Integration Manual - StaMd

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** |
| **ECUStatup.c** | StaMd\_Init0 needs to be called from EcuStartup\_Init2 via a trusted wrapper after Nvm ‘read all’ is complete, by calling Call\_StaMd\_Init0.  ECUStartup needs to also include the Ap\_StaMd.h header file.  The StaMd\_Init0 is defined outside of the RTE and is responsible for updating Type H memory across applications at start up. StaMd\_Init0 needs to be added to a non-trusted function list in a trusted application in the O.S. |
| **NtWrap.c, .h, O.S changes** | Add trusted function call to NtWrap :  /\* Trusted wrapper Function \*/  void TRUSTED\_NtWrapS\_StaMd\_Init0  (TrustedFunctionIndexType FunctionIndex, TrustedFunctionParameterRefType FunctionParams)  {  StaMd\_Init0();  }  …  void Call\_StaMd\_Init0(void)  {  (void) CallTrustedFunction  (NtWrapS\_StaMd\_Init0,  (TrustedFunctionParameterRefType)0);  } |
| **Optional NvM Fast Write implementation to meet program specific ECUReset or Programming session transition timing requirements.** | Whenever an ECU Reset (StaMd\_SCom\_EcuReset) or FlashBootloader/Programming session service request (StaMd\_SCom\_FBLTransitionReq ) the StaMd module is notified by the corresponding Scom functions, typically called directly from the diagnostic service handler. The system is then forced to the OFF state and an output flag, **InitiatePwrDnFastWrite\_Cnt\_lgc** is set to TRUE. This flag can be used to trigger a FAST NVM write as an option to meet ECU Rest and/or FlashBootloader/Programming session timing requirements.  The **InitiatePwrDnNormalWrite\_Cnt\_lgc** output flag will be set to TRUE in case of a normal transition to OFF.  The implementation of the “fast” NvM write is done by running the main functions responsible for handling the NvM processing continuously in a WHILE loop until all of NvM, including the closecheck flag used for NTC $BF, has been written. These main functions (NvM, Ea, NvMProxy etc…) are typically called from a SchM task. **Since the fast write occurs in a WHILE loop, it is important that the fast write is implemented in a low priority SchM task. Review the task priorities before implementation.**  For example, see the Ford S550/P552 fast NvM write implementation below. This is in a priority 2 SchM task. The only other tasks with a lower priority are the Background task and a SchM init task:  **do**  {  GetResource(OsRes\_MemStackTask);  Eep\_30\_At25128\_MainFunction();  Ea\_MainFunction();  NvMProxy\_MainFunction();  NvM\_MainFunction();  ReleaseResource(OsRes\_MemStackTask);  }**while**(CDD\_InitiatePwrDn**FastWrite**\_Cnt\_G\_lgc == TRUE);  In the case of a FlashBootloader/Programming session service request (**StaMd\_SCom\_FBLTransitionReq)**, once the NvM write is complete the output flag, **PwrDnFastWriteComplete\_Cnt\_lgc**, will be set to TRUE (**See the Data Dictionary**). The PwrDn**FastWriteComplete** flag can be used as follows to complete a FlashBootloader/Programming session transition request:  if (ProgrammingSessionEntered\_Cnt\_M\_lgc == TRUE)  {  if (CDD\_PwrDn**FastWriteComplete**\_Cnt\_G\_lgc == TRUE)  {  ProgrammingSessionEntered\_Cnt\_M\_lgc = FALSE;    /\* **Jump to Bootloader** \*/  Program Specific Vector to Bootloader  }  }  These status flags can be provided to complex device drivers, such as SchM, via CDD interface by creating the following global variables: **CDD\_PwrDnFastWriteComplete\_Cnt\_G\_lgc**, **CDD\_InitiatePwrDnFastWrite\_Cnt\_G\_lgc, CDD\_InitiatePwrDnNormalWrite\_Cnt\_G\_lgc**  **NOTE:** The StaMd\_SCom\_FBLTransitionReq should only be used to update NvM, if required, before transitioning to the bootloader in response to a diagnostic service request to transition to a flash programming session. Only use this Scom function as intended. Once this Scom function is called a reset or vector from the application is expected after the PwrDnFastWriteComplete flag has been set to true. |
|  |  |

Note : Referencing the external components should be avoided in most cases. Only in unavoidable circumstance external components should be refered. Developer should track the references.

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

extern FUNC(void, MCU\_CODE) **Mcu\_PerformReset**(void);

# Configuration

## Build Time Config

|  |  |  |
| --- | --- | --- |
| **Modules** | **Notes** |  |
| **<None>** |  |  |

## Configuration Files to be provided by Integration Project

**Ap\_StaMd\_Cfg.h**

### Da Vinci Parameter Configuration Changes

|  |  |  |
| --- | --- | --- |
| **Parameter** | **Notes** | **SWC** |
| **TypeHDataSize** | Total size of all Type H data in bytes |  |
| **StaMdCPEnable** | This container contains the configuration (parameters) for the StaMd Watchdog checkpoints. |  |
| **StaMdTODType** | This container defines the configuration for the type of TOD implementation used:  TOD\_2msToggle  TOD\_SteadyState (\*common setting)  TOD\_None |  |
| **StaMdNvMWriteAllAPI** | This container defines the API used for the NvM Write All function. (\*common setting is **NvMProxy\_WriteAll** if the NvM proxy is used). |  |
| **StaMdNvMGetErrorStatusAPI** | This container defines the API used for the NvM Get Error Status function. (\*common setting is **NvMProxy\_GetErrorStatus** if the NvM proxy is used). |  |
| **StaMdTrnsDiagMgrShtDwnTaskActivation** | This container defines the DiagMgr shutdown function. If StaMdCoreOsAppRef matches the application referenced for DiagMgrDemIfOsAppRef in DiagMgr then the generated output will be a client/server call and this field is ignored. If they do not match, then a task activation call is created and the task defined in this field is activated.  **NOTE:** Typical setting is Task\_TrnsB\_9 for the application 9 transition function, from which the StaMd9\_Trns\_DemShutdown function is called in some programs. |  |
| **GenerateExcludeOsAppRef** | This parameter defines the application(s) which do not require a States and Modes component. |  |
| **StaMdCoreOsAppRef** | This parameter defines the application which contains the core States and Modes component. |  |
| **StaMdsComOsAppRef** | This parameter defines the application which interfaces with the serial communications functions. |  |
| **StaMdSysCovOsAppRef** | This parameter defines the application which performs the states and modes systematic coverage. |  |

### DaVinci Interrupt Configuration Changes

|  |  |  |  |
| --- | --- | --- | --- |
| **ISR Name** | **VIM #** | **Priority Dependency** | **Notes** |
| **<None>** |  |  |  |

### Manual Configuration Changes

|  |  |  |
| --- | --- | --- |
| **Constant** | **Notes** | **SWC** |
| **<None>** |  |  |

# Integration

## Required Global Data Inputs

## Required Global Data Outputs

## Specific Include Path present

The **…StaMd/include** patch needs to be added to the include search path of the CCS project. Typical setting: **"${workspace\_loc:/FORD\_S550\_P552/StaMd/include}"**

# Runnable Scheduling

This section specifies the required runnable scheduling.

|  |  |  |
| --- | --- | --- |
| **Init** | **Scheduling Requirements** | **Trigger** |
| None | None | Init |

|  |  |  |
| --- | --- | --- |
| **Runnable** | **Scheduling Requirements** | **Trigger** |
|  |  | 10ms |

**.**

# Memory Mapping

## Mapping

|  |  |  |
| --- | --- | --- |
| **Memory Section** | **Contents** | **Notes** |
| STAMD\_START\_SEC\_VAR\_SAVED\_ZONEHGS\_32  STAMD\_START\_SEC\_VAR\_SAVED\_ZONEHGS\_8 |  |  |

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

## Usage

|  |  |  |
| --- | --- | --- |
| **Feature** | **RAM** | **ROM** |
| **<Memmap usuage info>** |  |  |

Table 1: ARM Cortex R4 Memory Usage

## Non RTE NvM Blocks

|  |
| --- |
| **Block Name** |
| **<None >** |

Note : Size of the NVM block if configured in developer

## RTE NvM Blocks

|  |
| --- |
| **Block Name** |
| None |

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 | 12-Dec-13 | BDO |
| 2 | Updated to FDD ES10B version 13 to address anomaly 5388. CR11347 | 07-Feb-14 | BDO |
| 3 | Anomaly 7307 - updates for consistency with FDD ES10B version 13 12602 | 28-Oct-14 | BDO |