# Module -- Active Pull Compensation

# High-Level Description

This module corrects for vehicle pull issues by compensation for both long and short term torque offsets.

# Figures

## Component Diagram

123.tif

# Variable Data Dictionary

For details on module input / output variable, refer to the Data Dictionary for the application. Input / output variable names are listed here for reference.

|  |  |  |
| --- | --- | --- |
| Module Inputs | Module Outputs | |
| HwTorque\_HwNm\_f32 | | PullCompCmd\_MtrNm\_f32 |
| HandwheelPosition\_HwDeg\_f32 | |  |
| HandwheelAuthority\_Uls\_f32 | |  |
| VehicleSpeed\_Kph\_f32 | |  |
| VehicleSpeedValid\_Cnt\_lgc | |  |
| HandwheelVelocity\_HwRadpS\_f32 | |  |
| SrlComYawRate\_DegpS\_f32 | |  |
| DisableLearning\_Cnt\_lgc | |  |
| DisableOutput\_Cnt\_lgc | |  |

## Module Internal Variables

This section identifies the name, range and resolutions for module specific data created by this module. If there are no range restrictions on the variable, the term “FULL” is placed into the table for legal range.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Variable Name | Resolution | Legal Range  (min) | Legal Range  (max) | Software Segment |
| DecGain\_Uls\_M\_f32 | Single Precision Float | 0.0000135 | 0.00054 | ACTIVEPULL\_START\_SEC\_VAR\_CLEARED\_32 |
| IncGain\_Uls\_M\_f32 | Single Precision Float | 0.0000135 | 0.00054 | ACTIVEPULL\_START\_SEC\_VAR\_CLEARED\_32 |
| LTIntGain\_Uls\_M\_f32 | Single Precision Float | 0.00001875 | 0.00045 | ACTIVEPULL\_START\_SEC\_VAR\_CLEARED\_32 |
| LTWindUpLimit\_HwNm\_M\_f32 | Single Precision Float | 0 | 4 | ACTIVEPULL\_START\_SEC\_VAR\_CLEARED\_32 |
| STStepSize\_HwNm\_M\_f32 | Single Precision Float | 0 | 20000 | ACTIVEPULL\_START\_SEC\_VAR\_CLEARED\_32 |
| PullCompStepSize\_HwNm\_M\_f32 | Single Precision Float | 0 | 0.2 | ACTIVEPULL\_START\_SEC\_VAR\_CLEARED\_32 |
| ResetPer1\_Cnt\_M\_lgc | n/a | FALSE | TRUE | ACTIVEPULL\_START\_SEC\_VAR\_CLEARED\_BOOLEAN |
| ResetPer2\_Cnt\_M\_lgc | n/a | FALSE | TRUE | ACTIVEPULL\_START\_SEC\_VAR\_CLEARED\_BOOLEAN |
| ResetPer3\_Cnt\_M\_lgc | n/a | FALSE | TRUE | ACTIVEPULL\_START\_SEC\_VAR\_CLEARED\_BOOLEAN |
| HwTorqueSV\_HwNm\_M\_Str | LPF32KSV\_Str | N/A | N/A | ACTIVEPULL\_START\_SEC\_VAR\_CLEARED\_UNSPECIFIED |
| HwTorqueSV\_HwNm\_M\_Str.K\_Uls\_f32 | Float32 | 0.001255848 | 0.715390457 |  |
| HwTorqueSV\_HwNm\_M\_Str.SV\_Uls\_f32 | Float32 | -10 | 10 |  |
| SrlComYawRateSV\_DegpS\_M\_Str | LPF32KSV\_Str | N/A | N/A | ACTIVEPULL\_START\_SEC\_VAR\_CLEARED\_UNSPECIFIED |
| SrlComYawRateSV\_DegpS\_M\_Str.K\_Uls\_f32 | Float32 | 0.001255848 | 0.715390457 |  |
| SrlComYawRateSV\_DegpS\_M\_Str.SV\_Uls\_f32 | Float32 | -128 | 127.9375 |  |
| EnableTime\_mS\_M\_u32 | 1 | 0 | 4294967295 | ACTIVEPULL\_START\_SEC\_VAR\_CLEARED\_32 |
| EnableLearn\_Cnt\_M\_lgc | n/a | FALSE | TRUE | ACTIVEPULL\_START\_SEC\_VAR\_CLEARED\_BOOLEAN |
| HwTorqueSTSV\_HwNm\_M\_Str | LPF32KSV\_Str | N/A | N/A | ACTIVEPULL\_START\_SEC\_VAR\_CLEARED\_UNSPECIFIED |
| HwTorqueSTSV\_HwNm\_M\_Str.K\_Uls\_f32 | Float32 | 0.001255848 | 0.715390457 |  |
| HwTorqueSTSV\_HwNm\_M\_Str.SV\_Uls\_f32 | Float32 | -10 | 10 |  |
| STComp\_HwNm\_M\_f32 | Single Precision Float | -4 | 4 | ACTIVEPULL\_START\_SEC\_VAR\_CLEARED\_32 |
| STOppSignTime\_mS\_M\_u32 | 1 | 0 | 4294967295 | ACTIVEPULL\_START\_SEC\_VAR\_CLEARED\_32 |
| PullCompCmd\_HwNm\_M\_f32 | Single Precision Float | -8.8 | 8.8 | ACTIVEPULL\_START\_SEC\_VAR\_CLEARED\_32 |
| LTComp\_HwNm\_M\_f32 | Single Precision Float | -4 | 4 | ACTIVEPULL\_START\_SEC\_VAR\_SAVED\_ZONEH\_32 |
| HwTorqueLTSV\_HwNm\_M\_Str | LPF32KSV\_Str | N/A | N/A | ACTIVEPULL\_START\_SEC\_VAR\_CLEARED\_UNSPECIFIED |
| HwTorqueLTSV\_HwNm\_M\_Str.K\_Uls\_f32 | Float32 | 0.006263487 | 1 |  |
| HwTorqueLTSV\_HwNm\_M\_Str.SV\_Uls\_f32 | Float32 | -14 | 14 |  |
| SComLTComp\_HwNm\_M\_f32 | Single Precision Float | -4 | 4 | ACTIVEPULL\_START\_SEC\_VAR\_CLEARED\_32 |
| SComLTCompSet\_Cnt\_M\_lgc | n/a | FALSE | TRUE | ACTIVEPULL\_START\_SEC\_VAR\_CLEARED\_BOOLEAN |
| SComSTComp\_HwNm\_M\_f32 | Single Precision Float | -4 | 4 | ACTIVEPULL\_START\_SEC\_VAR\_CLEARED\_32 |
| SComSTCompSet\_Cnt\_M\_lgc | n/a | FALSE | TRUE | ACTIVEPULL\_START\_SEC\_VAR\_CLEARED\_BOOLEAN |
| PrevLTLearnTime\_Min\_M\_u16 | 1 | 0 | 120 | ACTIVEPULL\_START\_SEC\_VAR\_CLEARED\_16 |
| PrevSTLearnTimeInc\_Sec\_M\_u12p4 | 0.0625 | 0 | 200 | ACTIVEPULL\_START\_SEC\_VAR\_CLEARED\_16 |
| PrevSTLearnTimeDec\_Sec\_M\_u12p4 | 0.0625 | 0 | 200 | ACTIVEPULL\_START\_SEC\_VAR\_CLEARED\_16 |
| HwTrqFilt\_HwNm\_D\_f32 | Single Precision Float | -10 | 10 | ACTIVEPULL\_START\_SEC\_VAR\_CLEARED\_32 |
| YawRateFilt\_DegpS\_D\_f32 | Single Precision Float | -128 | 127.9375 | ACTIVEPULL\_START\_SEC\_VAR\_CLEARED\_32 |
| STError\_HwNm\_D\_f32 | Single Precision Float | -10 | 10 | ACTIVEPULL\_START\_SEC\_VAR\_CLEARED\_32 |
| STIntGain\_Uls\_D\_f32 | Single Precision Float | 0.0000135 | 0.00054 | ACTIVEPULL\_START\_SEC\_VAR\_CLEARED\_32 |
| STReset\_Cnt\_D\_lgc | n/a | FALSE | TRUE | ACTIVEPULL\_START\_SEC\_VAR\_CLEARED\_BOOLEAN |
| LTError\_HwNm\_D\_f32 | Single Precision Float | -14 | 14 | ACTIVEPULL\_START\_SEC\_VAR\_CLEARED\_32 |
| PrevVehSpd\_Kph\_M\_f32 | Single Precision Float | 0 | 511 | ACTIVEPULL\_START\_SEC\_VAR\_CLEARED\_32 |
| VehSpdRate\_KphpS\_M\_f32 | Single Precision Float | -5110 | 5110 | ACTIVEPULL\_START\_SEC\_VAR\_CLEARED\_32 |

### User defined typedef definition/declaration

This section documents any user types uniquely used for the module.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Typedef Name | Element Name | User Defined Type | Legal Range  (min) | Legal Range  (max) |
| None |  |  |  |  |

# Constant Data Dictionary

## Calibration Constants

This section lists the calibrations used by the module. For details on calibration constants, refer to the Data Dictionary for the application.

|  |
| --- |
| Constant Name |
| k\_YawRateFilt\_Hz\_f32 |
| k\_HwTrqFilt\_Hz\_f32 |
| k\_STResetHwTrq\_HwNm\_f32 |
| k\_STResetHwPos\_HwDeg\_f32 |
| k\_STResetYawRate\_DegpS\_f32 |
| k\_EnableHwTrqMax\_HwNm\_f32 |
| k\_EnableHwPosMax\_HwDeg\_f32 |
| k\_EnableHwAuthMin\_Uls\_f32 |
| k\_EnableHwVelMax\_DegpS\_f32 |
| k\_EnableVehSpdRateMax\_KphpS\_f32 |
| k\_EnableVehSpdMin\_Kph\_f32 |
| k\_EnableVehSpdMax\_Kph\_f32 |
| k\_EnableYawRateMax\_DegpS\_f32 |
| k\_EnableTime\_mS\_u32 |
| k\_STLimit\_HwNm\_f32 |
| k\_STLearnTimeInc\_Sec\_f32 |
| k\_STLearnTimeDec\_Sec\_f32 |
| k\_STOppSignTime\_mS\_u32 |
| k\_STRampTime\_Sec\_f32 |
| k\_STIntInputLimit\_HwNm\_f32 |
| k\_STFilt\_Hz\_f32 |
| k\_FiltDeadband\_HwNm\_f32 |
| k\_LTLimit\_HwNm\_f32 |
| k\_LTLearnTime\_Min\_f32 |
| k\_LTFilt\_Hz\_f32 |
| k\_LTIntInputLimit\_HwNm\_f32 |
| k\_TotalLimit\_HwNm\_f32 |
| k\_HwNmToMtrNm\_Uls\_f32 |
| t\_VehSpdScaleTblX\_Kph\_u9p7[4] |
| t\_VehSpdScaleTblY\_Uls\_u2p14[4] |
| k\_OutputMaxRate\_HwNmpS\_f32 |

The filter constants were derived from the requirements in SF-09 in conjunction with the following filter analyses. Note that the upper frequency limits defined in the requirements for some values were not achievable. The data dictionary reflects the limits of both the requirements and the software limitations.

  

## Program(fixed) Constants

### Embedded Constants

All embedded constants whose values are provided in Eng units will be evaluated to the equivalent counts by using the FPM\_InitFixedPoint\_m() macro within the #define statement.

#### Local

|  |  |  |  |
| --- | --- | --- | --- |
| Constant Name | Resolution | Units | Value |
| D\_MINTOSEC\_SECPMIN\_F32 | Single Precision Float | SecPerMin | 60 |
| D\_STINTSCALER\_ULS\_F32 | Single Precision Float | Unitless | 1.35 |
| D\_STSAMPLETIME\_SEC\_F32 | Single Precision Float | Seconds | 0.002 |
| D\_LTINTSCALER\_ULS\_F32 | Single Precision Float | Unitless | 1.35 |
| D\_LTSAMPLETIME\_SEC\_F32 | Single Precision Float | Seconds | 0.1 |
| D\_PULLCOMPSAMPLETIME\_SEC\_F32 | Single Precision Float | Seconds | 0.002 |

#### Global

This section lists the global constants used by the module. For details on global constants, refer to the Data Dictionary for the application.

|  |
| --- |
| Constant Name |
| D\_FALSE\_CNT\_LGC |
| D\_180OVRPI\_ULS\_F32 |
| D\_2MS\_SEC\_F32 |
| D\_ZERO\_ULS\_F32 |
| D\_MTRTRQCMDLOLMT\_MTRNM\_F32 |
| D\_MTRTRQCMDHILMT\_MTRNM\_F32 |

### Module specific Lookup Tables Constants

|  |  |  |  |
| --- | --- | --- | --- |
| Constant Name | Resolution | Value | Software Segment |
| None |  |  |  |

# Functions/Macros used by the Sub-Modules

## Library Functions / Macros

The library and functions / Macros that are called by the various sub modules are identified below,

1. FPM\_FloatToFixed\_m
2. FPM\_FixedToFloat\_m
3. LPF\_SvUpdate\_s16InFixKTrunc\_m
4. Abs\_f32\_m
5. Min\_m
6. Max\_m
7. Sign\_f32\_m
8. Limit\_m
9. IntplVarXY\_u16\_u16Xu16Y\_Cnt

## Data Hiding Functions

1. <None>

## Global Functions/Macros Defined by this Module

None

## Local Functions/Macros Used by this MDD only

None

# Software Module Implementation

## Runtime Environment (RTE) Initial Values

This section lists the initial values of data written by this module but controlled by the RTE. After RTE initialization, the data in this table will contain these values.

|  |  |
| --- | --- |
| Data | Value |
| Rte\_InitValue\_DisableLearning\_Cnt\_lgc | FALSE |
| Rte\_InitValue\_DisableOutput\_Cnt\_lgc | FALSE |
| Rte\_InitValue\_HandwheelAuthority\_Uls\_f32 | 0 |
| Rte\_InitValue\_HandwheelPosition\_HwDeg\_f32 | 0 |
| Rte\_InitValue\_HandwheelVelocity\_HwRadpS\_f32 | 0 |
| Rte\_InitValue\_HwTorque\_HwNm\_f32 | 0 |
| Rte\_InitValue\_PullCompCmd\_MtrNm\_f32 | 0 |
| Rte\_InitValue\_SrlComYawRate\_DegpS\_f32 | 0 |
| Rte\_InitValue\_VehicleSpeed\_Kph\_f32 | 0 |
| Rte\_InitValue\_VehicleSpeedRate\_KphpS\_f32 | 0 |
| Rte\_InitValue\_VehicleSpeedValid\_Cnt\_lgc | FALSE |

## Initialization Functions

### Init: ActivePull\_Init1

#### Design Rationale

This initialization function is used to set values that are based solely on calibrations and constants (values which will not change over the course of an ignition cycle). This includes preliminary gain calculations, limits, and step sizes. It also initializes the LTComp\_HwNm\_M\_f32 module-internal variable with the appropriate value from NvM.

#### Module Internal

LTWindUpLimit\_HwNm\_M\_f32 = Min\_m(k\_TotalLimit\_HwNm\_f32, k\_LTLimit\_HwNm\_f32)

STStepSize\_HwNm\_M\_f32 = (D\_STSAMPLETIME\_SEC\_F32 \* k\_STLimit\_HwNm\_f32) / k\_STRampTime\_Sec\_f32

PullCompStepSize\_HwNm\_M\_f32 = k\_OutputMaxRate\_HwNmpS\_f32 \* D\_PULLCOMPSAMPLETIME\_SEC\_F32



## Periodic Functions

### Per: ActivePull\_Per1

#### Design Rationale

The requirements in SF-13 show a signal called Reset\_Svc. This is shown as an input flag to the function. However, the reset service has been implemented as a service call. In order to avoid any thread-based issues, the service sets a separate variable for each periodic (ResetPer1\_Cnt\_M\_lgc, in this case) to TRUE. Then, near the beginning of the execution of the periodic, this value is read. If it has been set to true, it is immediately set to FALSE, and a local copy (ResetSvc\_Cnt\_T\_lgc) is set to TRUE. In this way, each periodic uses its own local copy just as the design dictates using the input signal. The local copy will be set to true for one execution of each periodic function.

The SCom function to set the STComp is done in a similar fashion. The Scom function sets SComSTCompSet\_Cnt\_M\_lgc to TRUE and when ActivePull\_Per1 finds this value set to TRUE, it sets it back to FALSE and uses SComSTComp\_HwNm\_M\_f32 as the state variable (instead of STComp\_HwNm\_M\_f32, as it normally would). The state variable itself is never changed, but the output of the next execution of ActivePull\_Per1 will reflect the new value.

#### Program Flow Start

Rte\_Call\_ActivePull\_Per1\_CP0\_CheckpointReached()

#### Store Module Inputs to Local copies

DisableLearning\_Cnt\_T\_lgc = Rte\_Iread\_ActivePull\_Per1\_DisableLearning\_Cnt\_lgc()

DisableOutput\_Cnt\_T\_lgc = Rte\_Iread\_ActivePull\_Per1\_DisableOutput\_Cnt\_lgc()

HandwheelAuthority\_Uls\_T\_f32 = Rte\_Iread\_ActivePull\_Per1\_HandwheelAuthority\_Uls\_f32()

HandwheelPosition\_HwDeg\_T\_f32 = Rte\_Iread\_ActivePull\_Per1\_HandwheelPosition\_HwDeg\_f32()

HandwheelVelocity\_HwRadpS\_T\_f32 = Rte\_Iread\_ActivePull\_Per1\_HandwheelVelocity\_HwRadpS\_f32()

HwTorque\_HwNm\_T\_f32 = Rte\_Iread\_ActivePull\_Per1\_HwTorque\_HwNm\_f32()

SrlComYawRate\_DegpS\_T\_f32 = Rte\_Iread\_ActivePull\_Per1\_SrlComYawRate\_DegpS\_f32()

VehicleSpeedValid\_Cnt\_T\_lgc = Rte\_Iread\_ActivePull\_Per1\_VehicleSpeedValid\_Cnt\_lgc()

VehicleSpeed\_Kph\_T\_f32 = Rte\_Iread\_ActivePull\_Per1\_VehicleSpeed\_Kph\_f32()

LTComp\_HwNm\_T\_f32 = LTComp\_HwNm\_M\_f32

PrevSTComp\_HwNm\_T\_f32 = STComp\_HwNm\_M\_f32

#### Check for Scom Functions



#### Filter Inputs



#### Active Compensation Enable



#### Determine Enable Learning



#### Short Term Compensation Filter



#### Calculate Integrator Gains



#### Error Integrator & Active Limit



#### Store Local copy of outputs into Module Outputs

HwTrqFilt\_HwNm\_D\_f32 = HwTrqFilt\_HwNm\_T\_f32

YawRateFilt\_DegpS\_D\_f32 = YawRateFilt\_DegpS\_T\_f32

STError\_HwNm\_D\_f32 = STError\_HwNm\_T\_f32

STIntGain\_Uls\_D\_f32 = STIntGain\_Uls\_T\_f32

STReset\_Cnt\_D\_lgc = STReset\_Cnt\_T\_lgc

STComp\_HwNm\_M\_f32 = STComp\_HwNm\_T\_f32

EnableLearn\_Cnt\_M\_lgc = EnableLearning\_Cnt\_T\_lgc

#### Program Flow End

Rte\_Call\_ActivePull\_Per1\_CP1\_CheckpointReached()

### Per: ActivePull\_Per2

#### Design Rationale

The Reset\_Svc functionality is defined in section 6.3.1.1.

#### Program Flow Start

#### Rte\_Call\_ActivePull\_Per2\_CP0\_CheckpointReached()Store Module Inputs to Local copies

VehicleSpeed\_Kph\_T\_f32 = Rte\_Iread\_ActivePull\_Per2\_VehicleSpeed\_Kph\_f32()

DisableOutput\_Cnt\_T\_lgc = Rte\_Iread\_ActivePull\_Per2\_DisableOutput\_Cnt\_lgc()

PrevPullCompCmd\_HwNm\_T\_f32 = PullCompCmd\_HwNm\_M\_f32

STComp\_HwNm\_T\_f32 = STComp\_HwNm\_M\_f32

LTComp\_HwNm\_T\_f32 = LTComp\_HwNm\_M\_f32

#### Check for Reset



#### Calculate Active Compensation



#### Store Local copy of outputs into Module Outputs

PullCompCmd\_HwNm\_M\_f32 = PullCompCmd\_HwNm\_T\_f32

Rte\_Iwrite\_ActivePull\_Per2\_PullCompCmd\_MtrNm\_f32(PullCompCmd\_MtrNm\_T\_f32)

#### Program Flow End

### Rte\_Call\_ActivePull\_Per2\_CP1\_CheckpointReached()Per: ActivePull\_Per3

#### Design Rationale

The Reset\_Svc and state variable functionality are defined in section 6.3.1.1. Note that the Scom functions will have no effect until the next execution of ActivePull\_Per3, which could result in a propagation delay of up to 100 ms.

#### Program Flow Start

#### Rte\_Call\_ActivePull\_Per3\_CP0\_CheckpointReached()Store Module Inputs to Local copies

VehSpd\_Kph\_T\_f32 = Rte\_IRead\_ActivePull\_Per3\_VehicleSpeed\_Kph\_f32()

HwTorque\_HwNm\_T\_f32 = Rte\_Iread\_ActivePull\_Per3\_HwTorque\_HwNm\_f32()

EnableLearning\_Cnt\_T\_lgc = EnableLearn\_Cnt\_M\_lgc

STComp\_HwNm\_T\_f32 = STComp\_HwNm\_M\_f32

PrevLTComp\_HwNm\_T\_f32 = LTComp\_HwNm\_M\_f32

#### Check for Scom Functions



#### Long Term Compensation Filter



#### Error Integrator



#### Store Local copy of outputs into Module Outputs

LTError\_HwNm\_D\_f32 = LTError\_HwNm\_T\_f32

LTComp\_HwNm\_M\_f32 = LTComp\_HwNm\_T\_f32

#### Program Flow End

Rte\_Call\_ActivePull\_Per3\_CP1\_CheckpointReached()

## Fault Recovery Functions

None

## Shutdown Functions

None

## Interrupt Functions

None

## Serial Communication Functions

### Scom: ActivePull\_Scom\_Reset

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  | Type | Min | Max | UTP Tol. |
| **Arguments Passed** | None |  |  |  |  |
| **Return Value** | None |  |  | |  |

#### Design Rationale

#### Program Flow Start

N/A

#### Store Module Inputs to Local copies

#### Reset Service



#### Store Local copy of outputs into Module Outputs

None

#### Program Flow End

N/A

### Scom: ActivePull\_Scom\_SetLTComp

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  | Type | Min | Max | UTP Tol. |
| **Arguments Passed** | LTComp\_HwNm\_f32 | float32 | -10 | 10 |  |
| **Return Value** | None |  |  |  |  |

#### Design Rationale

This function helps to fulfill the requirement that the “Engineering interface tool shall provide ability to set state variable to desired value”. The state variable itself will not be updated until the next time ActivePull\_Per3 is run, but the NvM value is updated immediately.

#### Program Flow Start

N/A

#### Store Module Inputs to Local copies

None

#### 



#### Store Local copy of outputs into Module Outputs

None

#### Program Flow End

N/A

### Scom: ActivePull\_Scom\_SetSTComp

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  | Type | Min | Max | UTP Tol. |
| **Arguments Passed** | STComp\_HwNm\_f32 | float32 | -10 | 10 |  |
| **Return Value** | None |  |  |  |  |

#### Design Rationale

This function helps to fulfill the requirement that the “Engineering interface tool shall provide ability to set state variable to desired value”. The state variable itself will not be updated until the next time ActivePull\_Per1 is run.

#### Program Flow Start

N/A

#### Store Module Inputs to Local copies

None

#### 



#### Store Local copy of outputs into Module Outputs

None

#### Program Flow End

N/A

### Scom: ActivePull\_Scom\_ReadParam

#### Program Flow Start

N/A

#### Store Module Inputs to Local copies

None

#### Read Data

\*PullCompCmd\_HwNm\_f32 = PullCompCmd\_HwNm\_M\_f32

\*STComp\_HwNm\_f32 = STComp\_HwNm\_M\_f32

\*LTComp\_HwNm\_f32 = LTComp\_HwNm\_M\_f32

\*EnableLearn\_Cnt\_lgc = EnableLearn\_Cnt\_M\_lgc

#### Store Local copy of outputs into Module Outputs

None

#### Program Flow End

N/A

## Transition Functions

### Trns: ActivePull\_Trns1

#### Design Rationale

This function is run when entering the OPERATE state. The LTComp NvM block is set to write on shutdown (as ActivePull\_Per3 will be updating the NvM block), and the timers associated with ActivePull\_Per1 are initialized.

#### Program Flow Start

N/A

#### Store Module Inputs to Local copies

None

#### Initialization



#### Store Local copy of outputs into Module Outputs

None

#### Program Flow End

N/A

# Execution Requirements

## Execution Sequence of the Module

ActivePull\_Per1 and Per2 are run at 2ms intervals, while Per3 is run at a 100ms interval. However, while Per2 is run in all operation states, Per1 and Per3 are only run in the OPERATE state. ActivePull\_Trns1 is run upon entering the OPERATE state.

## Execution Rates for sub-modules called by the Scheduler

This table serves as reference for the Scheduler design

|  |  |  |
| --- | --- | --- |
| Function Name | Calling Frequency | System State(s) in which the function is called |
| ActivePull\_Init1 | On Event (once) | On Entering WARMINIT |
| ActivePull\_Per1 | 2 ms | OPERATE |
| ActivePull\_Per2 | 2 ms | All |
| ActivePull\_Per3 | 100 ms | OPERATE |
| ActivePull\_Trns1 | On Event | On Entering OPERATE |

## Execution Requirements for Serial Communication Functions

|  |  |
| --- | --- |
| Function Name | Sub-Module called by (Serial Comm Function Name) |
| ActivePull\_Scom\_Reset |  |
| ActivePull\_Scom\_SetLTComp |  |
| ActivePull\_Scom\_SetSTComp |  |
| ActivePull\_Scom\_ReadParam |  |

# Memory Map Definition Requirements

## Sub Modules (Functions)

This table identifies the software segments for functions identified in this module.

|  |  |
| --- | --- |
| Name of Sub Module | Software Segment |
| ActivePull\_Init1 | RTE\_START\_SEC\_AP\_ACTIVEPULL\_APPL\_CODE |
| ActivePull\_Per1 | RTE\_START\_SEC\_AP\_ACTIVEPULL\_APPL\_CODE |
| ActivePull\_Per2 | RTE\_START\_SEC\_AP\_ACTIVEPULL\_APPL\_CODE |
| ActivePull\_Per3 | RTE\_START\_SEC\_AP\_ACTIVEPULL\_APPL\_CODE |
| ActivePull\_Trns1 | RTE\_START\_SEC\_AP\_ACTIVEPULL\_APPL\_CODE |
| ActivePull\_Scom\_Reset | RTE\_START\_SEC\_AP\_ACTIVEPULL\_APPL\_CODE |
| ActivePull\_Scom\_SetLTComp | RTE\_START\_SEC\_AP\_ACTIVEPULL\_APPL\_CODE |
| ActivePull\_Scom\_SetSTComp | RTE\_START\_SEC\_AP\_ACTIVEPULL\_APPL\_CODE |
| ActivePull\_Scom\_ReadParam | RTE\_START\_SEC\_AP\_ACTIVEPULL\_APPL\_CODE |

## Local Functions

This table identifies the software segments for local functions identified in this module.

|  |  |
| --- | --- |
| Name of Sub Module | Software Segment |
|  |  |

# Known Issues / Limitations With Design

1. INLINE functions defined in “GlobalMacro.h” are not unit tested

# Revision Control Log

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Item #** | **Rev #** | **Change Description** | **Date** | **Author Initials** |
| 1 | 1.0 | Initial MDD | 01Aug11 | LWW |
| 2 | 2.0 | Updated to SF-13 rev 001 (started from scratch) | 02-Apr-12 | OT |
| 3 | 3.0 | Fixed buffered reads in Reset Scom function (changed to direct reads) | 18-Apr-12 | OT |
| 4 | 4.0 | Removed PIM from Scom and made LT learned variable to a typH. Added support for FDAD Common manufacturing srvc DID | 22-Apr-12 | VK |
| 5 | 5.0 | Updates to meet SF-13 rev 002 | 26-June-12 | VK |
| 6 | 6.0 | Corrected module internal variable ranges | 29-June-12 | VK |
| 7 | 7.0 | 1) Removed VehSpdRate global input and made necessary changes in Per1  2) Added VehicleSpeedRate logic in Per3 -Ver 003 updates  3) Changed LPF from fixed to float | 23-July-12 | NRAR |
| 8 | 8.0 | Inserted safe watchdog checkpoints | 15-Sept-12 | BWL |
| 9 | 9.0 | Corrected static variable to MDD format | 18-sep-12 | SSK |
| 10 | 10.0 | Updated calibration table Y datatype to u2p14 for anomaly correction, removed condition checks on SCom function | 20 Oct 12 | LWW |
| 11 | 11.0 | Anomaly 5379 fixed. | 07-Aug-13 | SP |
| 12 | 12.0 | Anomaly 5764 (to revert changes made as part of the previous Anomaly 5379 fix) | 16-Apr-14 | LK |