**RESTRICTED ACCESS**

|  |  |  |  |
| --- | --- | --- | --- |
| **DISTRIBUTION:** | **Firm** | **To** | **Copies** |
|  | AEE-C | C. Redon | 2 |
|  | AEE-C | X. Christmann, P. Vincent | 2 |

**PP Platform Software:**

**DESIGN OF BFE COMPONENT (Belt Function Execution)**

OBJECT: This document is the main document of the software design for *BFE*

SUMMARY: The aim of this document is to define the detailed design of the Belt Function Execution software component (BFE). The goal of the BFE is to compute the motor power order that shall be applied on the motor according to the selected cycle identifier, the hardware inputs (voltage/current/belt position/…) and the belt profile defined as post build parameters.

This module executes a part of the technical function G (To drive the motor) defined in the Designed Specification Document (see **[A0]**)

CONCLUSION: This document is applicable for the BFE sub-project available on MKS:

MKSProjects/ err\_generic/ Platform/ FoundationLibrary/ PHASE\_01/ ComponentLibrary/ BFE\_BeltFunctionExecution/ Design

EVOLUTION OF THE DOCUMENT

|  |  |  |  |
| --- | --- | --- | --- |
| **Issue** | **Date** | **Author** | **Motive and nature of the modifications** |
| 000 | 09.01.2012 | CSA | First release |
| 001 | 23.07.2013 | CSA | Traceability added against TF-G and architecture document  Emergency braking sequence preliminary design added for CPL issue  High Power preliminary design added  All others chapters reworked and reorganized in order to match with the TF-G doors module. |
| 002 | 23.07.2013 | CSA | Some COVERS tags updated due to numbering changes in Architecture document.  Bullet list style changed in chapter 2.1 (issue for Reqtify) |
| 003 | 01/08/2013 | CRE | Some modification have been done in the context of the design document review |
| 155 104 | 23/05/2014 | SFL | Create step option "belt payout abortion" for BPA cycles.  Select tensioning temperature adaptation interpolation tables according to cycle option. |
| 160 956 | 25/08/2014 | SFL | Management of motor stage status at the end of emergency braking sequence. |
| 167019 | 09/12/2014 | SFL | Little corrections about new BMM interface. |
| 215620 | 21/08/2015 | SFL | For standard current regulation steps, clamp motor command to zero to avoid change of sign. |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

This document contains **93** pages.

**CONTENTS**

1. Documentation 5

1.1. Upper level relevant documents 5

1.2. Design Specification documents 5

1.3. Other documents 5

1.4. Glossary and definition 5

2. overview 6

2.1. Description of the component 6

2.2. Context diagram 7

3. software interface 8

3.1. Diagrams 8

3.2. Exported functions 9

3.2.1. Function *BFE\_runScheduleStep* 9

3.2.2. Function *BFE\_runExecuteStandardSteps* 11

3.2.3. Function *BFE\_runExecuteHighPowerStep* 13

3.2.4. Function *BFE\_runExecuteEmergencyBrakingSequence* 15

3.2.5. Function *BFE\_runResetMotorStageStatus* 16

3.3. Exported variables 18

3.4. Exported constants 19

3.5. Exported types 20

3.6. Exported macros 21

4. Software description 22

4.1. Main data 22

4.1.1. Internal data used by the sub module in charge of step scheduling (10ms period) 22

4.1.2. BFE\_runScheduleStep main data 27

4.1.3. BFE\_runExecuteSteps main data 29

4.1.4. BFE\_runExecuteHighPowerStep main data 30

4.1.5. BFE\_runExecuteEmergencyBrakingSequence main data 32

4.1.6. Constants 33

4.1.6.1. Step order type values 33

4.1.6.2. Adaptive current interruption point 34

4.1.6.3. Motor current regulation 35

4.1.6.4. Motor current regulation with PID 36

4.1.6.5. Emergency motor braking sequence 38

4.1.6.6. Cycle scheduling 39

4.1.6.7. High Power regulation 42

4.2. Functioning overview 44

4.2.1. Global overview 44

4.2.2. Step scheduling management (10ms) 46

4.2.3. Standards steps management (2ms) 47

4.2.4. Management of steps with High Power regulation (400µs) 49

4.2.5. Management of the emergency motor stage braking and releasing (400µs) 50

4.3. Component functions 52

4.3.1. Functions called during the step scheduling management 52

4.3.1.1. M\_SelectStepId 52

4.3.1.2. M\_ProvideExecutedCycleStep 55

4.3.1.3. M\_ManageStepEvents 57

4.3.1.4. M\_ExtractStepParameters 61

4.3.1.5. M\_ControlStepExecution 66

4.3.1.6. M\_ExtractCycleOptions 68

4.3.2. Functions used to compute the motor command in case of standards steps 72

4.3.2.1. ManagePWMSteps 72

4.3.2.2. ManageCurrentSteps 74

4.3.2.3. ManagePIDCurrentSteps 77

4.3.2.4. ManageVoltageSteps 79

4.3.2.5. ManageNoStep 80

4.3.3. High Power regulation 81

4.3.3.1. Key data initialization 81

4.3.3.2. Battery current computation 83

4.3.3.3. Booster interruption condition evaluation 84

4.3.3.4. Integral action 85

4.3.3.5. Boost stage command and Power stage command elaboration 86

4.3.4. Critical sections to protect data or avoid re-entrant access 88

4.3.4.1. M\_START\_PROTECT\_STEP\_INIT\_DATA 88

4.3.4.2. M\_END\_PROTECT\_STEP\_INIT\_DATA 89

4.3.4.3. M\_START\_PROTECT\_POWER\_ORDER\_UPD 90

4.3.4.4. M\_END\_PROTECT\_POWER\_ORDER\_UPD 91

5. Post Build parameters 92

6. Compilation options 93

6.1. High Power feature key 93

6.2. PID current regulation feature key 93

6.3. Adaptive current interruption point feature key 93

6.4. Belt blocked feature key 93

6.5. Belt Movement feature key 93

7. Annex 94

7.1. Kilis temperature drift issue 94

# Documentation

## Upper level relevant documents

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

|  |  |  |  |
| --- | --- | --- | --- |
| Nb | **Document** | **Reference** | **Company** |
| **[A0]** | DSD – Design Specification Document | E-XXXXXXX | AEE-C |
| **[A1]** | Technical Function G Specification document | Doors | AEE-C |
| **[A2]** | *PP black box Architecture document* | MKS ref | AEE-C |
| **[A3]** | HWS\_ECU\_PP\_HPP\_Boost\_Design\_Module | PLM - E1120062 | AEE-C |

## Design Specification documents

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

|  |  |  |  |
| --- | --- | --- | --- |
| Nb | **Document** | **Reference** | **Company** |
| **[B0]** | PAL actuator design document | MKS | AEFC |
|  |  |  |  |
|  |  |  |  |

## Other documents

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

|  |  |  |  |
| --- | --- | --- | --- |
| Nb | **Document** | **Reference** | **Company** |
| **[C0]** | Post Builds Parameters specification | E-XXXXXXX | AEE-C |
|  |  |  |  |
|  |  |  |  |

## Glossary and definition

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

ADC Analogic-Digital Convertor

ATM Auto Tests Manager

BFS Belt Function Selection

BMM Belt Movement Monitoring

BSR Belt Slack Reduction

COTS Component On The Shell

ECU Electronic Controller Unit

MBD Model Based Design

NVM Non Volatile Memory

NVP Non Volatile Parameters

PAL Power Abstraction Layer

REL Release

SFR Software Function Recovery (i.e. release belt functions)

SW Software

VAL Vehicle Abstraction Layer

# overview

The purpose of this section is to present and define all the services realized by BFE.

## Description of the component

The BFE is used:

* To manage cycles execution
* To schedule and execute steps
* To increment executed cycles counters
* To provide the executed step and cycle and PWM consign for the other application parts

The prefix of the component is “**BFE**”.

BFE is a MBD component which has been designed with a main model that includes some referenced model.

Each referenced model will be generated in his own source/header file, like the main model. But these models (and source files) need to exchange some data so a BFE\_Internal module will contain shared data between all of these modules.

This implementation allows working in parallel on different parts of the BFE.

The BFE is split into many parts:

* An “engine” part that will be provided as a COTS to other projects
* Some configuration files outside of the folder containing the “core” and that shall be adapted by other projects (See Compilation options) to fulfil their needs (a template is available on the platform MKS project).

Static file organization of the BFE component:

\*.C  
\*.H

**BFE « engine » - directly provided by PP platform**

\*.C  
\*.H

\*.C  
\*.H

BFE\_Internal

\*.C  
\*.H

BFE\_BeltFunctionExecution

*(main files with the 3 exported functions)*

BFE\_ManageStepConstraint

BFE\_ProvideStepConfig

\*.C  
\*.H

*BFE sub modules*

BFE\_....

Data flows exchanged

\*.H

BFE\_Config

**BFE configuration to be adapted by project**

Compilation options

\*.C  
\*.H

BFE\_Interfaces

## Context diagram

This diagram describes the main flows between BFE modules and other modules.

The main flows exchanged between BFE and other modules shall be defined at the software architecture level. See the Software architecture document **[A2]** to have a complete and up to date view.

The following diagram is just given as information:

**BFE**

400 µs

Execute High Power steps

Execute Standards steps

ADC

Motor current

Battery current

Battery voltage

Temperature

BMM

Belt pay out

Belt speed

VAL

End of presafe situation

2 ms

Provide cycle & step number

Evaluate step interruption event

Provide:

First order

Increment step value

Manage step timer

10 ms

Post build parameters

Cycle/Steps parameters

Diag/Algo/BFS/…

Executed step

Executed cycle

PAL/Diag/ATM/…

Motor power

BFS

Selected Cycle

Motor resistance / T°C correction factor / etc…

# software interface

## Diagrams

All software interfaces are completely defined in the software architecture document (ref. **[A2]**):

* Data provider
* Data receiver
* Scaling associated to each data element
* Name of the macro used to set/get data elements

The following table is just given as information:

*Inputs:*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Macro | Arguments | Returned value | Data element description | Covered requirement |
| Rte\_Call\_pclVehicleAnalogSignals\_GetBatteryVoltage | (\* uint16) | NA | Battery voltage | TF\_G\_74 |
| Rte\_Call\_pclMotorCurrent\_ReadMotorCurrentInmA | (\* sint32) | NA | Motor current in mA | TF\_G\_84 |
| Rte\_Call\_pclMotorCurrent\_ReadSignedMotorCurrentInA | (\* sint8) | NA | Motor current in A | XXXXXXX |
| Rte\_Call\_pclInternalTemperature\_ReadFilteredInternalTemperature | (\* uint16) | NA | Temperature value | TF\_G\_86 |
| Rte\_Call\_pclBeltDeplSensor\_GetBeltSpeed\_mm\_s | (\* sint32) | NA | Belt speed | TF\_G\_82 |
| Rte\_Call\_pclBeltDeplSensor\_GetBeltPayOut\_mm | (\* sint16) | NA | Belt pay out | TF\_G\_83 |
| Rte\_Read\_prrSelectedCycle\_u8CycleNumber | (\* uint8) | NA | Selected cycle | TF\_G\_19 |
| Rte\_IRead\_BFE\_runScheduleStep\_prrVehAbstract\_b8IsCarAccelLow | NA | uint8 | End of presafe situation | XXXXXXX |
|  |  |  |  |  |

*Outputs:*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Macro | Arguments | Returned value | Data element description | Covered requirement |
| Rte\_Write\_psrExecutedStep\_u8StepNumber | uint8 | NA | Running step | XXXXXXX |
| Rte\_Write\_psrExecutedCycle\_u8CycleNumber | uint8 | NA | Running cycle | XXXXXXX |
| Rte\_Call\_pclMotorPowerOrder\_SetPowerOrder | sint16  boolean | NA  NA | Motor power  High Power activation | TF\_G\_70  TF\_G\_72  XXXXXXX |
|  |  |  |  |  |
|  |  |  |  |  |

## Exported functions

All the following functions are exported by the component.

### Function *BFE\_runScheduleStep*

* Object:

DSG\_BFE\_00001:

The aim of this function is multiple:

It shall monitor periodically the cycle identifier provided by the belt decisions algorithms. And following the state of this input, it shall extract the step parameters from the cycle definition structure (defined as post build parameters) and provide the required data to the BFE sub modules in charge of motor power order computation (see 3.2.2 & 3.2.3).

This function shall evaluate every 10ms the conditions that could lead to the interruption of the current step (based on step parameters) and the switch to the following step defined in the executed cycle parameters.

[COVERS: ARCH\_SW\_BFE\_0010]

* Entering and outgoing parameters:

None parameters.

* Dynamic aspect:

DSG\_BFE\_00002:

This function shall be called every 10ms.

[COVERS: ARCH\_SW\_BFE\_00300, DES\_TF\_G\_844]

* Who:

This function is called into an OS task or a time slot of the scheduler.

* External items used:
* RTE connections:

DSG\_BFE\_00003:

This function shall access with an Explicit Read access to the Selected Cycle Identifier to know which cycle shall be executed.

The following prototype shall be used:

Rte\_Read\_prrSelectedCycle\_u8CycleNumber ( uint8 \* pu8CycleNumber )

[COVERS: ARCH\_SW\_BFE\_0050, ARCH\_SW\_BFE\_0051]

DSG\_BFE\_00004:

This function shall access to the battery voltage level read at ECU connector.

The following prototype shall be used:

Rte\_Call\_pclKL30\_V\_Get ( uint16 \* ou16BatteryVoltage )

[COVERS: ARCH\_SW\_BFE\_0083]

DSG\_BFE\_00005:

This function shall access to the belt pay out measured by bobbin rotation sensors.

The following prototype shall be used:

Rte\_Call\_pclBelt\_D\_GetFromT0\_mm ( sint16 \* ps16BeltPayOut)

[COVERS:]

DSG\_BFE\_00006:

This function shall access to the belt speed computed by bobbin rotation sensors.

The following prototype shall be used:

Rte\_Call\_pseBelt\_S\_GetSpeed\_mm\_s ( sint32 \* ps32BeltSpeed)

[COVERS:]

DSG\_BFE\_00007:

This function shall access to the motor current measured in mA.

The following prototype shall be used:

Rte\_Call\_pclMotor\_I\_Get\_mA ( sint32 \* ps32MotorCurrent )

[COVERS:]

DSG\_BFE\_00008:

This function shall access to the ECU temperature measured by SBC internal probe.

The following prototype shall be used:

Rte\_Call\_pclECU\_T\_GetFiltered ( uint16 \* pu16Temperature )

[COVERS:]

DSG\_BFE\_00009:

This function shall provide to the rest of the application the cycle identifier in progress.

The following prototype shall be used:

Rte\_Write\_psrExecutedCycle\_u8CycleNumber( uint8 u8ExecutedCycle )

[COVERS: ARCH\_SW\_BFE\_0052]

DSG\_BFE\_00010:

This function shall provide to the rest of the application the step identifier in progress.

The following prototype shall be used:

Rte\_Write\_psrExecutedStep\_u8StepNumber( uint8 u8ExecutedStep )

[COVERS:]

DSG\_BFE\_00011:

This function shall access to the Pyrotechnical device activation status.

The following prototype shall be used:

Rte\_Read\_prrXXXXXXXXXXXXXXXXXX( YYYYY )

[COVERS:]

This function shall provide some internal results to the others BFE main functions in charge of the motor power order update. Due to the Autosar project context these data are exchanged by using RTE macros.

DSG\_BFE\_00012:

This function shall provide the first value of a step consign when starting a new step.

The following prototype shall be used:

Rte\_IrvWrite\_BFE\_runScheduleStep\_s32FirstOrderValue( sint32 s32FirstOrderValue )

[COVERS:]

DSG\_BFE\_00013:

This function shall provide the slope value of a step consign when starting a new step.

The following prototype shall be used:

Rte\_IrvWrite\_BFE\_runScheduleStep\_s32OrderIncrementValue ( sint32 s32IncrementValue )

[COVERS:]

DSG\_BFE\_00014:

This function shall provide the step duration when starting a new step (only required for high power steps purpose).

The following prototype shall be used:

Rte\_IrvWrite\_BFE\_runScheduleStep\_u16StepDuration ( uint16 u16StepDuration )

[COVERS:]

DSG\_BFE\_00015:

This function shall provide the regulation type that shall be applied when starting a new step.

The following prototype shall be used:

Rte\_IrvWrite\_BFE\_runScheduleStep\_u8OrderType ( uint8 u8OrderType )

[COVERS:]

* POST BUILD parameters:
* Returned values:

No returned value.

* Constraints:

The data exchanged internally by the BFE module are a “set” of data. Since a part of the BFE is called in an ISR context, there is a risk for this process not to have the complete set of data up to date when the ISR is executed.

DSG\_BFE\_00016:

The coherency of these data shall be ensured to avoid a wrong motor power order regulation for 400 or 800µs.

A critical section shall be created to protect the write access to the following key data:

* Rte\_IrvWrite\_BFE\_runScheduleStep\_s32FirstOrderValue
* Rte\_IrvWrite\_BFE\_runScheduleStep\_s32OrderIncrementValue
* Rte\_IrvWrite\_BFE\_runScheduleStep\_u16StepDuration
* Rte\_IrvWrite\_BFE\_runScheduleStep\_u8OrderType

[COVERS: ARCH\_SW\_BFE\_0206]

### Function *BFE\_runExecuteStandardSteps*

* Object:

DSG\_BFE\_00017:

The aim of this function is to compute the duty cycles consign that shall be applied on the motor stage in case of ‘standard’ steps. The standards steps are those that can be managed every 2ms:

* PWM order type steps
* Voltage order type steps
* Motor current order type steps (with the same regulation than 204 MOPF project)
* PID current order type (based on motor or supply current)

[COVERS: ARCH\_SW\_BFE\_0011]

* Entering and outgoing parameters:

None parameters.

* Dynamic aspect:

DSG\_BFE\_00018:

This function shall be called every 2ms.

[COVERS: ARCH\_SW\_BFE\_00301]

* Who:

This function is called into an OS task or a time slot of the scheduler.

* External items used:

RTE connections:

DSG\_BFE\_00019:

This function shall access to the motor current to perform the standard regulation in current or the PID current regulation

The following prototype shall be used:

Rte\_Call\_pclMotorCurrent\_ReadMotorCurrentInmA ( sint32 \* ps32MotorCurrent )

[COVERS: ARCH\_SW\_BFE\_0102]

DSG\_BFE\_00020:

This function shall access to the battery voltage to manage the steps configured in Voltage.

The following prototype shall be used:

Rte\_Call\_pclVehicleAnalogSignals\_GetBatteryVoltage ( uint16 \* pu16BatteryVoltage )

[COVERS: ARCH\_SW\_BFE\_0104]

DSG\_BFE\_00021:

This function shall access to the duty cycle applied by the power stage on the motor in order to know the rotation direction and to estimate the battery current.

The following prototype shall be used:

sint16 Rte\_IrvRead\_BFE\_runExecuteStandardSteps\_s16MotorPowerOrder( void )

[COVERS:]

DSG\_BFE\_00022:

This function shall access to the order type provided by BFE\_runScheduleStep to know the step type that shall be executed.

The following prototype shall be used:

uint8 Rte\_IrvRead\_BFE\_runExecuteStandardSteps\_u8OrderType( void )

[COVERS:]

DSG\_BFE\_00023:

This function shall access to the step first value provided by BFE\_runScheduleStep to know the reference value that shall be used to start the step regulation.

The following prototype shall be used:

sint32 Rte\_IrvRead\_BFE\_runExecuteStandardSteps\_s32FirstOrderValue ( void )

[COVERS:]

DSG\_BFE\_00024:

This function shall access to the slope value provided by BFE\_runScheduleStep to manage ramp step.

The following prototype shall be used:

sint32 Rte\_IrvRead\_BFE\_runExecuteStandardSteps\_s32OrderIncrementValue ( void )

[COVERS:]

DSG\_BFE\_00025:

This function shall send to the lower layers the consigns that shall be applied on the Half Bridges stage and on the boost stage.

The following prototype shall be used:

Rte\_Call\_pclMotorPowerOrder\_SetPowerOrder ( XXXXXXXXXXXXXXX ) 🡺 TBC with CPL

[COVERS: ARCH\_SW\_BFE\_0107]

DSG\_BFE\_00026:

This function shall access to the Pyrotechnical device activation status.

The following prototype shall be used:

Rte\_Read\_prrXXXXXXXXXXXXXXXXXX( YYYYY )

[COVERS: ARCH\_SW\_BFE\_0106]

POST BUILD parameters:

* Returned values:

No returned value.

* Constraints:

This function is one of the caller of the PAL actuator service in charge of the power stage consigns update. There is a risk of concurrent access to this service due to the BFE function in charge of High Power step or by the BFE function in charge of emergency motor braking and releasing sequence.

DSG\_BFE\_00027:

This function shall only perform his job if the following conditions are fulfilled:

* No High power step in progress
* No Pyro detected as triggered

[COVERS:]

Since the BFE function in charge of the emergency motor braking and the function in charge of Pyro activation detection are both called in the same ISR real time context, there is a risk to detect pyro triggering and to start the emergency motor braking sequence whereas the BFE\_runExecuteSteps is already started and in progress. In this situation the ISR will start the motor braking sequence and just after leaving the ISR context, the BFE\_runExecuteSteps will overwrite the power stage consigns.

DSG\_BFE\_00028:

This function shall only perform the function calls to:

* Rte\_Call\_pclMotorPowerOrder\_SetPowerOrder
* Rte\_IrvWrite\_BFE\_runExecuteStandardSteps\_s16MotorPowerOrder

under a critical section to prevent the call to the ISR in charge of Pyro activation detection (to avoid the situation described above).

[COVERS:]

### Function *BFE\_runExecuteHighPowerStep*

* Object:

DSG\_BFE\_00029:

The aim of this function is to compute the motor power orders that shall be applied on the power stage in case of a steps configured with High Power option.

[COVERS: ARCH\_SW\_BFE\_0012]

* Entering and outgoing parameters:

None parameters.

* Dynamic aspect:

DSG\_BFE\_00030:

This function shall be called every 400µs (at least).

[COVERS: ARCH\_SW\_BFE\_00350]

* Who:

This function shall be called in the ISR context in charge of the capture of ADC channels for the motor current, in order to have the last up to date values for the regulation.

* External items used:

RTE connections:

DSG\_BFE\_00031:

This function shall access to the motor current in order to perform High Power regulation.

The following prototype shall be used:

Rte\_Call\_pclMotorCurrent\_ReadMotorCurrentInmA ( sint32 \* ps32MotorCurrent )

[COVERS: ARCH\_SW\_BFE\_0202]

DSG\_BFE\_00032:

This function shall access to the battery voltage to manage the steps configured in Voltage.

The following prototype shall be used:

Rte\_Call\_pclVehicleAnalogSignals\_GetBatteryVoltage ( uint16 \* pu16BatteryVoltage )

[COVERS: ARCH\_SW\_BFE\_0204]

DSG\_BFE\_00033:

This function shall access to the duty cycle applied by the power stage on the motor in order to know the rotation direction and to estimate the battery current.

The following prototype shall be used:

sint16 Rte\_IrvRead\_BFE\_runExecuteHighPowerStep\_s16MotorPowerOrder( void )

[COVERS:]

DSG\_BFE\_00034:

This function shall access to the order type provided by BFE\_runScheduleStep to know the step type that shall be executed.

The following prototype shall be used:

uint8 Rte\_IrvRead\_BFE\_runExecuteHighPowerStep\_u8OrderType( void )

[COVERS:]

DSG\_BFE\_00035:

This function shall access to the step first value provided by BFE\_runScheduleStep to know the reference value that shall be used to start the step regulation.

The following prototype shall be used:

sint32 Rte\_IrvRead\_BFE\_runExecuteHighPowerStep\_s32FirstOrderValue ( void )

[COVERS:]

DSG\_BFE\_00036:

This function shall access to the step duration provided by BFE\_runScheduleStep to know when the boost stage shall be progressively turned off.

The following prototype shall be used:

Uint16 Rte\_IrvRead\_BFE\_runExecuteHighPowerStep\_u16StepDuration ( void )

[COVERS:]

DSG\_BFE\_00037:

This function shall access to the Pyrotechnical device activation status.

The following prototype shall be used:

Rte\_Read\_prrXXXXXXXXXXXXXXXXXX( YYYYY )

[COVERS:]

DSG\_BFE\_00038:

This function shall send to the lower layers the consigns that shall be applied on the Half Bridges stage and on the boost stage.

The following prototype shall be used:

Rte\_Call\_pclMotorPowerOrder\_SetPowerOrder ( XXXXXXXXXXXXXXX ) 🡺 TBC with CPL

[COVERS: ARCH\_SW\_BFE\_0205]

POST BUILD parameters:

None

* Returned values:

No returned value.

* Constraints:

DSG\_BFE\_00039:

This function shall only perform his job if the following conditions are fulfilled:

* High power step in progress
* No Pyro detected as triggered

[COVERS:]

### Function *BFE\_runExecuteEmergencyBrakingSequence*

* Object:

DSG\_BFE\_00040:

The aim of this function is to brake the motor rotation when it is in tensioning and then to reverse the motor rotation in releasing direction as quickly as possible.

[COVERS: DES\_TF\_G\_1413]

DSG\_BFE\_01000:

In addition to that, this function will deactivate the belt function execution at the first call.

It will prevent from concurrent access to the HW power stage.

[COVERS: DES\_TF\_G\_1342, DES\_TF\_G\_1343]

* Entering and outgoing parameters:

None parameters.

* Dynamic aspect:

DSG\_BFE\_00041:

This function shall be called every 400µs.

[COVERS: DES\_TF\_G\_1362]

* Who:

This function shall be called in the ISR context in charge of the capture of ADC channels for the motor current, in order to have the last up to date values for the regulation.

* External items used:

*RTE connections:*

DSG\_BFE\_00042:

This function shall access to the motor current in order to update the power stage duty cycle and avoid current peaks.

The following prototype shall be used:

Rte\_Call\_pclMotorCurrent\_ReadMotorCurrentInmA ( sint32 \* ps32MotorCurrent )

[COVERS: DES\_TF\_G\_1345, DES\_TF\_G\_1346]

DSG\_BFE\_00043:

This function shall access to the Pyrotechnical device activation status.

The following prototype shall be used:

Rte\_Read\_prrXXXXXXXXXXXXXXXXXX( YYYYY )

[COVERS: DES\_TF\_G\_1342, DES\_TF\_G\_1343]

DSG\_BFE\_00044:

This function shall access to the duty cycle applied by the power stage on the motor in order to know the rotation direction and to estimate the battery current.

The following prototype shall be used:

sint16 Rte\_IrvRead\_BFE\_runExecuteEmergencyBrakingSequence\_s16MotorPowerOrder

[COVERS: DES\_TF\_G\_1347, DES\_TF\_G\_1348]

DSG\_BFE\_00045:

This function shall send to the lower layers the consigns that shall be applied on the Half Bridges stage and on the boost stage.

The following prototype shall be used:

Rte\_Call\_pclMotorPowerOrder\_SetPowerOrder (s16MotorPowerOrder, b8HighPowerStepSet, u16BoostDutyCycle)

[COVERS: DES\_TF\_G\_1353, DES\_TF\_G\_1354, DES\_TF\_G\_1357, DES\_TF\_G\_1358]

DSG\_BFE\_01001:

This function shall have to update the “motor stage status” (BFE internal runnable variable) to deactivate the belt function execution.

The following prototype shall be used:

uint8 Rte\_IrvRead\_BFE\_runExecuteEmergencyBrakingSequence\_u8MotorStageStatus

uint8 Rte\_IrvWrite\_BFE\_runExecuteEmergencyBrakingSequence\_u8MotorStageStatus

[COVERS: DES\_TF\_G\_1342, DES\_TF\_G\_1343]

*POST BUILD parameters:*

2 look-up tables are involved in this motor command regulation. A look up table is defined by 2 arrays: the array for X axis and the array for coefficient that shall be applied. A linear interpolation will be performed between each points defined in this look up table. The X array shall be filled from the lower value to the higher value.

First look up table is defined to adapt the consign in function of the motor current measured:

DSG\_BFE\_00046:

This function shall access to the parameter: NVP\_au16\_PyroSequ\_MotorCurrent\_Xpoints

This size of this table is 3 (size not configurable).

[COVERS: DES\_TF\_G\_1371]

DSG\_BFE\_00047:

This function shall access to the parameter: NVP\_au8\_PyroSequ\_MotorCurrent\_Ypoints

This size of this table is 3 (size not configurable).

[COVERS: DES\_TF\_G\_1371]

Second look up table is defined to adapt the consign in function of the battery current estimated:

DSG\_BFE\_00048:

This function shall access to the parameter: NVP\_au16\_PyroSequ\_BattCurrent\_Xpoints

This size of this table is 3 (size not configurable).

[COVERS: DES\_TF\_G\_1371]

DSG\_BFE\_00049:

This function shall access to the parameter: NVP\_au8\_PyroSequ\_BattCurrent\_Ypoints

This size of this table is 3 (size not configurable).

[COVERS: DES\_TF\_G\_1371]

* Returned values:

No returned value.

* Constraints:

DSG\_BFE\_00050:

This function shall only perform his job if the following conditions are fulfilled:

* Pyro detected as triggered

[COVERS: DES\_TF\_G\_1363, DES\_TF\_G\_1364]

### Function *BFE\_runResetMotorStageStatus*

* Object:

DSG\_BFE\_00175:

The aim of this function is to reset the “motor stage status” (BFE internal runnable variable).

[COVERS:]

* Entering and outgoing parameters:

No parameters.

* Dynamic aspect:

None.

* Who:

This function shall be called in the ISR context in charge of the capture of ADC channels for the motor current, in order to be in the same context as *ExecuteEmergencyBrakingSequence* and *ExecuteHighPowerStep* BFE runnables.

* External items used:

None.

* Returned values:

No returned value.

* Constraints:

DSG\_BFE\_00176:

This function shall only perform his job if the following conditions are fulfilled:

* Once, when pyro status becomes inactive, after being activated during emergency braking sequence.

[COVERS:]

## Exported variables

All the following variables are exported by the component.

Since this component shall be AUTOSAR compliant, there are no exported variables.

|  |  |  |
| --- | --- | --- |
| **Variable name** | | |
|  | Type |  |
|  | Object |  |
|  | Unit |  |
|  | Range |  |
|  | Safety |  |
|  | Constraint |  |

## Exported constants

All the following constants are exported by the component.

Since this component shall be AUTOSAR compliant, there are no exported constants.

|  |  |  |
| --- | --- | --- |
| **Constant name** | | |
|  | Type |  |
|  | Object |  |
|  | Safety |  |
|  | Constraint |  |

## Exported types

All the following types are exported by the component.

Since this component shall be AUTOSAR compliant, there are no exported types.

|  |  |  |
| --- | --- | --- |
| **Type name** | | |
|  | Object |  |
|  | Safety |  |
|  | Constraint |  |

## Exported macros

All the following macros are exported by the component.

Since this component shall be AUTOSAR compliant, there are no exported macros.

|  |  |  |
| --- | --- | --- |
| **Macro name** | | |
|  | Type |  |
|  | Object |  |
|  | Unit |  |
|  | Range |  |
|  | Safety |  |
|  | Constraint |  |

# Software description

## Main data

### Internal data used by the sub module in charge of step scheduling (10ms period)

All following data are private to the BFE component. They are defined into the BFE\_Internal file and are available for all BFE sub-modules. All of these data are read/write by the 10ms periodic sub modules in the same task context.

DSG\_BFE\_00051:

[COVERS: ]

|  |  |  |
| --- | --- | --- |
| **stStepOrderOptions** | | |
|  | Type | Structure stStepOrderOptions\_Type  {  uint8 u8OrderType;  sint8 s8OrderValue;  uint16 u16StepDuration;  boolean bReleaseControlled;  boolean bRampStep;  boolean bWeightedStep;  boolean bPIDMotor;  boolean bPIDSupply;  boolean bHighPowerStep;  boolean bPowerDegradation;  } |
|  | Object | Contains current executed step parameters extracted from step library (Post build parameters) that will be used to define the increment step, the first order, and the order type when starting a new step. |
|  | Unit | u8OrderType:   * 0x00: PWM * 0x01: Current * 0x04: Voltage * All other values: No order type   s8OrderValue: depends of the u8OrderType value   * in case of PWM: 1 lsb = 1% * in case of voltage: 1 lsb = 0.25V * in case of current: 1 lsb = 0.5A   u16StepDuration: 1 lsb = 10 ms  bReleaseControlled: Boolean  bRampStep: Boolean  bWeightedStep: Boolean  bPIDMotor: Boolean  bPIDSupply: Boolean  bHighPowerStep: Boolean  bPowerDegradation: Boolean |
|  | Range | u8OrderType: [ 0x00 ; 0xFF ]  s8OrderValue: depends of the u8OrderType value   * in case of PWM: [ -100% ; 100% ] * in case of voltage: [ -32V ; 31.75V ] * in case of current: [ -47A ; 47A ] * in other cases: NA   u16StepDuration: [ 0x0000 ; 0xFFFF ]  bReleaseControlled: Boolean  bRampStep: Boolean  bWeightedStep: Boolean  bPIDMotor: Boolean  bPIDSupply: Boolean  bHighPowerStep: Boolean  bPowerDegradation: Boolean |
|  | Safety | NA |
|  | Constraint | This data is shared between several process handled by the BFE. The data integrity is ensured because all of these process are called in the same task context. |

DSG\_BFE\_00052:

[COVERS: ]

|  |  |  |
| --- | --- | --- |
| **stStepConstraints** | | |
|  | Type | Structure stOrderConstraints\_Type  {  uint8 u8OrderType;  sint8 s8OrderValue;  uint16 u16StepDuration;  boolean bTriggOffControlled;  boolean bMotorBlockedControlled;  boolean bOrderControlled;  boolean bBeltBlockedControlled;  uint8 u8AdaptCurrentLimit;  boolean bIsBeltMvtControlled;  boolean bBeltPayoutAbortion2;  boolean bBeltPayoutAbortion1;  } |
|  | Object | Contains current executed step parameters extracted from step library (Post build parameters) that will be used to evaluate the end of the current step execution. |
|  | Unit | u8OrderType:   * 0x00: PWM * 0x01: Current * 0x04: Voltage * All other values: No order type   s8OrderValue: depends of the u8OrderType value   * in case of PWM: 1 lsb = 1% * in case of voltage: 1 lsb = 0.25V * in case of current: 1 lsb = 0.5A   u16StepDuration: 1 lsb = 10 ms  bTriggOffControlled: Boolean  bMotorBlockedControlled: Boolean  bOrderControlled: Boolean  bBeltBlockedControlled: Boolean  u8AdaptCurrentLimit: NA  bIsBeltMvtControlled: Boolean  bBeltPayoutAbortion2 : Boolean  bBeltPayoutAbortion1: Boolean |
|  | Range | u8OrderType: [ 0x00 ; 0xFF ]  s8OrderValue: depends of the u8OrderType value   * in case of PWM: [ -100% ; 100% ] * in case of voltage: [ -32V ; 31.75V ] * in case of current: [ -47A ; 47A ] * in other cases: NA   u16StepDuration: [ 0x0000 ; 0xFFFF ]  bTriggOffControlled: Boolean  bMotorBlockedControlled: Boolean  bOrderControlled: Boolean  bBeltBlockedControlled: Boolean  u8AdaptCurrentLimit: [ 0 ; 63 ]  bIsBeltMvtControlled: Boolean  [bBeltPayoutAbortion2 ; bBeltPayoutAbortion1] :   * “00” = option not available * “01” = abortion of step if belt payout ≤ 1st threshold *NVP\_u16BeltPayoutAbortThrs1* * “10” = abortion of step if belt payout ≤ 2nd threshold *NVP\_u16BeltPayoutAbortThrs2* * “11” = option not available |
|  | Safety | NA |
|  | Constraint | This data is shared between several process handled by the BFE. The data integrity is ensured because all of these process are called in the same task context. |

DSG\_BFE\_00053:

[COVERS: ]

|  |  |  |
| --- | --- | --- |
| **u8IsEndOfPresafeSituation** | | |
|  | Type | Unsigned 8 bits |
|  | Object | Contains the end of presage situation information |
|  | Unit | NA |
|  | Range | 0x55: Presafe situation active  0xAA: Presafe situation finished |
|  | Safety | NA |
|  | Constraint | NA |

|  |  |  |
| --- | --- | --- |
| **s16MotorPower** | | |
|  | Type | Signed 16 bits |
|  | Object | Motor power to apply |
|  | Unit | % PWM |
|  | Range | [ -6400 ; 6912 ] |
|  | Safety | NA |
|  | Constraint | The full range for Half bridge power stage is [ -6400 ; 6400 ] (equivalent to [ -100 ; 100]  When the value is between 6400 and 6912 it means that the High power stage is activated. |

DSG\_BFE\_00054:

[COVERS: ]

|  |  |  |
| --- | --- | --- |
| **bIsTimeElapsed** | | |
|  | Type | Boolean |
|  | Object | Data used to know if the current executed step has reached his time out. |
|  | Unit | NA |
|  | Range | NA |
|  | Safety | NA |
|  | Constraint | NA |

DSG\_BFE\_00055:

[COVERS: ]

|  |  |  |
| --- | --- | --- |
| **bStartNewStep** | | |
|  | Type | Boolean |
|  | Object | Data used to know if a new step begins in this time slot. |
|  | Unit | NA |
|  | Range | NA |
|  | Safety | NA |
|  | Constraint | NA |

DSG\_BFE\_00056:

[COVERS: ]

|  |  |  |
| --- | --- | --- |
| **bIsRunningStep** | | |
|  | Type | Boolean |
|  | Object | Data used to know if a step is executed during this time slot. |
|  | Unit | NA |
|  | Range | NA |
|  | Safety | NA |
|  | Constraint | NA |

DSG\_BFE\_00057:

[COVERS: ]

|  |  |  |
| --- | --- | --- |
| **u16StepDuration** | | |
|  | Type | Unsigned 16 bits |
|  | Object | Contains the step duration of the current executed step. |
|  | Unit | 1lsb = 10 ms |
|  | Range | [ 0x0000 ; 0xFFFF] |
|  | Safety | NA |
|  | Constraint | NA |

DSG\_BFE\_00058:

[COVERS: ]

|  |  |  |
| --- | --- | --- |
| **bIntStepFlag** | | |
|  | Type | Boolean |
|  | Object | To know if the current executed step shall be interrupted at the next 10ms time slot. |
|  | Unit | NA |
|  | Range | NA |
|  | Safety | NA |
|  | Constraint | NA |

DSG\_BFE\_00059:

[COVERS: ]

|  |  |  |
| --- | --- | --- |
| **u16InternalSensorTemperature** | | |
|  | Type | Unsigned 16 bits |
|  | Object | ADC temperature value provided by SBC sensor |
|  | Unit | 1 lsb = 0.2325°C and offset = -53°C |
|  | Range | [ -53°C ; 185°C ] ( [ 0 lsb ; 1023lsb ] ) |
|  | Safety | NA |
|  | Constraint | NA |

DSG\_BFE\_00060:

[COVERS: ]

|  |  |  |
| --- | --- | --- |
| **u16BatteryVoltage** | | |
|  | Type | Unsigned 16 bits |
|  | Object | Battery voltage obtained from ADC register in mV |
|  | Unit | 1 lsb = 1mV |
|  | Range | [ 0x0000: 0xFFFF ] |
|  | Safety | NA |
|  | Constraint | NA |

DSG\_BFE\_00061:

[COVERS: ]

|  |  |  |
| --- | --- | --- |
| **s32MotorCurrent** | | |
|  | Type | Signed 32 bits |
|  | Object | Signed motor current in mA |
|  | Unit | 1 lsb = 1mA |
|  | Range | Full range |
|  | Safety | NA |
|  | Constraint | NA |

DSG\_BFE\_00062:

[COVERS: ]

|  |  |  |
| --- | --- | --- |
| **s32BeltSpeed** | | |
|  | Type | Signed 32 bits |
|  | Object | Signed belt speed |
|  | Unit | 1 lsb = 1 °/s |
|  | Range | Full range |
|  | Safety | NA |
|  | Constraint | NA |

DSG\_BFE\_01002:

[COVERS: ]

|  |  |  |
| --- | --- | --- |
| **u8MotorStageStatus** | | |
|  | Type | unsigned 8bits |
|  | Object | Contains the status of the motor stage |
|  | Unit | NA |
|  | Range | KU8\_MOTOR\_STAGE\_ACTIVATION: belt functions execution is allowed KU8\_MOTOR\_STAGE\_EMERGENCY\_DEACTIVATION: motor braking sequence started 🡪 belt function is no more allowed |
|  | Safety | NA |
|  | Constraint | NA |

### BFE\_runScheduleStep main data

DSG\_BFE\_00063:

[COVERS: DES\_TF\_G\_836]

|  |  |  |
| --- | --- | --- |
| **bPreviousEndStep** | | |
|  | Type | Boolean |
|  | Object | This data is the flag which states if the current executed step is finished and that leads to the next step execution or to the end of belt function execution (if no more steps are defined in the belt function parameters) |
|  | Unit | NA |
|  | Range | TRUE ; FALSE |
|  | Safety | NA |
|  | Constraint | Static data  This data is the memory of the ‘end of step status’ computed 10ms before by XXXXXX |

DSG\_BFE\_00064:

[COVERS: DES\_TF\_G\_838]

|  |  |  |
| --- | --- | --- |
| **u8PreviousExecutedStep** | | |
|  | Type | Unsigned 8 bits |
|  | Object | This data is the previous executed step that has been provided by the BFE to the rest of the application 10ms before. |
|  | Unit | NA |
|  | Range | [ 0 ; 7 ] U KU8\_UNDEFINED\_STEP |
|  | Safety | NA |
|  | Constraint | Static data  Init value = KU8\_UNDEFINED\_STEP |

DSG\_BFE\_00065:

[COVERS: DES\_TF\_G\_821]

|  |  |  |
| --- | --- | --- |
| **u8PreviousSelectedCycle** | | |
|  | Type | Unsigned 8 bits |
|  | Object | This data is the previous selected belt function that has been provided to the BFE by the BFS component 10ms before. |
|  | Unit | NA |
|  | Range | [ 0 ; KU8\_NB\_MAX\_BELT\_FCT\_LIBRARY [ U KU8\_NO\_BELT\_FUNCTION |
|  | Safety | NA |
|  | Constraint | Static data  Init value = KU8\_NO\_BELT\_FUNCTION |

DSG\_BFE\_00066:

[COVERS:]

|  |  |  |
| --- | --- | --- |
| **u8PowerDegradationFactor** | | |
|  | Type | Unsigned 8 bits |
|  | Object | This data contains the power degradation factor that shall be applied during belt function execution. |
|  | Unit | 1 lsb = 1/128 % (i.e. 128 lsb = 100% 🡺 no degradation applied on step consigns) |
|  | Range | [0 ; 128] |
|  | Safety | NA |
|  | Constraint | NA |

DSG\_BFE\_00067:

[COVERS:]

|  |  |  |
| --- | --- | --- |
| **eTemperatureCompensationType** | | |
|  | Type | Unsigned 8 bits |
|  | Object | This data contains the compensation type that shall be applied during step execution if they are configured with the “Weighted Step Options” |
|  | Unit | NA |
|  | Range | [0 ; 3] |
|  | Safety | NA |
|  | Constraint | NA |

### BFE\_runExecuteSteps main data

DSG\_BFE\_00068:

[COVERS:]

|  |  |  |
| --- | --- | --- |
| **s32PreviousFirstOrder2ms** | | |
|  | Type | Signed 32 bits data |
|  | Object | This data is the memory of the ‘step first order’ read every 2ms by the Function *BFE\_runExecuteSteps*. It will be used to detect a step starting. |
|  | Unit | NA |
|  | Range | Full signed 32 bits data range |
|  | Safety | NA |
|  | Constraint | Static data  This data is the memory of the ‘step first order’ read every 2ms by the Function *BFE\_runExecuteSteps*. |

DSG\_BFE\_00069:

[COVERS:]

|  |  |  |
| --- | --- | --- |
| **s32PreviousSlopeValue2ms** | | |
|  | Type | Signed 32 bits data |
|  | Object | This data is the memory of the ‘step slope value’ read every 2ms by the Function *BFE\_runExecuteSteps*. It will be used to detect a step starting. |
|  | Unit | NA |
|  | Range | Full range |
|  | Safety | NA |
|  | Constraint | Static data  This data is the memory of the ‘step slope value’ read every 2ms by the Function *BFE\_runExecuteSteps*. |

DSG\_BFE\_00070:

[COVERS:]

|  |  |  |
| --- | --- | --- |
| **s32PreviousStepTargetToReach** | | |
|  | Type | Signed 32 bits data |
|  | Object | This data contains the step consigns calculated during the previous call of the Function *BFE\_runExecuteSteps*  It will be used to compute the new step consign (only useful in case of a ramp step) |
|  | Unit | NA |
|  | Range | Full range |
|  | Safety | NA |
|  | Constraint | Static data |

### BFE\_runExecuteHighPowerStep main data

DSG\_BFE\_00071:

[COVERS:]

|  |  |  |
| --- | --- | --- |
| **s16CurrentConsign** | | |
|  | Type | Signed 16 bits data |
|  | Object | This data will contain the current consign value after rescaling in order to perform the high power regulation. |
|  | Unit | 1lsb = 10mA |
|  | Range | [0; 4000] (typical range since the upper limit for a High Power regulation is 40A) |
|  | Safety | NA |
|  | Constraint | Local data |

DSG\_BFE\_00072:

[COVERS:]

|  |  |  |
| --- | --- | --- |
| **u16BoostDurationTimer** | | |
|  | Type | Unsigned 16 bits data |
|  | Object | This data will contain the time elapsed since the beginning of the current executed High Power step. |
|  | Unit | 1lsb = 400µs |
|  | Range | NA |
|  | Safety | NA |
|  | Constraint | Local data  This data shall be a 16bits data length since the maximum duration for a High Power step is 500ms, so the counter can reach the value 1250. |

DSG\_BFE\_00073:

[COVERS:]

|  |  |  |
| --- | --- | --- |
| **u32SwitchPrevInteger** | | |
|  | Type | Unsigned 32 bits data |
|  | Object | Contains the value of the current integer that will be used to perform the Kilis drift compensation |
|  | Unit | 1lsb = 10mA |
|  | Range | NA |
|  | Safety | NA |
|  | Constraint | Local data |

DSG\_BFE\_00074:

[COVERS:]

|  |  |  |
| --- | --- | --- |
| **s16CorrectedMotorCurrent** | | |
|  | Type | Signed 16 bits data |
|  | Object | Contains the motor current value after Kilis drift correction. |
|  | Unit | 1lsb = 10mA |
|  | Range | NA |
|  | Safety | NA |
|  | Constraint | Local data |

DSG\_BFE\_00075:

[COVERS:]

|  |  |  |
| --- | --- | --- |
| **s16SupplyCurrent** | | |
|  | Type | Signed 16 bits data |
|  | Object | Contains the battery current value. |
|  | Unit | 1lsb = 10mA |
|  | Range | NA |
|  | Safety | NA |
|  | Constraint | Local data |

### BFE\_runExecuteEmergencyBrakingSequence main data

DSG\_BFE\_00076:

[COVERS: ]

|  |  |  |
| --- | --- | --- |
| **u8CounterEmergencySeq** | | |
|  | Type | unsigned 8 bits |
|  | Object | This counter is the timer used to know what step of the sequence shall be executed |
|  | Unit | 1lsb = 400µs |
|  | Range | [ 0 ; 255] |
|  | Safety | NA |
|  | Constraint | Static data |

DSG\_BFE\_00077:

[COVERS: DES\_TF\_G\_1370]

|  |  |  |
| --- | --- | --- |
| **u8AlphaFactorEmergencySeq** | | |
|  | Type | unsigned 8 bits |
|  | Object | Correction factor applied on the -100% nominal consign to limit over current on the motor |
|  | Unit | 1lsb = 100/64 % |
|  | Range | [ 0 ; 64] |
|  | Safety | NA |
|  | Constraint | Local data |

DSG\_BFE\_00078:

[COVERS: DES\_TF\_G\_1370]

|  |  |  |
| --- | --- | --- |
| **u8BetaFactorEmergencySeq** | | |
|  | Type | unsigned 8 bits |
|  | Object | Correction factor applied on the -100% nominal consign to limit over current on the battery |
|  | Unit | 1lsb = 100/64 % |
|  | Range | [ 0 ; 64] |
|  | Safety | NA |
|  | Constraint | Local data |

### Constants

#### Step order type values

Order type byte layout:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|  | HP | PID2 | PID1 |  |  |  |  |
|  |  |  |  |  | Order type (%, V, A) | | |

The following macro shall be used to access to these configuration bits:

DSG\_BFE\_00079:

[COVERS: ]

|  |  |  |
| --- | --- | --- |
| **KU8\_ORDER\_PWM** | | |
|  | Type | Unsigned 8 bits |
|  | Object | Value taken by Order type variable in case of step configured with PWM consign:   * u8WOrderType\_ScheduleStep * u8ROrderType\_StdStep * u8ROrderType\_HPStep |
|  | Safety | NA |
|  | Constraint | NA |
|  | Value | 0 |

DSG\_BFE\_00080:

[COVERS: ]

|  |  |  |
| --- | --- | --- |
| **KU8\_ORDER\_CURRENT** | | |
|  | Type | Unsigned 8 bits |
|  | Object | Value taken by Order type variable in case of step configured with current consign:   * u8WOrderType\_ScheduleStep * u8ROrderType\_StdStep * u8ROrderType\_HPStep |
|  | Safety | NA |
|  | Constraint | NA |
|  | Value | 1 |

DSG\_BFE\_00081:

[COVERS: ]

|  |  |  |
| --- | --- | --- |
| **KU8\_ORDER\_VOLTAGE** | | |
|  | Type | Unsigned 8 bits |
|  | Object | Value taken by Order type variable in case of step configured with voltage consign:   * u8WOrderType\_ScheduleStep * u8ROrderType\_StdStep * u8ROrderType\_HPStep |
|  | Safety | NA |
|  | Constraint | NA |
|  | Value | 4 |

DSG\_BFE\_00082:

[COVERS: ]

|  |  |  |
| --- | --- | --- |
| **KU8\_ORDER\_NO\_STEP** | | |
|  | Type | Unsigned 8 bits |
|  | Object | Additional bit set in order type variable in case of no step execution:   * u8WOrderType\_ScheduleStep * u8ROrderType\_StdStep * u8ROrderType\_HPStep |
|  | Safety | NA |
|  | Constraint | NA |
|  | Value | 7 |

DSG\_BFE\_00083:

[COVERS: ]

|  |  |  |
| --- | --- | --- |
| **KU8\_ORDER\_PID\_MOTOR** | | |
|  | Type | Unsigned 8 bits |
|  | Object | Additional bit set in order type variable in case of step configured with PID on motor current:   * u8WOrderType\_ScheduleStep * u8ROrderType\_StdStep * u8ROrderType\_HPStep   !!! This bit will only be taken into account if step is configured with current order !!! |
|  | Safety | NA |
|  | Constraint | NA |
|  | Value | 32 |

DSG\_BFE\_00084:

[COVERS: ]

|  |  |  |
| --- | --- | --- |
| **KU8\_ORDER\_PID\_SUPPLY** | | |
|  | Type | Unsigned 8 bits |
|  | Object | Additional bit set in order type variable in case of step configured with PID on battery current:   * u8WOrderType\_ScheduleStep * u8ROrderType\_StdStep * u8ROrderType\_HPStep   !!! This bit will only be taken into account if step is configured with current order !!! |
|  | Safety | NA |
|  | Constraint | NA |
|  | Value | 16 |

DSG\_BFE\_00085:

[COVERS: ]

|  |  |  |
| --- | --- | --- |
| **KU8\_ORDER\_HIGH\_POWER** | | |
|  | Type | Unsigned 8 bits |
|  | Object | Additional bit set in order type variable in case of step configured with High power option:   * u8WOrderType\_ScheduleStep * u8ROrderType\_StdStep * u8ROrderType\_HPStep   !!! This bit will only be taken into account if step is configured with current order !!! |
|  | Safety | NA |
|  | Constraint | NA |
|  | Value | 64 |

#### Adaptive current interruption point

DSG\_BFE\_00086:

[COVERS:]

|  |  |  |
| --- | --- | --- |
| **KAU16\_ADAPTATIVE\_CURRENT\_LIMIT** | | |
|  | Type | Unsigned 16 bits array (64 elements) |
|  | Object | Array containing the current interruption point values available for this feature. |
|  | Safety | NA |
|  | Constraint | NA |
|  | Value | TBD |
|  | Unit | 1lsb = 0.1 A |

#### Motor current regulation

DSG\_BFE\_00087:

[COVERS: DES\_TF\_G\_424]

|  |  |  |
| --- | --- | --- |
| **CAL\_KU16\_HIGH\_OFFSET** | | |
|  | Type | Unsigned 16 bits |
|  | Object | Contains the high offset value to apply on the motor power order (used to perform the slow current regulation) in function of the difference between the step consign and the motor current |
|  | Safety | NA |
|  | Constraint | This value shall be defined in the BFE configuration file |
|  | Value | 320 |
|  | Unit | 64lsb = 1% |

DSG\_BFE\_00088:

[COVERS: DES\_TF\_G\_424]

|  |  |  |
| --- | --- | --- |
| **CAL\_KU16\_LOW\_OFFSET** | | |
|  | Type | Unsigned 16 bits |
|  | Object | Contains the low offset value to apply on the motor power order (used to perform the slow current regulation) in function of the difference between the step consign and the motor current |
|  | Safety | NA |
|  | Constraint | This value shall be defined in the BFE configuration file |
|  | Value | 64 |
|  | Unit | 64lsb = 1% |

DSG\_BFE\_00089:

[COVERS: DES\_TF\_G\_424]

|  |  |  |
| --- | --- | --- |
| **CAL\_KU16\_HIGH\_THRS** | | |
|  | Type | Unsigned 16 bits |
|  | Object | Contains the high delta threshold used to decide if the low or high offset shall be applied on the motor power order during a slow current regulation |
|  | Safety | NA |
|  | Constraint | This value shall be defined in the BFE configuration file |
|  | Value | 2500 (2.5A) |
|  | Unit | 1lsb = 1mA |

DSG\_BFE\_00090:

[COVERS: DES\_TF\_G\_424]

|  |  |  |
| --- | --- | --- |
| **CAL\_KU16\_LOW\_THRS** | | |
|  | Type | Unsigned 16 bits |
|  | Object | Contains the low delta threshold used to decide if the low or no offset shall be applied on the motor power order during a slow current regulation |
|  | Safety | NA |
|  | Constraint | This value shall be defined in the BFE configuration file |
|  | Value | 500 (500mA) |
|  | Unit | 1lsb = 1mA |

DSG\_BFE\_00091:

[COVERS: DES\_TF\_G\_427]

|  |  |  |
| --- | --- | --- |
| **CAL\_KU8\_MAX\_CURRENT\_VALUE** | | |
|  | Type | Unsigned 8 bits |
|  | Object | Contains the maximum acceptable current on the motor |
|  | Safety | NA |
|  | Constraint | This value shall be defined in the BFE configuration file |
|  | Value | 80 (means 40A) |
|  | Unit | 1lsb = 0.5A |

#### Motor current regulation with PID

DSG\_BFE\_00092:

[COVERS: DES\_TF\_G\_432]

|  |  |  |
| --- | --- | --- |
| **KU8\_PID\_KP\_MOTOR** | | |
|  | Type | Unsigned 8 bits |
|  | Object | Contains the Kp factor used for proportional control in case of PID regulation based on motor current. |
|  | Safety | NA |
|  | Constraint | This value shall be defined as post build parameters |
|  | Value | 5 |

DSG\_BFE\_00093:

[COVERS: DES\_TF\_G\_432]

|  |  |  |
| --- | --- | --- |
| **KU8\_PID\_KI\_MOTOR** | | |
|  | Type | Unsigned 8 bits |
|  | Object | Contains the Kp factor used for integral control in case of PID regulation based on motor current. |
|  | Safety | NA |
|  | Constraint | This value shall be defined as post build parameters |
|  | Value | 27 |

DSG\_BFE\_00094:

[COVERS: DES\_TF\_G\_432]

|  |  |  |
| --- | --- | --- |
| **KU8\_PID\_KD\_MOTOR** | | |
|  | Type | Unsigned 8 bits |
|  | Object | Contains the Kp factor used for derivative control in case of PID regulation based on motor current. |
|  | Safety | NA |
|  | Constraint | This value shall be defined as post build parameters |
|  | Value | 0 |

DSG\_BFE\_00095:

[COVERS: DES\_TF\_G\_432]

|  |  |  |
| --- | --- | --- |
| **KU8\_PID\_KP\_SUPPLY** | | |
|  | Type | Unsigned 8 bits |
|  | Object | Contains the Kp factor used for proportional control in case of PID regulation based on battery current. |
|  | Safety | NA |
|  | Constraint | This value shall be defined as post build parameters |
|  | Value | 5 |

DSG\_BFE\_00096:

[COVERS: DES\_TF\_G\_432]

|  |  |  |
| --- | --- | --- |
| **KU8\_PID\_KI\_SUPPLY** | | |
|  | Type | Unsigned 8 bits |
|  | Object | Contains the Kp factor used for integral control in case of PID regulation based on battery current. |
|  | Safety | NA |
|  | Constraint | This value shall be defined as post build parameters |
|  | Value | 10 |

DSG\_BFE\_00097:

[COVERS: DES\_TF\_G\_432]

|  |  |  |
| --- | --- | --- |
| **KU8\_PID\_KD\_SUPPLY** | | |
|  | Type | Unsigned 8 bits |
|  | Object | Contains the Kp factor used for derivative control in case of PID regulation based on battery current. |
|  | Safety | NA |
|  | Constraint | This value shall be defined as post build parameters |
|  | Value | 0 |

DSG\_BFE\_00098:

[COVERS: DES\_TF\_G\_435]

|  |  |  |
| --- | --- | --- |
| **KU16\_PID\_VREF** | | |
|  | Type | Unsigned 16 bits |
|  | Object | Battery voltage reference used to compute the Correction factor to apply during the PID regulation. |
|  | Safety | NA |
|  | Constraint | This value shall be defined as post build parameters |
|  | Value | 13500 |
|  | Unit | 1lsb = 1mV |

#### Emergency motor braking sequence

DSG\_BFE\_00099:

[COVERS: DES\_TF\_G\_1408]

|  |  |  |
| --- | --- | --- |
| **KU8\_PYRO\_SEQ\_RAMPUP\_DURATION** | | |
|  | Type | Unsigned 8 bits |
|  | Object | Duration of the ramp up in number of function call ticks. |
|  | Safety | NA |
|  | Constraint | NA |
|  | Value | 25 lsb (means 10ms) |
|  | Unit | 1lsb = 400µs |

DSG\_BFE\_00100:

[COVERS: DES\_TF\_G\_1370]

|  |  |  |
| --- | --- | --- |
| **KU8\_PYRO\_SEQ\_CORR\_FACTOR\_LIMIT** | | |
|  | Type | Unsigned 8 bits |
|  | Object | Upper limit for the full range of correction factors Alpha and Beta applied during the motor braking sequence. |
|  | Safety | NA |
|  | Constraint | NA |
|  | Value | 64 |
|  | Unit | NA |

DSG\_BFE\_00101:

[COVERS: DES\_TF\_G\_1408]

|  |  |  |
| --- | --- | --- |
| **KU8\_PYRO\_SEQ\_BRAKING\_DURATION** | | |
|  | Type | Unsigned 8 bits (TBC) |
|  | Object | Duration of the motor braking and releasing step in number of function call ticks. |
|  | Safety | NA |
|  | Constraint | This value can be defined as POST BUILD parameter (TBC) |
|  | Value | 125 lsb (means 50ms) – Default value |
|  | Unit | 1lsb = 400µs |

#### Cycle scheduling

DSG\_BFE\_00102:

[COVERS:]

|  |  |  |
| --- | --- | --- |
| **KU8\_NB\_MAX\_BELT\_FCT\_LIBRARY** | | |
|  | Type | Unsigned 8 bits |
|  | Object | Number of belt functions which are available. |
|  | Safety | NA |
|  | Constraint | This parameter is a PRE BUILD parameter that shall be defined on the configuration file of the BFE. |
|  | Value | By default: 17 |

DSG\_BFE\_00103:

[COVERS:]

|  |  |  |
| --- | --- | --- |
| **KU8\_NB\_MAX\_STEP\_LIBRARY** | | |
|  | Type | Unsigned 8 bits |
|  | Object | Number of steps which are available in the step library. |
|  | Safety | NA |
|  | Constraint | This parameter is a PRE BUILD parameter that shall be defined on the configuration file of the BFE. |
|  | Value | By default: 80 |

DSG\_BFE\_00104:

[COVERS: DES\_TF\_G\_822, DES\_TF\_G\_837]

|  |  |  |
| --- | --- | --- |
| **KU8\_NO\_BELT\_FUNCTION** | | |
|  | Type | Unsigned 8 bits |
|  | Object | Value for belt cycle number in case of no belt function selected or in case of invalid cycle number (higher than KU8\_MAX\_BELT\_FUNCTION). |
|  | Safety | NA |
|  | Constraint | NA |
|  | Value | 0xFF |

DSG\_BFE\_00105:

[COVERS: DES\_TF\_G\_839, DES\_TF\_G\_840]

|  |  |  |
| --- | --- | --- |
| **KU8\_UNDEFINED\_STEP** | | |
|  | Type | Unsigned 8 bits |
|  | Object | Value for step number in case of no executed step. |
|  | Safety | NA |
|  | Constraint | NA |
|  | Value | 0xFF |

DSG\_BFE\_00106:

[COVERS:]

|  |  |  |
| --- | --- | --- |
| **KU8\_CYCLE\_SIZE** | | |
|  | Type | Unsigned 8 bits |
|  | Object | Size of one belt function parameters in the table. |
|  | Safety | NA |
|  | Constraint | NA |
|  | Value | 12 |

DSG\_BFE\_00107:

[COVERS:]

|  |  |  |
| --- | --- | --- |
| **KU8\_INDEX\_OF\_FIRST\_STEP** | | |
|  | Type | Unsigned 8 bits |
|  | Object | Byte position for the first step in a belt function parameters. |
|  | Safety | NA |
|  | Constraint | NA |
|  | Value | 4 |

DSG\_BFE\_00108:

[COVERS:]

|  |  |  |
| --- | --- | --- |
| **KU8\_MAX\_STEP\_INDEX** | | |
|  | Type | Unsigned 8 bits |
|  | Object | Upper limit of the number of steps that can be managed in one belt function . |
|  | Safety | NA |
|  | Constraint | NA |
|  | Value | 8 |

DSG\_BFE\_00109:

[COVERS:]

|  |  |  |
| --- | --- | --- |
| **KU8\_STEP\_SIZE** | | |
|  | Type | Unsigned 8 bits |
|  | Object | Size of one step parameters in the table. |
|  | Safety | NA |
|  | Constraint | NA |
|  | Value | 6 |

DSG\_BFE\_00110:

[COVERS:]

|  |  |  |
| --- | --- | --- |
| **KU8\_PWM\_ORDER\_TYPE** | | |
|  | Type | Unsigned 8 bits |
|  | Object | Constant used to detect if the step that shall be started is configured with a PWM order unit. |
|  | Safety | NA |
|  | Constraint | NA |
|  | Value | 0 |

DSG\_BFE\_00111:

[COVERS:]

|  |  |  |
| --- | --- | --- |
| **KU8\_CURRENT\_ORDER\_TYPE** | | |
|  | Type | Unsigned 8 bits |
|  | Object | Constant used to detect if the step that shall be started is configured with a current order unit. |
|  | Safety | NA |
|  | Constraint | NA |
|  | Value | 1 |

DSG\_BFE\_00112:

[COVERS:]

|  |  |  |
| --- | --- | --- |
| **KU8\_VOLTAGE\_ORDER\_TYPE** | | |
|  | Type | Unsigned 8 bits |
|  | Object | Constant used to detect if the step that shall be started is configured with a voltage order unit. |
|  | Safety | NA |
|  | Constraint | NA |
|  | Value | 4 |

DSG\_BFE\_00113:

[COVERS:]

|  |  |  |
| --- | --- | --- |
| **KU8\_NO\_POWER\_ADAPT** | | |
|  | Type | Unsigned 8 bits |
|  | Object | Constant used as power degradation factor for a belt function that does not required this feature. |
|  | Safety | NA |
|  | Constraint | NA |
|  | Value | 128 |

#### High Power regulation

DSG\_BFE\_00114:

[COVERS:]

|  |  |  |
| --- | --- | --- |
| **KU16\_PWM\_MAX\_BTN** | | |
|  | Type | Unsigned 16 bits |
|  | Object | Numerical value for the maximum motor power order that can be applied by the BFE on the power stage (without the Boost stage). |
|  | Safety | NA |
|  | Constraint | 64lsb = 1% |
|  | Value | 6400 |

DSG\_BFE\_00115:

[COVERS:]

|  |  |  |
| --- | --- | --- |
| **KU16\_PWM\_MAX\_BOOST** | | |
|  | Type | Unsigned 16 bits |
|  | Object | Numerical value for the maximum motor power order that can be applied by the BFE on the boost stage. |
|  | Safety | NA |
|  | Constraint | 1sb = 0.195 % |
|  | Value | 512 |

DSG\_BFE\_00116:

[COVERS:]

|  |  |  |
| --- | --- | --- |
| **KU16\_VBOOST\_MAX** | | |
|  | Type | Unsigned 16 bits |
|  | Object | Numerical value for the maximum voltage that can be applied on the power stage during a boost activation. |
|  | Safety | NA |
|  | Constraint | 1sb = 1mV |
|  | Value | 40000 (means 40V source: HW constraint introduced by BTN 8980) |

DSG\_BFE\_00117:

[COVERS:]

|  |  |  |
| --- | --- | --- |
| **KS16\_INTEGER\_MAX\_ABSOLUTE** | | |
|  | Type | Unsigned 16 bits |
|  | Object | Maximum value acceptable by the software for integral of the error whatever the battery voltage value is. |
|  | Safety | NA |
|  | Constraint | NA |
|  | Value | 28000 (this value is equivalent to a 77% command on the boost stage) |

DSG\_BFE\_00118:

[COVERS:]

|  |  |  |
| --- | --- | --- |
| **KS16\_DRIFT\_THRESHOLD** | | |
|  | Type | Signed 16 bits |
|  | Object | Current value threshold for Kilis drift compensation. Over this value the motor current shall be corrected. |
|  | Safety | NA |
|  | Constraint | 1lsb = 10mA |
|  | Value | 3000 ( = 30A) |

DSG\_BFE\_00119:

[COVERS: DES\_TF\_G\_669]

|  |  |  |
| --- | --- | --- |
| **KU8\_MIN\_CURRENT\_DURING\_BOOST** | | |
|  | Type | Unsigned 8 bits |
|  | Object | Battery current value threshold used to abort a high power regulation. |
|  | Safety | NA |
|  | Constraint | 1lsb = 10mA |
|  | Value | 35 (350mA) |

DSG\_BFE\_00120:

[COVERS:]

|  |  |  |
| --- | --- | --- |
| **KU8\_STEP\_DRUATION\_RESC\_HP** | | |
|  | Type | Unsigned 8 bits |
|  | Object | Factor used to rescale step duration value: 1lsb=10ms 🡺 1lsb=400µs |
|  | Safety | NA |
|  | Constraint | NA |
|  | Value | 25 |

DSG\_BFE\_00121:

[COVERS: DES\_TF\_G\_656]

|  |  |  |
| --- | --- | --- |
| **KU8\_BOOST\_DECREASE\_DURATION** | | |
|  | Type | Unsigned 8 bits |
|  | Object | Duration needed to perform the boost switch off sequence. |
|  | Safety | NA |
|  | Constraint | 1lsb = 400µs |
|  | Value | 100 |

DSG\_BFE\_00122:

[COVERS: DES\_TF\_G\_657]

|  |  |  |
| --- | --- | --- |
| **KU16\_STEP\_DECREASE\_PWMBOOST** | | |
|  | Type | Unsigned 16 bits |
|  | Object | Offset applied on the Integer maximum value when the boost interruption sequence is on-going. |
|  | Safety | NA |
|  | Constraint | NA |
|  | Value | 256 |

DSG\_BFE\_00123:

[COVERS: DES\_TF\_G\_624]

|  |  |  |
| --- | --- | --- |
| **KS16\_GAIN\_INTEGER\_BOOST** | | |
|  | Type | Signed 16 bits |
|  | Object | Division factor applied on the error during High power regulation, if the boost stage is activated |
|  | Safety | NA |
|  | Constraint | NA |
|  | Value | 2 |

DSG\_BFE\_00124:

[COVERS: DES\_TF\_G\_624]

|  |  |  |
| --- | --- | --- |
| **KS16\_GAIN\_INTEGER\_STANDARD** | | |
|  | Type | Signed 16 bits |
|  | Object | Division factor applied on the error during High power regulation, if the boost stage is NOT activated |
|  | Safety | NA |
|  | Constraint | NA |
|  | Value | 8 |

## Functioning overview

### Global overview

The BFE component is split into 3 modules with 3 different periods.

The 10ms periodic sub module is in charge of the step scheduling. It will periodically evaluate the step constraints to interrupt the current executed step, and in case of a step start it shall provide some data to other BFE sub modules in charge of the motor power regulation process.

In addition this sub module shall provide to other SW modules the executed cycle and the executed step.

This work repartition limits the CPU load taken by the BFE component with a small real time degradation: a new cycle can only start every 10ms.

The 2ms periodic sub module is in charge of the motor power regulation management for the ‘standards’ steps:

* For ‘PWM’ configured steps
* For ‘voltage’ configured step
* For ‘current’ configured step:
  + With the ‘classic’ current regulation (204 MOPF)
  + With the PID current regulation (based on motor and/or battery current)

The standards steps are the steps that don’t required a high frequency to perform the regulation.

And the 400µs periodic sub module is in charge of the motor power regulation management in case of step configured with High Power.

Due to Crash Pawl Locking feature issue, a new dedicated BFE periodic sub module can be called every 400µs to perform an emergency motor braking sequence and to reverse motor rotation direction to help the CPL mechanic.

Selected cycle (I)

Executed cycle (O)

Executed step (O)

No Cycle

Cycle N

No Cycle or new cycle Id

No Cycle

No Cycle or new cycle Id

Cycle N

No Step

No Step or Step 0

Step 0

Step 1

Step m

No Cycle

No Step

Apply a 0% PWM

Calculate new motor power consign based on HW inputs and belt function parameters

Apply a 0% PWM

Apply a 0% PWM or new motor power order

Extracts step parameters and provide them to 2ms and/or 400µs sub modules to perform the computation of motor power consign.

Evaluates that the current executed step shall be interrupted at the next 10ms function call.

Figure 1: BFE behaviour chronogram

### Step scheduling management (10ms)

The step scheduling management is done when the BFE\_runScheduleStep main function is called by the application OS.

Overview of the jobs performed by this function with Technical function references indication:

Perform a snapshot of all inputs data

Filter **Selected Cycle** based on **Pyro activation status**

Extract **cycle options**

TF\_G14

TF\_G113

New step to be started (due to new cycle or end of step in progress)

New Selected Cycle ?

Extract **step parameters**

TF\_G112

(based on Cycle/Step config.) Compute:

**Step first value / Step duration /**

**Slope value / Step order type**

TF\_G12

(based on Step config.)

Evaluate **Step ending flag**

Provide:

**Executed Cycle**

**Executed Step**

TF\_G111

With critical section to ensure data set integrity

### Standards steps management (2ms)

The standards steps management is done by the BFE\_runExecuteSteps main function every 2 ms.

Overview of the jobs performed by this function with Technical function references indication:

Perform a snapshot of all inputs data

**Slope value / First value / step order type / Pyro status**

**Etc …**

Compute Motor Power Order according to the **Order Type**

TF\_G14

TF\_G12

Pyro fired or HP step ?

Power stage consign update

(function call)

Under Disable IT / Enable IT

New cycle

Reset PID integral error

New Step

Order\_to\_apply(t) =  
First\_Value + Slope\_Value

Order\_to\_apply(t) =  
Order\_to\_apply(t-1) + Slope\_Value

Manage PWM Steps

Manage Current Steps

Manage PID Current Steps

Manage Voltage Steps

Manage No Step

Yes

No

No

Yes

Yes

No

Every 2ms

According to the order type value, the motor power order shall be updated with the good algorithm excepted if the step order type is High Power. In this case the motor power shall not be updated by this sub module (i.e. it will be updated by the sub module dedicated to High Power regulation).

The detection of new cycle and new step is based on the change of inputs data (first order / increment value / order type).

### Management of steps with High Power regulation (400µs)

This function call shall be synchronized with the ADC conversion interrupt to obtain the best motor power regulation.

The aim of this process is to calculate the motor power order that shall be applied on the motor in case of steps configured with High power option. This process is the only one that can request the high power stage activation (with the high power auto tests).

The current regulation performed is a PI regulation (Proportional – Integral)

Perform a snapshot of all inputs data

**First value / step order type / Pyro status / step duration**

TF\_G14

TF\_G12

Step configured with HP ?

Power stage consign update

(function call)

Manage High Power regulation

No

Yes

### Management of the emergency motor stage braking and releasing (400µs)

This function shall manage the motor power order only in critical situation such as a pyrotechnical device activation (during a car crash).

In this kind of situation, the “Step scheduling management”, the “Standard steps management” and the “High power steps management” will be deactivated (from a SW point of view) i.e. these sub modules will no more update the motor power order.

This function shall be called every 400µs in order to have the quickest possible response time (to brake the motor and to reverse the motor rotation direction in order to disengage the mechanic).

* Operations performed during this function execution:

The goal is to apply:

(Duty cycle applied on the booster stage)

(Duty cycle applied on the Half bridges stage)

With Alpha and Beta coefficients evaluated based on the motor current and the battery current by using look up tables defined as POST BUILD parameters.

This process is performed in a high frequency real time context (T = 400µs), so mathematical operations shall be limited to avoid an overload of the CPU load.

The Pictus micro controller is a 32bits µC and does not embed an APU. So the 64 bits operations shall be avoided.

The motor current input value is a 32 bits unsigned value. This value shall be saturated and cast into a unsigned 16bit range value 🡺 the motor current value will be between [ 0 ; 65 535 mA ].

This limitation is acceptable since we shall never exceed 60A.

To estimate the battery current the following operation shall be performed:

(The DutyCycle\_HB full range is [-6400 ; 6400])

This operation manages only 16bits data, so there is no risk of overflow during the operation.

The battery current value shall be ranged between [ 0 ; 65 535 mA ] (like the motor current).

The computation of Alpha and Beta factors are based on 2 linear interpolation lookup tables with an X-Axis unsigned 16bits range and the Y-Axis unsigned 8bits range. So all multiplication & division can be performed on 32 bits data.

Alpha and Beta are coefficients with values used to limit the duty cycle if the motor current or battery current are too high.

The full range (in term of lsb) of Alpha & Beta factors is [ 0 ; 64 ] (64 lsb means 1)

Note: to perform the linear interpolation, a dedicated MBD library will be used.

Then we are able to compute the duty cycle that shall be applied on the half bridge stage.

As reminder:

(Duty cycle applied on the Half bridge stage)

According to the PAL actuator design document, the 100% value is reached when the Half bridge duty cycle argument is set to 6400.

In terms of lsb:

The multiplication is done between 3 variables with a unsigned 8 bits size so no risk of overflow on a 32 bits data.

And the division can be done with a simple bit shift.

This regulation shall be performed for the first 50ms after the pyro activation detection.

A counter shall be implemented to manage this timer.

Since this function is called every 400µs, this counter can be implemented on an unsigned 8 bits data   
(50 / 0.4=125 function calls)

Then a ramp up shall be performed for 10 ms (10 / 0.4 = 25 function calls). The same counter can be used.

Finally at the end of the ramp up, the DutyCycle\_HB shall be fixed to 0% until the next application restart.

## Component functions

All the local functions of the components are listed below.

### Functions called during the step scheduling management

#### M\_SelectStepId

* **Its finality:**

The aim of this function is to update the step identifier that shall be executed by the BFE.

* **Exported functions used:**

None.

* **Exported variables used:**

None.

* **Object:**

This function provides the step identifier that will be used internally by the BFE.

The step identifier is a value between [0;8[ when a belt function is in progress and equal to KU8\_UNDEFINED\_STEP when no cycle is executed or if the step index read out from cycle parameters is invalid.

This function has no mathematical operations to manage, and just has to make basic checks in order to select the step identifier.

* **Entering and outgoing parameters:**

None.

* **Period:**

This function shall be called every 10ms.

* **Who:**

This function is called by the exported function in charge of the step scheduling (See 3.2.1 Function *BFE\_runScheduleStep*)

* **Items used:**

**M\_SelectStepId**

Selected cycle Id

Step Ending flag

Step identifier

Cycles parameters

BFE input

BFE internal data

BFE internal data

Post build parameters

BFE output

Previous Selected cycle Id

Previous Step Identifier

TF\_G\_111

Figure 2 M\_SelectStepId overview

The step ending flag is read thru **bPreviousEndStep** data (computed and provided by M\_ManageStep function).

The previous step identifier is read thru **u8PreviousExecutedStep** data.

The previous cycle identifier is read thru **u8PreviousSelectedCycle** data.

The selected cycle identifier is the input read thru **Rte\_Read\_prrSelectedCycle\_u8CycleNumber** service.

The ‘cycles parameters’ is a table accessed thru **NVP\_au8BeltProfilesDefinitions** alias.

DSG\_BFE\_00125:

The first check to perform is to test if the Selected Cycle Identifier is strictly lower than **KU8\_NB\_MAX\_BELT\_FCT\_LIBRARY**. If not the case, the Step Identifier shall be set to **KU8\_UNDEFINED\_STEP**. And the function can return.

[COVERS: DES\_TF\_G\_826, DES\_TF\_G\_843, DES\_TF\_G\_825]

If the previous test is Ok:

DSG\_BFE\_00126:

The start of a new cycle shall be detected:

🡺detection of a change between the Selected cycle and the Previous Selected Cycle

OR

🡺Same cycle but Previous Executed Step is equal to KU8\_UNDEFINED\_STEP (means that the belt function is restarted)

The *unfiltered step Identifier* data shall be set to **0**.

[COVERS: DES\_TF\_G\_832]

DSG\_BFE\_00127:

The end of the current executed step shall be taken into account:

🡺bPreviousEndStep is TRUE

The *unfiltered step Identifier* data shall be incremented.

[COVERS: DES\_TF\_G\_847]

DSG\_BFE\_00128:

In other cases:

🡺bPreviousEndStep is FALSE

The *unfiltered step Identifier* data shall not change.

[COVERS: DES\_TF\_G\_834]

Before updating the output of this function with the *unfiltered step identifier* value, the integrity of the step reference shall be checked.

DSG\_BFE\_00129:

The end of the 8th step of the belt function shall be detected:

🡺 *unfiltered step Identifier* has reached the configuration limit (KU8\_MAX\_STEP\_INDEX)

In this case, the Step Identifier shall be set to **KU8\_UNDEFINED\_STEP**. And the function can return.

[COVERS: DES\_TF\_G\_846]

As reminder the belt function array has the following format:



The step reference is the value of Step\_Id**n**\_**m** (with **n** the step identifier and **m** the belt function identifier.

The following formula shall be used to extract the step reference:

NVP\_au8BeltProfilesDefinitions[ *SelectedCycle* \* **KU8\_CYCLE\_SIZE** + **KU8\_INDEX\_OF\_FIRST\_STEP** + *Unfiltered\_Step\_Identifier*]

Note: There is no risk of array overrun since the *SelectedCycle* value has already been checked at the first check of the function and *Unfiltered\_Step\_Identifier* value has been tested against his valid range previously.

DSG\_BFE\_00130:

The step reference value shall be checked against his valid range:

🡺 The step reference is higher than **KU8\_NB\_MAX\_STEP\_LIBRARY**

In this case, the Step Identifier shall be set to **KU8\_UNDEFINED\_STEP**. And the function can return.

[COVERS: DES\_TF\_G\_848]

In all other cases, the step identifier can be set with the *unfiltered step identifier* value.

Note: a deeper analysis of the step content will be done during the extraction of step parameters that could lead to the “invalidation” of the step that shall be executed (refer to TF\_G\_112). This function reduces his analysis to the belt function parameters scope.

* **Returned values:**

None.

#### M\_ProvideExecutedCycleStep

* **Its finality:**

The aim of this function is to provide the executed cycle and executed step to the other software modules.

* **Exported functions used:**

None.

* **Exported variables used:**

None.

* **Object:**

In general the executed cycle will be the same than the selected cycle provided by the BFS.

If the current executed cycle is at his end (i.e. step identifier = KU8\_NO\_STEP), then the executed cycle shall take the KU8\_NO\_BELT\_FUNCTION value.

And the executed step is the executed step position on the cycle parameters.

The executed step is an integer with the range [0;7]. If there are no executed step the output value shall be KU8\_UNDEFINED\_STEP.

This function is a post processing module in charge of formatting the outputs for the other software components.

* **Entering and outgoing parameters:**

None.

* **Period:**

This function shall be called every 10ms.

* **Who:**

This function is called by the exported function in charge of the step scheduling (See 3.2.1 Function *BFE\_runScheduleStep*)

* **Items used:**

**M\_ProvideExecutedCycleStep**

Selected cycle Id

Step Identifier

Executed cycle

Executed step

Freewheeling state flag

Previous Executed step

Previous Executed cycle

Step Validity status

TF\_G\_111

Figure 3 M\_ProvideExecutedCycleStep overview

The selected cycle identifier is the input read thru **Rte\_Read\_prrSelectedCycle\_u8CycleNumber** service.

The step identifier is the data provided by the *M\_SelectStepId* function: **u8StepToExecute**

The previous executed step is read thru **u8PreviousExecutedStep** data.

The previous executed cycle is read thru **u8PreviousExecutedFunction** data.

The step validity status is the data provided by the *M\_ExtractStepParameters* function: **eStepValidity** data.

The freewheeling state flag is provided by the *M\_ControlStepExecution* function: **bIsStepFrozen** data. (to be renamed)

The executed cycle output shall be provided to the rest of the application by using the following API: Rte\_Write\_psrExecutedCycle\_u8CycleNumber( uint8 u8ExecutedCycle )

The executed step output shall be provided to the rest of the application by using the following API: Rte\_Write\_psrExecutedStep\_u8StepNumber( uint8 u8ExecutedStep )

According to these interfaces:

DSG\_BFE\_00131:

If the selected cycle is NO\_CYCLE:

🡺Executed cycle shall be KU8\_NO\_BELT\_FUNCTION

🡺Executed step shall be KU8\_UNDEFINED\_STEP

[COVERS: DES\_TF\_G\_826, DES\_TF\_G\_843]

DSG\_BFE\_00132:

If the step identifier is KU8\_UNDEFINED\_STEP (means that end of cycle has been reached whatever the reason: 8th step or invalid step reference or etc…):

🡺Executed cycle shall be KU8\_NO\_BELT\_FUNCTION

🡺Executed step shall be KU8\_UNDEFINED\_STEP

[COVERS: DES\_TF\_G\_848, DES\_TF\_G\_846, DES\_TF\_G\_831]

DSG\_BFE\_00133:

If the step validity status is invalid, the step shall not be executed, then the cycle will be aborted:

🡺Executed cycle shall be KU8\_NO\_BELT\_FUNCTION

🡺Executed step shall be KU8\_UNDEFINED\_STEP

[COVERS: DES\_TF\_G\_848, DES\_TF\_G\_846, DES\_TF\_G\_831]

DSG\_BFE\_00134:

If a free-wheeling phase has been started (due to motor rotation direction inversion detected):

🡺Executed cycle shall keep his last value

🡺Executed step shall keep his last value

[COVERS: XXXX]

In all other cases:

🡺Executed cycle shall take the Selected cycle value

🡺Executed step shall take the Step identifier value

* **Returned values:**

None.

#### M\_ManageStepEvents

* **Its finality:**

The aim of this function is to provide the *bIsStepFinished* status that will be taken into account 10ms later to stop the current executed step (if required).

This function is in charge of the part of TF\_G12 that defines how and when an executed step shall be stopped (“Steps constraints management”). The **Step ending flag** that is computed by this function will be internally used by the BFE in the *M\_SelectStepId* function.

* **Exported functions used:**

None.

* **Exported variables used:**

None.

* **Object:**

This function evaluates the step constraints in function of the step parameters and the ECU state or the vehicle state.

This flag shall be refreshed at each function call even if there are no step in progress.

DSG\_BFE\_00135:

If no step is in progress, the step ending flag shall be set to TRUE.

This is a design choice since we have no requirements.

[COVERS:]

In the other case (i.e. a step is in progress) all conditions that could lead to the step stop shall be evaluated one by one. Only one condition is sufficient to stop the current executed step.

Among all possible conditions available, only the step duration shall be checked unconditionally.

DSG\_BFE\_00136:

If the step timer has reached the limit configured in the step parameters.

🡺 **bIsTimerElapsed** is set to *TRUE*

Then the **step ending flag** shall be fixed to *TRUE*

[COVERS: DES\_TF\_G\_1310]

All other conditions shall only be evaluated if the related option is configured in the step parameters:

End of pre-pretensioning situation:

DSG\_BFE\_00137:

If the step is configured with the trigger Off option and the pre-pretensionning situation is no more present.

Then the **step ending flag** shall be fixed to *TRUE*

[COVERS: DES\_TF\_G\_111]

Belt blocked:

DSG\_BFE\_00138:

If the step is configured with the belt blocked detection option and if the belt speed is lower than the specified speed threshold for a specific time.

Then the **step ending flag** shall be fixed to *TRUE*

[COVERS:]

Note: the speed threshold is equal to 0 and the time to confirm the belt blocked situation is fixed to 10ms.

Motor blocked:

Belt movement detection:

DSG\_BFE\_00139:

If the step is configured with the Belt movement detection option and the belt is moving (i.e. belt speed different from 0).

Then the **step ending flag** shall be fixed to *TRUE*

[COVERS: DES\_TF\_G\_107]

Adaptive current limit reached:

This constraint is used for comfort belt functions in order to abort the step execution if the motor current overtakes a given limit. This current limit is configurable in the step parameters.

An array of 64 elements provide all available current limits values. Inside step parameters, 6 bits are allocated to configure an index. The association of this index and the array gives the nominal current limit value at 25°C.

The array to be considered is **KAU16\_ADAPTATIVE\_CURRENT\_LIMIT**.

Note: the array element values resolution is: 1lsb = 0.1A

Then this current limit shall be slightly corrected with a factor calculated by a lookup table in function of the temperature.

The lookup table takes the temperature as input and provides the correction factor as output.

The temperature references points are listed in the **NVP\_au16AdaptCurrentLimitTempIdx** data.

The correction factors references points are listed in the **NVP\_au8AdaptCurrentLimitFactors** data.

To avoid unnecessary rescaling operations the following resolutions shall be used:

For temperature points: the same resolution than the temperature read from RTE service. These values shall be coded on unsigned 16 bits data length.

For correction factors: 1 lsb = 1%, the values shall be coded on unsigned 8 bits data length.

Finally the motor current threshold is defined as below:

Due to the scaling of the 2 data involved in the multiplication, the *MotorCurrentThrs* resolution is 1lsb=1mA 🡺 This threshold can be directly compared with the measured motor current.

Note: if the Index read out from step parameters is 0, it means that the Adaptive current limit option shall not be taken into account to stop the current executed step.

DSG\_BFE\_00140:

If the step is configured with an index different from 0 and if the motor current measured overtakes the given motor current threshold (after temperature correction).

Then the **step ending flag** shall be fixed to *TRUE*

[COVERS:]

1st Low Belt Payout abortion:

DSG\_BFE\_00170:

If the step is configured with the “Belt payout abortion by 1st threshold” option and the compensated belt payout is lower than 1st NVP threshold *NVP\_u16BeltPayoutAbortThrs1*.

Then the **step ending flag** shall be fixed to *TRUE*

[COVERS: DES\_TF\_G\_1424]

2nd Low Belt Payout abortion:

DSG\_BFE\_00171:

If the step is configured with the “Belt payout abortion by 2nd threshold” option and the compensated belt payout is lower than 2nd NVP threshold *NVP\_u16BeltPayoutAbortThrs2*.

Then the **step ending flag** shall be fixed to *TRUE*

[COVERS: DES\_TF\_G\_1425]

* **Entering and outgoing parameters:**

None.

* **Period:**

This function shall be called every 10ms.

* **Who:**

This function is called by the exported function in charge of the step scheduling (See 3.2.1 Function *BFE\_runScheduleStep*)

* **Items used:**

**M\_ManageStepEvents**

End of pretensioning situation  
aka ‘Trigger Off’

Step status

Step ending flag

Step duration time out flag

Internal temperature sensor

Motor current measured

Battery voltage measured

Step parameters

Belt speed measured

Motor resistance table  
Adaptive current interruption point

Lookup table for temperature correction

Motor power order applied

TF\_G\_12

Belt pay out measured

Figure 4 M\_ManageStepEvents overview

The belt speed measured input is read by using the RTE interface:

The belt pay out measured input is read by using the RTE interface:

The motor power order input is read by using the RTE interface:

The battery voltage input is read by using the RTE interface:

The motor current input is read by using the RTE interface:

The internal temperature input is read by using the RTE interface:

The end of pre-pre-tensioning situation is read by using the RTE interface:

The step parameters (parameters applicable for the current executed step) are provided by a local data that is updated by ***Error! Not a valid bookmark self-reference.*** function: **stStepParameters**

The step validity status is provided by a local data that is updated by ***Error! Not a valid bookmark self-reference.*** function: **eStepValidity**

The step duration time out flag is provided by a local data that is updated by

*M*\_ControlStepExecution function: **bIsTimerElapsed**

POST BUILD parameters:

* **Returned values:**

None.

#### M\_ExtractStepParameters

* **Its finality:**

The aim of this function is to read out the step parameters when a new step shall be started and perform some basics consistency checks on the configuration.

Then a step validity status based on these checks will be computed that will be used to decide if the step shall be executed or not.

If the tests performed have been passed successfully, the step parameters will be decoded and stored in a local structure data and provided to the other BFE functions for their usage.

* **Exported functions used:**

None.

* **Exported variables used:**

None.

* **Object:**

Step validity status update

As described above: The goal of this function is to perform some checks to detect incoherent step configuration that could be dangerous for the power stage.

DSG\_BFE\_00141:

First of all: the values of Selected cycle and Step identifier inputs shall be checked against their validity range.

If Selected belt function is over the maximum number of belt function handled by the application (i.e. **KU8\_NB\_MAX\_BELT\_FCT\_LIBRARY**) OR if Step Identifier is out of its valid range (i.e. higher than **KU8\_MAX\_STEP\_INDEX**) 🡺 **eStepValidity** shall be set to *KE\_INVALID\_STEP*.

[COVERS: DES\_TF\_G\_915, DES\_TF\_G\_916, DES\_TF\_G\_922, DES\_TF\_G\_923]

If the previous check is Ok, it is possible to extract the step reference with the following formula:

Without risk of array overflow.

DSG\_BFE\_00142:

The values of the obtained step reference shall be checked against its validity range.

If *Step reference* is over the maximum number of steps handled by the application (i.e. **KU8\_NB\_MAX\_STEP\_LIBRARY**) 🡺 **eStepValidity** shall be set to *KE\_INVALID\_STEP*.

[COVERS: DES\_TF\_G\_917]

If the previous check is Ok, it is possible to extract the step parameters from the POST BUILD library.

As reminder the step library is organized as follow:



And the internal layout of a step is as follow:

|  |  |  |
| --- | --- | --- |
| Field | Description | Size |
| Duration | 1 LSB = 10ms | Coded on an unsigned 16 bits |
| Order value | 1 LSB = 1% PWM OR  1 LSB = 0.5 A OR  1 LSB = 0.25 V  (depends of the step order type) | Coded on a signed 8 bits |
| Order type & options | Bit field layout:   |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | | A | O | R | B1 | T |  |  |  | |  |  |  |  |  | Order type | | |   A: Ramp step  O: Order controlled step  R: Released controlled step  B1: Motor blocked event controlled  T: Trigger Off event controlled step  Order type:   * 0x00: PWM order type * 0x01: Current order type * 0x04: Voltage order type * All other values: invalid order type | Coded on 8 bits |
| Options | Bit field layout:   |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | |  |  |  |  |  |  | C2 | C1 | | Adaptive current limit | | | | | |  |  | |  | | | | | |  |  | |  |  | B2 | W | M | PO2 | PO1 | H |   C1: PID based on battery current  C2: PID based on motor current  B2: Belt blocked event controlled (HES)  W: Weighted step  M: Belt movement controlled  PO2 - PO1 : Belt Payout abortion  H: High Power step | Coded on 16 bits |

3 criterions have to be checked to invalidate the step execution:

🡪Step duration shall be compared against the value 0x0000

🡪Step duration shall be compared against the value 0xFFFF

🡪Step duration shall be checked against his valid range

Step duration is computed as follow:

Step order type can be extracted with the following formula:

DSG\_BFE\_00143:

If *Step duration* is 0 or 0xFFFF 🡺 **eStepValidity** shall be set to *KE\_INVALID\_STEP*.

[COVERS: DES\_TF\_G\_919, DES\_TF\_G\_920]

DSG\_BFE\_00144:

If *Step Order Type* is different from:

\* KU8\_PWM\_ORDER\_TYPE

\* KU8\_CURRENT\_ORDER\_TYPE

\* KU8\_VOLTAGE\_ORDER\_TYPE

🡺 **eStepValidity** shall be set to *KE\_INVALID\_STEP*.

[COVERS: DES\_TF\_G\_924]

If all of the previous checks have been passed successfully, it means that the step can be executed, then the step validity status can take the value *KE\_NEW\_STEP* or *KE\_ON\_GOING\_STEP*.

DSG\_BFE\_00145:

If the step has just been started (Step Identifier different from the previous executed step or new cycle is started)

🡺 **eStepValidity** shall be set to *KE\_NEW\_STEP*.

Else it means that the step has not changed since the last function call:

🡺 **eStepValidity** shall be set to *KE\_ON\_GOING\_STEP*.

[COVERS: DES\_TF\_G\_926, DES\_TF\_G\_1023]

Step parameters update:

DSG\_BFE\_00146:

To limit CPU load, the step parameters structure shall only be refreshed when a step is started (i.e. when **eStepValidity** = *KE\_NEW\_STEP*)

[COVERS: DES\_TF\_G\_928, DES\_TF\_G\_1022]

DSG\_BFE\_00147:

The step duration element in the step parameters structure shall be updated as follow:

[COVERS: DES\_TF\_G\_1011]

DSG\_BFE\_00148:

The step order element in the step parameters structure shall be updated as follow:

[COVERS: DES\_TF\_G\_1012]

* **Entering and outgoing parameters:**

None.

* **Period:**

This function shall be called every 10ms.

* **Who:**

This function is called by the exported function in charge of the step scheduling (See 3.2.1 Function *BFE\_runScheduleStep*)

* **Items used:**

**M\_ExtractStepParameters**

Step validity status

Step parameters table

Cycle parameters table

Step Identifier

Step parameters structure

Previous Executed Step

TF\_G\_112

Selected cycle

Previous Selected cycle

Figure 5 M\_ExtractStepParameters overview

The selected cycle input is read by using the RTE interface:

The previous selected cycle is read by using the static data **u8PreviousSelectedCycle** data.

The step identifier input is provided by *M\_SelectStepId* function (**u8StepIdentifier**).

The previous executed step is read by using the **u8PreviousExecutedStep** data.

The step validity status is provided to other BFE functions in order to know the status of the step identifier: **eStepValidity**.

The step parameters is a data structure that is used internally by the BFE to prepare the step execution: **stStepParameters**.

POST BUILD parameters:

Cycle parameters (aka Belt functions parameters) is the array that contains the configuration for all belt functions manages by the application. This table is accessed thru **NVP\_au8BeltProfilesDefinitions** alias.

Steps parameters is the array that contains the configuration of all steps managed by the application. This table is accessed thru **NVP\_au8StepsDefinitions** alias.

* **Returned values:**

None.

#### M\_ControlStepExecution

* **Its finality:**

The goal of this function is to prepare some key data that will be used during the regulation performed at 2ms or 400µs.

These key data are evaluated in this periodic function to reduce the CPU load.

* **Exported functions used:**

None.

* **Exported variables used:**

None.

* **Object:**
* **Entering and outgoing parameters:**

None.

* **Period:**

This function shall be called every 10ms.

* **Who:**

This function is called by the exported function in charge of the step scheduling (See 3.2.1 Function *BFE\_runScheduleStep*)

* **Items used:**

**M\_ControlStepExecution**

Step validity status

Slope value

Temperature correction type

Internal temperature sensor

Motor current measured

Battery voltage measured

Step parameters

Lookup tables for temperature correction

Motor power order applied

TF\_G\_12

Belt pay out measured

Power degradation factor

First step value

Step order type

Free Wheel state

Step time out flag

Step duration

Figure 6 M\_ControlStepExecution overview

The step parameters is provided by M\_ExtractStepParameters function (**stStepParameters**).

The step validity status is provided by M\_ExtractStepParameters function (**eStepValidity**).

The power degradation factor is provided by M\_ExtractCycleOptions function (**u8PowerDegradationFactor**).

The temperature correction type is provided by M\_ExtractCycleOptions function (**eTemperatureCompensationType**).

POST BUILD parameters:

Each lookup table for temperature correction is defined by using 2 tables: one for each axis.

According to the range of the temperature correction type date, the function shall be able to manage 4 lookup table in tensioning direction.

\* X-Axis for temperature: array of 7 unsigned 16 bits data elements:

**CAL\_AU16\_TEMP\_WEIGHT\_INDEX\_TENS\_X**

\* Y-Axis for correction factors: array of 7 unsigned 8 bits data elements:

**CAL\_AU8\_TEMP\_WEIGHT\_FACTORS\_TENS\_X**

With **X** a value between [0;3]

In addition, the application shall be able to manage only lookup table for steps in releasing direction:

\* X-Axis for temperature: array of 7 unsigned 16 bits data elements:

**CAL\_AU16\_TEMP\_WEIGHT\_INDEX\_RELEASE**

\* Y-Axis for correction factors: array of 7 unsigned 8 bits data elements:

**CAL\_AU8\_TEMP\_WEIGHT\_FACTORS\_RELEASE**

* **Returned values:**

None.

#### M\_ExtractCycleOptions

* **Its finality:**

The aim of this function is:

* to compute and provide the power degradation factor that shall be applied during belt function execution
* and to provide the temperature correction type in order to apply the correct lookup table if the steps are configured with the Weighted step option.
* **Exported functions used:**

None.

* **Exported variables used:**

None.

* **Object:**

This function is in charge of the extraction of cycle options contained in the cycles parameters headers when a new belt function starts and then to provide these data to the other BFE local function involved in the motor power order computation.

The outputs shall only be re-evaluated in the case of the start of a new belt function.

DSG\_BFE\_00149:

If a new belt function starts, the power degradation factor and temperature compensation type shall be re-evaluated. Else the outputs values shall not be changed.

i.e. if the **selected cycle** is different from the **previous selected cycle** and the selected cycle is valid (means that selected cycle value is lower than **KU8\_NB\_MAX\_BELT\_FCT\_LIBRARY**)

[COVERS: DES\_TF\_G\_963, DES\_TF\_G\_972]

If the previous condition has been checked successfully, the option cycle byte of the new belt function shall be read and decoded.

As reminder: belt functions are coded as follow:



And the option cycle byte is coded as:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| T1 | T0 |  | P |  |  |  |  |
| Temperature compensation type | | |  |  |  |  |  |

According to the belt function parameters layout:

There is no risk of array overflow since the Selected cycle value has already been checked against its valid range.

DSG\_BFE\_00150:

If the *PowerDegradationOption* bit is unset, then the power degradation feature is not required for this belt function: **u8PowerDegradationFactor** data shall be set to **KU8\_NO\_POWER\_ADAPT**.

[COVERS: DES\_TF\_G\_965]

DSG\_BFE\_00151:

If the *PowerDegradationOption* bit is set, then the power degradation factor shall be evaluated based on vehicle battery level and the lookup table defined as Post build parameters.

[COVERS: DES\_TF\_G\_966]

*Note: the linear interpolation is directly managed by a specific MBD library.*

The temperature compensation cycle option is managed as follows:

DSG\_BFE\_00172:

If the step is not a Weighted step (if bit W of step option is not set), then no temperature compensation is applied for this step.

[COVERS: DES\_TF\_G\_974]

DSG\_BFE\_00173:

If the step is a Weighted step (if bit W of step option is set), and if the step order is negative, then a temperature compensation based on the NVP Releasing interpolation table is applied:

(NVP\_au16WeightTempIdxRel; NVP\_au8WeightFactorsRel)

[COVERS: DES\_TF\_G\_1092; DES\_TF\_G\_955; DES\_TF\_G\_1093]

DSG\_BFE\_00174:

If the step is a Weighted step (if bit W of step option is set), and if the step order is positive, then a temperature compensation based on a NVP Tensioning interpolation table, chosen by the T1-T0 cycle option, is applied:

* If T1-T0 = “00”, the 1st Tensioning interpolation table is used:

(NVP\_au16WeightTempIdxTens0; NVP\_au8WeightFactorsTens0)

* If T1-T0 = “01”, the 2nd Tensioning interpolation table is used:

(NVP\_au16WeightTempIdxTens1; NVP\_au8WeightFactorsTens1)

* If T1-T0 = “00”, the 3rd Tensioning interpolation table is used:

(NVP\_au16WeightTempIdxTens2; NVP\_au8WeightFactorsTens2)

* If T1-T0 = “00”, the 4th Tensioning interpolation table is used:

(NVP\_au16WeightTempIdxTens3; NVP\_au8WeightFactorsTens3)

[COVERS: DES\_TF\_G\_1091; DES\_TF\_G\_957; DES\_TF\_G\_955; DES\_TF\_G\_1093]

* **Entering and outgoing parameters:**

None.

* **Period:**

This function shall be called every 10ms.

* **Who:**

This function is called by the exported function in charge of the step scheduling (See 3.2.1 Function *BFE\_runScheduleStep*

* **Items used:**

**M\_ExtractCycleOptions**

Selected cycle Id

Previous cycle to execute

Power degradation factor

Temperature adaptation type

Vehicle battery level

TF\_G\_113

Lookup table for Power degradation factor.

Cycle parameters.

Figure 7 M\_ExtractCycleOptions overview

The selected cycle identifier is the input read thru **Rte\_Read\_prrSelectedCycle\_u8CycleNumber** service.

The vehicle battery level is the input read thru **XXXX** service.

*/!\ this battery level is NOT the voltage read at the ECU connector by the microcontroller ADC peripheral.*

The previous cycle identifier is read thru **u8PreviousSelectedCycle** data.

The power degradation coefficient is provided to the others local functions of the BFE module: **u8PowerDegradationFactor**

The temperature adaptation type is provided to the others local functions of the BFE module: **eTemperatureCompensationType**

POST BUILD parameters:

Cycle parameters (aka Belt functions parameters) is the array that contains the configuration for all belt functions manages by the application. This table is accessed thru **NVP\_au8BeltProfilesDefinitions** alias.

Lookup table for power degradation factors computation is defined by using 2 tables: one for each axis.

\* X-Axis for battery voltage: array of 6 unsigned 16 bits data elements:

**NVP\_au16PowerDegradVoltIdx**

\* Y-Axis for power degradation factors: array of 6 unsigned 8 bits data elements:

**NVP\_au8PowerDegradFactors**

* **Returned values:**

None.

### Functions used to compute the motor command in case of standards steps

The following functions can be called during the execution of the BFE sub module in charge of standards steps.

At each execution of the Function *BFE\_runExecuteSteps* only one of the next functions can be called. And in case of a High Power step none of them shall be called, because the High Power regulation is managed in a specific dedicated BFE main function that is called every 400µs.

Before calling a specific function to manage the computation of motor command, some common operations between all regulation types shall be performed.

DSG\_BFE\_00152:

Update of the step consign to reach for the next 2ms:

If a step has just started the consign to reach will be computed as follow:

If a step is in progress, the consign to reach will be computed as follow:

To detect that a step is started the step first value and the slope value shall be compared respectively to the values memorized 2ms before.

i.e. a comparison shall be done between the First step value retrieved form the RTE IRV read service and **s32PreviousFirstOrder2ms** data. And the same job shall be done between Slope value retrieved from the RTE IRV read service and **s32PreviousSlopeValue2ms** data.

As soon as the checks have been done and the expected strategy applied, the 2 statics data shall be refreshed with the IRV values.

When the step consign has been calculated, then the **s32PreviousStepTargetToReach** data can be refreshed with this new value.

[COVERS: DES\_TF\_G\_1094, DES\_TF\_G\_1104, DES\_TF\_G\_1101]

The *StepConsign* value obtained with the previous formula shall be stored in XXXXX and will be used by all local functions listed below.

DSG\_BFE\_00153:

Whatever the order type (excepted for High Power), the booster power order shall be set to 0%.

[COVERS: DES\_TF\_G\_1099, DES\_TF\_G\_1103, DES\_TF\_G\_1098, DES\_TF\_G\_1089, DES\_TF\_G\_1061]

#### ManagePWMSteps

* **Its finality:**

The aim of this function is to compute the motor command in case of steps configured with PWM order type.

* **Exported functions used:**

None.

* **Exported variables used:**

None.

* **Object:**

DSG\_BFE\_00154:

According to the scaling of the data computed by M\_ControlStepExecution function in case of a step configured with PWM order type:

The first order step value and slope value have the same high resolution scaling:

Step parameter consign value full range is [-100 ; 100] (i.e. 1lsb=1%)

After applying the power degradation factor and temperature correction factor the full range will be [-819200 ; 819200]

*Note: the full range could be greater is temperature correction factor is over 1 (means over than 64lsb). But since it’s not possible to apply a power order over 100% on the power stage, all values out of this range will be saturated to the limits.*

The goal is to compute a motor power order with a [-6400 ; 6400] full range.

Then the following operation shall be performed to rescale the *StepConsign* and match with the expected output resolution:

[COVERS: DES\_TF\_G\_1100]

* **Entering and outgoing parameters:**

None.

* **Period:**

This function shall be called every 2ms, when the step order type is ‘PWM’.

* **Who:**

This function is called by the exported function in charge of the standard step execution (See 3.2.2 Function *BFE\_runExecuteSteps*)

* **Items used:**

None.

* **Returned values:**

None.

#### ManageCurrentSteps

* **Its finality:**

The aim of this function is to compute the motor command in case of steps configured with standard current regulation type.

* **Exported functions used:**

None.

* **Exported variables used:**

None.

* **Object:**

According to the scaling of the data computed by M\_ControlStepExecution function in case of a step configured with voltage order type:

The first order step value and slope value have the same high resolution scaling:

Initially step parameter consign value full range is [-40A ; 40A] with physical values or [-80;80] with logical values (i.e. 1lsb=0.5A)

*Note: ‘40A’ value is just given as example, it will depend from a POST-BUILD parameter:* ***CAL\_KU8\_MAX\_CURRENT\_VALUE***

After applying the power degradation factor and temperature correction factor the full range will be [-655360 ; 655360] (with nominal conditions: No power degradation=128 and no temperature correction = 64)

Power degradation factor resolution is 1/128

Temperature correction factor resolution is 1/64

The resolution of the multiplication result will be .

So: *StepFirstOrderValue* and *SlopeValue* have this resolution. And *StepConsign* data will inherit this resolution.

The goal is to compute a motor power order with a [-6400 ; 6400] full range.

In case of a new belt function started with standard current regulation, the previous power motor order and the previous motor current are both null. In this kind of situation, this regulation will require a lot of time to reach the target since the initial power motor order is 0%. To reduce this time, an initial power motor order is computed to start the regulation.

DSG\_BFE\_00155:

The *TargetInAmper* is obtained from the *StepConsign* with a rescaling operation.

The *MaxMotorCurrent* is ***CAL\_KU8\_MAX\_CURRENT\_VALUE***.

The values given by this division are typically between 0 and 1 (excepted if the target is over than the maximum motor current, but in this case the power order cannot exceed 100% so 100% will be applied).

And to match with the expected output resolution:

When switching with lsb values and rescaling to perform the division between data with same units:

[COVERS: DES\_TF\_G\_426]

In all other cases the standard current regulation is applied as follow:

This function shall evaluate the gap between the step consign and the motor current value. Then an offset shall be applied on the motor power consign based on the post build parameters and the gap value.

Step Order value – Motor current measured

PWM offset to apply

Delta THRS low

Delta THRS High

- Delta THRS High

- Delta THRS Low

- Low Offset

- High Offset

High Offset

Low Offset

DSG\_BFE\_00156:

To compute the delta between the target to reach and the measured motor current, a rescaling shall be performed first:

Then this delta is compared to the thresholds defined in the BFE configuration files and according to these comparisons, the related offset will be applied on the previous motor power order:

|  |  |
| --- | --- |
| **Delta** | Offset to apply on motor power order |
|  | -CAL\_KU16\_HIGH\_OFFSET |
|  | -CAL\_KU16\_LOW\_OFFSET |
|  | 0 |
|  | CAL\_KU16\_LOW\_OFFSET |
|  | CAL\_KU16\_HIGH\_OFFSET |

The offset defined in BFE configuration part have the same resolution than the motor power order retrieved from RTE service. So they can be applied without any additional operations.

[COVERS: DES\_TF\_G\_756, DES\_TF\_G\_1088, DES\_TF\_G\_421]

DSG\_BFE\_00177:

The motor command is clamped to zero in case of change of sign. The goal is to prevent from unexpected negative current during some standard motor current regulation steps.

Here is the detailed design of this clamping operation:

If the last motor power order applied on power stage is positive or equal to zero, and if the new calculated motor command is negative or equal to zero, then this motor command shall be clamped to zero. Otherwise it is provided without any clamping correction.

[COVERS: DES\_TF\_G\_1531]

* **Entering and outgoing parameters:**

None.

* **Period:**

This function shall be called every 2ms when the step order type is ‘Current with standard regulation’.

* **Who:**

This function is called by the exported function in charge of the standard step execution (See 3.2.2 Function *BFE\_runExecuteSteps*)

* **Items used:**
  + Last motor power order applied on power stage
  + Measured motor current in mA
  + Step consign to reach (as described at the beginning of the chapter 4.3.2)
  + ‘New cycle started’ flag
  + Post build parameters:
    - Maximum current supported by ECU: **CAL\_KU8\_MAX\_CURRENT\_VALUE**
    - High & Low offsets values
    - High & Low delta threshold values for the gap between step consign and measured current
* **Returned values:**

None.

#### ManagePIDCurrentSteps

* **Its finality:**

The aim of this function is to compute the motor power order that should be applied in case of step configured with current order type with a PID option.

* **Exported functions used:**

None.

* **Exported variables used:**

None.

* **Object:**

According to the scaling of the data computed by M\_ControlStepExecution function in case of a step configured with voltage order type:

The first order step value and slope value have the same high resolution scaling:

Initially step parameter consign value full range is [-40A ; 40A] with physical values or [-80;80] with logical values (i.e. 1lsb=0.5A)

*Note: ‘40A’ value is just given as example, it will depend from a POST-BUILD parameter:* ***CAL\_KU8\_MAX\_CURRENT\_VALUE***

After applying the power degradation factor and temperature correction factor the full range will be [-655360 ; 655360] (with nominal conditions: No power degradation=128 and no temperature correction = 64)

Power degradation factor resolution is 1/128

Temperature correction factor resolution is 1/64

The resolution of the multiplication result will be .

So: *StepFirstOrderValue* and *SlopeValue* have this resolution. And *StepConsign* data will inherit this resolution.

*(Note: in case of PID regulation, the slope value is 0 since it’s forbidden to perform a PID regulation on a ‘moving’ target)*

The goal is to compute a motor power order with a [-6400 ; 6400] full range.

The step consign is received with 1lsb=1mA resolution. According to the static error, the error history: the motor command shall be computed with a PID algorithm. This regulation can be based on motor current or on the battery current.

The PID algorithm that shall be implemented is not a strict PID, Due to HW constraints some deviations are required.

* **Entering and outgoing parameters:**

None.

* **Period:**

This function shall be called every 2ms when the step order type is ‘Current’ with at least one PID bit active.

* **Who:**

This function is called by the exported function in charge of the step scheduling (See 3.2.2 Function

* **Items used:**
  + Motor current value in mA
  + Previous integral error value
  + New cycle flag
  + Post build parameters:
    - Reference voltage value to get correction factor
    - Kp/Ki/Kd parameters for battery and motor PID regulation
* **Returned values:**

None.

#### ManageVoltageSteps

* **Its finality:**

The aim of this function is to compute the motor power command in case of steps configured with voltage order type.

* **Exported functions used:**

None.

* **Exported variables used:**

None.

* **Object:**

DSG\_BFE\_00157:

According to the scaling of the data computed by M\_ControlStepExecution function in case of a step configured with voltage order type:

The first order step value and slope value have the same high resolution scaling:

Initially step parameter consign value full range is [-32V ; 31.75V] with physical values or [-128;127] with logical values (i.e. 1lsb=0.25V)

After applying the power degradation factor and temperature correction factor the full range will be [--1048576 ; 1040384] (with nominal conditions: No power degradation=128 and no temperature correction = 64)

Power degradation factor resolution is 1/128

Temperature correction factor resolution is 1/64

The resolution of the multiplication result will be .

So: *StepFirstOrderValue* and *SlopeValue* have this resolution. And *StepConsign* data will inherit this resolution.

The goal is to compute a motor power order with a [-6400 ; 6400] full range.

Basically the motor power order is obtained with the following formula:

The *TargetInMilliVolt* is obtained from the *StepConsign* with a rescaling operation.

The *BatteryVoltageInMilliVolt* is obtained from the battery voltage retrieved by RTE macro.

The values given by this division are typically between 0 and 1 (excepted if the target is over than the battery voltage, but in this case the power order cannot exceed 100% so 100% will be applied).

And to match with the expected output resolution:

When switching with lsb values and rescaling to perform the division between data with same units:

However to obtain the best rounding value a little adaptation shall be done based on linear approximation formula:

[COVERS: DES\_TF\_G\_1104]

* **Entering and outgoing parameters:**

None.

* **Period:**

This function shall be called every 2ms when the step order type is ‘Voltage’.

* **Who:**

This function is called by the exported function in charge of the standard step execution (See 3.2.2 Function *BFE\_runExecuteSteps*)

* **Items used:**
  + Battery voltage
  + Step consign obtained from the
* **Returned values:**

None.

#### ManageNoStep

* **Its finality:**

The aim of this function is to force a null motor power order when no step is currently in progress or the step order type is invalid.

* **Exported functions used:**

None.

* **Exported variables used:**

None.

* **Object:**

DSG\_BFE\_00158:

When this function is called, the motor power order shall be set to 0% whatever the *StepConsign* value is.

[COVERS: DES\_TF\_G\_1060]

* **Entering and outgoing parameters:**

None.

* **Period:**

This function shall be called every 2ms when the step order type is ‘No Order type’ or when the order type is not a supported value.

* **Who:**

This function is called by the exported function in charge of the standards steps execution (See 3.2.2 Function *BFE\_runExecuteSteps*).

* **Items used:**

None.

* **Returned values:**

None.

### High Power regulation

As reminder the High Power algorithm can be split as follow:

Current consign

**+**

**-**

Integral action

Booster Interruption conditions

ε

Boost stage command & Power stage command computation

Integral

Half-Bridges HW components drivers

Booster HW components drivers

Supply current computation

Motor current

**BFE function scope**

Figure 8 High Power algorithm description

As described in the 4.3.2.2 chapter the StepFirstOrder Value inherits a high resolution in order to reduce the computation error in case of ramp steps management. But for this kind of regulation this high resolution is not required. On the contrary to limit the size of data during the High power regulation and reduce the CPU load, the consign resolution shall be reduced.

The *StepFirstOrder* data resolution is 8192lsb for 0.5A (refer to chapter 4.3.2.2 or 4.3.2.3 for more details). A resolution of 1lsb for 10mA is enough to perform this regulation.

The following formula shall be applied to rescale the current consign:

This Current Consign will be used in this chapter and sub-chapters. This result will be stored in the **s16CurrentConsign** local function data.

Since this regulation is a closed loop control system, it’s mandatory to manage the data initialization when starting the step execution.

#### Key data initialization

2 situations shall be managed: a new belt function starts with a first step configured with High Power regulation (nominal situation)

*Initialization for integral of the error:*

DSG\_BFE\_00159:

The integral of the error shall be initialized according to the following formula:

*Vbattery* is the battery voltage level measured at the ECU connector. Its resolution is 1lsb=1mV

*FullScaleHB* is the numerical value of the upper limit that can be applied on the power stage (without the boost). This is a constant value fixed to 6400 (**KU16\_PWM\_MAX\_BTN**).

*CurrentConsign* is the step consign. Its resolution should be 1lsb=1mA (obtained from **s16CurrentConsign** with a rescaling operation to have consistent units).

*Rmotor* is the motor resistance. Its resolution should be 1lsb=1Ω. The motor resistance value depends of the temperature. Then a lookup table should be used to compute the numerical value in function of the temperature. To limit the CPU load consumption, it has been decided to use the lower value of motor resistance (i.e. 120mΩ) for this operation.

*Note: This 120mΩ value shall be adjusted if the motor type changes !!!*

Finally:

[COVERS: DES\_TF\_G\_627]

All operations can be done on a signed 32bits data without risks of overflow. And the final result can be stored on the static signed 16bits data **s16OldIntegerError**. This static data contains usually the integral of the error value computed at the previous function call (400µs before), and is used to compute the next motor power order that shall be applied on Half bridges and boost stages. But since we are on the first function call for the High Power regulation, we have no error history.

*Initialization of the maximum value for the integral of the error:*

DSG\_BFE\_00160:

When starting a new High Power regulation step, the maximum value of the integral of the error shall be initialized according to the following formula:

*Vbattery* is the battery voltage level measured at the ECU connector. Its resolution is 1lsb=1mV

*FullScaleHB* is the numerical value of the upper limit that can be applied on the power stage (without the boost). This is a constant value fixed to 6400 (**KU16\_PWM\_MAX\_BTN**).

*VboostMax* is the numerical value of the maximum voltage that can be applied with the boost stage on the power stage. This is a constant value fixed to 40000mV (**KU16\_VBOOST\_MAX**).

Since the 2 voltage data have the same resolution, the operation can be done directly.

[COVERS: DES\_TF\_G\_625]

With this operation, lower the battery voltage is, higher is the Integral Max value. This could lead to a maximum integral value incoherent with the requirement DES\_TF\_G\_640 which requires that the boost stage power order shall not exceed 80%.

DSG\_BFE\_00161:

According to DES\_TF\_G\_530 requirement, the power order applied on boost stage is computed as follow:

The goal is to compute the maximum Integral of error value to reach a 80% power order on the boost stage.

*FullScaleHB* is the numerical value of the upper limit that can be applied on the power stage (without the boost). This is a constant value fixed to 6400 (**KU16\_PWM\_MAX\_BTN**).

*FullScaleBoost* is the numerical value of the upper limit that can be applied on the boost stage. This is a constant value fixed to 512 (**KU16\_PWM\_MAX\_BOOST**).

*BoostPowerOrderMax* is the numerical value of the boost power order when reaching 80% 🡺 80%\*512=409

After computation: *IntegralMaxAbsolute* shall never exceed 31813.

It has been decided to take a margin and to limit the integer maximum value to 28000 (equivalent to a boost power order of 77%).

If the *IntegralMax* value is over **KS16\_INTEGER\_MAX\_ABSOLUTE** (=28000), then *IntegralMax* shall be fixed to **KS16\_INTEGER\_MAX\_ABSOLUTE** in order to protect the booster HW components.

Else *IntegralMax* value is the result of the computation performed above.

[COVERS: DES\_TF\_G\_640]

The *IntegralMax* data shall be stored in **s16IntegerMax** variable.

*Initialization for step timer control:*

In order to control the end of a step controlled with High Power (boost stage shall not be switched off brutally), the time elapsed since the step start shall be monitored.

DSG\_BFE\_00162:

When a new step is started, the step timer shall be reinitialized to 0

**u16BoostDurationTimer** = 0

[COVERS: DES\_TF\_G\_656, DES\_TF\_G\_657]

*Initialization for Half Bridges Kilis temperature drift compensation:*

Note: this should be normally managed directly at the PAL or HBA software components level. But since this behaviour has been detected with High Power steps, the software compensation is performed by the High Power function. This has been observed with Infineon BTN 8980 HW parts and for motor current values over 30A.

🡺 To be reworked and moved to the PAL or HBA software components

🡺 no requirements anywhere to justify the implementation and the low pass filter.

**u32SwitchPrevInteger** variable shall be set to 0 when beginning a High Power step in order to reinitialize the Kilis drift compensation.

#### Battery current computation

The battery current is a function of motor current and power order applied on power and boost stages.

Before applying the transfer function, the motor current shall be rescaled with a resolution 1lsb = 10mA (as already done for the step consign) and a correction of the motor current shall be performed because of the Kilis temperature drift.

DSG\_BFE\_00163:

The motor current provided by the RTE service has a resolution of 1lsb = 1mA. This value shall be first rescaled then corrected.

Refer to Annex 7.1 for the Drift compensation algorithm. This correction returns **s16CorrectedMotorCurrent** data.

According to the Boost proof of design (**[A3]**):

With *Ibatt*: the battery current

*Iboost*: the boost stage current

*Dboost*: the power order applied on boost stage

With *Imot*: the motor current

*Dbridge*: the power order applied on power stage

🡺

|  |  |
| --- | --- |
| **Boost stage deactivated** | Boost stage activated |
| *Dboost* = 0 | *Dbridge* = 1 |
| Then: | Then: |
| When the boost stage is deactivated: | When the boost stage is activated: |
| In both cases: | |

The battery current computed will be stored in **s16SupplyCurrent** variable with a 1lsb=10mA resolution (as Imot).

[COVERS: DES\_TF\_G\_524, DES\_TF\_G\_525, DES\_TF\_G\_530, DES\_TF\_G\_531]

#### Booster interruption condition evaluation

As described in the requirements, there are 2 conditions that could lead to a booster activation interruption.

🡪 if the High Power step execution is closed to his time out.

🡪 if the boost stage is driven at his maximum value, but the current is too low (means that there is a HW or mechanic issue)

*Boost interruption criterion based on step duration limit:*

DSG\_BFE\_00164:

High Power step timer management:

The step duration value get from RTE macro is coded with 1lsb=10ms resolution.

The step timer value is managed with a 1lsb=400µs resolution.

So a rescaling shall be done to compare data with same resolution.

The step duration value shall be multiplied by 25 (**KU8\_STEP\_DRUATION\_RESC\_HP**)

The boost stage shall not be stopped too quickly, the power order shall decrease softly before the switch off. Then the start of the boost interruption sequence shall begin 40ms before the step time out. So duration threshold to take into account is StepDuration – 40ms.

🡺

When step execution counter is over this threshold, the boost stage deactivation sequence shall be engaged.

i.e.:

If : the boost interruption sequence is started.

[COVERS: DES\_TF\_G\_656]

*Boost interruption criterion based on a low current measured:*

DSG\_BFE\_00165:

If the battery current computed is lower than the battery current threshold (350mA) and the Integral of the error is over or equal to the Integral maximum value, the boost stage deactivation sequence shall be engaged.

i.e.:

If (**s16SupplyCurrent** < **KU8\_MIN\_CURRENT\_DURING\_BOOST**

AND

**s16IntegerErrorLimited** **s16IntegerMax)**

[COVERS: DES\_TF\_G\_669]

#### Integral action

The integral action can be split in 3 parts:

1. Boost deactivation sequence
2. Boost regulation in normal situation
3. Limitation of the Integral of the error

The items 1 and 2 are exclusive. And the point 3 is always executed.

1. *Boost deactivation sequence*

To perform the deactivation sequence of the boost stage, the Integer maximum value will be reduced at each High power function call in order to reach **KU16\_PWM\_MAX\_BTN** value (means that the boost power order will be 0 and only the Half bridge stage will be activated).

DSG\_BFE\_00166:

The boost deactivation sequence is started when one of the boost interruption conditions had been validated (refer to chapter above: 4.3.3.3).

The switch off sequence shall be completely managed and finished in 40ms (due boost interruption condition based on step duration limit).

In the worst case, the Integer of the error is equal to maximum absolute value reachable by the software: **KS16\_INTEGER\_MAX\_ABSOLUTE** (= 28000).

The goal is to reach **KU16\_PWM\_MAX\_BTN** value (= 6400) in 40ms.

40ms is equivalent to 100 function calls.

Then: (28000-6400)/100 = 216

To ensure a correct boost deactivation sequence, the integer maximum value shall decrease by 216 at each function call.

It has been decided to take a margin, the offset applied by the application will be finally 256 (**KU16\_STEP\_DECREASE\_PWMBOOST**)

🡺 At each function call:

The Integer maximum value lower limit is **KU16\_PWM\_MAX\_BTN**. A comparison against this constant will be performed to limit the range.

*Note: This part will not change directly the Integer of the error value. But since the Integer Maximum value is reduced, this will finally impact the Integer of the error value during the part: 3)Limitation of the integral error.*

🡺

[COVERS: DES\_TF\_G\_657]

1. *Boost regulation in normal situation*

This part will only be executed if none of the deactivation conditions has been set.

DSG\_BFE\_00167:

This part is in charge of computing the integral of the error between the step consign and the battery current computed before. The high power regulation is like a PI regulator with a variable gain.

The error is the difference between the Step consign and the battery current. Both data have the same resolution, then:

The new integer value is computed as follow:

*OldInteger* data is stored in **s16OldIntegerError** variable.

*Error* data is stored in **s16CurrentError** variable.

*Ki* gain value depends if the boost stage is activated (Previous integer value over **KU16\_PWM\_MAX\_BTN**) or not.

|  |  |
| --- | --- |
| **KU16\_PWM\_MAX\_BTN s16OldIntegerError**  **Boost stage not activated** | KU16\_PWM\_MAX\_BTN s16OldIntegerError  Boost stage activated |
|  |  |
|  |  |

[COVERS: DES\_TF\_G\_623, DES\_TF\_G\_624]

1. *Limitation of the integral error*

DSG\_BFE\_00168:

Since the integer of the error is directly used to compute the power orders applied on the boost stage and on the Half bridge stage, this key data shall always be saturated between [0 ; Integer Maximum value]

In terms of software variables:

If , then

If , then

In all other cases:

[COVERS: DES\_TF\_G\_626, DES\_TF\_G\_627]

When the limited integer value has been computed, the static data containing the integer of the error (and that will be used at the next high power function call) shall be updated with this new value (i.e. **s16OldIntegerError**):

🡺

#### Boost stage command and Power stage command elaboration

DSG\_BFE\_00169:

The computation of the both power order is based on the Integral of the error after limitation: **s16IntegerErrorLimited**.

If the Integer is lower than **KU16\_PWM\_MAX\_BTN**: The power order of the boost stage shall be fixed to 0 and the power order on the power stage shall be fixed to **s16IntegerErrorLimited**.

Else (Integer over the **KU16\_PWM\_MAX\_BTN**): the power order on the power stage shall be fixed to **KU16\_PWM\_MAX\_BTN** and the power order on the boost stage shall be calculated with the following formula:

[COVERS: DES\_TF\_G\_530, DES\_TF\_G\_531]

### Critical sections to protect data or avoid re-entrant access

The aim of the following macro-functions is to protect some critical data or some function calls in order to avoid motor power order corruption due to real time issues.

The final implementation of these function or macro is under the responsibility of the final project according to the technical possibilities given by operating system and/or scheduler.

#### M\_START\_PROTECT\_STEP\_INIT\_DATA

* **Its finality:**

The aim of this function is to start the critical section that has to protect the data written by Function *BFE\_runScheduleStep* and exchanged with: Function *BFE\_runExecuteSteps* and Function *BFE\_runExecuteHighPowerStep*.

* **Exported functions used:**

None.

* **Exported variables used:**

None.

* **Object:**

The goal is to protect the step order type / the first value of the step / the step duration / the slope value when these 4 data are refreshed.

There is a risk that the Interrupt in charge of High Power regulation function call could be triggered during these data update and then this regulation could perform his job with an incoherent data set.

This function-like macro shall be implemented in the configuration part of the BFE.

* **Entering and outgoing parameters:**

None.

* **Period:**

This function shall be called every 10ms.

* **Who:**

This function is called by the exported function in charge of the step scheduling (See 3.2.1 Function *BFE\_runScheduleStep*Function )

* **Items used:**

None.

* **Returned values:**

None.

#### M\_END\_PROTECT\_STEP\_INIT\_DATA

* **Its finality:**

The aim of this function is to finish the critical section that has to protect the data written by Function *BFE\_runScheduleStep* and exchanged with: Function *BFE\_runExecuteSteps* and Function *BFE\_runExecuteHighPowerStep*.

* **Exported functions used:**

None.

* **Exported variables used:**

None.

* **Object:**

The goal is to protect the step order type / the first value of the step / the step duration / the slope value when these 4 data are refreshed.

There is a risk that the Interrupt in charge of High Power regulation function call could be triggered during these data update and then this regulation could perform his job with an incoherent data set.

This function-like macro shall be implemented in the configuration part of the BFE.

The function call shall be done just after the update of the 4 data mentioned above.

* **Entering and outgoing parameters:**

None.

* **Period:**

This function shall be called every 10ms.

* **Who:**

This function is called by the exported function in charge of the step scheduling (See 3.2.1 Function *BFE\_runScheduleStep*Function )

* **Items used:**

None.

* **Returned values:**

None.

#### M\_START\_PROTECT\_POWER\_ORDER\_UPD

* **Its finality:**

The aim of this function is to start the critical section that has to protect the data written by Function *BFE\_runScheduleStep* and exchanged with: Function *BFE\_runExecuteSteps* and Function *BFE\_runExecuteHighPowerStep*.

* **Exported functions used:**

None.

* **Exported variables used:**

None.

* **Object:**

This function shall ensure that the call to the service in charge of motor power order update will not be interrupted by the interrupt in charge of High Power and emergency braking management.

This function-like macro shall be implemented in the configuration part of the BFE.

The function call shall be done just before the function call to the PAL service in charge of motor power order update.

* **Entering and outgoing parameters:**

None.

* **Period:**

This function shall be called every 2ms.

* **Who:**

This function is called by the exported function in charge of the standard step execution (See 3.2.2 Function *BFE\_runExecuteSteps*)

* **Items used:**

None.

* **Returned values:**

None.

#### M\_END\_PROTECT\_POWER\_ORDER\_UPD

* **Its finality:**

The aim of this function is to finish the critical section that has to protect the data written by Function *BFE\_runScheduleStep* and exchanged with: Function *BFE\_runExecuteSteps* and Function *BFE\_runExecuteHighPowerStep*

* **Exported functions used:**

None.

* **Exported variables used:**

None.

* **Object:**

This function shall ensure that the call to the service in charge of motor power order update will not be interrupted by the interrupt in charge of High Power and emergency braking management.

This function-like macro shall be implemented in the configuration part of the BFE.

The function call shall be done just after the function call to the PAL service in charge of motor power order update.

* **Entering and outgoing parameters:**

None.

* **Period:**

This function shall be called every 2ms.

* **Who:**

This function is called by the exported function in charge of the standard step execution (See 3.2.2 Function *BFE\_runExecuteSteps*)

* **Items used:**

None.

* **Returned values:**

None.

# Post Build parameters

Here are listed the post build parameters used by the component that are not defined in a specification document.

All post build parameters are defined in **[C0]**

# Compilation options

Here is listed all the compilation options used for this component.

Some preprocessing compilation options can be used to adapt the compiled BFE source code to the project needs. This will reduce the memory print of the BFE for a low end project for instance.

## High Power feature key

|  |  |  |
| --- | --- | --- |
| **BFE\_CFG\_OPT\_HIGH\_POWER** | | |
|  | Object | This compilation key shall be defined if the Hardware supports the high power feature.  If not, the Function will be compiled without his content. |
|  | Constraint |  |

## PID current regulation feature key

|  |  |  |
| --- | --- | --- |
| **BFE\_CFG\_OPT\_PID** | | |
|  | Object | This compilation key shall be defined if the Hardware supports the PID current regulation feature.  If not, the  ManagePIDCurrentSteps function will be compiled without his content. |
|  | Constraint |  |

## Adaptive current interruption point feature key

|  |  |  |
| --- | --- | --- |
| **BFE\_CFG\_OPT\_ADAPT\_CURRENT** | | |
|  | Object | The compilation allow the activation/deactivation of the interpolation table in charge of the adaptive current interruption point feature.  If this option is not defined: the 2 tables used for the linear interpolation are not used, the temperature is not used. |
|  | Constraint |  |

## Belt blocked feature key

|  |  |  |
| --- | --- | --- |
| **BFE\_CFG\_OPT\_BELT\_BLOCKED\_HES** | | |
|  | Object | This option can be deactivated if the HES sensors are absent from the PCB, or if the BSR function is not required by the project. |
|  | Constraint |  |

## Belt Movement feature key

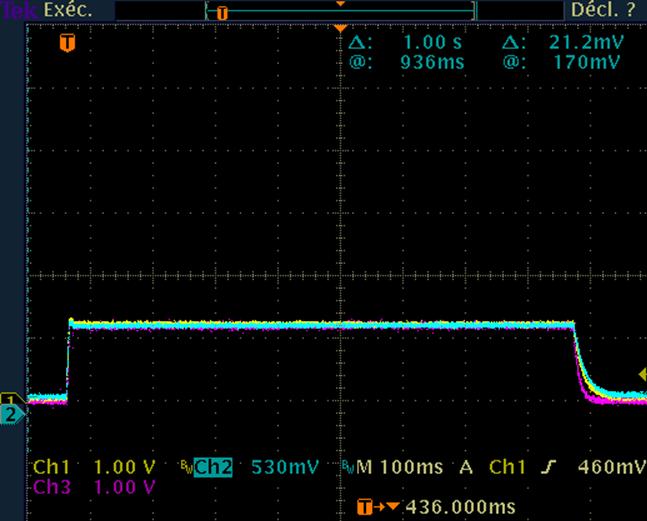
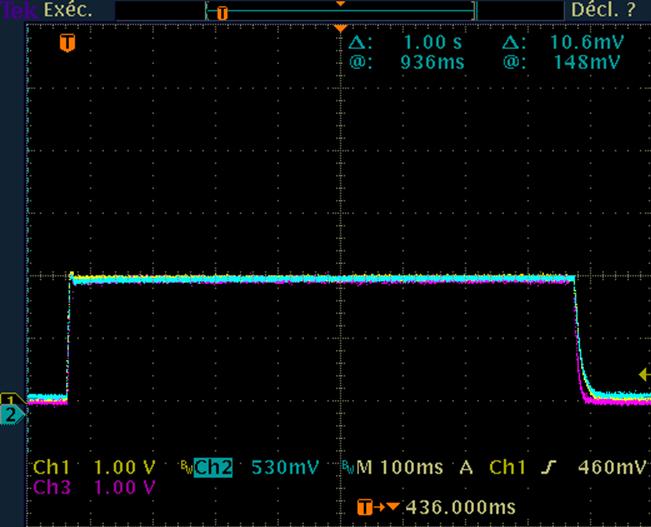
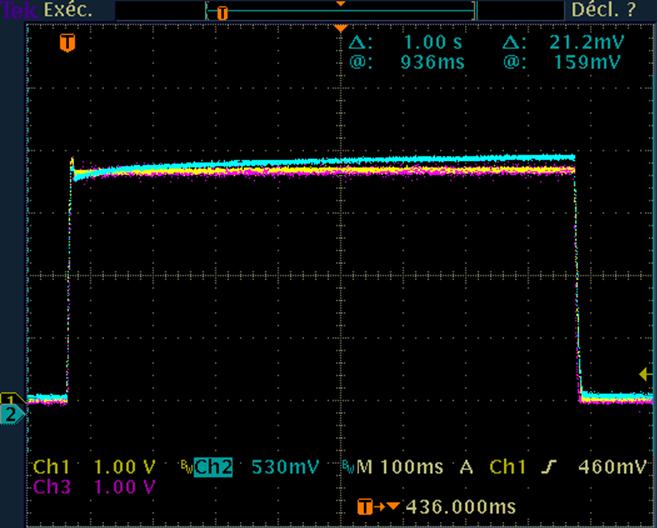
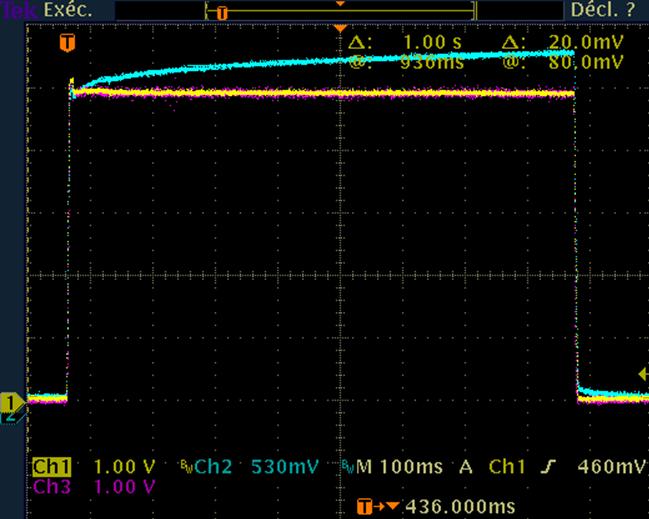
|  |  |  |
| --- | --- | --- |
| **BFE\_CFG\_OPT\_BELT\_MVT\_DETECT** | | |
|  | Object | This option can be deactivated if the HES sensors are absent from the PCB, or if the BSR function is not required by the project. |
|  | Constraint |  |

# Annex

## Kilis temperature drift issue

Ch2 in blue is the voltage at Is output

Ch1 and ch3 are the current measured on motor and on supply (with PWM = 100%, controlled current supply)



I = 12A, no drift

I = 20A, no drift

I = 36A, drift

I = 50A, drift

30A level

30A level

Above 30A, the measured current is too high with a drift response like a low pass filter.

30A level

Current

Voltage on Is output

The principle is to subtract from the measure a value proportional to the over current over 30A (KS16\_DRIFT\_THRESHOLD) after a low pass filtering

Over Current: I-30A

Filtered over current = (Filtered over current \* 1023 + Over Current)/ 1024

τ= 400ms

(Over Current is called u16DriftCurrent in the program)

if (s16MotorCurrent > KS16\_DRIFT\_THRESHOLD)

u16DriftCurrent = s16MotorCurrent - KS16\_DRIFT\_THRESHOLD;

else

u16DriftCurrent = 0;

u32IntegerCurrent = u32SwitchPrevInteger - (u32SwitchPrevInteger >>10) + u16DriftCurrent;

Note: u32SwitchPrevInteger contains the previous integer of the current value computed at the previous function call or is equal to 0 if it’s the first function call when starting a new step.

s16FilteredOverCurrent = u32IntegerCurrent >>10 ;

This filtered over current is multiplied by an empiric factor (Current/ 214) to get the right value of current without drift

s16Correction = ((long) s16MotorCurrent \* s16FilteredOverCurrent)>>14;

s16CorrectedMotorCurrent = s16MotorCurrent - s16Correction;