-M: To generate time base | /RevAS/30\_DES\_Requirements/Technical Functions/DES\_TF\_M\_To\_Generate\_Time\_Base | RBE/FCE |
|  | TF-N: To evaluate belt data | /RevAS/30\_DES\_Requirements/Technical Functions/DES\_TF\_N\_To\_Evaluate\_Belt\_Data | RBE/FCE |
|  | TF-O: To schedule the SW | /RevAS/30\_DES\_Requirements/Technical Functions/DES\_TF\_O\_To\_Run\_SW | RBE/FCE |
|  | TF-P: To handle network management | /RevAS/30\_DES\_Requirements/Technical Functions/DES\_TF\_P\_To Handle\_Network\_Management | RBE/FCE |
|  | TF-Q: To Provide Data For Expertise | /RevAS/30\_DES\_Requirements/Technical Functions/DES\_TF\_Q\_To\_Provide\_Data\_For\_Expertise | RBE/FCE |
|  | TF-R: To Decide Belt Function Execution | /RevAS/30\_DES\_Requirements/Technical Functions/DES\_TF\_R\_To\_Decide\_Belt\_Function\_Execution | RBE/FCE |
|  | TF-S: To drive the boost | /RevAS/30\_DES\_Requirements/Technical Functions/DES\_TF\_S\_To\_Drive\_Boost | RBE/FCE |
|  | TF-X: To generate time base | /RevAS/30\_DES\_Requirements/Technical Functions/DES\_TF\_M\_To\_Generate\_Time\_Base | RBE/FCE |

## Design interface description Documents

This section presents all the documents that are linked to this software architecture design document.

Note: All links are related to S:\drive, to have them functional, please mount the S:\drive on your Audi Tr6 extended platform sandbox.

|  |  |  |  |
| --- | --- | --- | --- |
| **Nb.** | **Document** | **Reference** | **Company** |
|  | EEPROM parameters | [SBE\_4G\_NVP\_layout.xls](file:///C:\Users\alexandru.paval\Documents\Sandboxes\Daimler_MMA_ReVAS\Phase_02\View_Development\Components\Application\Autoliv\NVP\Config\SBE_4G_NVP_layout.xls) | RBE/FCE |
|  | Design Interface description of AdcIf | N/A | RBE/FCE |
|  | Design Interface Description of Auto Tests Manager | [ATM - Design Interface Description.docx](ATM%20-%20Design%20Interface%20Description.docx) | RBE/FCE |
|  | Design Interface Description of Belt Function Decision | N/A | RBE/FCE |
|  | Design Interface Description of Belt Function Execution | [BFE - Design Interface Description.docx](file:///S:\Architectures\Application\Description\Associated_Documents\BFE%20-%20Design%20Interface%20Description%20.docx) | RBE/FCE |
|  | Design Interface Description of Belt Function Selection | [BFS - Design Interface Description.docx](file:///S:\Architectures\Application\Description\Associated_Documents\BFS%20-%20Design%20Interface%20Description%20.docx) | RBE/FCE |
|  | Design Interface Description of Belt Movement Monitoring | [BMM - Design Interface Description.docx](file:///C:\Users\alexandru.paval\Documents\Sandboxes\Daimler_MMA_ReVAS\Phase_02\View_Development\Architectures\Application\Description\Associated_Documents\BMM%20-%20Design%20Interface%20Description.docx) | RBE/FCE |
|  | Design Interface Description of Belt Parking Algorithm | N/A | RBE/FCE |
|  | Design Interface Description of Belt Slack Reduction | [BSR - Design Interface Description.docx](file:///S:\Architectures\Application\Description\Associated_Documents\BFS%20-%20Design%20Interface%20Description%20.docx) | RBE/FCE |
|  | Design Interface Description of Basic Software Manager | N/A | RBE/FCE |
|  | Design Interface Description of Basic Software Manager Interface | N/A | RBE/FCE |
|  | Design Interface Description of Can Tranceiver Interface | N/A | RBE/FCE |
|  | Design Interface Description of Communication Interaction Layer | [CIL - Design Interface Description.docx](file:///S:\Architectures\Application\Description\Associated_Documents\CIL%20-%20Design%20Interface%20Description.docx) | RBE/FCE |
|  | Design Interface Description of Diagnostic Communication Manager Interface | N/A | RBE/FCE |
|  | Design Interface Description of Diagnostic Event Manager Interface | N/A | RBE/FCE |
|  | Design Interface Description of DiagOnCAN services management | [DIA - Design Interface Description.docx](file:///S:\Architectures\Application\Description\Associated_Documents\DIA%20-%20Design%20Interface%20Description.docx) | RBE/FCE |
|  | Design Interface Description of Electronic Control Unit Manager | N/A | RBE/FCE |
|  | Design Interface Description of Electronic Control Unit Manager Interface | N/A | RBE/FCE |
|  | Design Interface Description of End of life | [EOL - Design Interface Description.docx](file:///S:\Architectures\Application\Description\Associated_Documents\DIA%20-%20Design%20Interface%20Description.docx) | RBE/FCE |
|  | Design Interface Description of Error Handler | [ERH - Design Interface Description.docx](ERH%20-%20Design%20Interface%20Description.docx) | RBE/FCE |
|  | Design Interface Description of Haptic Warning | [HWA - Design Interface Description.docx](file:///S:\Architectures\Application\Description\Associated_Documents\DIA%20-%20Design%20Interface%20Description.docx) | RBE/FCE |
|  | Design Interface Description of Memory Integrity Control | N/A | RBE/FCE |
|  | Design Interface Description of Mode Management | [MMG - Design Interface Description.docx](file:///S:\Architectures\Application\Description\Associated_Documents\MMG%20-%20Design%20Interface%20Description.docx) | RBE/FCE |
|  | Design Interface Description of Network Management Interface | N/A | RBE/FCE |
|  | Design Interface Description of Non-Volatile Memory Interface | N/A | RBE/FCE |
|  | Design Interface Description of Non-Volatile Parameters | [NVP - Design Interface Description.docx](file:///S:\Architectures\Application\Description\Associated_Documents\NVP%20-%20Design%20Interface%20Description.docx) | RBE/FCE |
|  | Design Interface Description of Operating System Interface | N/A | RBE/FCE |
|  | Design Interface Description of Power Abstraction Layer | [PAL - Design Interface Description.docx](file:///S:\Architectures\Application\Description\Associated_Documents\PAL%20-%20Design%20Interface%20Description.docx) | RBE/FCE |
|  | Design Interface Description of Pre-Crash Master | N/A | RBE/FCE |
|  | Design Interface Description of Physical Measures Provider | [PMP - Design Interface Description.docx](file:///S:\Architectures\Application\Description\Associated_Documents\PMP%20-%20Design%20Interface%20Description.docx) | RBE/FCE |
|  | Design Interface Description of Port Interface | N/A | RBE/FCE |
|  | Design Interface Description of Pre-Tensioning | [PRE - Design Interface Description.docx](file:///S:\Architectures\Application\Description\Associated_Documents\PMP%20-%20Design%20Interface%20Description.docx) | RBE/FCE |
|  | Design Interface Description of Production cycle function | N/A | RBE/FCE |
|  | Design Interface Description of Pulse Width Modulation Interface | N/A | RBE/FCE |
|  | Design Interface Description of Reset Cause Management | N/A | RBE/FCE |
|  | Design Interface Description of SBC | N/A | RBE/FCE |
|  | Design Interface Description of System Context Management | N/A | RBE/FCE |
|  | Design Interface Description of Standard Function Recovery (releasing function) | [SFR - Design Interface Description.docx](file:///S:\Architectures\Application\Description\Associated_Documents\SFR%20-%20Design%20Interface%20Description.docx) | RBE/FCE |
|  | Design Interface Description of Serial Peripheral Interface Interface | N/A | RBE/FCE |
|  | Design Interface Description of Startup | N/A | RBE/FCE |
|  | Design Interface Description of System Time Management | N/A | RBE/FCE |
|  | Design Interface Description of Vehicle Dynamics algorithm | N/A | RBE/FCE |

## Design Specification Documents

This section presents all the documents that complete this software architecture design document.

Note: All links are related to S:\drive, to have them functional, please mount the S:\drive on your sandbox.

|  |  |  |  |
| --- | --- | --- | --- |
| **Nb** | **Document** | **Reference** | **Company** |
|  | Design document of AdcIf | N/A | RBE/FCE |
|  | Design document of Auto Tests Manager | [ATM - Detailed Design Document.docx](../../../../Components/Application/Autoliv/ATM/Design/ATM%20-%20Detailed%20Design%20Document.docx) | RBE/FCE |
|  | Design document of Belt Function Decision | N/A | RBE/FCE |
|  | Design document of Belt Function Execution | [BFE - Detailed Design Document.docx](../../../../Components/Application/Autoliv/BFE/Design/BFE%20-%20Detailed%20Design%20Document.docx) | RBE/FCE |
|  | Design document of Belt Function Selection | [BFS - Detailed Design Document.docx](../../../../Components/Application/Autoliv/BFS/Design/BFS%20-%20Detailed%20Design%20Document.docx) | RBE/FCE |
|  | Design document of Belt Movement Monitoring | [BMM - Detailed Design Document.docx](file:///C:\Users\alexandru.paval\Documents\Sandboxes\Daimler_MMA_ReVAS\Phase_02\View_Development\Components\Application\Autoliv\BMM\Design\BMM%20-%20Detailed%20Design%20Document.docx) | RBE/FCE |
|  | Design document of Belt Parking Algorithm | N/A | RBE/FCE |
|  | Design document of Basic Software Manager Interface | N/A | RBE/FCE |
|  | Design document of Belt Slack Reduction | [BSR - Detailed Design Document.docx](../../../../Components/Application/Autoliv/BSR/Design/BSR%20-%20Detailed%20Design%20Document.docx) |  |
|  | Design document of Communication Interaction Layer | [CIL - Detailed Design Document.docx](../../../../Components/Application/Autoliv/CIL/Design/CIL%20-%20Detailed%20Design%20Document.docx) | RBE/FCE |
|  | Design document of Diagnostic Communication Manager Interface | N/A | RBE/FCE |
|  | Design document of Diagnostic Event Manager Interface | N/A | RBE/FCE |
|  | Design document of DiagOnCAN services management | [DIA - Detailed Design Document.docx](../../../../Components/Application/Autoliv/DIA/Design/DIA%20-%20Detailed%20Design%20Document.docx) | RBE/FCE |
|  | Design document of End of life | [EOL - Detailed Design Document.docx](../../../../Components/Application/Autoliv/EOL/Design/EOL%20-%20Detailed%20Design%20Document.docx) | RBE/FCE |
|  | Design document of Error Handler | [ERH - Detailed Design Document.docx](../../../../Components/Application/Autoliv/ERH/Design/ERH%20-%20Detailed%20Design%20Document.docx) | RBE/FCE |
|  | Design document of Haptic Warning | [HWA - Detailed Design Document.docx](../../../../Components/Application/Autoliv/HWA/Design/HWA%20-%20Detailed%20Design%20Document.docx) | RBE/FCE |
|  | Design document of Memory Integrity Control | N/A | RBE/FCE |
|  | Design document of Mode Management | [MMG - Detailed Design Document.docx](../../../../Components/Application/Autoliv/MMG/Design/MMG%20-%20Detailed%20Design%20Document.docx) | RBE/FCE |
|  | Design document of Network Management Interface | N/A | RBE/FCE |
|  | Design document of Non-Volatile Memory Interface | N/A | RBE/FCE |
|  | Design document of Non-Volatile Parameters | N/A | RBE/FCE |
|  | Design document of Power Abstraction Layer | [PAL - Detailed Design Document.docx](../../../../Components/Application/Autoliv/PAL/Design/PAL%20-%20Detailed%20Design%20Document.docx) | RBE/FCE |
|  | Design document of Physical Measures Provider | [PMP - Detailed Design Document.docx](../../../../Components/Application/Autoliv/PMP/Design/PMP%20-%20Detailed%20Design%20Document.docx) | RBE/FCE |
|  | Design document of Port Interface | N/A | RBE/FCE |
|  | Design document of Production cycle function | N/A | RBE/FCE |
|  | Design document of Reset Cause Management | N/A | RBE/FCE |
|  | Design document of RTE If | N/A | RBE/FCE |
|  | Design document of System Context Management | N/A | RBE/FCE |
|  | Design document of Standard Function Recovery (releasing function) | [SFR - Detailed Design Document.docx](../../../../Components/Application/Autoliv/SFR/Design/SFR%20-%20Detailed%20Design%20Document.docx) | RBE/FCE |
|  | Design document of Serial Peripheral Interface Interface | N/A | RBE/FCE |

## Freescale Documents

This section presents all the documents that complete this software architecture design document.

|  |  |  |  |
| --- | --- | --- | --- |
| Nb | **Document** | **Reference** | **Company** |
|  | S32K1xx Series Reference Manual | S32K14x\_RM\_Rev2.pdf | NXP |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

## Tier2 Documents

This section presents all the documents that complete this software architecture design document.

|  |  |  |  |
| --- | --- | --- | --- |
| Nb | **Document** | **Reference** | **Company** |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

## HW Datasheet

This section presents all the documents related to the HW components that complete this software architecture design document.

|  |  |  |  |
| --- | --- | --- | --- |
| Nb | **Document** | **Reference** | **Company** |
|  | BTN8984TA datasheet | BTN8984TA\_TDS\_051 | Infineon |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

## Other Documents

This section presents all the documents that also have been needed to write this software architecture design document.

|  |  |  |  |
| --- | --- | --- | --- |
| Nb | **Document** | **Reference** | **Company** |
|  | Unified Modelling Language | 2.1.1 | OMG |
|  | MCU RFQ | [E2581849](https://plm.autoliv.int/linkto/latest/ProductDescription/E2581849/*) | FCE |
|  |  |  |  |

## Glossary And Definition

This section presents all the definitions and/or abbreviations used in this document.

*List of terms in alphabetical order:*

|  |  |
| --- | --- |
| ***Term*** | ***Meaning*** |
| ADC | Analog Digital Converter |
| AEC | Autoliv Error Code |
| API | Application Programming Interface |
| ASDM | Active Safety Domain Master |
| ASIC | Application Specific Integrated Circuit |
| ASY | Active SafetY |
| BSW | Basic SW modules |
| CAN | Controller Area Network |
| C/S | Chip Select |
| COP | Computer Operating Properly |
| eCPL | Electronic Crash Pole Locking |
| DART | Ditch - Airborne - Rough Terrain |
| DFLASH | Data FLASH |
| ECC | Error Code Correction |
| ECU | Electronic Control Unit |
| EOL | End Of Life |
| EEPROM | Electric Erasable and Programmable Read only Memory |
| HFPP | High Force Pre-Pre-Tensioning belt function |
| HF-PRE | High Force PRE pre-tensioning |
| HR | Hard Releasing |
| I/O | Input/Output |
| IMU | Inartial Measurements Unit |
| ISS | Integrated Safing System |
| LFPP | Low Force Pre-Pre-Tensioning belt function |
| MSA | Motor Start/Stop Automatic |
| MCAL | Micro-Controller Abstraction Layer |
| MCU | Micro-controller Unit |
| NMG | Mode ManaGement |
| NVM | Non Volatile Memory |
| OS | Operating System |
| PCM | Pre-Crash Master |
| PFLASH | Program FLASH |
| PIT | Periodic Interrupt Timer |
| PLL | Phase-locked loop |
| RAM | Random Access Memory |
| RCWM | Rear Collision Warning and Mitigation |
| RML | Left PP ECU |
| RMR | Right PP ECU |
| RMx | Both PP ECU |
| ROM | Read Only Memory |
| RSU | Remote Sensor Unit |
| RTE | Real Time Environment |
| RTOS | Real Time Operating System |
| SFR | Standard Function Recovery |
| SODL | Side Obstacle Detection Left |
| SPI | Serial Peripheral Interface |
| SRS | Supplementary Restraint System |
| TBC | To be confirmed |
| TBD | To be defined |
| TF | Technical Function |
| TFLASH | Test FLASH of the Pictus MCU (“one time programmable” memory) |
| W/D | Watchdog |

# Description

The purpose of PAL is to manage the full power bridge that is to say H-bridge + booster (if equipped). It is composed of:

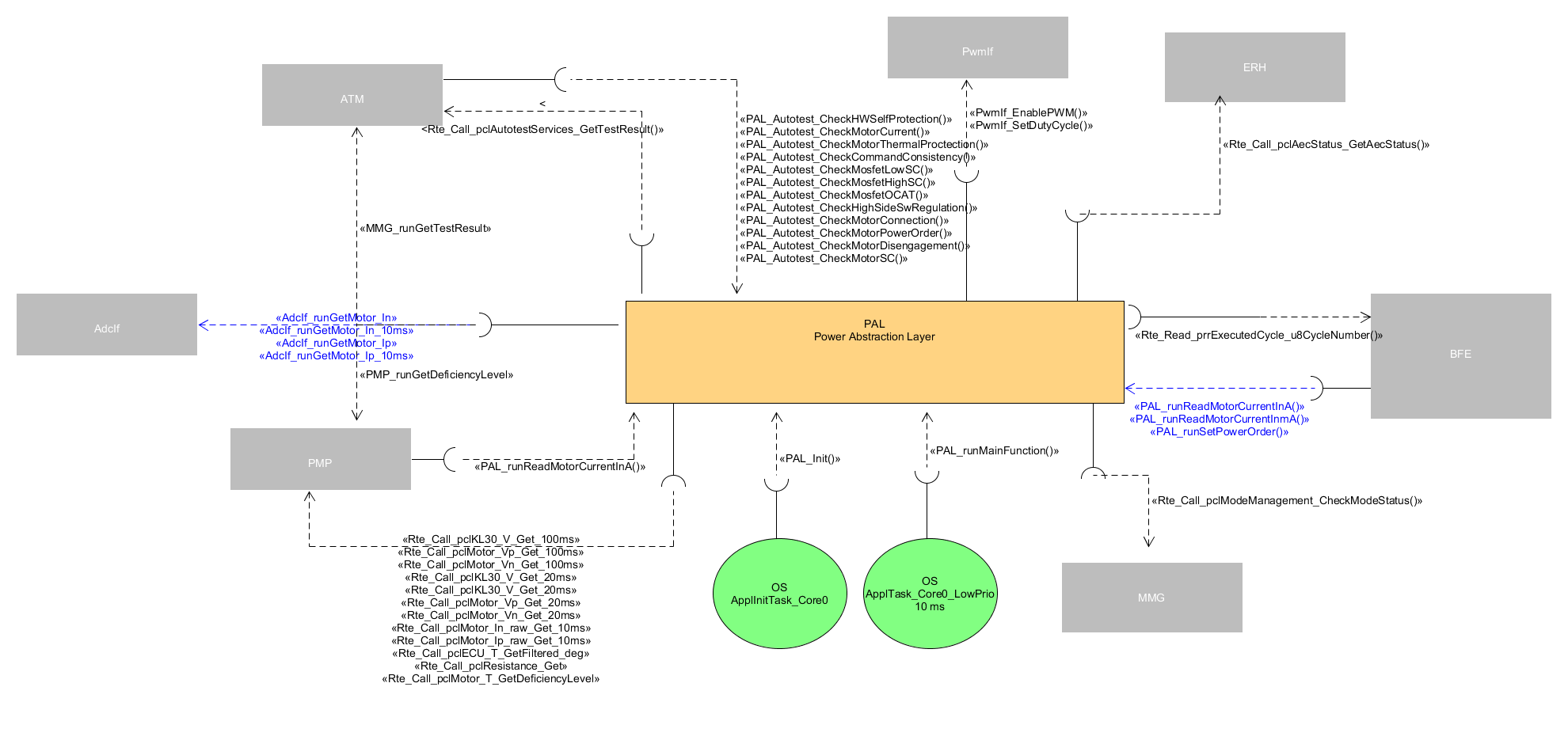
* The actuator part – To control the motor (based on the power order computed by BFE)
* The sensor part – To measure the motor current
* The auto-tests – To diagnos the full power bridge (current & activation signals monitoring, HW and SW thermal protection…)

PAL is the **core** SW unit of this ECU, part of the so called “ECU abstraction” layer.

# Technical functions - Overview

|  |  |  |  |
| --- | --- | --- | --- |
| **Requirements** | **Criteria** | **Linked Runnable** | **Source** |
| **ARCH\_SW\_PAL\_0010** | The componenet shall be initialized | PAL\_Init() | ALV\_EXT\_TF\_G\_833; ALV\_EXT\_TF\_G\_835 |
| **ARCH\_SW\_PAL\_0450** | The component shall have a main function | PAL\_runMainFunction() |  |
| **ARCH\_SW\_PAL\_0020** | The component shall control the power bridge based on the requested power order | PAL\_runSetPowerOrder() | ALV\_EXT\_TF\_G\_782; ALV\_EXT\_TF\_G\_783; ALV\_EXT\_TF\_G\_788; ALV\_EXT\_TF\_G\_789; ALV\_EXT\_TF\_G\_798; ALV\_EXT\_TF\_G\_808; ALV\_EXT\_TF\_G\_809; ALV\_EXT\_TF\_G\_814; ALV\_EXT\_TF\_G\_816; ALV\_EXT\_TF\_G\_817; ALV\_EXT\_TF\_G\_819; ALV\_EXT\_TF\_G\_820; ALV\_EXT\_TF\_G\_821; ALV\_EXT\_TF\_G\_822; ALV\_EXT\_TF\_G\_826; ALV\_EXT\_TF\_G\_877; ALV\_EXT\_TF\_G\_878; ALV\_EXT\_TF\_G\_843; ALV\_EXT\_TF\_G\_847; ALV\_EXT\_TF\_G\_494; ALV\_EXT\_TF\_G\_495; ALV\_EXT\_TF\_G\_496;  ALV\_EXT\_TF\_G\_498;ALV\_EXT\_TF\_G\_502; ALV\_EXT\_TF\_G\_503; ALV\_EXT\_TF\_G\_504 |
| **ARCH\_SW\_PAL\_0025** | The component shall be able to activate/deactivate the half bridge connected to the Vp pin of the motor | PAL\_runSetPowerOrder() | ALV\_EXT\_TF\_G\_777 |
| **ARCH\_SW\_PAL\_0026** | The component shall be able to activate/deactivate the half bridge connected to the Vn pin of the motor | PAL\_runSetPowerOrder() | ALV\_EXT\_TF\_G\_779 |
| **ARCH\_SW\_PAL\_0027** | The component shall be able to control the MOSFETs of the half bridge connected to the Vp pin of the motor | PAL\_runSetPowerOrder() | ALV\_EXT\_TF\_G\_801 |
| **ARCH\_SW\_PAL\_0028** | The component shall be able to control the MOSFETs of the half bridge connected to the Vn pin of the motor | PAL\_runSetPowerOrder() | ALV\_EXT\_TF\_G\_803 |
| **ARCH\_SW\_PAL\_0029** | The component shall be able to control MOSFET of the high power | PAL\_runSetPowerOrder() | ALV\_EXT\_TF\_G\_805; ALV\_EXT\_TF\_G\_829; ALV\_EXT\_TF\_G\_831 |
| **ARCH\_SW\_PAL\_0070** | The component shall be able to set the power state to IDLE | PAL\_DisablePowerStage() |  |
| ARCH\_SW\_PAL\_0075 | The component shall be able to deactivate the half bridge connected to the Vp pin of the motor | PAL\_DisablePowerStage() |  |
| ARCH\_SW\_PAL\_0076 | The component shall be able to deactivate the half bridge connected to the Vn pin of the motor | PAL\_DisablePowerStage() |  |
| **ARCH\_SW\_PAL\_0090** | The component shall be able to init the hardware | PAL\_Cfg\_Init() |  |
| **ARCH\_SW\_PAL\_0095** | The component shall be able to disable the H-bridge at startup | PAL\_Cfg\_Init() |  |
| **ARCH\_SW\_PAL\_0110** | The component shall initialize the auto tests data | PAL\_AT\_Init() |  |
| **ARCH\_SW\_PAL\_0080** | The component shall provide the current in miliamps | PAL\_runReadMotorCurrentInmA() | ALV\_EXT\_TF\_F\_83; ALV\_EXT\_TF\_F\_84; ALV\_EXT\_TF\_F\_269; ALV\_EXT\_TF\_F\_270; ALV\_EXT\_TF\_F\_274; ALV\_EXT\_TF\_F\_275; ALV\_EXT\_TF\_F\_276; ALV\_EXT\_TF\_F\_284; ALV\_EXT\_TF\_F\_285; ALV\_EXT\_TF\_F\_289; ALV\_EXT\_TF\_F\_290; ALV\_EXT\_TF\_F\_294; ALV\_EXT\_TF\_F\_295; ALV\_EXT\_TF\_F\_296; ALV\_EXT\_TF\_F\_304; ALV\_EXT\_TF\_F\_305; ALV\_EXT\_TF\_F\_309; ALV\_EXT\_TF\_F\_310; ALV\_EXT\_TF\_F\_314; ALV\_EXT\_TF\_F\_315; ALV\_EXT\_TF\_F\_316; ALV\_EXT\_TF\_F\_320; ALV\_EXT\_TF\_F\_323; ALV\_EXT\_TF\_F\_327; ALV\_EXT\_TF\_F\_328; ALV\_EXT\_TF\_F\_329; ALV\_EXT\_TF\_F\_330; ALV\_EXT\_TF\_F\_331; ALV\_EXT\_TF\_G\_476; ALV\_EXT\_TF\_G\_477; ALV\_EXT\_TF\_G\_478 |
| **ARCH\_SW\_PAL\_0085** | The component shall be able to read the current measured at the Vp motor pin | PAL\_runReadMotorCurrentInmA() | ALV\_EXT\_TF\_F\_280 |
| **ARCH\_SW\_PAL\_0086** | The component shall be able to read the current measured at the Vn motor pin | PAL\_runReadMotorCurrentInmA() | ALV\_EXT\_TF\_F\_300 |
| **ARCH\_SW\_PAL\_0100** | The component shall provide the current in amps | PAL\_runReadMotorCurrentInA() | ALV\_EXT\_TF\_F\_83; ALV\_EXT\_TF\_F\_84; ALV\_EXT\_TF\_F\_269; ALV\_EXT\_TF\_F\_270; ALV\_EXT\_TF\_F\_274; ALV\_EXT\_TF\_F\_275; ALV\_EXT\_TF\_F\_276; ALV\_EXT\_TF\_F\_284; ALV\_EXT\_TF\_F\_285; ALV\_EXT\_TF\_F\_289; ALV\_EXT\_TF\_F\_290; ALV\_EXT\_TF\_F\_294; ALV\_EXT\_TF\_F\_295; ALV\_EXT\_TF\_F\_296; ALV\_EXT\_TF\_F\_304; ALV\_EXT\_TF\_F\_305; ALV\_EXT\_TF\_F\_309; ALV\_EXT\_TF\_F\_310; ALV\_EXT\_TF\_F\_314; ALV\_EXT\_TF\_F\_315; ALV\_EXT\_TF\_F\_316; ALV\_EXT\_TF\_F\_320; ALV\_EXT\_TF\_F\_323; ALV\_EXT\_TF\_F\_327; ALV\_EXT\_TF\_F\_328; ALV\_EXT\_TF\_F\_329; ALV\_EXT\_TF\_F\_330; ALV\_EXT\_TF\_F\_331 |
| **ARCH\_SW\_PAL\_0105** | The component shall be able to read the current measured at the Vp motor pin in amps | PAL\_runReadMotorCurrentInA() | ALV\_EXT\_TF\_F\_280 |
| **ARCH\_SW\_PAL\_0106** | The component shall be able to read the current measured at the Vn motor pin in amps | PAL\_runReadMotorCurrentInA() | ALV\_EXT\_TF\_F\_300 |
| **ARCH\_SW\_PAL\_0140** | The component shall check for HW thermanl protection | PAL\_Autotest\_CheckHWSelfProtection() | ALV\_EXT\_TF\_H\_684; ALV\_EXT\_TF\_H\_685; |
| **ARCH\_SW\_PAL\_0340** | The component shall check the current used by the motor | PAL\_AutotestCheckMotorCurrent() | ALV\_EXT\_TF\_H\_1029; ALV\_EXT\_TF\_H\_1030; ALV\_EXT\_TF\_H\_1031; ALV\_EXT\_TF\_H\_1032; ALV\_EXT\_TF\_H\_1033; ALV\_EXT\_TF\_H\_1034; ALV\_EXT\_TF\_H\_1035;ALV\_EXT\_TF\_H\_1037; ALV\_EXT\_TF\_H\_1038; |
| **ARCH\_SW\_PAL\_0360** | The component shall check if an over temperature of the power stage is reached by returning a SW self-protection status | PAL\_Autotest\_CHeckMotorThermalProtection() | ALV\_EXT\_TF\_H\_1074; |
| **ARCH\_SW\_PAL\_0366** | The component shall check the state of specific autotests | PAL\_Autotest\_CHeckMotorThermalProtection() | ALV\_EXT\_TF\_H\_1067; |
| **ARCH\_SW\_PAL\_0368** | The component shall check if high switch regulation works well. | PAL\_Autotest\_CheckHighSideSwRegulation() | ALV\_EXT\_TF\_H\_1248; ALV\_EXT\_TF\_H\_1249; ALV\_EXT\_TF\_H\_1250; ALV\_EXT\_TF\_H\_1251; ALV\_EXT\_TF\_H\_1255; ALV\_EXT\_TF\_H\_1259; ALV\_EXT\_TF\_H\_2473; |
| **ARCH\_SW\_PAL\_0369** | The component shall check if the power stage activation state is consistent with the order. | PAL\_Autotest\_CheckCommandConsistency() | ALV\_EXT\_TF\_H\_646; |
| **ARCH\_SW\_PAL\_0370** | The component shall check the Short Circuit failure for Low Mosfets and set the result of autotest. | PAL\_Autotest\_CheckMosfetLowSC() | DAI\_EXT\_TF\_H\_2431; ALV\_EXT\_TF\_H\_748; DAI\_EXT\_TF\_H\_2430; |
| **ARCH\_SW\_PAL\_0372** | The component shall check the Short Circuit failure for High Mosfets and set the result of autotest. | PAL\_Autotest\_CheckMosfetHighSC() | DAI\_EXT\_TF\_H\_2432; ALV\_EXT\_TF\_H\_744; |
| **ARCH\_SW\_PAL\_0373** | The component shall check is the power bridge is open circuit | PAL\_Autotest\_CheckMosfetOCAT() | DAI\_EXT\_TF\_H\_2247; DAI\_EXT\_TF\_H\_2248;  DAI\_EXT\_TF\_H\_2249; DAI\_EXT\_TF\_H\_2250; DAI\_EXT\_TF\_H\_2251; DAI\_EXT\_TF\_H\_2252; ALV\_EXT\_TF\_H\_843; ALV\_EXT\_TF\_H\_844; ALV\_EXT\_TF\_H\_847; ALV\_EXT\_TF\_H\_848; ALV\_EXT\_TF\_H\_849; ALV\_EXT\_TF\_H\_850; ALV\_EXT\_TF\_H\_852; ALV\_EXT\_TF\_H\_853; ALV\_EXT\_TF\_H\_854; ALV\_EXT\_TF\_H\_855; ALV\_EXT\_TF\_H\_859; ALV\_EXT\_TF\_H\_866; ALV\_EXT\_TF\_H\_860; ALV\_EXT\_TF\_H\_861; ALV\_EXT\_TF\_H\_862; |
| **ARCH\_SW\_PAL\_0374** | The Autotest shall be performed in 4 steps in order to set the result. | PAL\_Autotest\_CheckMosfetOCAT() | ALV\_EXT\_TF\_H\_797; ALV\_EXT\_TF\_H\_798;  ALV\_EXT\_TF\_H\_799;  ALV\_EXT\_TF\_H\_800; |
| **ARCH\_SW\_PAL\_0376** | The component shall check if the motor is present and connected | PAL\_Autotest\_CheckMotorConnection() | ALV\_EXT\_TF\_H\_906; ALV\_EXT\_TF\_H\_907; |
| **ARCH\_SW\_PAL\_0380** | The component shall check if the motor is coupled with the mechanic. | PAL\_Autotest\_CheckMotorDisengagement() | ALV\_EXT\_TF\_H\_958; ALV\_EXT\_TF\_H\_975; ALV\_EXT\_TF\_H\_976; |
| **ARCH\_SW\_PAL\_0382** | The component shall check if a high PWM duty cycle will not be applied for a too long time | PAL\_Autotest\_CheckMotorPowerOrder() | ALV\_EXT\_TF\_H\_1122; ALV\_EXT\_TF\_H\_1126; ALV\_EXT\_TF\_H\_1127; ALV\_EXT\_TF\_H\_1128; ALV\_EXT\_TF\_H\_1129; ALV\_EXT\_TF\_H\_1130;  ALV\_EXT\_TF\_H\_1131; |
| **ARCH\_SW\_PAL\_0386** | The component shall check if the motor is not shortcut with Vbat or GND | PAL\_Autotest\_CheckMotorSC() | ALV\_EXT\_TF\_H\_1181; ALV\_EXT\_TF\_H\_1197; ALV\_EXT\_TF\_H\_1198; |
| **ARCH\_SW\_PAL\_0077** | This function shall set the power stage in IDLE state | PAL\_StartFreeWheelingState() |  |

This first figure gives an overview of PAL SW unit, including the 3 parts mentioned above.



**Figure 1: PAL - Static description**

In summary, from the left to the right:

* The actuator part

It will start from BFE which will use a PAL service to request power order. PAL will then distribute this power order over the different HW parts of the full power bridge, using PWM and DIO services.

As reminder, the full power bridge is composed of 2 main HW stages:

* + H-bridge
  + Booster (if equipped)
* The sensor part

On demand, PAL will scale and provide the measured current based on current sampled by ADC.

* The auto-tests part

PAL will be responsible for testing the full power bridge.

PAL will provide auto-test callback functions which will be executed by ATM.

# Technical functions - Actuator

## Motor power order scaling

From a function point of view, the motor power order scaling is [+6400; +6912].

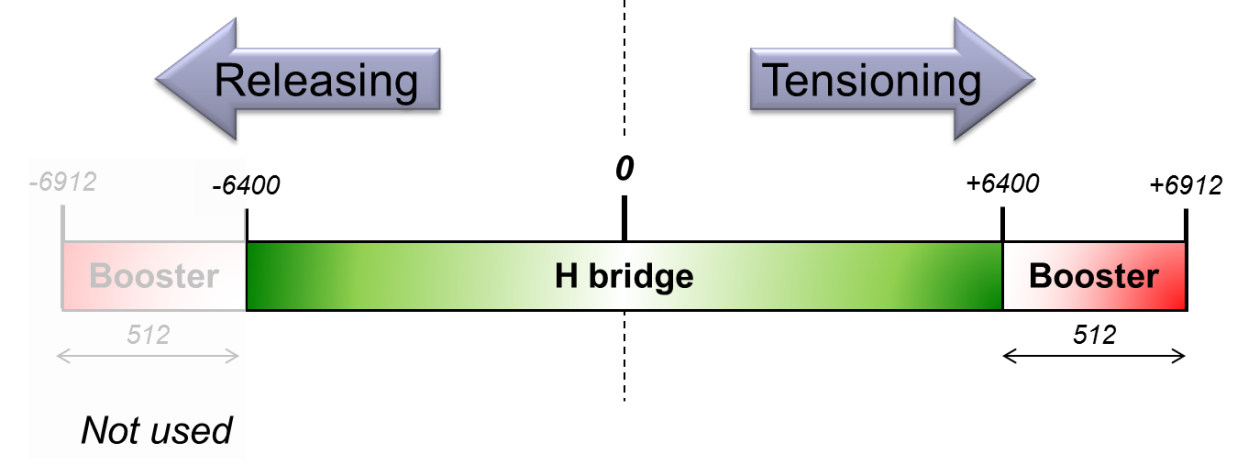
Actually this scaling is the union of 2 scalings:

* H-bridge power order scaling: [-6400 ; +6400]

Within this range, only the H-bridge is used to drive the motor in both directions

* Booster control order scaling [+0; +512]Booster

Within this range, the booster will provide additional power to the power, in complement to the H-bridge



**Figure 2: Pal: Motor power order scaling**

For an implementation point of view, the motor power order will be split up in 2 orders corresponding to respectively the H-bridge and booster order scalings.

By this way, this will give the possibility to control independently the H-bridge and booster HW stages (even if in practise the booster functions will be always used in complement to the H-bridge).

Then, the PAL\_runSetPowerOrder function will have 2 arguments:

* The first argument will allow controlling the power supplied by the H-bridge [-6400; +6400]
* The second argument will allow controlling the additional power supplied by the booster HW stage [0; +512]

Besides, the motor power order will be signed to drive the motor in both directions which means:

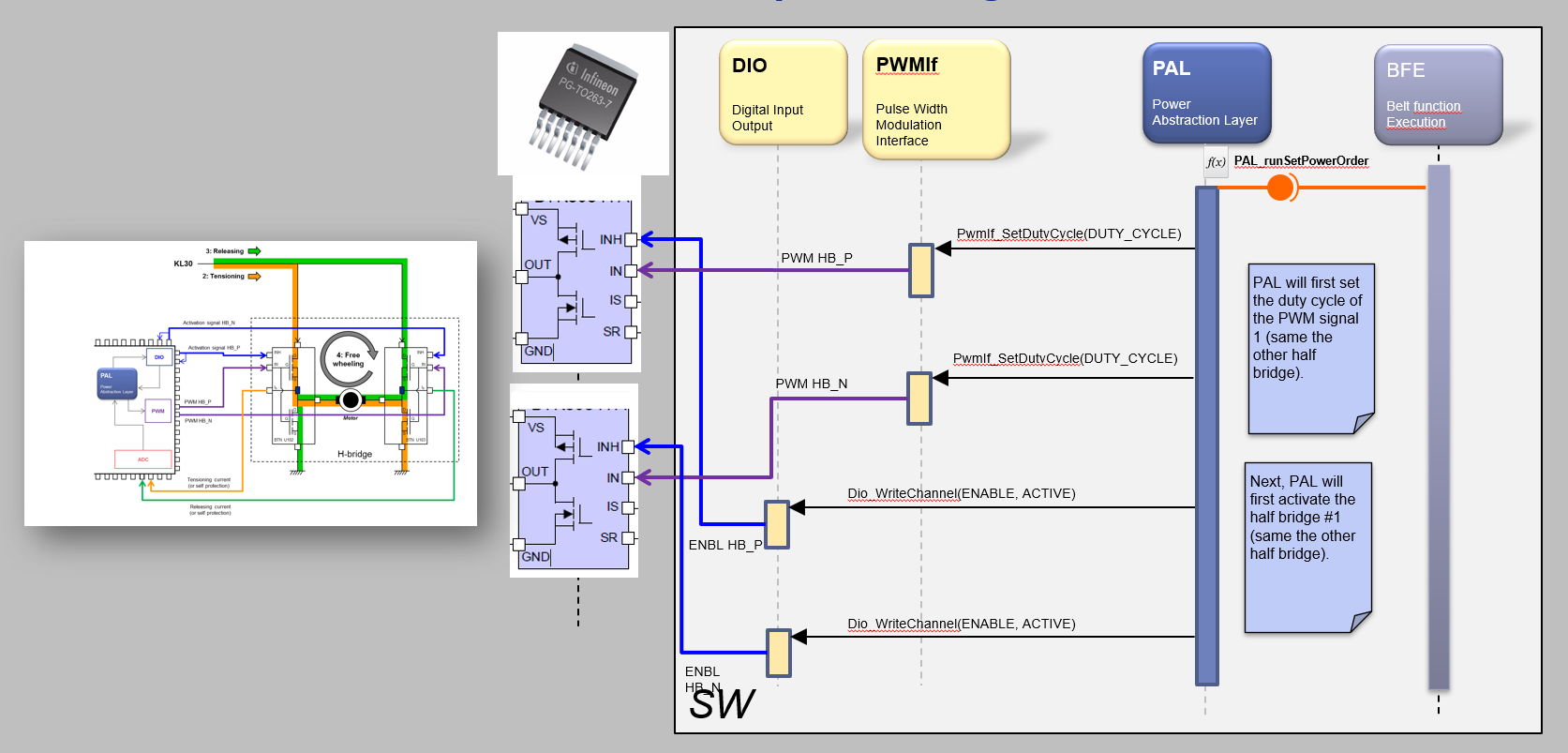
* If positive, the motor will be driven in the TENSIONING direction
* If negative, the motor will be driven in the RELEASING direction

## To control the motor WITHOUT booster

Most of the time, the motor will be driven without activating the booster function.

In this case, only the H-bridge stage will be periodically and synchronously controlled from BFE, all along the belt function cycle execution.

PAL will then control the 2 half bridges using PWM and DIO services. By this way, PAL will apply the requested motor power order in both directions.



**Figure 3: Pal - To control the Motor power (without booster)**

### To control the motor in tensioning direction

As reminder, 2 PWM signals associated to 2 activation signals (controlled by the MCU) are respectively connected to the half bridges #P and #N.

When the motor is activated in the TENSIONING direction, the 2 PWM are controlled as follows:

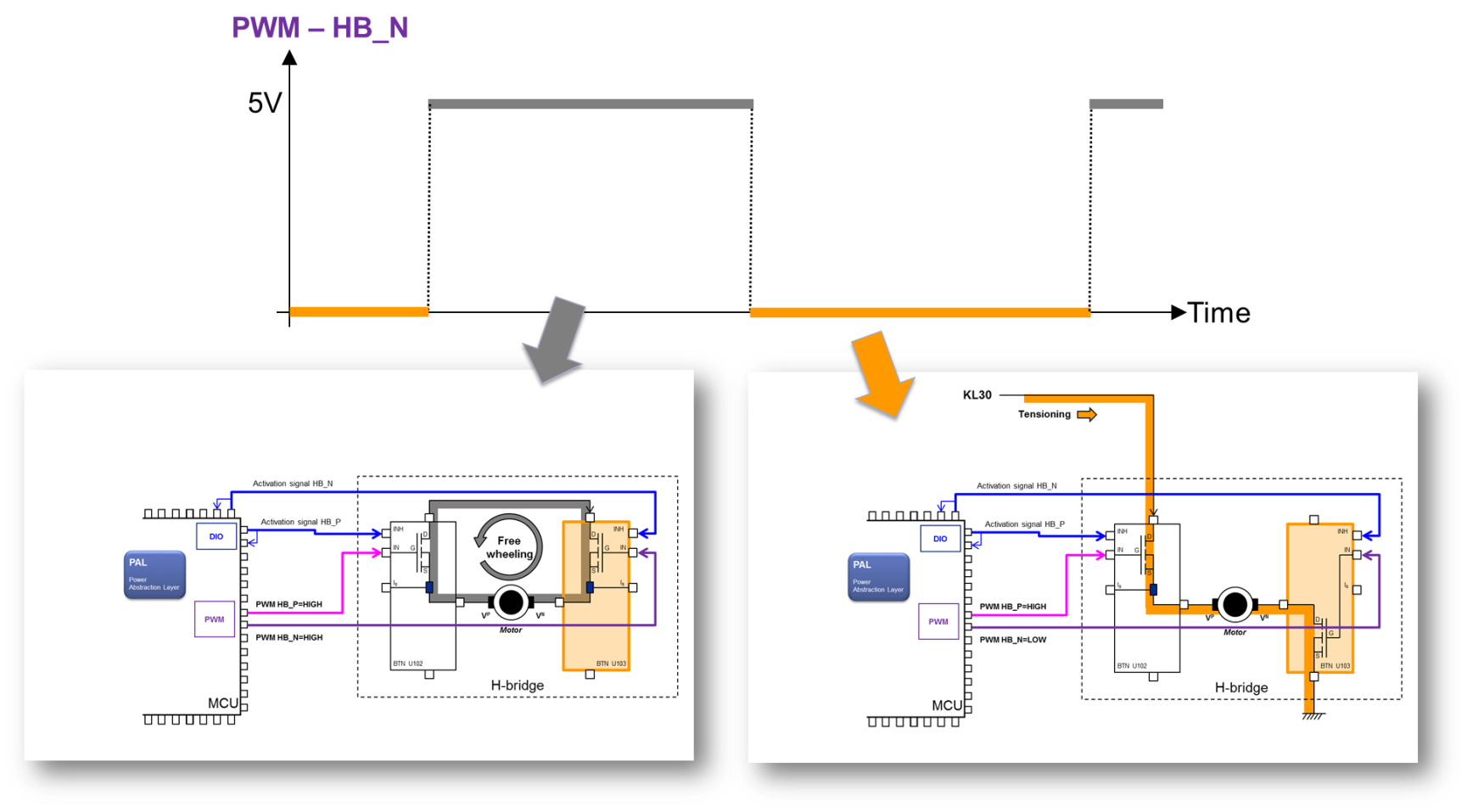
1. The duty cycle of the PWM signal connected to the half bridge #P is permanently set to 100% during the tensioning cycle.

By this way the high MOSFET of the half bridge #P will be permanently turned on.

1. The duty cycle of the PWM signal connected to the half bridge #N is controlled according to the tensioning cycle profile parameters:

* When the PWM signal level is low (the low MOFSET is closed), the battery voltage is applied at motor terminals (current flows to the ground)
* When the PWM signal level is high (the high MOFSET is closed), the motor is in free-wheeling state, dissipating the stored energy through the 2 high MOFSET

By this way it will allow adjusting the amount of power given to the motor (see the figure below).



**Figure 4: Pal - To drive the motor in TENSIONING direction without booster**

Practically, the PAL component will proceed as follows to drive the motor in TENSIONING direction:

1. It will call the [PwmIf\_SetDutyCycle](#_Hlk386110566" \s "1,181583,181599,5,,Pwm_SetDutyCycle) service with a duty cycle equal to 100% to permanently close the high MOFSET of the half bridge #P
2. It will call the [PwmIf\_SetDutyCycle](#_Hlk386110566) service to apply the requested duty cycle to the HBA\_N component. It will allow controlling the power given to the motor in TENSIONING direction
3. It will call the [Dio\_WriteChannel](#_Hlk387853153) service to activate both half bridge components

### To control the motor in releasing direction

When the motor is activated in the RELEASING direction, the principle is the opposite of the TENSIONING one:

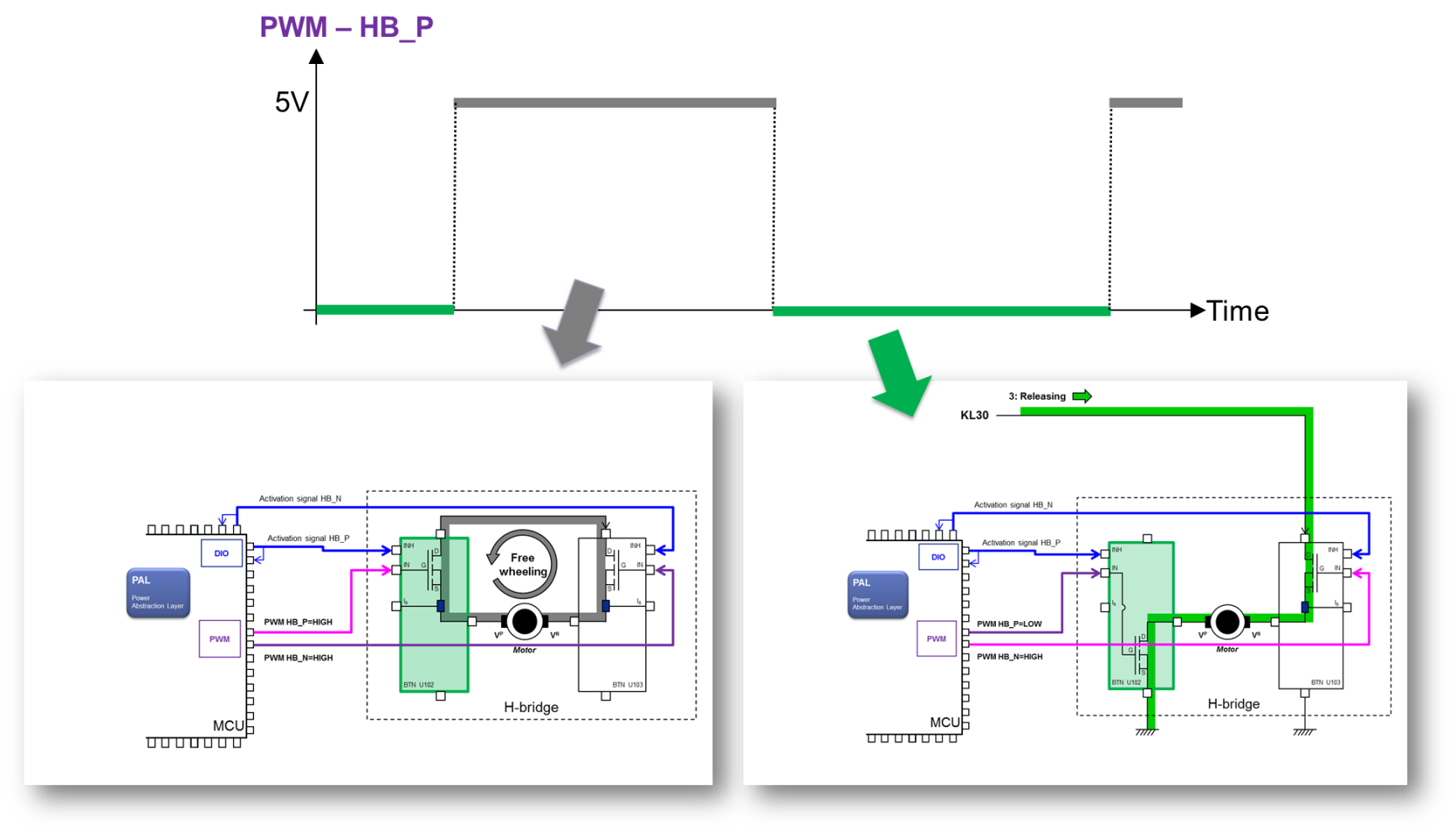
1. The duty cycle of the PWM signal connected to the half bridge #N is permanently set to 100% during the tensioning cycle.

By this way the high MOSFET of the half bridge #N will be permanently turned on.

1. The duty cycle of the PWM signal connected to the half bridge #P is controlled according to the tensioning cycle profile parameters:

* When the PWM signal level is low (the low MOFSET is closed), the battery voltage is applied at motor terminals (but in the other direction comparing to the tensioning one)
* When the PWM signal level is high (the high MOFSET is closed), the motor is in free-wheeling state, dissipating the stored energy through the 2 high MOFSET

By this way it will allow adjusting the amount of power given to the motor (see the figure below).



**Figure 5: Pal - To drive the motor in RELEASING direction without booster**

Practically, the PAL component will proceed as follows to drive the motor in RELEASING direction:

1. It will call the [PwmIf\_SetDutyCycle](#_Hlk386110566) service with a duty cycle equal to 100% to permanently close the high MOFSET of the half bridge #N
2. It will call the [PwmIf\_SetDutyCycle](#_Hlk386110566) service to apply the requested duty cycle to the HBA\_P component. It will allow controlling the power given to the motor in RELEASING direction
3. It will call the [Dio\_WriteChannel](#_Hlk387853153) service to activate both half bridge components

# Technical functions - Sensor

The other aspect of PAL is to take about information related to the HW power stage like current measurement, H-bridge protections status…

## To provide the motor current

The motor current acquisition chain can be divided in 5 steps:

* Step #1 (HW)

An image of the real motor current is provided by the current sense of the half bridge components to the MCU ADC unit (through the other HW components mounted between the half bridge components and the MCU).

Depending of the motor rotation direction, the current is provided either by the ‘P’ half bridge or the ‘N’ one.

* Step #2 (HW)

The 2 motor current HW signals are sampled by the MCU ADC unit

* Step #3 (SW)

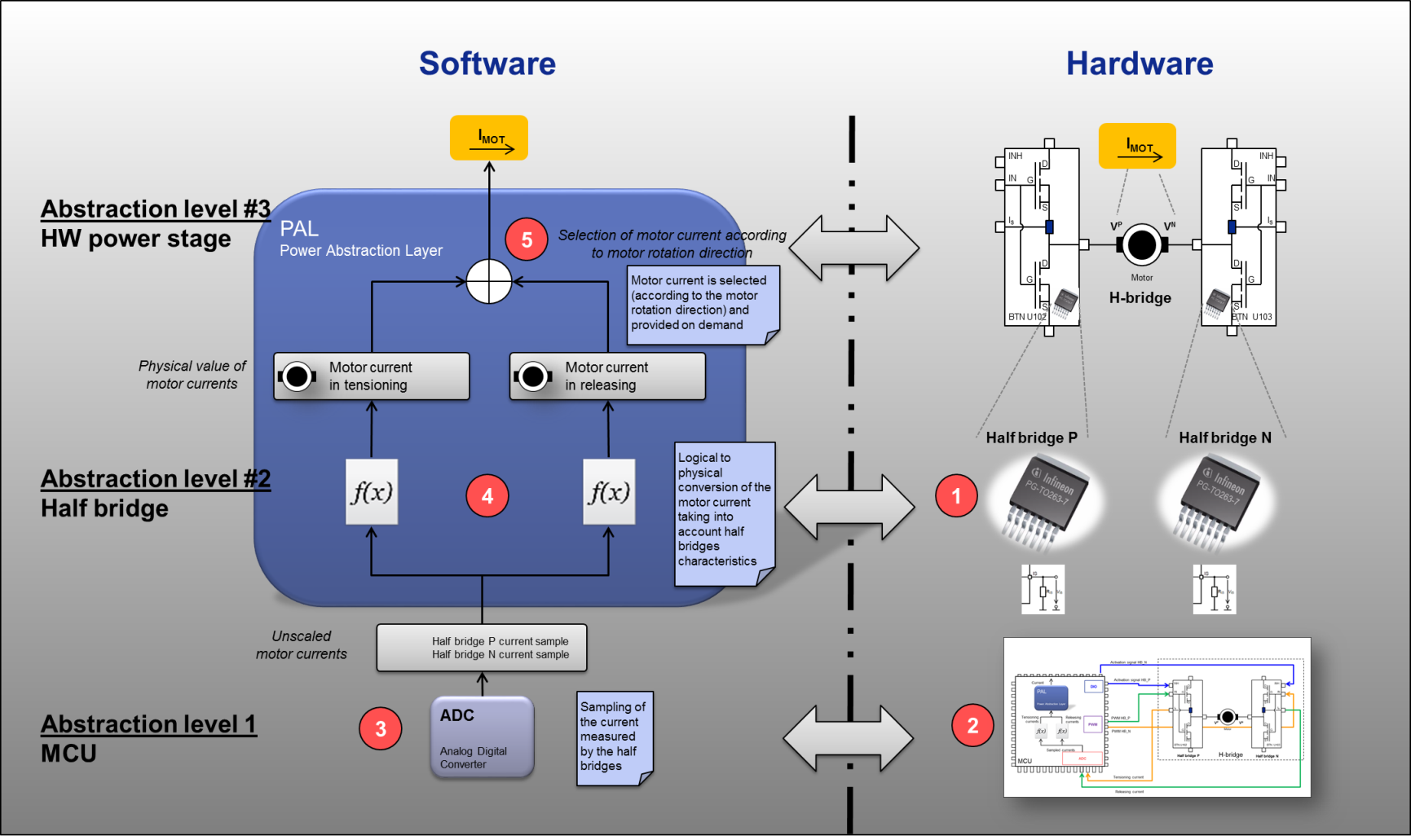
The ADC SW unit periodically trigs and records the motor current signals sampling.

* Step #4 (SW)

The low level of PAL will get and rescal the sampled motor currents in (mill) amps, based on the characteristics of each half bridge component

* Step #5 (SW)

The high level of PAL will provide (on demand) the motor current in (mill) amps (taking into account the motor rotation direction).



**Figure 9: Pal - Motor current acquisition chain**

The next figure describes a bit more about the principle to measure the motor current.

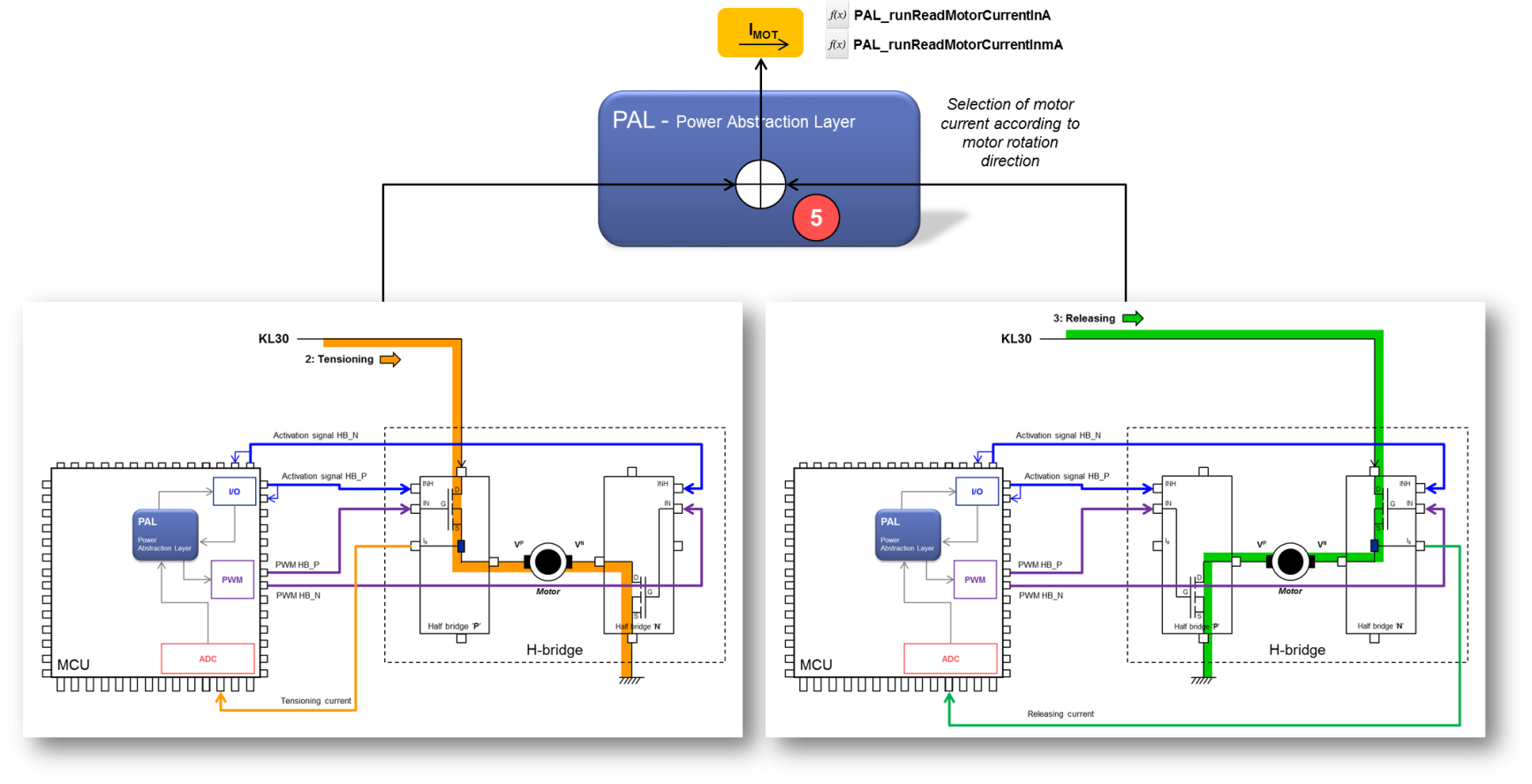
As reminder, the current is provided by

* The ‘P’ half-bridge of the motor is driven in tensioning direction
* The ‘N’ half-bridge of the motor is driven in releasing direction

Based on that, PAL will select and provide (on demand) the current according to the motor direction rotation.

Note:

If the half-bridge component is in HW self-protection, PAL will provide a null current.



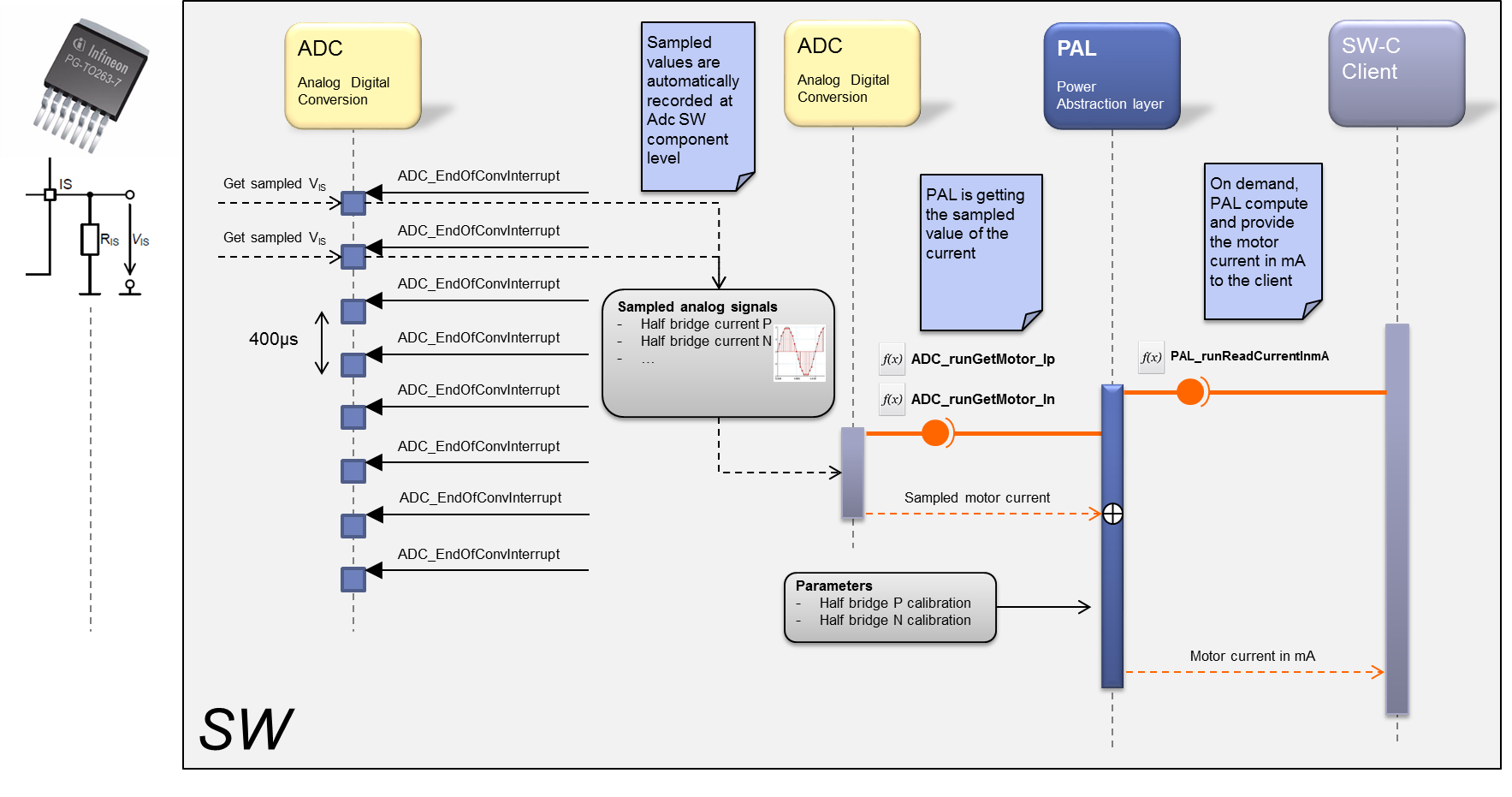
**Figure 10: Pal - Motor current selection**

The last figure below describes the real aspect of the motor current measurement:

PAL will then:

* Get the mirror of the motor current provided by the Infineon BTN chip
* Take into account the motor rotation direction
* Take into account the calibration parameters of the Infineon BTN current sense
* Provide the motor current to the SW client

This computation is done one demand in the real time context of the requesting client.



**Figure 11: Pal - Motor current computation**

# Technical functions – Auto-tests

The last aspect of PAL is the diagnosis of the complete power bridge.

## To check the HW self-protection state

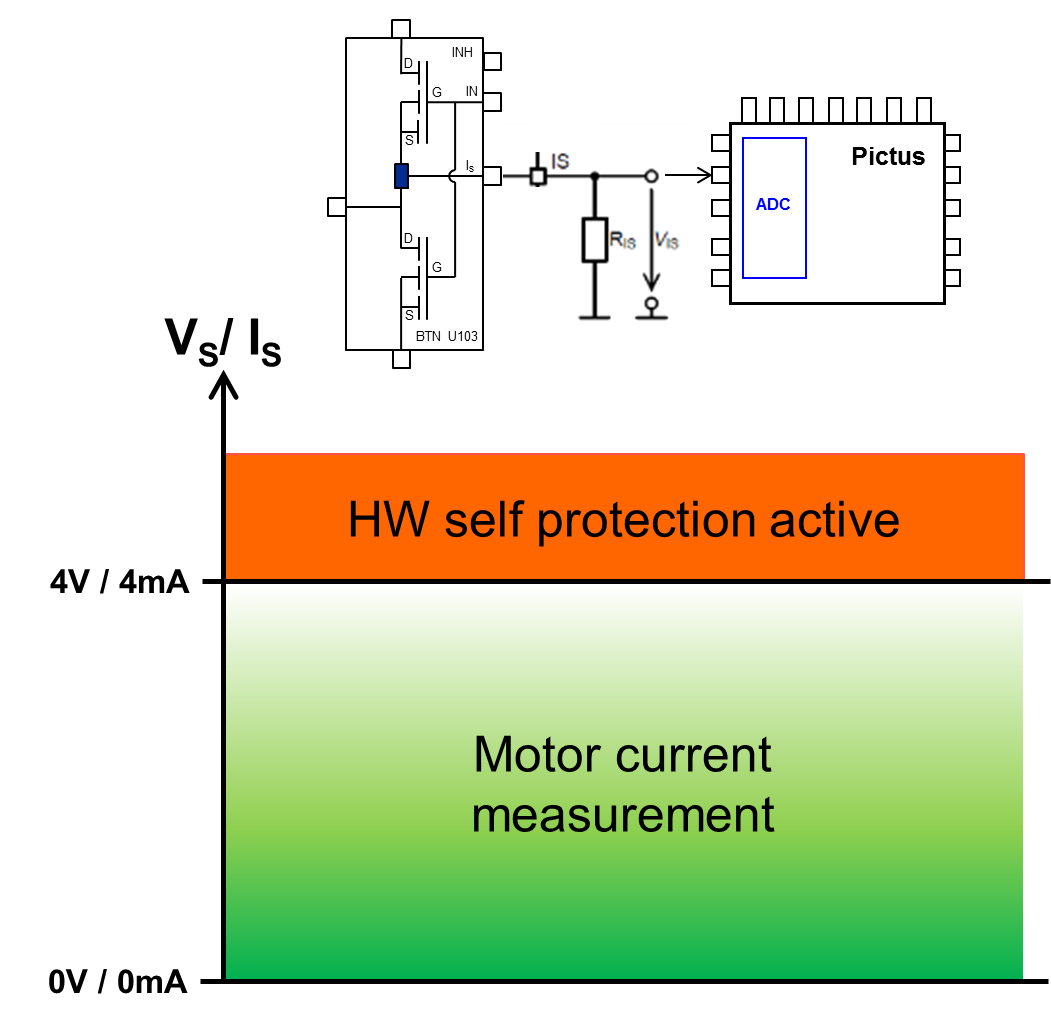
The 2 Infineon half-bridges are equipped with an integrated self-protection function to protect the chip against over-current, over-temperature… refer to [F1] for more details.

When active, the self-protection state will be transmitted to the SW application through the “motor current” HW signal.

As shown by the figure below, the distinction between the current measurement and the HW self-protection state is a matter of signal level.

According to [F1], if the half-bridge is in self-protection state, it will provide a current over 4mA (≡ 4V at ADC channel level) through the IS HW signal.

By this way, the PP SW will be able to distinguish the HW self-protection from the current measurement.



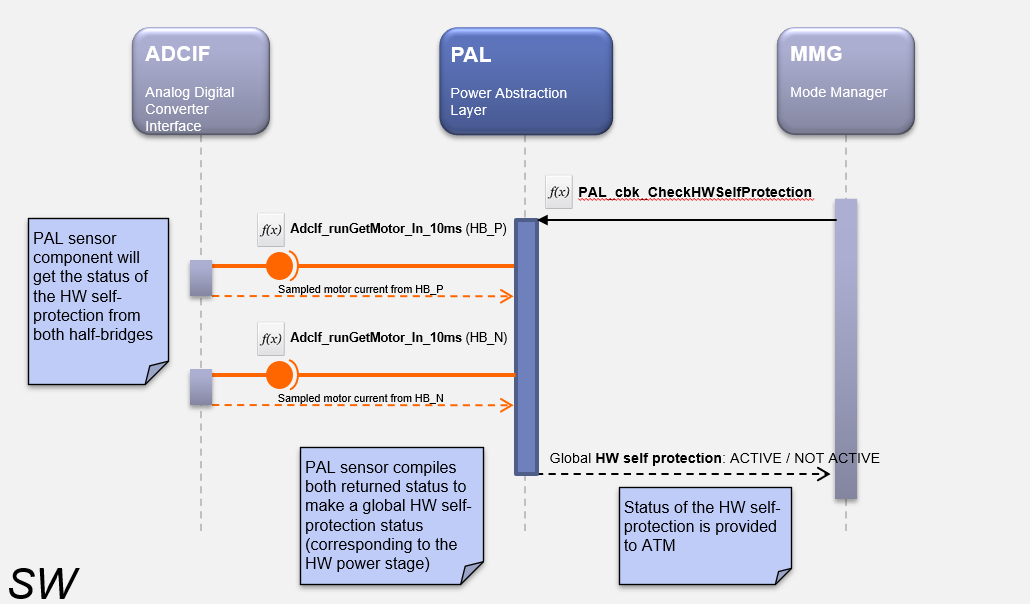
**Figure 12: Current / HW self-protection area**

This HW self-protection will be periodically monitored by PAL (auto-test functions):

* To detect if the self-protection is active at power bridge level
* To know if the motor current can be computed

PAL will compare the current sampled by ADC to the “4mA” threshold.

If above, the self-protection state will be validated by PAL and communicated to ATM.



**Figure 13: Pal- to monitor the HW self-protection**

## Runnables - Main

## PAL\_Init

### Definition

|  |  |  |  |
| --- | --- | --- | --- |
| **Prototype** | | | |
| void **PAL\_Init** (void) | | | |
| **Object** | | | |
| This function shall initialize the PAL SW unit. | | | |
| **Parameters** | | | |
| Name | Type | Direction | Description |
| NA | NA | NA | NA |
| **Returned value** | | | |
| Name | Description | | |
| NA | NA | | |
| **Dynamic aspect** | | | |
| Synchronous server operation  Non-reentrant | | | |
| **Covered requirements** | | | |
| ARCH\_SW\_PAL\_0010 | | | |

### Data flow / Parameters

NA

### Called functions

NA

## PAL\_runMainFunction

### Definition

|  |  |  |  |
| --- | --- | --- | --- |
| **Prototype** | | | |
| void **PAL\_runMainFunction** (void) | | | |
| **Object** | | | |
| Cyclic runnable in charge of the internal state machine. Handles free wheeling timer and disabling power stage. | | | |
| **Parameters** | | | |
| Name | Type | Direction | Description |
| NA | NA | NA | NA |
| **Returned value** | | | |
| Name | Description | | |
| NA | NA | | |
| **Dynamic aspect** | | | |
| Periodic - 2ms  Non-reentrant | | | |
| **Covered requirements** | | | |
| ARCH\_SW\_PAL\_0450; ARCH\_SW\_PAL\_0456; ARCH\_SW\_PAL\_0457 | | | |

### Data flow / Parameters

NA

### Called functions

|  |  |  |  |
| --- | --- | --- | --- |
| **Requirements** | **Criteria** | **Linked Runnable** | **Source** |
| ARCH\_SW\_PAL\_0456 | The PAL\_runMainFunction function shall allow the application to switch to the OFF\_MODE if a Shut Down is requested or if a critical autotest has failed during startup. |  |  |
| ARCH\_SW\_PAL\_0457 | The PAL\_runMainFunction function shall allow the application to enable control for autotest for motor command |  |  |

# Runnables - Actuator

From a dynamic perspective, the control of the power HW stage is performed in the real time context of the demander.

In other words, the PAL interface is completely composed of synchronous server operations to allow the client to control the full HW power bridge synchronously.

The present section will describe these server operations.

## PAL\_runSetPowerOrder

### Definition

|  |  |  |  |
| --- | --- | --- | --- |
| **Prototype** | | | |
| void **PAL\_runSetPowerOrder** (s16MotorPowerOrderType s16MotorPowerOrder,  u16BoostPWMOrderType u16BoostDutyCycle) | | | |
| **Object** | | | |
| This function shall control the full power bridge based on the requested power order. | | | |
| **Parameters** | | | |
| Name | Type | Direction | Description |
| s16MotorPowerOrder | s16MotorPowerOrderType | In | Motor power order |
| u16BoostDutyCycle | u16BoostPWMOrderType | In | High power order |
| **Returned value** | | | |
| Name | Description | | |
| NA | NA | | |
| **Dynamic aspect** | | | |
| Synchronous server operation  Non-reentrant | | | |
| **Covered requirements** | | | |
| ARCH\_SW\_PAL\_0020; ARCH\_SW\_PAL\_0025; ARCH\_SW\_PAL\_0026; ARCH\_SW\_PAL\_0027; ARCH\_SW\_PAL\_0028; ARCH\_SW\_PAL\_0029 | | | |

### Data flow / Parameters

NA

### Called functions

|  |  |  |  |
| --- | --- | --- | --- |
| **Requirements** | **Criteria** | **Linked Runnable** | **Source** |
| **ARCH\_SW\_PAL\_0025** | The [Dio\_WriteChannel](#_Hlk387853153) function shall be called to activate/deactivate the half bridge connected to the VP pin of the motor. | PAL\_runSetPowerOrder() |  |
| **ARCH\_SW\_PAL\_0026** | The [Dio\_WriteChannel](#_Hlk387853153) function shall be called to activate/deactivate the half bridge connected to the VN pin of the motor. | PAL\_runSetPowerOrder() |  |
| **ARCH\_SW\_PAL\_0027** | The [PwmIf\_SetDutyCycle](#_Hlk386110566) function shall be called to control MOSET’s of the half bridge connected to the VP pin of the motor. | PAL\_runSetPowerOrder() |  |
| **ARCH\_SW\_PAL\_0028** | The [PwmIf\_SetDutyCycle](#_Hlk386110566) function shall be called to control MOSET’s of the half bridge connected to the VN pin of the motor. | PAL\_runSetPowerOrder() |  |
| **ARCH\_SW\_PAL\_0029** | The [PwmIf\_SetDutyCycle](#_Hlk386110566) function shall be called to control the MOSET’s of the high power stage. | PAL\_runSetPowerOrder() |  |

## PAL\_DisablePowerStage

### Definition

|  |  |  |  |
| --- | --- | --- | --- |
| **Prototype** | | | |
| void **PAL\_DisablePowerStage** (void) | | | |
| **Object** | | | |
| * External service that shall disable the HW components that drive the motor. | | | |
| **Parameters** | | | |
| Name | Type | Direction | Description |
| NA | NA | NA | NA |
| **Returned value** | | | |
| Name | Description | | |
| NA | NA | | |
| **Dynamic aspect** | | | |
| Synchronous server operation  Non-reentrant | | | |
| **Covered requirements** | | | |
| ARCH\_SW\_PAL\_0070; ARCH\_SW\_PAL\_0075; ARCH\_SW\_PAL\_0076 | | | |

### Called functions

|  |  |  |  |
| --- | --- | --- | --- |
| **Requirements** | **Criteria** | **Linked Runnable** | **Source** |
| **ARCH\_SW\_PAL\_0075** | The [Dio\_WriteChannel](#_Hlk387853153) function shall be called to deactivate the half bridge connected to the VP pin of the motor. | **PAL\_DisablePowerStage()** |  |
| **ARCH\_SW\_PAL\_0076** | The [Dio\_WriteChannel](#_Hlk387853153) function shall be called to deactivate the half bridge connected to the VN pin of the motor. | **PAL\_DisablePowerStage()** |  |

## PAL\_ StartFreeWheelingState

### Definition

|  |  |  |  |
| --- | --- | --- | --- |
| **Prototype** | | | |
| void **PAL\_StartFreeWheelingState** (void) | | | |
| **Object** | | | |
| This function shall set the power stage in IDLE state | | | |
| **Parameters** | | | |
| Name | Type | Direction | Description |
| NA | NA | NA | NA |
| **Returned value** | | | |
| Name | Description | | |
| NA | NA | | |
| **Dynamic aspect** | | | |
| Synchronous server operation  Non-reentrant | | | |
| **Covered requirements** | | | |
| ARCH\_SW\_PAL\_0077; ARCH\_SW\_PAL\_0078; ARCH\_SW\_PAL\_0079 | | | |

### Called functions

|  |  |  |  |
| --- | --- | --- | --- |
| **Requirements** | **Criteria** | **Linked Runnable** | **Source** |
| **ARCH\_SW\_PAL\_0078** | The [Dio\_WriteChannel](#_Hlk387853153) function shall be called to activate the half bridge connected to the VP pin of the motor. | **PAL\_StartFreeWheelingState()** |  |
| **ARCH\_SW\_PAL\_0079** | The [Dio\_WriteChannel](#_Hlk387853153) function shall be called to activate the half bridge connected to the VN pin of the motor. | **PAL\_StartFreeWheelingState()** |  |

## PAL\_Cfg\_Init

### Definition

|  |  |  |  |
| --- | --- | --- | --- |
| **Prototype** | | | |
| void **PAL\_Cfg\_Init** (void) | | | |
| **Object** | | | |
| This function shall init the hardware according to current config | | | |
| **Parameters** | | | |
| Name | Type | Direction | Description |
| NA | NA | NA | NA |
| **Returned value** | | | |
| Name | Description | | |
| NA | NA | | |
| **Dynamic aspect** | | | |
| Synchronous server operation  Non-reentrant | | | |
| **Covered requirements** | | | |
| ARCH\_SW\_PAL\_0090; ARCH\_SW\_PAL\_0095 | | | |

### Called functions

|  |  |  |  |
| --- | --- | --- | --- |
| **Requirements** | **Criteria** | **Linked Runnable** | **Source** |
| **ARCH\_SW\_PAL\_0095** | The [Dio\_WriteChannel](#_Hlk387853153) function shall be called to deactivate the half bridge | **PAL\_Cfg\_Init()** |  |

## PAL\_AT\_Init

### Definition

|  |  |  |  |
| --- | --- | --- | --- |
| **Prototype** | | | |
| void **PAL\_AT\_Init** (void) | | | |
| **Object** | | | |
| This function shall initialize the static data used during PAL auto tests | | | |
| **Parameters** | | | |
| Name | Type | Direction | Description |
| NA | NA | NA | NA |
| **Returned value** | | | |
| Name | Description | | |
| NA | NA | | |
| **Dynamic aspect** | | | |
| Synchronous server operation  Non-reentrant | | | |
| **Covered requirements** | | | |
| ARCH\_SW\_PAL\_0110 | | | |

# Runnables - Sensor

## PAL\_runReadMotorCurrentInmA

### Definition

|  |  |  |  |
| --- | --- | --- | --- |
| **Prototype** | | | |
| void **PAL\_runReadMotorCurrentInmA** (s32MotorCurrentInmAType \* ps32MotorCurrentInmA) | | | |
| **Object** | | | |
| This function shall provide the current in milliamps. | | | |
| **Parameters** | | | |
| Name | Type | Direction | Description |
| ps32MotorCurrentInmA | s32MotorCurrentInmAType | Out | Signed current in milliamps:  Positive 🡪 tensioning current  Negative 🡪 releasing current |
| **Returned value** | | | |
| Name | Description | | |
| NA | NA | | |
| **Dynamic aspect** | | | |
| Synchronous server operation  Reentrant | | | |
| **Covered requirements** | | | |
| ARCH\_SW\_PAL\_0080; ARCH\_SW\_PAL\_0085; ARCH\_SW\_PAL\_0086 | | | |

### Called functions

|  |  |  |  |
| --- | --- | --- | --- |
| **Requirements** | **Criteria** | **Linked Runnable** | **Source** |
| **ARCH\_SW\_PAL\_0085** | If the motor rotation direction is “tensioning”, the AdcIf\_runGetMotor\_Ip function shall be called to read the motor current measured by the half bridge connected to the VP pin of the motor. | PAL\_runReadMotorCurrentInmA() |  |
| **ARCH\_SW\_PAL\_0086** | If the motor rotation direction is “releasing”, the AdcIf\_runGetMotor\_In function shall be called to read the motor current measured by the half bridge connected to the VN pin of the motor. | PAL\_runReadMotorCurrentInmA() |  |

## PAL\_runReadMotorCurrentInA

### Definition

|  |  |  |  |
| --- | --- | --- | --- |
| **Prototype** | | | |
| void **PAL\_runReadMotorCurrentInA** (s8MotorCurrentInAType \* ps8MotorCurrentInA) | | | |
| **Object** | | | |
| This function shall provide the current in amps. | | | |
| **Parameters** | | | |
| Name | Type | Direction | Description |
| ps8MotorCurrentInA | s8MotorCurrentInAType | Out | Signed current in amps:  Positive 🡪 tensioning current  Negative 🡪 releasing current |
| **Returned value** | | | |
| Name | Description | | |
| NA | NA | | |
| **Dynamic aspect** | | | |
| Synchronous server operation  Reentrant | | | |
| **Covered requirements** | | | |
| ARCH\_SW\_PAL\_0100; ARCH\_SW\_PAL\_0105; ARCH\_SW\_PAL\_0106 | | | |

### Called functions

|  |  |  |  |
| --- | --- | --- | --- |
| **Requirements** | **Criteria** | **Linked Runnable** | **Source** |
| **ARCH\_SW\_PAL\_0105** | If the motor rotation direction is “tensioning”, the [AdcIf\_runGetMotor\_Ip](#_Hlk424735356) function shall be called to read the motor current measured by the half bridge connected to the VP pin of the motor. | PAL\_runReadMotorCurrentInA() |  |
| **ARCH\_SW\_PAL\_0106** | If the motor rotation direction is “releasing”, the [AdcIf\_runGetMotor\_In](#_Hlk424735356) function shall be called to read the motor current measured by the half bridge connected to the VN pin of the motor. | PAL\_runReadMotorCurrentInA() |  |

# Runnables – To check power-bridge

## PAL\_Autotest\_CheckHighSideSwRegulation

### Definition

|  |  |  |  |
| --- | --- | --- | --- |
| **Prototype** | | | |
| void PAL\_Autotest\_CheckHighSideSwRegulation (u8TestResultType \*pu8TestResult) | | | |
| **Object** | | | |
| This function shall checkif high switch regulation works well. | | | |
| **Parameters** | | | |
| Name | Type | Direction | Description |
| pu8TestResult | u8TestResultType\* (uint8) | OUT | Status of the test |
| **Returned value** | | | |
| Name | Description | | |
| NA | - | | |
| **Dynamic aspect** | | | |
| Synchronous server operation  Non-reentrant | | | |
| **Requirements** | | | |
| **ARCH\_SW\_PAL\_0368;** | | | |

## PAL\_Autotest\_CheckHWSelfProtection

### Definition

|  |  |  |  |
| --- | --- | --- | --- |
| **Prototype** | | | |
| void **PAL\_Autotest\_CheckHWSelfProtection** (u8TestResultType \*pu8TestResult) | | | |
| **Object** | | | |
| This function shall check if the HW thermal protection is active. | | | |
| **Parameters** | | | |
| Name | Type | Direction | Description |
| pu8TestResult | u8TestResultType\* (uint8) | OUT | Status of the test |
| **Returned value** | | | |
| Name | Description | | |
| NA | - | | |
| **Dynamic aspect** | | | |
| Synchronous server operation  Reentrant | | | |
| **Covered requirements** | | | |
| ARCH\_SW\_PAL\_0140; ARCH\_SW\_PAL\_0147; | | | |

### Called functions

|  |  |  |  |
| --- | --- | --- | --- |
| **Requirements** | **Criteria** | **Linked Runnable** | **Source** |
| **ARCH\_SW\_PAL\_0147** | ATM\_runGetTestResult shall be called to get the status ot specific tests (inhibiting the execution of the present one if failed). | PAL\_Autotest\_CheckHWSelfProtection() | ALV\_EXT\_TF\_H\_677, |

## PAL\_Autotest\_CheckCommandConsistency

### Definition

|  |  |  |  |
| --- | --- | --- | --- |
| **Prototype** | | | |
| void **PAL\_Autotest\_CheckCommandConsistency** (u8TestResultType \*pu8TestResult) | | | |
| **Object** | | | |
| This function shall indicate if the power stage activation state is consistent with the order. | | | |
| **Parameters** | | | |
| Name | Type | Direction | Description |
| pu8TestResult | u8TestResultType (uint8) | OUT | Status of the test |
| **Returned value** | | | |
| Name | Description | | |
| NA | - | | |
| **Dynamic aspect** | | | |
| Synchronous server operation  Reentrant | | | |
| **Requirements** | | | |
| **ARCH\_SW\_PAL\_0369;** | | | |

## PAL\_Autotest\_CheckMosfetLowSC

### Definition

|  |  |  |  |
| --- | --- | --- | --- |
| **Prototype** | | | |
| void **PAL\_Autotest\_CheckMosfetLowSC** (u8TestResultType \* pu8TestResult) | | | |
| **Object** | | | |
| This function shall test if the low MOFSET of the power bridge are in short circuit. | | | |
| **Parameters** | | | |
| Name | Type | Direction | Description |
| pu8TestResult | u8TestResultType\* (uint8) | OUT | Status of the test |
| **Returned value** | | | |
| Name | Description | | |
| NA | - | | |
| **Dynamic aspect** | | | |
| Server operation  Non Reentrant | | | |
| **Requirements** | | | |
| ARCH\_SW\_PAL\_0370; ARCH\_SW\_PAL\_0391; ARCH\_SW\_PAL\_0390; | | | |

### Data flow

|  |  |  |  |
| --- | --- | --- | --- |
| **Requirements** | **Criteria** | **Levels/Tolerances** | **Source** |
|  | NVP parameters shall be used by this auto-tests |  |  |

### Called functions

|  |  |  |  |
| --- | --- | --- | --- |
| **Requirements** | **Criteria** | **Linked Runnable** | **Source** |
| ARCH\_SW\_PAL\_0390; | [ATM\_runGetTestResult](#_Ref410741888) shall be called to get the status of specific tests (inhibiting the execution of the present one if failed). | PAL\_Autotest\_CheckMosfetLowSC() | ALV\_EXT\_TF\_H\_728; ALV\_EXT\_TF\_H\_731;  ALV\_EXT\_TF\_H\_735; |
| ARCH\_SW\_PAL\_0391; | PMP\_runGetMotorVp\_100ms and PMP\_runGetMotorVn\_100ms shall be called to check battery voltage. | PAL\_Autotest\_CheckMosfetLowSC() | ALV\_EXT\_TF\_H\_730; |

## PAL\_Autotest\_CheckMosfetHighSC

### Definition

|  |  |  |  |
| --- | --- | --- | --- |
| **Prototype** | | | |
| void **PAL\_Autotest\_CheckMosfetHighSC** (u8TestResultType \*pu8TestResult) | | | |
| **Object** | | | |
| This function shall test if the high MOFSET of the power bridge are in short circuit. | | | |
| **Parameters** | | | |
| Name | Type | Direction | Description |
| pu8TestResult | u8TestResultType\* (uint8) | OUT | Status of the test |
| **Returned value** | | | |
| Name | Description | | |
| NA | - | | |
| **Dynamic aspect** | | | |
| Server operation  Non Reentrant | | | |
| **Requirements** | | | |
| ARCH\_SW\_PAL\_0372; ARCH\_SW\_PAL\_0395; ARCH\_SW\_PAL\_0396; | | | |

### Data flow

|  |  |  |  |
| --- | --- | --- | --- |
| **Requirements** | **Criteria** | **Levels/Tolerances** | **Source** |
|  | NVP parameters shall be used by this auto-tests |  |  |

### Called functions

|  |  |  |  |
| --- | --- | --- | --- |
| **Requirements** | **Criteria** | **Linked Runnable** | **Source** |
| ARCH\_SW\_PAL\_0395; | [ATM\_runGetTestResult](#_Ref410741888) shall be called to get the status of specific tests (inhibiting the execution of the present one if failed). | PAL\_Autotest\_CheckMosfetHighSC() | ALV\_EXT\_TF\_H\_728; ALV\_EXT\_TF\_H\_731;  ALV\_EXT\_TF\_H\_735; |
| ARCH\_SW\_PAL\_0396; | PMP\_runGetBatteryVoltage\_100ms , PMP\_runGetMotorVp\_100ms and PMP\_runGetMotorVn\_100ms shall be called to check battery voltage. | **P**AL\_Autotest\_CheckMosfetHighSC() | ALV\_EXT\_TF\_H\_730; |

## PAL\_Autotest\_CheckMosfetOCAT

### Definition

|  |  |  |  |
| --- | --- | --- | --- |
| **Prototype** | | | |
| void **PAL\_Autotest\_CheckMosfetOCAT** (u8TestResultType \* pu8TestResult) | | | |
| **Object** | | | |
| This function shall test if MOFSET of the power bridge stay in open circuit when ordered. | | | |
| **Parameters** | | | |
| Name | Type | Direction | Description |
| pu8TestResult | u8TestResultType\* (uint8) | OUT | Status of the test |
| **Returned value** | | | |
| Name | Description | | |
| NA | - | | |
| **Dynamic aspect** | | | |
| Server operation  Non Reentrant | | | |
| **Requirements** | | | |
| ARCH\_SW\_PAL\_0374; ARCH\_SW\_PAL\_0375; ARCH\_SW\_PAL\_0373; | | | |

### Data flow

|  |  |  |  |
| --- | --- | --- | --- |
| **Requirements** | **Criteria** | **Levels/Tolerances** | **Source** |
|  | NVP parameters shall be used by this auto-tests |  |  |

### Called functions

|  |  |  |  |
| --- | --- | --- | --- |
| **Requirements** | **Criteria** | **Linked Runnable** | **Source** |
| ARCH\_SW\_PAL\_0375 | [ATM\_runGetTestResult](#_Ref410741888) shall be called to get the status ot specific tests (inhibiting the execution of the present one if failed). | PAL\_Autotest\_CheckMosfetOCAT() | ALV\_EXT\_TF\_H\_786; ALV\_EXT\_TF\_H\_789;  ALV\_EXT\_TF\_H\_790; ALV\_EXT\_TF\_H\_791;  ALV\_EXT\_TF\_H\_795; |
|  | [PMP\_runGetMotorVp\_20ms](#_Hlk412189179)function shall be called to get the value of the motor V+ voltage. | PAL\_Autotest\_CheckMosfetOCAT() |  |
|  | [PMP\_runGetMotorVn\_20ms](#_Hlk412189278)function shall be called to get the value of the motor V- voltage. | PAL\_Autotest\_CheckMosfetOCAT() |  |
|  | [PMP\_runGetBatteryVoltage\_20ms](#_Hlk411940329)function shall be called to get the value of ground signal. | PAL\_Autotest\_CheckMosfetOCAT() |  |

# Runnables – To check motor

## PAL\_Autotest\_CheckMotorConnection

### Definition

|  |  |  |  |
| --- | --- | --- | --- |
| **Prototype** | | | |
| void **PAL\_Autotest\_CheckMotorConnection** (u8TestResultType \*pu8TestResult) | | | |
| **Object** | | | |
| This function shall test if the motor is well connected. | | | |
| **Parameters** | | | |
| Name | Type | Direction | Description |
| pu8TestResult | u8TestResultType\* | OUT | Status of the test |
| **Returned value** | | | |
| Name | Description | | |
| NA | - | | |
| **Dynamic aspect** | | | |
| Server operation  Non Reentrant | | | |
| **Requirements** | | | |
| ARCH\_SW\_PAL\_0376; ARCH\_SW\_PAL\_0377; ARCH\_SW\_PAL\_0410; | | | |

### Called functions

|  |  |  |  |
| --- | --- | --- | --- |
| **Requirements** | **Criteria** | **Linked Runnable** | **Source** |
| ARCH\_SW\_PAL\_0377 | [ATM\_runGetTestResult](#_Ref410741888) shall be called to get the status ot specific tests (inhibiting the execution of the present one if failed). | PAL\_Autotest\_CheckMotorConnection() | ALV\_EXT\_TF\_H\_892; ALV\_EXT\_TF\_H\_893; ALV\_EXT\_TF\_H\_894; |
| ARCH\_SW\_PAL\_0410; | PMP\_runGetMotorVp\_100ms and PMP\_runGetMotorVn\_100ms shall be called to check battery voltage. | PAL\_Autotest\_CheckMotorConnection() |  |

## PAL\_Autotest\_CheckMotorDisengagement

### Definition

|  |  |  |  |
| --- | --- | --- | --- |
| **Prototype** | | | |
| void **PAL\_Autotest\_CheckMotorDisengagement** (u8TestResultType \*pu8TestResult) | | | |
| **Object** | | | |
| This function shall check whether the motor is blocked while performing a releasing mode | | | |
| **Parameters** | | | |
| Name | Type | Direction | Description |
| pu8TestResult | u8TestResultType\* | OUT | Status of the test |
| **Returned value** | | | |
| Name | Description | | |
| NA | - | | |
| **Dynamic aspect** | | | |
| Server operation  Non Reentrant | | | |
| **Requirements** | | | |
| ARCH\_SW\_PAL\_0378; ARCH\_SW\_PAL\_0379; ARCH\_SW\_PAL\_0380; ARCH\_SW\_PAL\_0400; ARCH\_SW\_PAL\_0401; ARCH\_SW\_PAL\_0402; ARCH\_SW\_PAL\_0403; | | | |

### Data flow

|  |  |  |  |
| --- | --- | --- | --- |
| **Requirements** | **Criteria** | **Linked Runnable** | **Source** |
| ARCH\_SW\_PAL\_0378; | The current executed cycle shall be read from BFE | PAL\_Autotest\_CheckMotorDisengagement() | ALV\_EXT\_TF\_H\_965; |

### Called functions

|  |  |  |  |
| --- | --- | --- | --- |
| **Requirements** | **Criteria** | **Linked Runnable** | **Source** |
| ARCH\_SW\_PAL\_0379; | [ATM\_runGetTestResult](#_Ref410741888) shall be called to get the status ot specific tests (inhibiting the execution of the present one if failed). | PAL\_Autotest\_CheckMotorDisengagement() | ALV\_EXT\_TF\_H\_962; |
| ARCH\_SW\_PAL\_0400; | PMP\_runGetFilteredTemperature\_deg shall be called to get the ECU temperature | PAL\_Autotest\_CheckMotorDisengagement() |  |
| ARCH\_SW\_PAL\_0401; | PMP\_runGetBatteryVoltage\_10ms shall be called to get the battery voltage | PAL\_Autotest\_CheckMotorDisengagement() |  |
| ARCH\_SW\_PAL\_0402; | PMP\_runGetMotorResistance shall be called to get motor resistance | PAL\_Autotest\_CheckMotorDisengagement() |  |
| ARCH\_SW\_PAL\_0403; | PAL\_runReadMotorCurrentInmA shall be called to get motor current. | PAL\_Autotest\_CheckMotorDisengagement() |  |

## PAL\_Autotest\_CheckMotorCurrent

### Definition

|  |  |  |  |
| --- | --- | --- | --- |
| **Prototype** | | | |
| void **PAL\_Autotest\_CheckMotorCurrent** (u8TestResultType \*pu8TestResult) | | | |
| **Object** | | | |
| This function shall check the current of the motor | | | |
| **Parameters** | | | |
| Name | Type | Direction | Description |
| pu8TestResult | u8TestResultType\* | OUT | Status of the test |
| **Returned value** | | | |
| Name | Description | | |
| NA | - | | |
| **Dynamic aspect** | | | |
| Server operation  Non Reentrant | | | |
| **Covered requirements** | | | |
| ARCH\_SW\_PAL\_0340; ARCH\_SW\_PAL\_0341; ARCH\_SW\_PAL\_0342; ARCH\_SW\_PAL\_0343; | | | |

### Data flow

|  |  |  |  |
| --- | --- | --- | --- |
| **Requirements** | **Criteria** | **Linked Runnable** | **Source** |
| ARCH\_SW\_PAL\_0343; | The current executed cycle shall be read from BFE | PAL\_Autotest\_CheckMotorCurrent() | ALV\_EXT\_TF\_H\_1023; |

### Called functions

|  |  |  |  |
| --- | --- | --- | --- |
| **Requirements** | **Criteria** | **Linked Runnable** | **Source** |
| ARCH\_SW\_PAL\_0341 | Rte\_BFE\_psrExecutedCycle\_u8CycleNumber shall be called to get the current executed cycle number | PAL\_Autotest\_CheckMotorCurrent () |  |
| ARCH\_SW\_PAL\_0342 | [ATM\_runGetTestResult](#_Ref410741888) shall be called to get the status ot specific tests (inhibiting the execution of the present one if failed). | PAL\_Autotest\_CheckMotorCurrent () |  |

## PAL\_Autotest\_CheckMotorThermalProctection

### Definition

|  |  |  |  |
| --- | --- | --- | --- |
| **Prototype** | | | |
| void **PAL\_Autotest\_CheckMotorThermalProctection** (u8TestResultType \*pu8TestResult) | | | |
| **Object** | | | |
| This function shall check if an over temperature of the power stage is reached by returning a SW self-protection status | | | |
| **Parameters** | | | |
| Name | Type | Direction | Description |
| pu8TestResult | u8TestResultType\* | OUT | Status of the test |
| **Returned value** | | | |
| Name | Description | | |
| NA | - | | |
| **Dynamic aspect** | | | |
| Server operation  Non Reentrant | | | |
| **Covered requirements** | | | |
| ARCH\_SW\_PAL\_0360; ARCH\_SW\_PAL\_0361; **ARCH\_SW\_PAL\_0366;** | | | |

### Called functions

|  |  |  |  |
| --- | --- | --- | --- |
| **Requirements** | **Criteria** | **Linked Runnable** | **Source** |
| ARCH\_SW\_PAL\_0361 | [PMP\_runGetDeficiencyLevel](#_Hlk412209786)function shall be called to get the estimation of the energy stored in the motor. | PAL\_Autotest\_CheckMotorThermalProctection () | ALV\_EXT\_TF\_H\_1361; |

## PAL\_Autotest\_CheckMotorPowerOrder

### Definition

|  |  |  |  |
| --- | --- | --- | --- |
| **Prototype** | | | |
| void **PAL\_Autotest\_CheckMotorPowerOrder** (u8TestResultType \*pu8TestResult) | | | |
| **Object** | | | |
| This function shall check if the PWM order is greater than 2 different thresholds during a specified time | | | |
| **Parameters** | | | |
| Name | Type | Direction | Description |
| pu8TestResult | u8TestResultType\* | OUT | Status of the test |
| **Returned value** | | | |
| Name | Description | | |
| NA | - | | |
| **Dynamic aspect** | | | |
| Server operation  Non Reentrant | | | |
| **Requirements** | | | |
| ARCH\_SW\_PAL\_0381; ARCH\_SW\_PAL\_0382; ARCH\_SW\_PAL\_0390; ARCH\_SW\_PAL\_0391; ARCH\_SW\_PAL\_0392; | | | |

### Data flow

|  |  |  |  |
| --- | --- | --- | --- |
| **Requirements** | **Criteria** | **Linked Runnable** | **Source** |
| ARCH\_SW\_PAL\_0381 | The current executed data from BFE shall be an input | PAL\_Autotest\_CheckMotorPowerOrder() | ALV\_EXT\_TF\_H\_1123; |

### Called functions

|  |  |  |  |
| --- | --- | --- | --- |
| **Requirements** | **Criteria** | **Linked Runnable** | **Source** |
| ARCH\_SW\_PAL\_0390 | The ERH\_runGetAecStatus function shall be called to get the motor Order AEC status | PAL\_Autotest\_CheckMotorPowerOrder() |  |
| ARCH\_SW\_PAL\_0391 | [ATM\_runGetTestResult](#_Ref410741888) shall be called to get the status ot specific tests (inhibiting the execution of the present one if failed). | PAL\_Autotest\_CheckMotorPowerOrder() |  |
| ARCH\_SW\_PAL\_0392 | MMG\_runCheckModeStatus shall be called to get the context status | PAL\_Autotest\_CheckMotorPowerOrder() |  |
| ARCH\_SW\_PAL\_0393 | Rte\_Read\_prrExecutedCycle\_u8CycleNumber shall be called to check if no PRO cycle is running | PAL\_Autotest\_CheckMotorPowerOrder() | ALV\_EXT\_TF\_H\_1120; |

## PAL\_Autotest\_CheckMotorSC

### Definition

|  |  |  |  |
| --- | --- | --- | --- |
| **Prototype** | | | |
| void **PAL\_Autotest\_CheckMotorSC** (u8TestResultType \*pu8TestResult) | | | |
| **Object** | | | |
| This function shall check if the motor is in short circuit during a start-up release cycle | | | |
| **Parameters** | | | |
| Name | Type | Direction | Description |
| pu8TestResult | u8TestResultType\* | OUT | Status of the test |
| **Returned value** | | | |
| Name | Description | | |
| NA | - | | |
| **Dynamic aspect** | | | |
| Server operation  Non Reentrant | | | |
| **Requirements** | | | |
| ARCH\_SW\_PAL\_0383; RCH\_SW\_PAL\_0384; ARCH\_SW\_PAL\_0385; ARCH\_SW\_PAL\_0386; ARCH\_SW\_PAL\_0387; ARCH\_SW\_PAL\_0388; | | | |

### Data flow

|  |  |  |  |
| --- | --- | --- | --- |
| **Requirements** | **Criteria** | **Linked Runnable** | **Source** |
| ARCH\_SW\_PAL\_0383 | The current executed cycle data from BFE shall be an input | PAL\_Autotest\_CheckMotorSC() | ALV\_EXT\_TF\_H\_1188; |

### Called functions

|  |  |  |  |
| --- | --- | --- | --- |
| **Requirements** | **Criteria** | **Linked Runnable** | **Source** |
| ARCH\_SW\_PAL\_0384 | The [PMP\_runGetBatteryVoltage\_10ms](#_Hlk410804423) function shall be called to get the battery voltage. | PAL\_Autotest\_CheckMotorSC() | ALV\_EXT\_TF\_H\_1186; |
| ARCH\_SW\_PAL\_0385 | [ATM\_runGetTestResult](#_Ref410741888) shall be called to get the status ot specific tests (inhibiting the execution of the present one if failed). | PAL\_Autotest\_CheckMotorSC() | ALV\_EXT\_TF\_H\_1185; |
| ARCH\_SW\_PAL\_0386 | PAL\_runReadMotorCurrentInmA shall be called to get the Motor current | PAL\_Autotest\_CheckMotorSC() |  |
| ARCH\_SW\_PAL\_0387 | PMP\_runGetBatteryVoltage\_10ms shall be called to mesure motor SC current | PAL\_Autotest\_CheckMotorSC() |  |
| ARCH\_SW\_PAL\_0388 | PMP\_runGetFilteredTemperature\_deg shall be called to get the filtred value of the ECU temperature | PAL\_Autotest\_CheckMotorSC() |  |
| ARCH\_SW\_PAL\_0389 | PMP\_runGetMotorResistance shall be called to get the value for motor resitance | PAL\_Autotest\_CheckMotorSC() |  |

# MCU resources

The following requirements on resource consumption objectives apply to the module/package:

|  |  |  |  |
| --- | --- | --- | --- |
| **Requirements** | **Criteria** | **Levels/Tolerances** | **Source** |
| ARCH\_SW\_PAL\_9997 | The ROM size consumed by this component shall not exceed 5K bytes. |  |  |
| ARCH\_SW\_PAL\_9998 | The heap size consumed by this component shall be less than 200 bytes. |  |  |