# Module -- NvM Proxy

# High-Level Description

(Description must be within 8-10 lines.)

# Figures

## Diagram – Function Data Sharing

This diagram depicts the physical memory allocation for the various parts of the NvM Proxy system. 3 application RAM areas are shown for illustrative purposes, however, this module can handle any number of application RAM areas.

The memory stack components below the NvM are not shown in this diagram to promote clarity.

The NvMProxy\_CmdQueue is required to be allocated to global shared memory to provide write access to the Proxy server function that is designed to be called from any application.



### Diagram – NvM Data Initialization

Depiction of the Nv Data initialization sequence from the perspective of which application is active (i.e. MPU configuration at the time of operation execution)

Only pertinent initialization functions and steps are shown to promote clarity.



### Diagram – NvM Runtime

Following is a depiction of the write Motor Position EOL calibrations via diagnostic service request. The MtrPos component is assumed to be running in the ASIL D application and its server runnable for processing an EOL motor cal write request is assumed to invoke the NvM\_WriteBlock operation.

The lifelines in this diagram represent execution within the Os or an application. The details of the diagnostic service request are omitted from this diagram for clarity purposes.



# 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 | |
| Configured by NvMProxyCfg | | None |
|  | |  |

## 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 |
| NvMPWriteRqst\_Cnt\_M\_Str[D\_NUMPRXYBLOCKS\_CNT\_U16] | See NvMPWriteBuff\_Type | See NvMPWriteBuff\_Type | See NvMPWriteBuff\_Type | NVMPROXY\_START\_SEC\_VAR\_CLEARED\_UNSPECIFIED |
| NvMPSetRBSRqst\_Cnt\_M\_Str[D\_NUMPRXYBLOCKS\_CNT\_U16] | See NvMPSetRBSBuff\_Type | See NvMPSetRBSBuff\_Type | See NvMPSetRBSBuff\_Type | NVMPROXY\_START\_SEC\_VAR\_CLEARED\_UNSPECIFIED |
| NvMPBlkShtdwnSts\_Cnt\_M\_lgc[D\_NUMPRXYBLOCKS\_CNT\_U16] | 1 | 0 | 1 | NVMPROXY\_START\_SEC\_VAR\_CLEARED\_UNSPECIFIED |

### 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) |
| NvMProxyCfg\_Type | NvMBlock | NvM\_BlockIdType | 0 | FULL |
|  | unsecurePtr | constant uint8\* to variable data | NA | NA |
|  | securePtr | constant uint8\* to variable data | NA | NA |
|  | secureSize | uint16 | 0 | FULL |
|  | initHandling | NvMProxy\_InitHandling | See Datatype | See Datatype |
|  | failResponse | NvMProxy\_FailResponse | See Datatype | See Datatype |
|  | failActData | NvMP\_FailActionDataType | See Datatype | See Datatype |
|  | failActFunc | NvMP\_FailActionFuncType | See Datatype | See Datatype |
| NvMProxy\_InitHandling | NVMPROXY\_NONE | uint8 | 0 | 0 |
|  | NVMPROXY\_CRC16 | uint8 | 1 | 1 |
|  | NVMPROXY\_REDUNDANT | uint8 | 2 | 2 |
|  | NVMPROXY\_ZERODATA | uint8 | 3 | 3 |
| NvMProxy\_FailResponse | NVMPROXY\_NOTAPPLICABLE | uint8 | 0 | 0 |
|  | NVMPROXY\_NTC\_0A | uint8 | 1 | 1 |
|  | NVMPROXY\_NTC\_08\_ROMDEF | uint8 | 2 | 2 |
|  | NVMPROXY\_NTC\_08\_NOTIFFUNC | uint8 | 3 | 3 |
|  | NVMPROXY\_NTC\_07\_ROMDEF | uint8 | 4 | 4 |
|  | NVMPROXY\_NTC\_07\_NOTIFFUNC | uint8 | 5 | 5 |
|  | NVMPROXY\_NTC\_06\_ROMDEF | uint8 | 6 | 6 |
|  | NVMPROXY\_NTC\_06\_NOTIFFUNC | uint8 | 7 | 7 |
| NvMP\_FailActFuncType | Pointer to void function | pointer | N/A | N/A |
| NvMP\_FailActionDataType | Pointer to uint8 | pointer | N/A | N/A |
| NvMPWriteBuff\_Type | Pend | boolean | 0 | FULL |
|  | BlkStatus | NvM\_RequestResultType | 0 | FULL |
|  | SrcPtr | uint8\* | NA | NA |
| NvMPSetRBSBuff\_Type | Pend | boolean | 0 | FULL |
|  | BlockChanged | boolean | NA | NA |

# 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 |
| <None> |
|  |

## Configuration Constants

This section lists the configuration constants used by the module. For details on configuration constants, refer to the Module User Guide. The values are set by the integration project specific configuration files Cd\_NvMProxy\_Cfg.h and Cd\_NvMProxy\_PBcfg.c

|  |  |
| --- | --- |
| Constant Name | Type |
| NvMProxyCfg [D\_NUMPRXYBLOCKS\_CNT\_U16] | NvMProxyCfg\_Type |

## 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\_NUMPRXYBLOCKS\_CNT\_U16 | 1 | Count | Configured in integration project |
| NVMPROXY\_EXCLUSIVE\_AREA\_0 | NA | NA | Generated by SchM |
| D\_CRC16SIZE\_CNT\_U16 | 1 | Count | 2 |

#### 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 |
| <None> |
|  |

### 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,

## Data Hiding Functions

1. NvM\_WriteAll()
2. NvM\_WriteBlock()
3. NvM\_SetRamBlockStatus()
4. NvM\_GetErrorStatus()
5. SchM\_Enter\_NvMProxy()
6. SchM\_Exit\_NvMProxy()

## Global Functions/Macros Defined by this Module

### NvMProxy\_WriteAll

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Function Name** | NvMProxy\_WriteAll | Type | Min | Max | UTP Tol. |
| **Arguments Passed** | None |  |  |  |  |
| **Return Value** | N one |  |  |  |  |

#### Description

This function implements the AUTOSAR standard API for the NvM BSW WriteAll service. The interface must adhere to the AUTOSAR standard to allow mapping service port needs of SWC’s located in outside the QM application to this component’s service interface via the Rte.



### NvMProxy\_WriteBlock

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Function Name** | NvMProxy\_WriteBlock | Type | Min | Max | UTP Tol. |
| **Arguments Passed** | Block | NvM\_BlockIdType |  |  |  |
|  | SrcPtr | uint8 |  |  |  |
| **Return Value** | result | Std\_ReturnType |  |  |  |

#### Description

This function implements the AUTOSAR standard API for the NvM BSW WriteBlock service. The interface must adhere to the AUTOSAR standard to allow mapping service port needs of SWC’s located in outside the QM application to this component’s service interface via the Rte.



### NvMProxy\_GetErrorStatus

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Function Name** | NvMProxy\_GetErrorStatus | Type | Min | Max | UTP Tol. |
| **Arguments Passed** | Block | NvM\_BlockIdType |  |  |  |
|  | RequestResultPtr | uint8\* |  |  |  |
| **Return Value** | N one |  |  |  |  |

#### Description

This function implements the AUTOSAR standard API for the NvM BSW GetErrorStatus service. The interface must adhere to the AUTOSAR standard to allow mapping service port needs of SWC’s located in outside the QM application to this component’s service interface via the Rte.



### NvMProxy\_SetRamBlockStatus

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Function Name** | NvMProxy\_SetRamBlockStatus | Type | Min | Max | UTP Tol. |
| **Arguments Passed** | Block | NvM\_BlockIdType |  |  |  |
|  | BlockChanged | boolean |  |  |  |
| **Return Value** | N one |  |  |  |  |

#### Description

This function implements the AUTOSAR standard API for the NvM BSW SetRamBlockStatus service. The interface must adhere to the AUTOSAR standard to allow mapping service port needs of SWC’s located in outside the QM application to this component’s service interface via the Rte.



## Local Functions/Macros Used by this MDD only

# 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 |
| <None> |  |

## Initialization Functions

### Init: NvMProxy\_Init

#### Design Rationale

Transfer the data from the unsecured Nv Data memory buffer to the secured Nv Data memory buffer and initialize the block status shadow to the values returned by the NvM API.

#### Module Outputs



#### Module Internal

NvMPWriteRqst\_Cnt\_M\_Str = 0\*

NvMPSetRBSRqst\_Cnt\_M\_Str = 0\*

*\* Cleared by memory clear function and not explicitly cleared in this init function*

## Periodic Functions

### Per: NvMProxy\_MainFunction

#### Design Rationale

This function is responsible for forwarding the queued NvM requests to the NvM driver in a SchM task running the NvM BSW.

This function copies the secured data area to the unsecured data area.

#### Program Flow Start

None

#### Store Module Inputs to Local copies

None

#### Processing of function





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

None

#### Program Flow End

None

## Fault Recovery Functions

None

## Shutdown Functions

None

## Interrupt Functions

None

## Serial Communication Functions

None

# Execution Requirements

## Execution Sequence of the Module

NvMProxy\_Init must be scheduled after to NvM\_ReadAll is completed. Additionaly NvMProxy\_Init must be executed as a trusted function to grant rights for initializing the secured memory. This would typically be accomplished via an Os trusted function call API.

NvMProxy\_MainFunction should be scheduled prior to NvM\_MainFunction. This provides the minimal amount of lag in forwarding and processing NvM service requests.

## 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 |
| NvMProxy\_Init | Once during startup | COLD INIT |
| NvMProxy\_MainFunction | Same as NvM\_MainFunction | Same as NvM\_MainFunction |

## Execution Requirements for Serial Communication Functions

|  |  |
| --- | --- |
| Function Name | Sub-Module called by (Serial Comm Function Name) |
| <None> |  |

# 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 |
| NvMProxy\_Init | NVMPROXY\_START\_SEC\_CODE |
| NvMProxy\_MainFunction | NVMPROXY\_START\_SEC\_CODE |
| NvMProxy\_WriteAll | NVMPROXY\_START\_SEC\_CODE |
| NvMProxy\_WriteBlock | NVMPROXY\_START\_SEC\_CODE |
| NvMProxy\_SetRamBlockStatus | NVMPROXY\_START\_SEC\_CODE |
| NvMProxy\_GetErrorStatus | NVMPROXY\_START\_SEC\_CODE |

## Local Functions

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

|  |  |
| --- | --- |
| Name of Sub Module | Software Segment |
| None |  |

# Known Issues / Limitations With Design

# Revision Control Log

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Item #** | **Rev #** | **Change Description** | **Date** | **Author Initials** |
| 1 |  | Initial creation | 21-Mar-12 | JJW |
| 2 |  | Corrected anomaly 4437 in NvMProxy init routine | 01-Mar-13 | KJS |
| 3 |  |  | 30-May-13 | JJW |
| 4 |  | Added CRC and Redundant block checking ability | 22-Nov-13 | LWW |
| 5 |  | Added corrections for set ram block status for anomaly EA3#3817 | 17-May-16 | KJS |