# Module --

# High-Level Description

This module implements the polarity assignments for the EPS systems to allow for various configurations of input and output signals for proper alignment.

# Figures

## Diagram – Component Diagram

Polarity.tif

## Diagram – Function Data Sharing

No shared data.

# 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 | |
| DiagHwTrqPolarity\_Cnt\_s08 | | HwTrqPolarity\_Cnt\_s08 |
| DiagHwPosPolarity\_Cnt\_s08 | | HwPosPolarity\_Cnt\_s08 |
| DiagMtrPosPolarity\_Cnt\_s08 | | MtrPosPolarity\_Cnt\_s08 |
| DiagMtrVelPolarity\_Cnt\_s08 | | MtrVelPolarity\_Cnt\_s08 |
| DiagMtrElecMechPolarity\_Cnt\_s08 | | MtrElecMechPolarity\_Cnt\_s08 |
| DiagAssistAssyPolarity\_Cnt\_s08 | | AssistAssyPolarity\_Cnt\_s08 |
|  | | SysC\_ MtrElecMechPolarity\_Cnt\_s32 |

## 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 | Data Type | Resolution | Legal Range  (min) | Legal Range  (max) | Software Segment |
| Polarity\_Cnt\_Str | Polarity\_DataType | N/A | See DataType | See DataType | \*\*NvM/Per-Instance Memory\*\* |

### 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) |
| Polarity\_DataType | Polarity\_Cnt\_b08 | uint8 | Full | Full |
|  | R\_Polarity\_Cnt\_b08 | uint8 | Full | Full |

# 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 |
|  |

## 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\_HWTRQPOL\_CNT\_B08 | 1 | Counts | 0x01 |
| D\_HWPOSPOL\_CNT\_B08 | 1 | Counts | 0x02 |
| D\_MTRPOSPOL\_CNT\_B08 | 1 | Counts | 0x04 |
| D\_MTRVELPOL\_CNT\_B08 | 1 | Counts | 0x08 |
| D\_ASSTASSEMPOL\_CNT\_B08 | 1 | Counts | 0x10 |
| D\_MTRELECMECHPOL\_CNT\_B08 | 1 | Counts | 0x20 |

#### 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\_NEGONE\_CNT\_S8 |
| D\_ONE\_CNT\_S8 |

### Module specific Lookup Tables Constants

(This is for lookup tables (arrays) with fixed values, same name as other tables)

|  |  |  |  |
| --- | --- | --- | --- |
| 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. None

## Data Hiding Functions

1. Rte\_Pim\_Polarity\_Cnt\_Str()
2. Rte\_Call\_Polarity\_WriteBlock()

## Global Functions/Macros Defined by this Module

N/A

## Local Functions/Macros Used by this MDD only

### GetPolarity

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Function Name** | GetPolarity | Type | Min | Max | UTP Tol. |
| **Arguments Passed** | Polarity\_Cnt\_T\_b08 | uint8 | FULL | FULL |  |
|  | PolarityMask\_Cnt\_T\_b08 | uint8 | FULL | FULL |  |
| **Return Value** | Polarity\_Cnt\_T\_s08 | sint8 | -1 | 1 | 0 |

#### Description



# 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 |
| AssistAssyPolarity\_Cnt\_s08 | 0 |
| HwPosPolarity\_Cnt\_s08 | 0 |
| HwTrqPolarity\_Cnt\_s08 | 0 |
| MtrElecMechPolarity\_Cnt\_s08 | 0 |
| MtrPosPolarity\_Cnt\_s08 | 0 |
| MtrVelPolarity\_Cnt\_s08 | 0 |
| SysC\_ MtrElecMechPolarity\_Cnt\_s32 | 0 |
| DiagHwTrqPolarity\_Cnt\_s08 | 0 |
| DiagHwPosPolarity\_Cnt\_s08 | 0 |
| DiagMtrPosPolarity\_Cnt\_s08 | 0 |
| DiagMtrVelPolarity\_Cnt\_s08 | 0 |
| DiagMtrElecMechPolarity\_Cnt\_s08 | 0 |
| DiagAssistAssyPolarity\_Cnt\_s08 | 0 |

## Initialization Functions

### Init: \_Init1

#### Design Rationale

During cold initialization, set all output ports to their values based on the stored polarity calibration.

#### Program Flow Start

N/A

#### Store Module Inputs to Local Copies

N/A

#### Processing



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

N/A

#### Program Flow End

N/A

## Periodic Functions

### Per: Polarity\_Per1

#### Design Rationale

This function is checking the polarity signals sent out to the application against the redundantly stored polarity bitfield. The polarity outputs need to be fed back into this module as an input to insure the check is done on the exact memory locations that are being used during run-time to compute the polarity adjusted signals.

#### Program Flow Start

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

#### Store Module Inputs to Local copies

*HwTrqPolarity\_Cnt\_T\_s08* = Rte\_IRead\_Polarity\_Per1\_DiagHwTrqPolarity\_Cnt\_s08()

*HwPosPolarity\_Cnt\_T\_s08* = Rte\_IRead\_Polarity\_Per1\_DiagHwPosPolarity\_Cnt\_s08()

*MtrPosPolarity\_Cnt\_T\_s08* = Rte\_IRead\_Polarity\_Per1\_DiagMtrPosPolarity\_Cnt\_s08()

*MtrVelPolarity\_Cnt\_T\_s08* = Rte\_IRead\_Polarity\_Per1\_DiagMtrVelPolarity\_Cnt\_s08()

*AssistAssyPolarity\_Cnt\_T\_s08* = Rte\_IRead\_Polarity\_Per1\_DiagAssistAssyPolarity\_Cnt\_s08()

*MtrElecMechPolarity\_Cnt\_T\_s08* = Rte\_IRead\_Polarity\_Per1\_DiagMtrElecMechPolarity\_Cnt\_s08()

#### Processing



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

None

#### Program Flow End

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

## Serial Communication Functions

### SCom: Polarity\_SCom\_ReadPolarity

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Function Name** | Polarity\_SCom\_ReadPolarity | Type | Min | Max |
| **Arguments Passed** | Polarity\_Ptr\_T\_b08 | uint8 pointer | FULL | FULL |
| **Return Value** | N/A |  |  |  |

#### Design Rationale

Read and return current polarity calibration to the testing tool.

#### Program Flow Start

N/A

#### Store Module Inputs to Local copies

N/A

#### (Processing of function)



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

N/A

#### Program Flow End

N/A

### SCom: Polarity\_SCom\_SetPolarity

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Function Name** | Polarity\_SCom\_SetPolarity | Type | Min | Max |
| **Arguments Passed** | Polarity\_Cnt\_T\_b08 | uint8 | FULL | FULL |
| **Return Value** | N/A |  |  | |

### Design Rationale

Suspend and Resume all interrupts are used around the updating of the polarity outputs to prevent the function running the diagnostics (Per1) from interrupting this function before all of the outputs are updated.

#### Program Flow Start

N/A

#### Store Module Inputs to Local copies

N/A

#### Processing



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

N/A

#### Program Flow End

N/A

# Execution Requirements

## Execution Sequence of the Module

N/A

## 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 |
| Polarity\_Init1 | Executed Once after RTE is started | ColdInit |
| Polarity\_Per1 | 10ms | All |

## Execution Requirements for Serial Communication Functions

|  |  |
| --- | --- |
| Function Name | Sub-Module called by (Serial Comm Function Name) |
| Polarity\_SCom\_ReadPolarity |  |
| Polarity\_SCom\_SetPolarity |  |

# 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 |
| Polarity\_Init1 |  |
| Polarity\_Per1 |  |
| Polarity\_SCom\_ReadPolarity |  |
| Polarity\_SCom\_SetPolarity |  |

## Local Functions

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

|  |  |
| --- | --- |
| Name of Sub Module | Software Segment |
| GetPolarity | Inline with calling function |

# Known Issues / Limitations With Design

# Additional Comments

1. The polarity bits for hand wheel torque, hand wheel position, motor position, and motor velocity (bits 0 - 4) are assumed to be used at the “sensor” level to place the signals in the proper orientation for use within the ECU. For example, hand wheel torque bit should be used to get the torque signals into the Nexteer CRF described in the polarity FDD-25.
2. Porting a signal from CRF to MRF, or from MRF to CRF, shall be done with the assist assembly polarity bit (bit 5).

# Revision Control Log

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Item #** | **Rev #** | **Change Description** | **Date** | **Author Initials** |
| 1 | 1 | Initial MDD | 23May11 | NRAR |
| 2 | 2 | Carried over from for component based design | 14Nov11 | JWW |
| 3 | 3 | Updates and initial release for component based design | 21Apr2012 | KJS |
| 4 | 4 | Update for GSOD redundant output – FDDv006 | 22-Oct-12 | JWJ |
| 7 | 7 | Updated per FDD requirements | 19-Feb-13 | LWW |