# Integration Manual

for S32K3 I2C Driver

Document Number: IM34I2CASRR21-11 Rev0000R3.0.0 Rev. 1.0

# Chapter 1

# Revision History

| Revision | Date | Author | Description |
|----------|------|--------|-------------|
| 1.0 | 31.03.2023 | NXP RTD Team | Prepared for release RTD S32K3 3.0.0 |

# Chapter 2

# Introduction

- Supported Derivatives

- Overview

- About This Manual

- Acronyms and Definitions

- Reference List

This integration manual describes the integration requirements for I2c Driver for S32K3XX microcontrollers.

## 2.1  Supported Derivatives

The software described in this document is intended to be used with the following microcontroller devices of NXP Semiconductors:

- s32k310_mqfp100

- s32k310_lqfp48

- s32k311_mqfp100 / MWCT2015S_mqfp100

- s32k311_lqfp48

- s32k312_mqfp100 / MWCT2016S_mqfp100

- s32k312_mqfp172 / MWCT2016S_mqfp172

- s32k314_mqfp172

- s32k314_mapbga257

- s32k322_mqfp100 / MWCT2D16S_mqfp100

- s32k322_mqfp172 / MWCT2D16S_mqfp172

- s32k324_mqfp172 / MWCT2D17S_mqfp172

- s32k324_mapbga257

- s32k341_mqfp100

- s32k341_mqfp172

- s32k342_mqfp100

- s32k342_mqfp172

- s32k344_mqfp172

- s32k344_mapbga257

- s32k394_mapbga289

- s32k396_mapbga289

- s32k358_mqfp172

- s32k358_mapbga289

- s32k328_mqfp172

- s32k328_mapbga289

- s32k338_mqfp172

- s32k338_mapbga289

- s32k348_mqfp172

- s32k348_mapbga289

- s32m274_lqfp64

- s32m276_lqfp64

All of the above microcontroller devices are collectively named as S32K3.

Note: MWCT part numbers contain NXP confidential IP for Qi Wireless Power.

## 2.2 Overview

**AUTOSAR (AUTomotive Open System ARchitecture)** is an industry partnership working to establish standards for software interfaces and software modules for automobile electronic control systems.

AUTOSAR:

- paves the way for innovative electronic systems that further improve performance, safety and environmental friendliness.

- is a strong global partnership that creates one common standard: "Cooperate on standards, compete on implementation".

- is a key enabling technology to manage the growing electrics/electronics complexity. It aims to be prepared for the upcoming technologies and to improve cost-efficiency without making any compromise with respect to quality.

- facilitates the exchange and update of software and hardware over the service life of the vehicle.

## 2.3   About This Manual

This Technical Reference employs the following typographical conventions:

- **Boldface** style: Used for important terms, notes and warnings.

- *Italic* style: Used for code snippets in the text. Note that C language modifiers such "const" or "volatile" are sometimes omitted to improve readability of the presented code.

Notes and warnings are shown as below:

Note

   This is a note.

Warning

   This is a warning

## 2.4 Acronyms and Definitions

| Term | Definition |
|------|------------|
| API | Application Programming Interface |
| ASM | Assembler |
| BSMI | Basic Software Make file Interface |
| CAN | Controller Area Network |
| C/CPP | C and C++ Source Code |
| CS | Chip Select |
| CTU | Cross Trigger Unit |
| DEM | Diagnostic Event Manager |
| DET | Development Error Tracer |
| DMA | Direct Memory Access |
| ECU | Electronic Control Unit |
| FIFO | First In First Out |
| LSB | Least Signifigant Bit |
| MCU | Micro Controller Unit |
| MIDE | Multi Integrated Development Environment |
| MSB | Most Significant Bit |
| N/A | Not Applicable |
| RAM | Random Access Memory |
| SIU | Systems Integration Unit |
| SWS | Software Specification |
| VLE | Variable Length Encoding |
| XML | Extensible Markup Language |

## 2.5 Reference List

| # | Title | Version |
|---|-------|---------|
| 1 | Reference Manual | S32K3xx Reference Manual, Rev.6, Draft B, 01/2023 |
| | | S32K39 and S32K37 Reference Manual, Rev. 2 Draft A, 11/2022 |
| | | S32M27x Reference Manual, Rev.2, Draft A, - 02/2023 |
| 2 | Datasheet | S32K3xx Data Sheet, Rev. 6, 11/2022 |
| | | S32K396 Data Sheet, Rev. 1.1 — 08/2022 |
| | | S32M2xx Data Sheet, Rev. 2 RC — 12/2022 |
| 3 | Errata | S32K358_0P14E Mask Set Errata – Rev. 28, 9/2022 |
| | | S32K396_0P40E Mask Set Errata, Rev. DEC2022, 12/2022 |
| | | S32K311_0P98C Mask Set Errata, Rev. 6/March/2023, 3/2023 |
| | | S32K312: Mask Set Errata for Mask 0P09C, Rev. 25/April/2022 |
| | | S32K342: Mask Set Errata for Mask 0P97C, Rev. 10, 11/2022 |
| | | S32K3x4: Mask Set Errata for Mask 0P55A/1P55A, Rev. 14/Oct/2022 |

# Chapter 3

# Building the driver

- Build Options
- Files required for compilation
- Setting up the plugins

This section describes the source files and various compilers, linker options used for building the driver.

It also explains the EB Tresos Studio plugin setup procedure.

## 3.1 Build Options

- GCC Compiler/Assembler/Linker Options
- DIAB Compiler/Assembler/Linker Options
- GHS Compiler/Assembler/Linker Options
- IAR Compiler/Assembler/Linker Options

The RTD driver files are compiled using:

- NXP GCC 10.2.0 20200723 (Build 1728 Revision g5963bc8)
- Wind River Diab Compiler 7.0.4
- Compiler Versions: Green Hills Multi 7.1.6d / Compiler 2021.1.4
- Compiler Versions: IAR ANSI C/C++ Compiler V8.50.10 (safety version)

The compiler, assembler, and linker flags used for building the driver are explained below.

The TS_T40D34M30I0R0 part of the plugin name is composed as follows:

- T = Target_Id (e.g. T40 identifies Cortex-M architecture)
- D = Derivative_Id (e.g. D34 identifies S32K3 platform)
- M = SW_Version_Major and SW_Version_Minor
- I = SW_Version_Patch
- R = Reserved

## 3.1.1  GCC Compiler/Assembler/Linker Options

### 3.1.1.1  GCC Compiler Options

| Compiler Option | Description |
| --- | --- |
| -mcpu=cortex-m7 | Targeted ARM processor for which GCC should tune the performance of the code |
| -mthumb | Generates code that executes in Thumb state |
| -mlittle-endian | Generate code for a processor running in little-endian mode |
| -mfpu=fpv5-sp-d16 | Specifies the floating-point hardware available on the target |
| -mfloat-abi=hard | Specifies the floating-point ABI to use. "hard" allows generation of floating-point instructions and uses FPU-specific calling conventions |
| -std=c99 | Specifies the ISO C99 base standard |
| -Os | Optimize for size. Enables all -O2 optimizations except those that often increase code size |
| -ggdb3 | Produce debugging information for use by GDB using the most expressive format available, including GDB extensions if at all possible. Level 3 includes extra information, such as all the macro definitions present in the program |
| -Wall | Enables all the warnings about constructions that some users consider questionable, and that are easy to avoid (or modify to prevent the warning), even in conjunction with macros |
| -Wextra | This enables some extra warning flags that are not enabled by -Wall |
| -pedantic | Issue all the warnings demanded by strict ISO C. Reject all programs that use forbidden extensions. Follows the version of the ISO C standard specified by the aforementioend -std option |
| -Wstrict-prototypes | Warn if a function is declared or defined without specifying the argument types |
| -Wundef | Warn if an undefined identifier is evaluated in an #if directive. Such identifiers are replaced with zero |
| -Wunused | Warn whenever a function, variable, label, value, macro is unused |
| -Werror=implicit-function-declaration | Make the specified warning into an error. This option throws an error when a function is used before being declared |
| -Wsign-compare | Warn when a comparison between signed and unsigned values could produce an incorrect result when the signed value is converted to unsigned. |
| -Wdouble-promotion | Give a warning when a value of type float is implicitly promoted to double |
| -fno-short-enums | Specifies that the size of an enumeration type is at least 32 bits regardless of the size of the enumerator values. |
| -funsigned-char | Let the type char be unsigned by default, when the declaration does not use either signed or unsigned |
| -funsigned-bitfields | Let a bit-field be unsigned by default, when the declaration does not use either signed or unsigned |

| Compiler Option | Description |
|---|---|
| -fno-common | Makes the compiler place uninitialized global variables in the BSS section of the object file. This inhibits the merging of tentative definitions by the linker so you get a multiple-definition error if the same variable is accidentally defined in more than one compilation unit |
| -fstack-usage | This option is only used to build test for generation Ram/↵ Stack size report. Makes the compiler output stack usage information for the program, on a per-function basis |
| -fdump-ipa-all | This option is only used to build test for generation Ram/↵ Stack size report. Enables all inter-procedural analysis dumps |
| -c | Stop after assembly and produce an object file for each source file |
| -DS32K3XX | Predefine S32K3XX as a macro, with definition 1 |
| -D $ (DERIVATIVE) | Predefine S32K3's derivative as a macro, with definition 1. For example: Predefine for S32K344 will be -DS32K344. |
| -DGCC | Predefine GCC as a macro, with definition 1 |
| -DUSE_SW_VECTOR_MODE | Predefine USE_SW_VECTOR_MODE as a macro, with definition 1. By default, the drivers are compiled to handle interrupts in Software Vector Mode |
| -DD_CACHE_ENABLE | Predefine D_CACHE_ENABLE as a macro, with definition 1. Enables data cache initalization in source file system.↵ c under the Platform driver |
| -DI_CACHE_ENABLE | Predefine I_CACHE_ENABLE as a macro, with definition 1. Enables instruction cache initalization in source file system.c under the Platform driver |
| -DENABLE_FPU | Predefine ENABLE_FPU as a macro, with definition 1. Enables FPU initalization in source file system.c under the Platform driver |
| -DMCAL_ENABLE_USER_MODE_SUPPORT | Predefine MCAL_ENABLE_USER_MODE_SUPPO↵ RT as a macro, with definition 1. Allows drivers to be configured in user mode. |
| –sysroot= | Specifies the path to the sysroot, for Cortex-M7 it is /arm-none-eabi/newlib |
| -specs=nano.specs | Use Newlib nano specs |
| -specs=nosys.specs | Do not use printf/scanf |

### 3.1.1.2 GCC Assembler Options

| Assembler Option | Description |
|---|---|
| -Xassembler-with-cpp | Specifies the language for the following input files (rather than letting the compiler choose a default based on the file name suffix) |
| -mcpu=cortexm7 | Targeted ARM processor for which GCC should tune the performance of the code |
| -mfpu=fpv5-sp-d16 | Specifies the floating-point hardware available on the target |
| -mfloat-abi=hard | Specifies the floating-point ABI to use. "hard" allows generation of floating-point instructions and uses FPU-specific calling conventions |
| -mthumb | Generates code that executes in Thumb state |

| Assembler Option | Description |
|---|---|
| -c | Stop after assembly and produce an object file for each source file |

### 3.1.1.3 GCC Linker Options

| Linker Option | Description |
|---|---|
| -Wl,-Map,filename | Produces a map file |
| -T linkerfile | Use linkerfile as the linker script. This script replaces the default linker script (rather than adding to it) |
| –entry=Reset_Handler | Specifies that the program entry point is Reset_Handler |
| -nostartfiles | Do not use the standard system startup files when linking |
| -mcpu=cortexm7 | Targeted ARM processor for which GCC should tune the performance of the code |
| -mthumb | Generates code that executes in Thumb state |
| -mfpu=fpv5-sp-d16 | Specifies the floating-point hardware available on the target |
| -mfloat-abi=hard | Specifies the floating-point ABI to use. "hard" allows generation of floating-point instructions and uses FPU-specific calling conventions |
| -mlittle-endian | Generate code for a processor running in little-endian mode |
| -ggdb3 | Produce debugging information for use by GDB using the most expressive format available, including GDB extensions if at all possible. Level 3 includes extra information, such as all the macro definitions present in the program |
| -lc | Link with the C library |
| -lm | Link with the Math library |
| -lgcc | Link with the GCC library |
| -specs=nano.specs | Use Newlib nano specs |
| -specs=nosys.specs | Do not use printf/scanf |

## 3.1.2 DIAB Compiler/Assembler/Linker Options

### 3.1.2.1 DIAB Compiler Options

| Compiler Option | Description |
|---|---|
| -tARMCORTEXM7MG:simple | Selects target processor (hardware single-precision, software double-precision floating-point) |
| -mthumb | Selects generating code that executes in Thumb state |
| -std=c99 | Follows the C99 standard for C |
| -Oz | Like -O2 with further optimizations to reduce code size |
| -g | Generates DWARF 4.0 debug information |
| -fstandalone-debug | Emits full debug info for all types used by the program |
| -Wstrict-prototypes | Warn if a function is declared or defined without specifying the argument types |
| -Wsign-compare | Produce warnings when comparing signed type with unsigned type |
| -Wdouble-promotion | Give a warning when a value of type float is implicitly promoted to double |

| Compiler Option | Description |
|---|---|
| -Wunknown-pragmas | Issues a warning for unknown pragmas |
| -Wundef | Warns if an undefined identifier is evaluated in an #if directive. Such identifiers are replaced with zero |
| -Wextra | Enables some extra warning flags that are not enabled by '-Wall' |
| -Wall | Enables all of the most useful warnings (for historical reasons this option does not literally enable all warnings) |
| -pedantic | Emits a warning whenever the standard specified by the -std option requires a diagnostic |
| -Werror=implicit-function-declaration | Generates an error whenever a function is used before being declared |
| -fno-common | Compile common globals like normal definitions |
| -fno-signed-char | Char is unsigned |
| -fno-trigraphs | Do not process trigraph sequences |
| -V | Displays the current version number of the tool suite |
| -c | Stop after assembly and produce an object file for each source file |
| -DS32K3XX | Predefine S32K3XX as a macro, with definition 1 |
| -D $ (DERIVATIVE) | Predefine S32K3's derivative as a macro, with definition 1 |
| -DDIAB | Predefine DIAB as a macro, with definition 1 |
| -DUSE_SW_VECTOR_MODE | Predefine USE_SW_VECTOR_MODE as a macro, with definition 1. By default, the drivers are compiled to handle interrupts in Software Vector Mode |
| -DD_CACHE_ENABLE | Predefine D_CACHE_ENABLE as a macro, with definition 1. Enables data cache initalization in source file system.↩c under the Platform driver |
| -DI_CACHE_ENABLE | Predefine I_CACHE_ENABLE as a macro, with definition 1. Enables instruction cache initalization in source file system.c under the Platform driver |
| -DENABLE_FPU | Predefine ENABLE_FPU as a macro, with definition 1. Enables FPU initalization in source file system.c under the Platform driver |
| -DMCAL_ENABLE_USER_MODE_SUPPORT | Predefine MCAL_ENABLE_USER_MODE_SUPPO↩RT as a macro, with definition 1. Allows drivers to be configured in user mode |

### 3.1.2.2 DIAB Assembler Options

| Assembler Option | Description |
|---|---|
| -mthumb | Selects generating code that executes in Thumb state |
| -Xpreprocess-assembly | Invokes C preprocessor on assembly files before running the assembler |
| -Xassembly-listing | Produces an .lst assembly listing file |
| -c | Stop after assembly and produce an object file for each source file |
| -tARMCORTEXM7MG:simple | Selects target processor (hardware single-precision, software double-precision floating-point) |

### 3.1.2.3    DIAB Linker Options

| Linker Option | Description |
|---|---|
| -e Reset_Handler | Make the symbol Reset_Handler be treated as a root symbol and the start label of the application |
| linker_script_file.dld | Use linker_script_file.dld as the linker script. This script replaces the default linker script (rather than adding to it) |
| -m30 | m2 + m4 + m8 + m16 |
| -Xstack-usage | Gathers and display stack usage at link time |
| -Xpreprocess-lecl | Perform pre-processing on linker scripts |
| -Llibrary_path | Points to the libraries location for ARMV7EMMG to be used for linking |
| -lc | Links with the standard C library |
| -lm | Links with the math library |
| -tARMCORTEXM7MG:simple | Selects target processor (hardware single-precision, software double-precision floating-point) |

## 3.1.3    GHS Compiler/Assembler/Linker Options

### 3.1.3.1    GHS Compiler Options

| Compiler Option | Description |
|---|---|
| -cpu=cortexm7 | Selects target processor: Arm Cortex M7 |
| -thumb | Selects generating code that executes in Thumb state |
| -fpu=vfpv5_d16 | Specifies hardware floating-point using the v5 version of the VFP instruction set, with 16 double-precision floating-point registers |
| -fsingle | Use hardware single-precision, software double-precision FP instructions |
| -C99 | Use (strict ISO) C99 standard (without extensions) |
| –ghstd=last | Use the most recent version of Green Hills Standard mode (which enables warnings and errors that enforce a stricter coding standard than regular C and C++) |
| -Osize | Optimize for size |
| –gnu_asm | Enables GNU extended asm syntax support |
| -dual_debug | Generate DWARF 2.0 debug information |
| -G | Generate debug information |
| -keeptempfiles | Prevents the deletion of temporary files after they are used. If an assembly language file is created by the compiler, this option will place it in the current directory instead of the temporary directory |
| -Wimplicit-int | Produce warnings if functions are assumed to return int |
| -Wshadow | Produce warnings if variables are shadowed |
| -Wtrigraphs | Produce warnings if trigraphs are detected |
| -Wundef | Produce a warning if undefined identifiers are used in #if preprocessor statements |

| Compiler Option | Description |
| --- | --- |
| –unsigned_chars | Let the type char be unsigned, like unsigned char |
| –unsigned_fields | Bitfelds declared with an integer type are unsigned |
| –no_commons | Allocates uninitialized global variables to a section and initializes them to zero at program startup |
| –no_exceptions | Disables C++ support for exception handling |
| –no_slash_comment | C++ style // comments are not accepted andgenerate errors |
| –prototype_errors | Controls the treatment of functions referenced or called when no prototype has been provided |
| –incorrect_pragma_warnings | Controls the treatment of valid #pragma directives that use the wrong syntax |
| -c | Stop after assembly and produce an object file for each source file |
| -DS32K3XX | Predefine S32K3XX as a macro, with definition 1 |
| -D $ (DERIVATIVE) | Predefine S32K3's derivative as a macro, with definition 1. For example: Predefine for S32K344 will be -DS32K344. |
| -DGHS | Predefine GHS as a macro, with definition 1 |
| -DUSE_SW_VECTOR_MODE | Predefine USE_SW_VECTOR_MODE as a macro, with definition 1. By default, the drivers are compiled to handle interrupts in Software Vector Mode |
| -DD_CACHE_ENABLE | Predefine D_CACHE_ENABLE as a macro, with definition 1. Enables data cache initalization in source file system.↩c under the Platform driver |
| -DI_CACHE_ENABLE | Predefine I_CACHE_ENABLE as a macro, with definition 1. Enables instruction cache initalization in source file system.c under the Platform driver |
| -DENABLE_FPU | Predefine ENABLE_FPU as a macro, with definition 1. Enables FPU initalization in source file system.c under the Platform driver |
| -DMCAL_ENABLE_USER_MODE_SUPPORT | Predefine MCAL_ENABLE_USER_MODE_SUPPO↩RT as a macro, with definition 1. Allows drivers to be configured in user mode |

### 3.1.3.2 GHS Assembler Options

| Assembler Option | Description |
| --- | --- |
| -cpu=cortexm7 | Selects target processor: Arm Cortex M7 |
| -fpu=vfpv5_d16 | Specifies hardware floating-point using the v5 version of the VFP instruction set, with 16 double-precision floating-point registers |
| -fsingle | Use hardware single-precision, software double-precision FP instructions |
| -preprocess_assembly_files | Controls whether assembly files with standard extensions such as .s and .asm are preprocessed |
| -list | Creates a listing by using the name and directory of the object file with the .lst extension |
| -c | Stop after assembly and produce an object file for each source file |

### 3.1.3.3 GHS Linker Options

| Linker Option | Description |
|---|---|
| -e Reset_Handler | Make the symbol Reset_Handler be treated as a root symbol and the start label of the application |
| -T linker_script_file.ld | Use linker_script_file.ld as the linker script. This script replaces the default linker script (rather than adding to it) |
| -map | Produce a map file |
| -keepmap | Controls the retention of the map file in the event of a link error |
| -Mn | Generates a listing of symbols sorted alphabetically/numerically by address |
| -delete | Instructs the linker to remove functions that are not referenced in the final executable. The linker iterates to find functions that do not have relocations pointing to them and eliminates them |
| -ignore_debug_references | Ignores relocations from DWARF debug sections when using -delete. DWA↩RF debug information will contain references to deleted functions that may break some third-party debuggers |
| -Llibrary_path | Points to library_path (the libraries location) for thumb2 to be used for linking |
| -larch | Link architecture specific library |
| -lstartup | Link run-time environment startup routines. The source code for themodules in this library is provided in the src/libstartup directory |
| -lind_sd | Link language-independent library, containing support routines for features such as software floating point, run-time error checking, C99 complex numbers, and some general purpose routines of the ANSI C library |
| -v | Prints verbose information about the activities of the linker, including the libraries it searches to resolve undefined symbols |
| -keep=C40_Ip_AccessCode | Avoid linker remove function C40_Ip_AccessCode from Fls module because it is not referenced explicitly |
| -nostartfiles | Controls the start files to be linked into the executable |

### 3.1.4  IAR Compiler/Assembler/Linker Options

#### 3.1.4.1  IAR Compiler Options

| Compiler Option | Description |
|---|---|
| --cpu Cortex-M7 | Targeted ARM processor for which IAR should tune the performance of the code |
| --cpu_mode thumb | Generates code that executes in Thumb state |
| --endian little | Generate code for a processor running in little-endian mode |
| --fpu VFPv5-SP | Use this option to generate code that performs floating-point operations using a Floating Point Unit (FPU). Single-precision variant. |
| -e | Enables all IAR C language extensions |
| -Ohz | Optimize for size. the compiler will emit AEABI attributes indicating the requested optimization goal. This information can be used by the linker to select smaller or faster variants of DLIB library functions |
| --debug | Makes the compiler include debugging information in the object modules. Including debug information will make the object files larger |

| Compiler Option | Description |
|---|---|
| –no_clustering | Disables static clustering optimizations. Static and global variables defined within the same module will not be arranged so that variables that are accessed in the same function are close to each other |
| –no_mem_idioms | Makes the compiler not optimize certain memory access patterns |
| –do_explicit_zero_opt_in_named_sections | Disable the exception for variables in user-named sections, and thus treat explicit initializations to zero as zero initializations, not copy initializations |
| –require_prototypes | Force the compiler to verify that all functions have proper prototypes. Generates an error otherwise |
| –no_wrap_diagnostics | Does not wrap long lines in diagnostic messages |
| –diag_suppress Pa050 | Suppresses diagnostic message Pa050 |
| -DS32K3XX | Predefine S32K3XX as a macro, with definition 1 |
| -D $ (DERIVATIVE) | Predefine S32K3's derivative as a macro, with definition 1. For example: Predefine for S32K344 will be -DS32K344. |
| -DIAR | Predefine IAR as a macro, with definition 1 |
| -DUSE_SW_VECTOR_MODE | Predefine USE_SW_VECTOR_MODE as a macro, with definition 1. By default, the drivers are compiled to handle interrupts in Software Vector Mode. |
| -DD_CACHE_ENABLE | Predefine D_CACHE_ENABLE as a macro, with definition 1. Enables data cache initalization in source file system.↩c under the Platform driver |
| -DI_CACHE_ENABLE | Predefine I_CACHE_ENABLE as a macro, with definition 1. Enables instruction cache initalization in source file system.c under the Platform driver |
| -DENABLE_FPU | Predefine ENABLE_FPU as a macro, with definition 1. Enables FPU initalization in source file system.c under the Platform driver |
| -DMCAL_ENABLE_USER_MODE_SUPPORT | Predefine MCAL_ENABLE_USER_MODE_SUPPO↩RT as a macro, with definition 1. Allows drivers to be configured in user mode. |

#### 3.1.4.2 IAR Assembler Options

| Assembler Option | Description |
|---|---|
| –cpu Cortex-M7 | Targeted ARM processor for which IAR should generate the instruction set |
| –fpu VFPv5-SP | Use this option to generate code that performs floating-point operations using a Floating Point Unit (FPU). Single-precision variant. |
| –cpu_mode thumb | Selects the thumb mode for the assembler directive CODE |
| -g | Disables the automatic search for system include files |
| -r | Generates debug information |

#### 3.1.4.3 IAR Linker Options

| Linker Option | Description |
|---|---|
| –map filename | Produces a map file |
| –config linkerfile | Use linkerfile as the linker script. This script replaces the default linker script (rather than adding to it) |
| –cpu=Cortex-M7 | Selects the ARM processor variant to link the application for |
| –fpu VFPv5-SP | Use this option to generate code that performs floating-point operations using a Floating Point Unit (FPU). Single-precision variant. |
| –entry __start | Treats __start as a root symbol and start label |
| –enable_stack_usage | Enables stack usage analysis. If a linker map file is produced, a stack usage chapter is included in the map file |
| –skip_dynamic_initialization | Dynamic initialization (typically initialization of C++ objects with static storage duration) will not be performed automatically during application startup |
| –no_wrap_diagnostics | Does not wrap long lines in diagnostic messages |

## 3.2   Files required for compilation

This section describes the include files required to compile, assemble and link the I2c Driver for S32K3XX microcontrollers.

To avoid integration of incompatible files, all the include files from other modules shall have the same AR_MAJO↩R_VERSION and AR_MINOR_VERSION, i.e. only files with the same AUTOSAR major and minor versions can be compiled.

#### 3.2.0.0.1   I2c Driver Files:

- I2c_TS_T40D34M30I0R0\src\CDD_I2c.c

- I2c_TS_T40D34M30I0R0\src\Flexio_I2c_Ip.c

- I2c_TS_T40D34M30I0R0\src\Flexio_Ip_hw_access.h

- I2c_TS_T40D34M30I0R0\src\I2c_Ipw.c

- I2c_TS_T40D34M30I0R0\src\Lpi2c_Ip.c

- I2c_TS_T40D34M30I0R0\src\Lpi2c_Ip_HwAccess.c

- I2c_TS_T40D34M30I0R0\src\Lpi2c_Ip_Irq.c

- I2c_TS_T40D34M30I0R0\include\CDD_I2c.h

- I2c_TS_T40D34M30I0R0\include\Flexio_I2c_Ip.h

- I2c_TS_T40D34M30I0R0\include\Flexio_I2c_Ip_Types.h

- I2c_TS_T40D34M30I0R0\include\Flexio_I2c_Ip_Features.h

- I2c_TS_T40D34M30I0R0\include\I2c_Ip_Callbacks.h

- I2c_TS_T40D34M30I0R0\include\I2c_Ipw.h

- I2c_TS_T40D34M30I0R0\include\I2c_Ipw_Types.h

- I2c_TS_T40D34M30I0R0\include\I2c_Types.h

- I2c_TS_T40D34M30I0R0\include\Lpi2c_Ip.h

- I2c_TS_T40D34M30I0R0\include\Lpi2c_Ip_Features.h

- I2c_TS_T40D34M30I0R0\include\Lpi2c_Ip_HwAccess.h

- I2c_TS_T40D34M30I0R0\include\Lpi2c_IpIrq.h

- I2c_TS_T40D34M30I0R0\include\Lpi2c_Ip_Types.h


**3.2.0.0.2  I2c Driver Generated Files (must be generated by the user using a configuration tool):**

- CDD_I2c_Cfg.h

- CDD_I2c_Cfg.c

- CDD_I2c_PBcfg.h

- CDD_I2c_PBcfg.c

- I2c_Ipw_PBcfg.h

- I2c_Ipw_PBcfg.c

- Lpi2c_Ip_Cfg.h

- Lpi2c_Ip_PBcfg.c

- Lpi2c_Ip_PBcfg.h

- Flexio_I2c_Ip_Cfg.h

- Flexio_I2c_Ip_PBcfg.h

- Flexio_I2c_Ip_PBcfg.c


**3.2.0.0.3  Base Files:**

- BaseNXP_TS_T40D34M30I0R0\include\Mcal.h

- BaseNXP_TS_T40D34M30I0R0\include\I2c_MemMap.h

- BaseNXP_TS_T40D34M30I0R0\include\Platform_Types.h

- BaseNXP_TS_T40D34M30I0R0\include\Soc_Ips.h

- BaseNXP_TS_T40D34M30I0R0\include\Std_Types.h

- BaseNXP_TS_T40D34M30I0R0\include\OsIf.h

- BaseNXP_TS_T40D34M30I0R0\generate_PC\include\modules.h

- BaseNXP_TS_T40D34M30I0R0\header\S32K312_FLEXIO.h

- BaseNXP_TS_T40D34M30I0R0\header\S32K314_FLEXIO.h

- BaseNXP_TS_T40D34M30I0R0\header\S32K324_FLEXIO.h

- BaseNXP_TS_T40D34M30I0R0\header\S32K322_FLEXIO.h

- BaseNXP_TS_T40D34M30I0R0\header\S32K344_FLEXIO.h

- BaseNXP_TS_T40D34M30I0R0\header\S32K341_FLEXIO.h

- BaseNXP_TS_T40D34M30I0R0\header\S32K342_FLEXIO.h

- BaseNXP_TS_T40D34M30I0R0\header\S32K396_FLEXIO.h

- BaseNXP_TS_T40D34M30I0R0\header\S32K312_LPI2C.h

- BaseNXP_TS_T40D34M30I0R0\header\S32K314_LPI2C.h

- BaseNXP_TS_T40D34M30I0R0\header\S32K324_LPI2C.h

- BaseNXP_TS_T40D34M30I0R0\header\S32K322_LPI2C.h

- BaseNXP_TS_T40D34M30I0R0\header\S32K344_LPI2C.h

- BaseNXP_TS_T40D34M30I0R0\header\S32K341_LPI2C.h

- BaseNXP_TS_T40D34M30I0R0\header\S32K342_LPI2C.h

- BaseNXP_TS_T40D34M30I0R0\header\S32K396_LPI2C.h

### 3.2.0.0.4  DEM Files:

- Dem_TS_T40D34M30I0R0\include\Dem.h

- Dem_TS_T40D34M30I0R0\include\Dem_Types.h

- Dem_TS_T40D34M30I0R0\generate_PC\include\Dem_IntErrId.h

- Dem_TS_T40D34M30I0R0\src\Dem.c

### 3.2.0.0.5  DET Files:

- Det_TS_T40D34M30I0R0\include\Det.h

- Det_TS_T40D34M30I0R0\src\Det.c

### 3.2.0.0.6  RTE Files:

- Rte_TS_T40D34M30I0R0\include\SchM_I2c.h

- Rte_TS_T40D34M30I0R0\src\SchM_I2c.c

## 3.3   Setting up the plugins

The I2c Driver was designed to be configured by using the EB Tresos Studio (version 29.0.0 or later)

**3.3.0.0.1 Location of various files inside the I2C module folder:**

- VSMD (Vendor Specific Module Definition) file in EB Tresos Studio XDM format:
    - I2c_TS_T40D34M30I0R0\config\I2c.xdm
- VSMD (Vendor Specific Module Definition) file(s) in AUTOSAR compliant EPD format:
    - I2c_TS_T40D34M30I0R0\autosar\I2c_<subderivative_name>.epd
- Code Generation Templates for variant aware parameters:
    - I2c_TS_T40D34M30I0R0\generate_PB\src\CDD_I2c_PBcfg.c
    - I2c_TS_T40D34M30I0R0\generate_PB\src\I2c_Ipw_PBcfg.c
    - I2c_TS_T40D34M30I0R0\generate_PB\src\Lpi2c_Ip_PBcfg.c
    - I2c_TS_T40D34M30I0R0\generate_PB\src\Flexio_I2c_Ip_PBcfg.c
    - I2c_TS_T40D34M30I0R0\generate_PB\src\CDD_I2c_PBcfg.h
    - I2c_TS_T40D34M30I0R0\generate_PB\src\I2c_Ipw_PBcfg.h
    - I2c_TS_T40D34M30I0R0\generate_PB\src\Lpi2c_Ip_PBcfg.h
    - I2c_TS_T40D34M30I0R0\generate_PB\src\Flexio_I2c_Ip_PBcfg.h
- Code Generation Templates for parameters without variation points:
    - I2c_TS_T40D34M30I0R0\generate_PC\src\CDD_I2c_Cfg.c
    - I2c_TS_T40D34M30I0R0\generate_PC\include\CDD_I2c_Cfg.h

**3.3.0.0.2 Steps to generate the configuration:**

1. Copy the following module folders into the Tresos plugins folder:
    - BaseNXP_TS_T40D34M30I0R0
    - Dem_TS_T40D34M30I0R0
    - Det_TS_T40D34M30I0R0
    - EcuC_TS_T40D34M30I0R0
    - Rte_TS_T40D34M30I0R0
    - Resource_TS_T40D34M30I0R0
    - Mcl_TS_T40D34M30I0R0
    - Platform_TS_T40D34M30I0R0
    - Mcu_TS_T40D34M30I0R0
    - I2c_TS_T40D34M30I0R0
2. Set the desired Tresos Output location folder for the generated sources and header files.
3. Use the EB Tresos Studio GUI to modify ECU configuration parameters values.
4. Generate the configuration files

# Chapter 4

# Function calls to module

- Function Calls during Start-up
- Function Calls during Shutdown
- Function Calls during Wake-up

## 4.1 Function Calls during Start-up

I2C shall be initialized during STARTUP phase of EcuM initialization. The API to be called for this is I2c_Init(). The MCU module should be initialized before the I2C is initialized. The API to be called for this purpose is I2c↩ _Init(). The PORT, MCL (if the DMA option is used) and OsIf (if blocking function are used) modules shall be initialized before I2C is initialized.

## 4.2 Function Calls during Shutdown

I2C can be silenced by calling I2c_DeInit().

## 4.3 Function Calls during Wake-up

N/A

# Chapter 5

# Module requirements

- Exclusive areas to be defined in BSW scheduler

- Exclusive areas not available on this platform

- Peripheral Hardware Requirements

- ISR to configure within AutosarOS - dependencies

- ISR Macro

- Other AUTOSAR modules - dependencies

- Data Cache Restrictions

- User Mode support

- Multicore support

## 5.1 Exclusive areas to be defined in BSW scheduler

In the current implementation, I2c is using the services of Schedule Manager (SchM) for entering and exiting the exclusive areas. The following critical regions are used in the I2c driver:

**I2C_EXCLUSIVE_AREA_00** is used in function I2c_SyncTransmit to protect I2C_aeChannelStatus. **I2C↩_EXCLUSIVE_AREA_01** is used in function I2c_AsyncTransmit to protect I2C_aeChannelStatus.

**Critical Region Exclusive Matrix** Below is the table depicting the exclusivity between different critical region IDs from the I2c driver. If there is an "X" in a table, it means that those 2 critical regions cannot interrupt each other.

| EXCLUSIVE_AREA | I2C_EXCLUSIVE_AREA_00 | I2C_EXCLUSIVE_AREA_01 |
|---|---|---|
| I2C_EXCLUSIVE_AREA_00 | x | x |
| I2C_EXCLUSIVE_AREA_01 | x | x |

## 5.2 Exclusive areas not available on this platform

List of exclusive areas which are not available on this platform (or blank if they're all available).

## 5.3 Peripheral Hardware Requirements

N/A.

## 5.4 ISR to configure within AutosarOS - dependencies

| ISR Name | Hardware interrupt vector |
|---|---|
| LPI2C0_Master_Slave_IRQHandler | 161 |
| LPI2C1_Master_Slave_IRQHandler | 162 |

## 5.5 ISR Macro

RTD drivers use the ISR macro to define the functions that will process hardware interrupts. Depending on whether the OS is used or not, this macro can have different definitions.

### 5.5.1 Without an Operating System   The macro *USING_OS_AUTOSAROS* must not be defined.

#### 5.5.1.1 Using Software Vector Mode

The macro *USE_SW_VECTOR_MODE* must be defined and the ISR macro is defined as:

#define ISR(IsrName) void IsrName(void)

In this case, the drivers' interrupt handlers are normal C functions and their prologue/epilogue will handle the context save and restore.

#### 5.5.1.2 Using Hardware Vector Mode

The macro *USE_SW_VECTOR_MODE* must not defined and the ISR macro is defined as:

#define ISR(IsrName) INTERRUPT_FUNC void IsrName(void)

In this case, the drivers' interrupt handlers must also handle the context save and restore.

### 5.5.2 With an Operating System   Please refer to your OS documentation for description of the ISR macro.

## 5.6    Other AUTOSAR modules - dependencies

- **Port**: Needed to configure the pins to be used by the I2c driver.

- **DET**: Needed for detecting and reporting development errors. The API function used is *Det_ReportError*. The detection can be configured using the *EthDevErrorDetect* configuration parameter

- **DEM**: Needed for detecting and reporting extended production errors.

- **RTE**: Needed for implementing data consistency through exclusive areas.

- **MCU**: Needed to configure the clocks to be used by the I2c driver.

- **MCL**: Needed to configure the DMA channels used by the I2c driver.

- **EcuC**: Needed to retrieve information about post-build variants.

- **Base**: Needed for common files/definitions which are used by all RTD modules.

- **OS**: Needed to define a mapping between EcuC partitions and EcuC core ids when multicore support is enabled.

## 5.7    Data Cache Restrictions

In the DMA transfer mode, DMA transfers may issue cache coherency problems. To avoid possible coherency issues when D-CACHE is enabled, the user shall ensure that the buffers used as TCD source and destination are allocated in the NON-CACHEABLE area (by means of I2c_Memmap).

## 5.8    User Mode support

- User Mode configuration in the module

- User Mode configuration in AutosarOS

### 5.8.1    User Mode configuration in the module
No special measures need to be taken to run I2C module from user mode. The I2C driver code can be executed at any time from both supervisor and user mode.

### 5.8.2    User Mode configuration in AutosarOS

When User mode is enabled, the driver may has the functions that need to be called as trusted functions in AutosarOS context. Those functions are already defined in driver and declared in the header <IpName>_Ip↩ _TrustedFunctions.h. This header also included all headers files that contains all types definition used by parameters or return types of those functions. Refer the chapter User Mode configuration in the module for more detail about those functions and the name of header files they are declared inside. Those functions will be called indirectly with the naming convention below in order to AutosarOS can call them as trusted functions.

`Call_<Function_Name>_TRUSTED(parameter1,parameter2,...)`

That is the result of macro expansion `OsIf_Trusted_Call` in driver code:

#define OsIf_Trusted_Call[1-6params](name,param1,...,param6) Call_##name##_TRUSTED(param1,...,param6)

So, the following steps need to be done in AutosarOS:

- Ensure `MCAL_ENABLE_USER_MODE_SUPPORT` macro is defined in the build system or somewhere global.

- Define and declare all functions that need to call as trusted functions follow the naming convention above in Integration/User code. They need to visible in `Os.h` for the driver to call them. They will do the marshalling of the parameters and call `CallTrustedFunction()` in OS specific manner.

- `CallTrustedFunction()` will switch to privileged mode and call `TRUSTED_<Function_Name>()`.

- `TRUSTED_<Function_Name>()` function is also defined and declared in Integration/User code. It will unmarshalling of the parameters to call `<Function_Name>()` of driver. The `<Function_Name>()` functions are already defined in driver and declared in `<IpName>_Ip_TrustedFunctions.h`. This header should be included in OS for OS call and indexing these functions.

See the sequence chart below for an example calling `Linflexd_Uart_Ip_Init_Privileged()` as a trusted function.
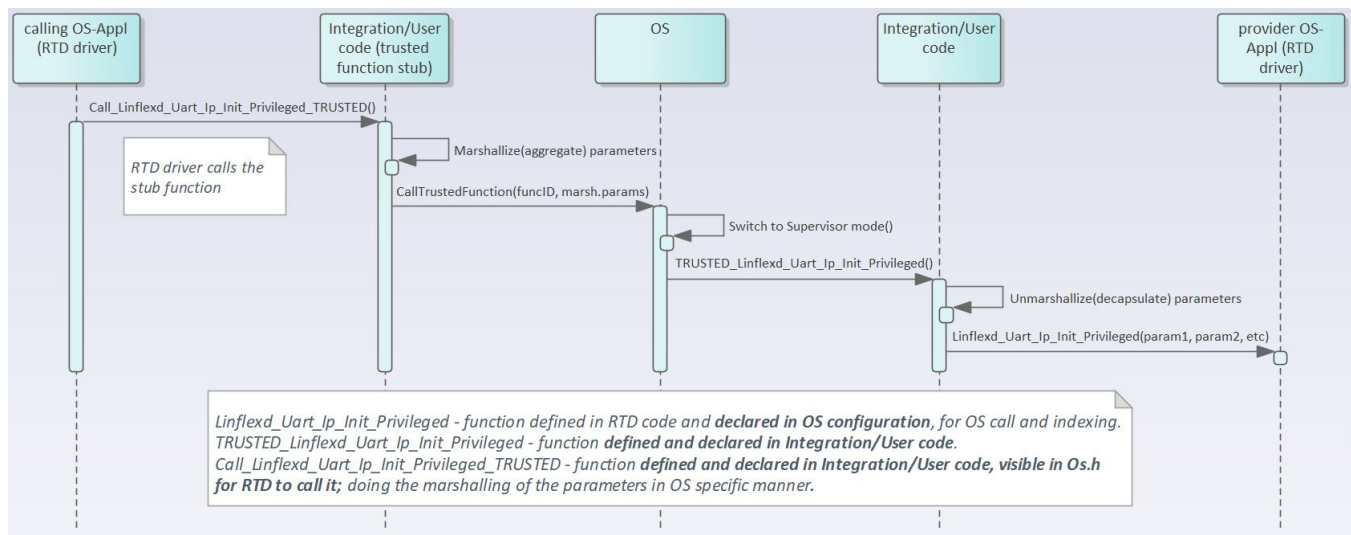


**Figure 5.1 Example sequence chart for calling `Linflexd_Uart_Ip_Init_Privileged` as trusted function**

## 5.9 Multicore support

The I2C implements the "Autosar 4.4 MCAL Multicore Distribution" according to type II, in which the mappable element is set to (Channel / HW Unit / Job ,etc). For additional details, please refer to AUTOSAR_EXP_BSW↩ DistributionGuide. The I2c and the mappable elements can be allocated to zero, one or several ECUC partitions, by means of "I2cEcucPartionRef". If the I2c is mapped to zero ECUC partitions, the I2c behavior reverts to single-core implementation, similar to previous Autosar versions. If the I2c is mapped to one or more ECUC partitions, the I2c enforces the following multi-core assumptions:

- The I2c assumes there is a single EcucPartition allocated per core. Internally, the module will use the Core ID returned by GetCoreID API to reference the appropriate global data and configuration elements.

- The I2c assumes the EcucCoreIDs are defined in a compact/consecutive order, starting from zero. The rationale is that the number of EcucPartitions is used for dimensioning the I2c internal variables and the EcucCoreIDs are used for indexing those variables. (AR-86601 Zero based and dense IDs for OS-Cores and OSApplications).

- The I2c assumes that initialization is performed on each core, I2c_Init() is called separately for each core, using a different configguration structure. (Type II)

- The I2c initialization expects the upper layer will pass the correct initialization pointer, specific to the partition in which the driver is to be used. For example: EcucPartition_1 is assigned to CoreID 1; I2c_Init function will be called with I2c_Config_EcucPartition_1 configuration structure, on Core 1.

- The I2c will check upon each API call if the requested resource is configured to be available on the current core, if DET error reporting is enabled.

- The I2c requires that all variables in NonCacheable MemMap sections be allocated accordingly, to avoid data corruption in multicore context.

- The I2c assumes that RTE module implements the EXCLUSIVE AREAS to be coreaware only. The rationale is that the module implementation ensures data integrity by separating the mappable elements for different cores already, thus implementing the EXCLUSIVE AREAS in a blocking manner (ex: spin-lock) on a multicore scope, might affect the performance of the drivers on the two cores, although they might access separate HW elements. For single-core scope, the EXCLUSIVE AREAS keep the same purpose as on previous AUTOSAR implementations. (∗ - to be updated per I2c usecase, to be detailed/removed if some modules require such kind of functionality for critical features which cannot be atomically shared among cores)

- The I2c assumes that each interrupt is routed by the system only to the core on which is supposed to be serviced.

**Chapter 6**

# Main API Requirements

## 6.1   Main function calls within BSW scheduler

N/A.

## 6.2   API Requirements

N/A.

## 6.3   Calls to Notification Functions, Callbacks, Callouts

**Call-back Notifications:** N/A

**User Notification:** The I2C Handler & Driver provides notifications per job and sequence in synchronous mode. The notifications can be configured as pointers to user defined functions. If notification is not desired, the appropriate EndNotification field shall be left blank. For asynchronous transmissions, job and sequences notifications are performed before the scheduling of the next job. In this way, call I2c_PrepareSlaveBuffer() can be targeted on the next schedulable jobs, before the starting of the job transfer.

# Chapter 7

# Memory allocation

- Sections to be defined in I2c_MemMap.h

- Linker command file

## 7.1 Sections to be defined in I2c_MemMap.h

| Section name | Type of section | Description |
|---|---|---|
| I2C_START_SEC_CONFIG_DA←TA_UNSPECIFIED | Configuration Data | Start of Memory Section for Config Data |
| I2C_STOP_SEC_CONFIG_DAT←A_UNSPECIFIED | Configuration Data | End of Memory Section for Config Data |
| I2C_START_SEC_CONST_32 | Configuration Data | Start of Memory Section for Config Data. |
| I2C_STOP_SEC_CONST_32 | Configuration Data | End of Memory Section for Config Data |
| I2C_START_SEC_CODE | Code | Start of memory Section for Code |
| I2C_STOP_SEC_CODE | Code | End of memory Section for Code |
| I2C_START_SEC_VAR_CLEA←RED_32 | Variables | Used for variables which have to be aligned to 32 bit. For instance used for variables of size 32 bit or used for composite data types:arrays, structures containing elements of maximum 32 bits. These variables are never cleared and never initialized by start-up code. |
| I2C_STOP_SEC_VAR_CLEAR←ED_32 | Variables End of above section. | End of above section. |
| I2C_START_SEC_VAR_CLEA←RED_UNSPECIFIED | Variables | Used for variables, structures, arrays when the SIZE (alignment) does not fit the criteria of 8,16 or 32 bit. These variables are never cleared and never initialized by start-up code. |
| I2C_STOP_SEC_VAR_CLEAR←ED_UNSPECIFIED | Variables | End of above section. |

| Section name | Type of section | Description |
|---|---|---|
| I2C_START_SEC_VAR_CLEA←RED_UNSPECIFIED_NO_CAC←HEABLE | Variables | Used for variables, structures, arrays when the SIZE (alignment) does not fit the criteria of 8,16 or 32 bit, and that have to be stored in a non-cacheable memory section. These variables are never cleared and never initialized by start-up code. |
| I2C_STOP_SEC_VAR_CLEAR←ED_UNSPECIFIED_NO_CACH←EABLE | Variables | End of above section. |

## 7.2 Linker command file

Memory shall be allocated for every section defined in the driver's "<Module>"_MemMap.h.

# Chapter 8

# Integration Steps

This section gives a brief overview of the steps needed for integrating this module:

1. Generate the required module configuration(s). For more details refer to section Files Required for Compilation

2. Allocate the proper memory sections in the driver's memory map header file ("<Module>"_MemMap.h) and linker command file. For more details refer to section Sections to be defined in <Module>_MemMap.h

3. Compile & build the module with all the dependent modules. For more details refer to section Building the Driver

# Chapter 9

# External assumptions for driver

The section presents requirements that must be complied with when integrating the I2C driver into the application.

| External Assumption Req ID | External Assumption Text |
| --- | --- |
| EA_RTD_00071 | If interrupts are locked, a centralized function pair to lock and unlock interrupts shall be used. |
| EA_RTD_00081 | The integrator shall assure that <MSN>_Init() and <MSN>_DeInit() functions do not interrupt each other. |
| EA_RTD_00082 | When caches are enabled and data buffers are allocated in cacheable memory regions the buffers involved in DMA transfer shall be aligned with both start and end to cache line size. Note: **Rationale**: This ensures that no other buffers/variables compete for the same cache lines. |
| EA_RTD_00092 | The integrator shall allocate a single EcucPartition per core or the partition in which the I2c is allocated shall be exclusively mapped to a core. Note↩: Internally, the I2c will use the Core ID returned by GetCoreID API to reference the appropriate global data and configuration elements, that is why a core should reference only one configured partition. |
| EA_RTD_00093 | The application shall define EcucCoreIDs in a compact/consecutive order, starting from zero. |
| EA_RTD_00094 | When multicore support is enabled, the application shall call I2c_Init() for each core, using the dedicated configuration pointer for that core. |
| EA_RTD_00096 | The application shall pass the correct initialization pointer, specific to the partition in which the driver is to be used. |
| EA_RTD_00106 | Standalone IP configuration and HL configuration of the same driver shall be done in the same project |
| EA_RTD_00107 | The integrator shall use the IP interface only for hardware resources that were configured for standalone IP usage. Note: The integrator shall not directly use the IP interface for hardware resources that were allocated to be used in HL context. |
| EA_RTD_00108 | The integrator shall use the IP interface to a build a CDD, therefore the BSWMD will not contain reference to the IP interface |
| EA_RTD_00113 | When RTD drivers are integrated with AutosarOS and User mode support is enabled, the integrator shall assure that the definition and declaration of all RTD functions needed to be called as trusted functions follow the naming convention Call<Function_Name>TRUSTE↩D(parameter1,parameter2,...) in Integration/User code. They need to visible in Os.h for the driver to call them. They will call RTD <Function_↩Name>() as trusted functions in OS specific manner. |