**Module Design Document**

**For**

**Active Pull Compensation**

**Oct 20, 2017**

**Prepared By:**

**Matthew Leser,**

**Nexteer Automotive,**

**Saginaw, MI, USAChange History**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Sl. No.** | **Description** | **Author** | **Version** | **Date** |
| 1 | Initial MDD | LWW | 1.0 | 01Aug11 |
| 2 | Updated to SF-13 rev 001 (started from scratch) | OT | 2.0 | 02-Apr-12 |
| 3 | Fixed buffered reads in Reset Scom function (changed to direct reads) | OT | 3.0 | 18-Apr-12 |
| 4 | Removed PIM from Scom and made LT learned variable to a typH. Added support for FDAD Common manufacturing srvc DID | VK | 4.0 | 22-Apr-12 |
| 5 | Updates to meet SF-13 rev 002 | VK | 5.0 | 26-June-12 |
| 6 | Corrected module internal variable ranges | VK | 6.0 | 29-June-12 |
| 7 | 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 | NRAR | 7.0 | 23-July-12 |
| 8 | Inserted safe watchdog checkpoints | BWL | 8.0 | 15-Sept-12 |
| 9 | Corrected static variable to MDD format | SSK | 9.0 | 18-sep-12 |
| 10 | Updated calibration table Y datatype to u2p14 for anomaly correction, removed condition checks on SCom function | LWW | 10.0 | 20 Oct 12 |
| 11 | Anomaly 5379 fixed. | SP | 11.0 | 07-Aug-13 |
| 12 | Anomaly 5764 (to revert changes made as part of the previous Anomaly 5379 fix) | LK | 12.0 | 16-Apr-14 |
| 13 | Updated to version 4 and 5 of FDD - Added estimated lateral acceleration, Pull command Enable as a new inputs. And Updated to new template | SB | 13.0 | 09-Dec-15 |
| 14 | Updated the SCom function arguments section | SB | 14.0 | 02-May-16 |
| 15 | Added new Trans function | SB | 15.0 | 19-May-16 |
| 17 | Updated to SF13A FDD version 6.0.0 | SB | 17.0 | 05-Aug-16 |
| 18 | Updated to SF13A FDD version 6.2.0 | SR | 18.0 | 03-Oct-16 |
| 19 | Updated to SF13A FDD version 6.3.0 | KByrski | 19.0 | 24-Jan-17 |
| 20 | Updated Diagram | ML | 20.0 | 20-Oct-17 |

Table of Contents

[1 ActivePull & High-Level Description 4](#_Toc463264832)

[2 Design details of software module 5](#_Toc463264833)

[2.1 Graphical representation of ActivePull 5](#_Toc463264834)

[2.2 Data Flow Diagram 5](#_Toc463264835)

[2.2.1 Module level DFD 5](#_Toc463264836)

[2.2.2 Sub-Module level DFD 5](#_Toc463264837)

[2.3 Component diagram 5](#_Toc463264838)

[2.4 Variable Data Dictionary 5](#_Toc463264839)

[2.4.1 User defined ‘typedef’ definition/declaration 5](#_Toc463264840)

[2.4.2 Variable definition for enumerated types 5](#_Toc463264841)

[2.5 Constant Data Dictionary 6](#_Toc463264842)

[2.5.1 Program Constants 6](#_Toc463264843)

[2.5.2 Module Specific Lookup Tables 6](#_Toc463264844)

[2.6 Software Module Implementation 6](#_Toc463264845)

[2.6.1 Sub-Module Functions 6](#_Toc463264846)

[2.6.2 Interrupt Service Routines 8](#_Toc463264847)

[2.6.3 \_SCOMM () Functions 8](#_Toc463264848)

[2.6.4 Module Internal (Local) Functions 10](#_Toc463264849)

[2.6.5 Transition Functions 10](#_Toc463264850)

[3 Known Limitations with Design 12](#_Toc463264851)

[4 UNIT TEST CONSIDERATION 13](#_Toc463264852)

[Appendix A Abbreviations and Acronyms 14](#_Toc463264853)

[Appendix B Glossary 15](#_Toc463264854)

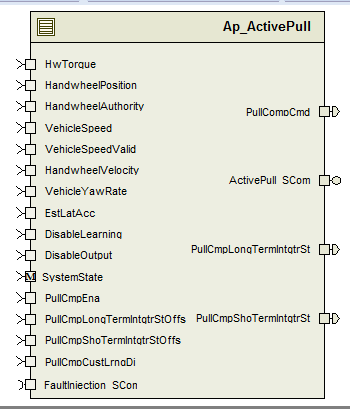
[Appendix C References 16](#_Toc463264855)

# ActivePull & High-Level Description

*The Active Pull Compensation Function corrects vehicle pull issues by compensating for HW torque offsets detected by the steering system. These torque offsets are classified as short-term and long-term, each of which is compensated for independently by the algorithm. When the compensation is applied, the need for the driver to provide a constant input torque to counter these offsets is reduced.*

# Design details of software module

## Graphical representation of ActivePull



## Data Flow Diagram

### Module level DFD

N/A

### Sub-Module level DFD

N/A

## Component diagram

N/A

## Variable Data Dictionary

### User defined ‘typedef’ definition/declaration

N/A

### Variable definition for enumerated types

N/A

## Constant Data Dictionary

### Program Constants

#### Local Constants

Refer .m file

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.

#### Global Constants

Refer .m file

### Module Specific Lookup Tables

None

## Software Module Implementation

<The detailed design of the function is provided in the FDD. The detail design shall only be add to the MDD when it is not provided in the FDD or the FDD is not adequate and clarification is needed.>

### Sub-Module Functions

#### Initialization sub-module 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**Store Module Inputs to Local copies**

*Refer to FDD*

##### **(Processing of function)………**

*Refer to FDD*

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

*Refer to FDD*

#### Periodic sub-module ActivePull\_Per1()

Refer ‘ActvPullCmpPer1’ block in the Simulink model.

##### **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.

##### **Store Module Inputs to Local copies**

Refer ‘ActvPullCmpPer1’ block in the Simulink model.

##### **(Processing of function)………**

Refer ‘ActvPullCmpPer1’ block in the Simulink model.

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

Refer ‘ActvPullCmpPer1’ block in the Simulink model.

#### Periodic sub-module ActivePull\_Per2()

Refer ‘ActvPullCmpPer2’ block in the Simulink model.

##### **Design Rationale**

None.

##### **Store Module Inputs to Local copies**

Refer ‘ActvPullCmpPer2’ block in the Simulink model.

##### **(Processing of function)………**

Refer ‘ActvPullCmpPer2’ block in the Simulink model.

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

Refer ‘ActvPullCmpPer2’ block in the Simulink model.

#### Periodic sub-module ActivePull\_Per3()

Refer ‘ActvPullCmpPer3’ block in the Simulink model.

##### **Design Rationale**

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.

##### **Store Module Inputs to Local copies**

*Refer ‘ActvPullCmpPer3’ block in the Simulink model.*

##### **(Processing of function)………**

*Refer ‘ActvPullCmpPer3’ block in the Simulink model.*

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

*Refer ‘ActvPullCmpPer3’ block in the Simulink model.*

#### Non Periodic sub-module {\_NONPer()}

None

### Interrupt Service Routines

None

### \_SCOMM () Functions

#### ActivePull\_Scom\_ReadParam ()

*Refer ‘ActivePull\_SCom\_ReadParam’ block in the Simulink model.*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  | Type | Min | Max | UTP Tol. |
| **Arguments Passed** | PullCompCmd\_HwNm\_f32  STComp\_HwNm\_f32  LTComp\_HwNm\_f32  EnableLearn\_Cnt\_lgc | Float32  Float32  Float32  Float32 | -8.8  -10  -10  FALSE | 8.8  10  10  TRUE |  |
| **Return Value** | None |  |  | |  |

##### **Design Rationale**

None

##### **Store Module Inputs to Local copies**

*Refer ‘ActivePull\_SCom\_ReadParam’ block in the Simulink model.*

##### **(Processing of function)………**

*Refer ‘ActivePull\_SCom\_ReadParam’ block in the Simulink model.*

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

*Refer ‘ActivePull\_SCom\_ReadParam’ block in the Simulink model.*

#### ActivePull\_Scom\_Reset ()

*Refer ‘ActivePull\_SCom\_Reset’ block in the Simulink model.*

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

##### **Design Rationale**

None

##### **Store Module Inputs to Local copies**

*Refer ‘ActivePull\_SCom\_Reset’ block in the Simulink model.*

##### **(Processing of function)………**

*Refer ‘ActivePull\_SCom\_Reset’ block in the Simulink model.*

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

*Refer ‘ActivePull\_SCom\_Reset’ block in the Simulink model.*

#### ActivePull\_Scom\_SetLTComp ()

*Refer ‘ActivePull\_SCom\_Reset’ block in the Simulink model.*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  | 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.

##### **Store Module Inputs to Local copies**

*Refer ‘ActivePull\_SCom\_SetLTComp’ block in the Simulink model.*

##### **(Processing of function)………**

*Refer ‘ActivePull\_SCom\_SetLTComp’ block in the Simulink model.*

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

*Refer ‘ActivePull\_SCom\_SetLTComp’ block in the Simulink model.*

#### ActivePull\_Scom\_SetSTComp ()

*Refer ‘ActivePull\_SCom\_SetSTComp’ block in the Simulink model.*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  | 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.

##### **Store Module Inputs to Local copies**

*Refer ‘ActivePull\_SCom\_SetSTComp’ block in the Simulink model.*

##### **(Processing of function)………**

*Refer ‘ActivePull\_SCom\_SetSTComp’ block in the Simulink model.*

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

*Refer ‘ActivePull\_SCom\_SetSTComp’ block in the Simulink model.*

### Module Internal (Local) Functions

#### Local Function ActvCmpEna\_lgc

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Function Name** | ActvCmpEna\_lgc | Type | Min | Max |
| **Arguments Passed** | HwTrqFilt\_HwNm\_T\_f32,  YawRateFilt\_DegpS\_T\_f32,  HandwheelPosition\_HwDeg\_T\_f32,  EstLatAcc\_MpSecSq\_T\_f32,  HandwheelVelocity\_HwRadpS\_T\_f32,  HandwheelAuthority\_Uls\_T\_f32,  VehicleSpeed\_Kph\_T\_f32,  VehicleSpeedValid\_Cnt\_T\_lgc,  DisableLearning\_Cnt\_T\_lgc,  DisableOutput\_Cnt\_T\_lgc  PullCmpCustLrngDi\_Cnt\_T\_lgc | Float32  Float32  Float32  Float32  Float32  Float32  Float32  Boolean  Boolean  Boolean  Boolean | -10  -120  -1440.11  -10  -42  0  0  FALSE  FALSE  FALSE  FALSE | 10  120  1440.11  10  42  1  511  TRUE  TRUE  TRUE  TRUE |
|  |  |  |  |  |
| **Return Value** | EnableLearning\_Cnt\_T\_lgc | Boolean | FALSE | TRUE |

##### **Design Rationale**

Refer "Active Compensation Enable" Block in Simulink Model

Removed the display variable “PullCmp\_STReset\_Cnt\_D\_lgc” from the code because it is redundant with the module level variable “PullCmp\_STReset\_Cnt\_M\_lgc”.

#### Local Function CalcIntGain\_f32

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Function Name** | CalcIntGain\_f32 | Type | Min | Max |
| **Arguments Passed** | HwTorque\_HwNm\_T\_f32,  PrevSTComp\_HwNm\_T\_f32 | Float32  Float32 | -10  -10 | 10  10 |
|  |  |  |  |  |
| **Return Value** | STIntGain\_Uls\_T\_f32 | Float32 | 0 | 1 |

##### **Design Rationale**

Refer " Calculate Integrator Gains" Block in Simulink Model

### Transition Functions

#### ActivePull\_Trns1()

*Refer ‘ActivePull\_Trns1’ block in the Simulink model.*

##### **Design Rationale**

This function is run when entering the OPERATE state. The timers associated with ActivePull\_Per1 are initialized.

##### **Store Module Inputs to Local copies**

*Refer ‘ActivePull\_Trns1’ block in the Simulink model.*

##### **(Processing of function)………**

*Refer ‘ActivePull\_Trns1’ block in the Simulink model.*

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

*Refer ‘ActivePull\_Trns1’ block in the Simulink model.*

#### \_Trns2()

*Refer ‘ActivePull\_Trns2’ block in the Simulink model.*

##### **Design Rationale**

This function is run when entering the OFF state.

##### **Store Module Inputs to Local copies**

*Refer ‘ActivePull\_Trns2’ block in the Simulink model.*

##### **(Processing of function)………**

*Refer ‘ActivePull\_Trns2’ block in the Simulink model.*

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

*Refer ‘ActivePull\_Trns2’ block in the Simulink model.*

# Known Limitations with Design

None

# UNIT TEST CONSIDERATION

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

Abbreviations and Acronyms

| **Abbreviation or Acronym** | **Description** |
| --- | --- |
|  |  |
|  |  |

Glossary

**Note**: Terms and definitions from the source “Nexteer Automotive” take precedence over all other definitions of the same term. Terms and definitions from the source “Nexteer Automotive” are formulated from multiple sources, including the following:

* ISO 9000
* ISO/IEC 12207
* ISO/IEC 15504
* Automotive SPICE® Process Reference Model (PRM)
* Automotive SPICE® Process Assessment Model (PAM)
* ISO/IEC 15288
* ISO 26262
* IEEE Standards
* SWEBOK
* PMBOK
* Existing Nexteer Automotive documentation

| **Term** | **Definition** | **Source** |
| --- | --- | --- |
| MDD | Module Design Document |  |
| DFD | Data Flow Diagram |  |

References

| **Ref. #** | **Title** | **Version** |
| --- | --- | --- |
| 1 | AUTOSAR Specification of Memory Mapping (Link:[AUTOSAR\_SWS\_MemoryMapping.pdf](http://www.autosar.org/download/R4.0/AUTOSAR_SWS_MemoryMapping.pdf)) | v1.3.0 R4.0 Rev 2 |
| 2 | MDD Guideline | Process release 04.02.01 |
| 3 | [Software Naming Conventions.doc](http://misagweb01.nexteer.com/eRoomReq/Files/erooms8/NextGeneration/0_fc55f/Software%20Naming%20Conventions%2003x(In%20Work).doc) | 2.0 |
| 4 | [Software Design and Coding Standards.doc](http://eroom1.nexteer.com/eRoomReq/Files/erooms8/NextGeneration/0_1a67a9/Software%20Design%20and%20Coding%20Standards.doc) | 2.1 |
| 5 | FDD SF013A\_PullCmpActv\_Design | 6.3.0 |