Integration Manual – NvM Proxy

Table of Contents

[1 Dependencies 2](#_Toc357692818)

[1.1 SWCs 2](#_Toc357692819)

[1.2 Functions to be provided to Integration Project 2](#_Toc357692820)

[2 Configuration 3](#_Toc357692821)

[2.1 Build Time Config 3](#_Toc357692822)

[2.2 Configuration Files to be provided by Integration Project 3](#_Toc357692823)

[2.2.1 Da Vinci Config generation 3](#_Toc357692824)

[2.2.2 Manual Configuration Changes 3](#_Toc357692825)

[3 Integration 4](#_Toc357692826)

[3.1 Required Global Data Inputs 4](#_Toc357692827)

[3.2 Optional Global Data Inputs 4](#_Toc357692828)

[3.3 Specific Include Path present 4](#_Toc357692829)

[4 Runnable Scheduling 5](#_Toc357692830)

[5 Memory Mapping 6](#_Toc357692831)

[5.1 Mapping 6](#_Toc357692832)

[5.2 Usage 6](#_Toc357692833)

[5.3 NvM Blocks 6](#_Toc357692834)

[6 Compiler Settings 6](#_Toc357692835)

[6.1 Preprocessor MACRO 6](#_Toc357692836)

[6.2 Optimization Settings 6](#_Toc357692837)

[7 Revision Control Log 7](#_Toc357692838)

# Dependencies

## SWCs

|  |  |
| --- | --- |
| Module | Required Feature |
| NvM | NvM\_WriteBlock()  NvM\_GetBlockStatus() |
| DiagMgr | NxtrDiagMgr#\_ReportNTCStatus() |
| Crc | Crc\_CalculateCRC16() |

## Global Functions(Non RTE) to be provided to Integration Project

NvMProxy\_Init

NvMProxy\_MainFunction

NvMProxy\_WriteBlock

NvMProxy\_WriteAll

NvMProxy\_GetErrorStatus

NvMProxy\_SetRamBlockStatus

# Configuration

## Build Time Config

|  |  |  |
| --- | --- | --- |
| Modules | Notes |  |
| None |  |  |

## Configuration Files to be provided by Integration Project

### Da Vinci Parameter Configuration Changes

|  |  |  |
| --- | --- | --- |
| Parameter | Notes | SWC |
| *NvMProxyConfigSet/NvMProxyBlock/*NvmRamBlockDataAddressSecure | The symbol name of the secured buffer location for the block data  NOTE: For blocks defined by PIM memory in the Rte, this parameter is the symbol name that Developer inserts into the NvM Ram block configuration parameter for the associated NvM block) | NvMProxy |
| *NvMProxyConfigSet/NvMProxyBlock/*InitBlockHandling | This parameter chooses the type of protection handling done on the block at initialization:  **None:** No specific handling needed  **CRC16:** Run a 16 bit CRC on the NvM block and check it against the CRC stored in the NvM block (last two bytes). Failures will trigger the fail action specified in the “InitCheckFailResponse” configuration  **Redundant:** Run redundant storage check on the NvM block. The 1’s compliment of the block data is stored in the NvM block to protect against corruption. Failures will trigger the fail action specified in the “InitCheckFailResponse” configuration  **ZeroData:** Ignore what is stored in NvM and always over-ride the NvM RAM buffer with zeros. This is useful for blocks that may be un | NvMProxy |
| *NvMProxyConfigSet/NvMProxyBlock/*InitCheckFailResponse | Defines the type of response to occur if the NvM block fails either the CRC16 or Redundant check at initialization:  **N/A:** This should be chosen if the block doesn’t have CRC16 or Redundant InitBlockHandling turned on  **SetNTC\_0x0A:** This should be chosen to set NTC 0x0A (should be calibrated to a critical shutdown fault- F1)  **SetNTC\_0x08\_LoadROMDefaults:** This should be chosen to set NTC 0x08 (should be calibrated to be a non-shutdown fault (F3) and mapped to a CTC that lights lamp). Also, default values will be loaded from FLASH memory at the symbolic location specified in the “ROMDefault\_Or\_NotificationFunction\_Symbol” configuration parameter.  **SetNTC\_0x08\_CallNotificationFunction:** This should be chosen to set NTC 0x08 (should be calibrated to be a non-shutdown fault (F3) and mapped to a CTC that lights lamp). Also, a user defined function will be called. The function name is to be specified in the the “ROMDefault\_Or\_NotificationFunction\_Symbol” configuration parameter.  **SetNTC\_0x07\_LoadROMDefaults:** This should be chosen to set NTC 0x07 (should be calibrated to be a non-shutdown fault (F3) and mapped to a CTC that does not light lamp). Also, default values will be loaded from FLASH memory at the symbolic location specified in the “ROMDefault\_Or\_NotificationFunction\_Symbol” configuration parameter.  **SetNTC\_0x07\_CallNotificationFunction:** This should be chosen to set NTC 0x07 (should be calibrated to be a non-shutdown fault (F3) and mapped to a CTC that does not light lamp). Also, a user defined function will be called. The function name is to be specified in the the “ROMDefault\_Or\_NotificationFunction\_Symbol” configuration parameter.  **SetNTC\_0x06\_LoadROMDefaults:** This should be chosen to set NTC 0x06 (should be calibrated to be a non-shutdown fault (F3) and not mapped to a CTC). Also, default values will be loaded from FLASH memory at the symbolic location specified in the “ROMDefault\_Or\_NotificationFunction\_Symbol” configuration parameter.  **SetNTC\_0x06\_CallNotificationFunction:** This should be chosen to set NTC 0x06 (should be calibrated to be a non-shutdown fault (F3) and not mapped to a CTC). Also, a user defined function will be called. The function name is to be specified in the the “ROMDefault\_Or\_NotificationFunction\_Symbol” configuration parameter. | NvMProxy |
| *NvMProxyConfigSet/NvMProxyBlock/*ROMDefault\_Or\_NotificationFunction\_Symbol | This parameter defines the symbolic name of the FLASH constant that contains the default values to load into NvM RAM if the block fails its CRC16 or Redundant check at initialization in the case of the “InitCheckFailResponse” parameter being either “SetNTC\_0x08\_LoadROMDefaults”, “SetNTC\_0x07\_LoadROMDefaults” or “SetNTC\_0x06\_LoadROMDefaults”. It defines the symbolic name of the notification function in the case of the “InitCheckFailResponse” parameter being either “SetNTC\_0x08\_CallNotificationFunction”, “SetNTC\_0x07\_CallNotificationFunction” or “SetNTC\_0x06\_CallNotificationFunction”.  This parameter should be set to “NULL\_PTR” if CRC16 or Redundant block initialization handling is not turned on. | NvMProxy |
| *NvMProxyConfigSet/NvMProxyBlock/*NvMRamGlobalShared | Set to “True” if the NvM block’s RAM is already in “Global Shared” memory. This avoids creating another buffer. This is typically the case for “TypeH” blocks and the EEPROM close check block. | NvMProxy |
| *NvMProxyConfigSet/NvMProxyBlock/*NvMBlockDescriptorRef | Reference to the NvMBlockDescriptor container that defines the NvM block linked to this proxy configuration. | NvMProxy |
| *NvMProxyGeneral/* NvMProxyIncludes | This contains a list of project specific include files that need to be compiled into the NvMProxy | NvMProxy |
| *NvMProxyGeneral/* FailureAPI | This should be set to the diagnostic manager’s API for setting faults. This should be set to the “ReportNTCStatus” API since it is called during initialization. It is configurable because the API depends on the application number. Typically this should be set to “NxtrDiagMgr10\_ReportNTCStatus” (assuming Ap 10 is the ASILD application) | NvMProxy |
| *DiagMgrConfigSet/DiagMgrEventParameter* | This module needs four DiagMgr NTCs added to the configuration: NTC 0x0A, NTC 0x08, NTC 0x07, and NTC 0x06. These need to be configured as “DIAGMGR\_EVENT\_KIND\_BSW” | DiagMgr |

### DaVinci Interrupt Configuration Changes

|  |  |  |  |
| --- | --- | --- | --- |
| ISR Name | VIM # | Priority Dependency | Notes |
| N/A |  |  |  |

### Manual Configuration Changes

|  |  |  |
| --- | --- | --- |
| Constant | Notes | SWC |
| NVMPROXY\_EXCLUSIVE\_AREA\_0 | This exclusive are covers the areas of execution within the component that are operating on the request buffer. The buffer is operated on by the MainFunction and the WriteBlock functions. An appropriate level of protection must be employed to maintain exclusive usage of the buffer data. | SchM |

# Integration

The following import steps must be completed :

1. Place CBD project structure to appropriate integration folder
2. Execute the “Integrate.bat” script from the Tools directory of this component to perform the necessary integration steps:
   1. The script creates the required directories in the integration project, “Generators/Artt/NvMProxy” and “Generators/Components/\_Schemes/NvMProxy/bswmd”
   2. The script then copies the required files from the CBD generate directory into the new directories.
3. If this is the first time integration, then perform the Davinci Configurator 3rd party component integration procedure.
4. Configure NvM proxy component per program needs
5. Generate NvM proxy and import generated Cd\_NvMProxy\_swc.arxml into davinci developer and map all NvM service needs on the blocks needing proxies to the NvM Proxy service component (instead of the NvM service component)

**SPECIAL INTEGRATION NOTES:**

NvM Block Sizes

If CRC16 protection is chosen on a block, the NvM configuration needs to be increased by “2” to hold the CRC. The CRC value will be stored in the last two bytes of the data block.

Similarly, if Redundant protection is chosen on a block, the NvM configuration needs to be doubled to hold the redundant data.

Because of compiler alignment restrictions, it is HIGHLY recommended to use the debugger in Code Composer to analyze the compiled size of the NvM block. This can be done by using the sizeof(<NvMRamShadowName>) in the expressions window. The resulting size shown should get the added “2” or doubling in the NvM Configuration.

NvM Configuration

The NvM configuration parameter “NvMRamBlockDataAddress” configuration for all blocks using the NvM proxy need to have “NvMP\_” pre-pended to the normal name of the RAM shadow symbol. **Please note that if the block is linked through Davinci Developer Per-Instance Memory Mapping, this name will automatically revert back (remove the “NvMP\_”) every time the Davinci Developer project is saved and the block size may be reverted to the original size (without the added CRC size or doubling for redundant store).**

ROM Defaults and Notification Functions

If “ROM defaults” or “Notification Functions” are configured via the “InitCheckFailResponse” parameter, it is up to the integration project to provide the ROM default data or the notification function named per the “ROMDefault\_Or\_NotificationFunction\_Symbol” parameter. The ROM default option will use a blind memory copy, so it is important that the same NvM RAM Shadow datatype is used for ROM constant to ensure proper data alignment.

## Required Global Data Inputs

N/A

## Required Global Data Outputs

N/A

## Specific Include Path present

Yes

# Runnable Scheduling

This section specifies the required runnable scheduling.

|  |  |  |
| --- | --- | --- |
| Init | Scheduling Requirements | Trigger |
| NvMProxy\_Init() | Must be executed after NvM driver has initialized the unsecured block data to be forwarded to the secured memory by this component. | Init |

|  |  |  |
| --- | --- | --- |
| Runnable | Scheduling Requirements | Trigger |
| NvMProxy\_MainFunction() | Run prior to NvM\_MainFunction for minimal request processing latency | Same as NvM\_MainFunction |

**.**

# Memory Mapping

## Mapping

|  |  |  |
| --- | --- | --- |
| Memory Section | Contents | Notes |
| NVMPROXY\_START\_SEC\_VAR\_NOINIT\_8 |  | Typically allocated to application in which NvM driver resides.  Not required to be allocated to Global shared memeory. |
| NVMPROXY\_START\_SEC\_VAR\_CLEARED\_16 |  | Must be allocated into Global Shared memory |
| NVMPROXY\_START\_SEC\_VAR\_CLEARED\_UNSPECIFIED |  | Must be allocated into Global Shared memory |
| NVMPROXY\_START\_SEC\_CODE |  |  |
| NVMPROXY\_START\_SEC\_CONST\_UNSPECIFIED |  |  |

\* Each …START\_SEC… constant is terminated by a …STOP\_SEC… constant as specified in the AUTOSAR Memory Mapping requirements.

## Usage

|  |  |  |
| --- | --- | --- |
| Feature | RAM | ROM |
|  |  |  |

Table 1: ARM Cortex R4 Memory Usage

## Non RTE NvM Blocks

|  |
| --- |
| Block Name |
| <NVM block used Non RTE functions > |

Note : Size of the NVM block if configured in developer

## RTE NvM Blocks

|  |
| --- |
| Block Name |
| <NVM block used in RTE functions > |

Note : Size of the NVM block if configured in developer

# Compiler Settings

## Preprocessor MACRO

<Define all the preprocessor Macros needed and conditions when needed>.

## Optimization Settings

<Define Optimization levels that are needed and conditions when needed>.

# Revision Control Log

|  |  |  |  |
| --- | --- | --- | --- |
| **Rev #** | **Change Description** | **Date** | **Author** |
| 1 | Initial version |  | JJW |
| 2 | Updates per generation definition | 10/18/12 | JJW |
| 3 | Updates for CRC and Redundant checking feature | 12/02/13 | LWW |