# Integration Manual

for S32K1 RM Driver

Document Number: IM2RMASR4.4 Rev0000R1.0.1 Rev. 1.0

---

**S32K1 RM Driver**

# Chapter 1

# Revision History

| Revision | Date | Author | Description |
|---|---|---|---|
| 1.0 | 24.02.2022 | NXP RTD Team | Prepared for release RTD S32K1 Version 1.0.1 |

# Chapter 2

# Introduction

- Supported Derivatives

- Overview

- About This Manual

- Acronyms and Definitions

- Reference List

This integration manual describes the integration requirements for RM CDD Driver for S32K1 microcontrollers.

## 2.1   Supported Derivatives

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

- s32k116_qfn32

- s32k116_lqfp48

- s32k118_lqfp48

- s32k118_lqfp64

- s32k142_lqfp48

- s32k142_lqfp64

- s32k142_lqfp100

- s32k142w_lqfp48

- s32k142w_lqfp64

- s32k144_lqfp48

- s32k144_lqfp64

- s32k144_lqfp100

- s32k144_mapbga100

- s32k144w_lqfp48

- s32k144w_lqfp64

- s32k146_lqfp64

- s32k146_lqfp100

- s32k146_mapbga100

- s32k146_lqfp144

- s32k148_lqfp100

- s32k148_mapbga100

- s32k148_lqfp144

- s32k148_lqfp176

All of the above microcontroller devices are collectively named as S32K1.

## 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

**S32K1 RM Driver**

## 2.4   Acronyms and Definitions

| Term | Definition |
|---|---|
| API | Application Programming Interface |
| ASM | Assembler |
| BSMI | Basic Software Make file Interface |
| C/CPP | C and C++ Source Code |
| CDD | Complex Device Driver |
| DEM | Diagnostic Event Manager |
| DET | Development Error Tracer |
| ECU | Electronic Control Unit |
| LSB | Least Signifigant Bit |
| MCU | Micro Controller Unit |
| MIDE | Multi Integrated Development Environment |
| MPU | Memory Protection Unit |
| MRC | Memory Region Controller |
| MSB | Most Significant Bit |
| N/A | Not Applicable |
| RAM | Random Access Memory |
| RM | Resource Manager |
| SIU | Systems Integration Unit |
| SWS | Software Specification |
| XML | Extensible Markup Language |

## 2.5   Reference List

| # | Title | Version |
|---|---|---|
| 1 | S32K1 Series Reference Manual | Rev. 14, 09/2021 |
| 2 | S32K116_0N96V | Rev. 22/OCT/2021 |
| 3 | S32K118_0N97V | Rev. 22/OCT/2021 |
| 4 | S32K142_0N33V | Rev. 22/OCT/2021 |
| 5 | S32K144_0N57U | Rev. 22/OCT/2021 |
| 6 | S32K144W_0P64A | Rev. 22/OCT/2021 |
| 7 | S32K146_0N73V | Rev. 22/OCT/2021 |
| 8 | S32K148_0N20V | Rev. 22/OCT/2021 |
| 9 | S32K1xx Data Sheet | Rev. 14, 08/2021 |

# 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
- GHS Compiler/Assembler/Linker Options
- IAR Compiler/Assembler/Linker Options

The RTD driver files are compiled using:

- NXP GCC 9.2.0 20190812 (Build 1649 Revision gaf57174)
- IAR ANSI C/C++ Compiler V8.40.3.228/W32 for ARM Functional Safety
- Green Hills Multi 7.1.6d / Compiler 2020.1.4

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

The TS_T40D2M10I1R0 part of the plugin name is composed as follows:

- T = Target_Id (e.g. T40 identifies Cortex-M architecture)
- D = Derivative_Id (e.g. D2 identifies S32K1 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-m4 | Targeted ARM processor for which GCC should tune the performance of the code (for S32K14x devices) |
| -mcpu=cortex-m0plus | Targeted ARM processor for which GCC should tune the performance of the code (for S32K11x devices) |
| -mthumb | Generates code that executes in Thumb state |
| -mlittle-endian | Generate code for a processor running in little-endian mode |
| -mfpu=fpv4-sp-d16 | Specifies the floating-point hardware available on the target (for S32K14x devices) |
| -mfloat-abi=hard | Specifies the floating-point ABI to use. "hard" allows generation of floating-point instructions and uses FPU-specific calling conventions (for S32K14x devices) |
| -mfpu=auto | Specifies the floating-point hardware available on the target (for S32K11x devices) |
| -mfloat-abi=soft | Specifies the floating-point ABI to use. Specifying "soft" causes GCC to generate output containing library calls for floating-point operations (for S32K11x devices) |
| -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. |

**S32K1 RM Driver**

**Building the driver**

| Compiler Option | Description |
|---|---|
| -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 |
| -fomit-frame-pointer | Omit the frame pointer in functions that dont need one. This avoids the instructions to save, set up and restore the frame pointer; on many targets it also makes an extra register available. |
| -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 | Makes the compiler output stack usage information for the program, on a per-function basis |
| -fdump-ipa-all | Enables all inter-procedural analysis dumps |
| -c | Stop after assembly and produce an object file for each source file |
| -DS32K1XX | Predefine S32K1XX as a macro, with definition 1 |
| -DS32K148 | Predefine S32K148 as a macro, with definition 1 |
| -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 |
| -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 (for S32K14x devices) |
| -DENABLE_FPU | Predefine ENABLE_FPU as a macro, with definition 1. Enables FPU initalization in source file system.c under the Platform driver (for S32K14x devices) |
| -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. |

### 3.1.1.2 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=cortex-m4 | Targeted ARM processor for which GCC should tune the performance of the code (for S32K14x devices) |
| -mcpu=cortex-m0plus | Targeted ARM processor for which GCC should tune the performance of the code (for S32K11x devices) |
| -mthumb | Generates code that executes in Thumb state |
| -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=cortex-m4 | Targeted ARM processor for which GCC should tune the performance of the code (for S32K14x devices) |
| -mcpu=cortex-m0plus | Targeted ARM processor for which GCC should tune the performance of the code (for S32K11x devices) |
| -mthumb | Generates code that executes in Thumb state |
| -mfpu=fpv4-sp-d16 | Specifies the floating-point hardware available on the target (for S32K14x devices) |
| -mfloat-abi=hard | Specifies the floating-point ABI to use. "hard" allows generation of floating-point instructions and uses FPU-specific calling conventions (for S32K14x devices) |
| -mfpu=auto | Specifies the floating-point hardware available on the target (for S32K11x devices) |
| -mfloat-abi=soft | Specifies the floating-point ABI to use. Specifying "soft" causes GCC to generate output containing library calls for floating-point operations (for S32K11x devices) |
| -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 |
| -n | Turn off page alignment of sections, and disable linking against shared libraries |

## 3.1.2 GHS Compiler/Assembler/Linker Options

### 3.1.2.1 GHS Compiler Options

| Compiler Option | Description |
| --- | --- |
| -cpu=cortexm4 | Selects target processor: Arm Cortex M4 (for S32K14x devices) |
| -cpu=cortexm0plus | Selects target processor: Arm Cortex M0+ (for S32K11x devices) |
| -thumb | Selects generating code that executes in Thumb state |
| -fpu=vfpv4_d16 | Specifies hardware floating-point using the v4 version of the VFP instruction set, with 16 double-precision floating-point registers (for S32K14x devices) |
| -fsingle | Use hardware single-precision, software double-precision FP instructions (for S32K14x devices) |

| Compiler Option | Description |
|---|---|
| -fsoft | Specifies software floating-point (SFP) mode. This setting causes your target to use integer registers to hold floating-point data and use library subroutine calls to emulate floating-point operations (for S32K11x devices) |
| -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 |
| –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 |
| -DS32K1XX | Predefine S32K1XX as a macro, with definition 1 |
| -DS32K148 | Predefine S32K148 as a macro, with definition 1 |
| -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 |
| -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 (for S32K14x devices) |
| -DENABLE_FPU | Predefine ENABLE_FPU as a macro, with definition 1. Enables FPU initalization in source file system.c under the Platform driver (for S32K14x devices) |

| Compiler Option | Description |
|---|---|
| -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 GHS Assembler Options

| Assembler Option | Description |
|---|---|
| -cpu=cortexm4 | Selects target processor: Arm Cortex M4 (for S32K14x devices) |
| -cpu=cortexm0plus | Selects target processor: Arm Cortex M0+ (for S32K11x devices) |
| -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 |

### 3.1.2.3 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 (for S32K14x devices) |
| -lind_sf | 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 (for S32K11x devices) |
| -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 |

**S32K1 RM Driver**

| Linker Option | Description |
|---|---|
| -nostartfiles | Controls the start files to be linked into the executable |

## 3.1.3   IAR Compiler/Assembler/Linker Options

### 3.1.3.1   IAR Compiler Options

| Compiler Option | Description |
|---|---|
| –cpu=Cortex-M4 | Targeted ARM processor for which IAR should tune the performance of the code (for S32K14x devices) |
| –cpu=Cortex-M0+ | Targeted ARM processor for which IAR should tune the performance of the code (for S32K11x devices) |
| –cpu_mode=thumb | Generates code that executes in Thumb state |
| –endian=little | Generate code for a processor running in little-endian mode |
| –fpu=FPv4-SP | Use this option to generate code that performs floating-point operations using a Floating Point Unit (FPU). Single-precision variant. (for S32K14x devices) |
| –fpu=none | Use this option to generate code that performs floating-point operations using a Floating Point Unit (FPU). No FPU. (for S32K11x devices) |
| -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 |
| –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 |
| –no_explicit_zero_opt | Do not treat explicit initializations to zero of static variables as zero 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 |
| -DS32K1XX | Predefine S32K1XX as a macro, with definition 1 |
| -DS32K148 | Predefine S32K148 as a macro, with definition 1 |
| -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. |

**S32K1 RM Driver**

| Compiler Option | Description |
|---|---|
| -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 (for S32K14x devices) |
| -DENABLE_FPU | Predefine ENABLE_FPU as a macro, with definition 1. Enables FPU initalization in source file system.c under the Platform driver (for S32K14x devices) |
| -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 IAR Assembler Options

| Assembler Option | Description |
|---|---|
| –cpu=Cortex-M4 | Targeted ARM processor for which IAR should tune the performance of the code (for S32K14x devices) |
| –cpu=Cortex-M0+ | Targeted ARM processor for which IAR should tune the performance of the code (for S32K11x devices) |
| –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 |

### 3.1.3.3 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-M4 | Targeted ARM processor for which IAR should tune the performance of the code (for S32K14x devices) |
| –cpu=Cortex-M0+ | Targeted ARM processor for which IAR should tune the performance of the code (for S32K11x devices) |
| –fpu=FPv4-SP | Use this option to generate code that performs floating-point operations using a Floating Point Unit (FPU). Single-precision variant. (for S32K14x devices) |
| –fpu=none | Use this option to generate code that performs floating-point operations using a Floating Point Unit (FPU). No FPU. (for S32K11x devices) |
| –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 AUTOSAR Rm Driver for S32K1 microcontrollers.

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

### 3.2.1   RM Driver Files

- Source files:

    – Rm_TS_T40D2M10I1R0\src\CDD_Rm.c

    – Rm_TS_T40D2M10I1R0\src\CDD_Rm_Ipw.c

    – Rm_TS_T40D2M10I1R0\src\Mpu_Ip.c

- Include file:

    – Rm_TS_T40D2M10I1R0\include\CDD_Rm.h

    – Rm_TS_T40D2M10I1R0\include\CDD_Rm_Ipw.h

    – Rm_TS_T40D2M10I1R0\include\Mpu_Ip_TypesDef.h

    – Rm_TS_T40D2M10I1R0\include\Mpu_Ip.h

    – Rm_TS_T40D2M10I1R0\include\Mpu_Ip_TrustedFunctions.h

    Note

       These files should be generated by the user using a configuration/generation tool

       – CDD_Rm_Cfg.c
       – CDD_Rm_PBcfg.c
       – CDD_Rm_Ipw_PBcfg.c
       – Mpu_Ip_PBcfg.c
       – CDD_Rm_Cfg.h
       – CDD_Rm_PBcfg.h
       – CDD_Rm_Ipw_Cfg.h
       – CDD_Rm_Ipw_PBcfg.h
       – Mpu_Ip_Cfg.h
       – Mpu_Ip_Cfg_Defines.h
       – Mpu_Ip_PBcfg.h

### 3.2.2   Other includes files

**S32K1 RM Driver**

#### 3.2.2.1    Files from Base common folder

- Base\_TS\_T40D2M10I1R0\include\Mcal.h

- Base\_TS\_T40D2M10I1R0\include\BasicTypes.h

- Base\_TS\_T40D2M10I1R0\include\StandardTypes.h

- Base\_TS\_T40D2M10I1R0\include\Devassert.h

- Base\_TS\_T40D2M10I1R0\include\Rm\_MemMap.h

- Base\_TS\_T40D2M10I1R0\include\Osif.h

- Base\_TS\_T40D2M10I1R0\header\S32K148\_MPU.h (depending on the derivatives)

#### 3.2.2.2    Files from Det folder

- Det\_TS\_T40D2M10I1R0\include\Det.h

#### 3.2.2.3    Files from RTE Files

- Rte\_TS\_T40D2M10I1R0\include\SchM\_Rm.h

## 3.3    Setting up the plugins

The RM CDD driver was designed to be configured by using the EB Tresos Studio (version 27.1.0 b200625-0900 or later)

#### 3.3.0.0.1    Location of various files inside the RM module folder:

- VSMD (Vendor Specific Module Definition) file in EB Tresos Studio XDM format:
    - Rm\_TS\_T40D2M10I1R0\config\Rm.xdm
- VSMD (Vendor Specific Module Definition) file(s) in AUTOSAR compliant EPD format:
    - Rm\_TS\_T40D2M10I1R0\autosar\Rm\_<subderivative\_name>.epd
- Code Generation Templates :
    - Rm\_TS\_T40D2M10I1R0\generate\_PC\src\CDD\_Rm\_Cfg.c
    - Rm\_TS\_T40D2M10I1R0\generate\_PB\src\CDD\_Rm\_Ipw\_PBcfg.c
    - Rm\_TS\_T40D2M10I1R0\generate\_PB\src\CDD\_Rm\_PBcfg.c
    - Rm\_TS\_T40D2M10I1R0\generate\_PC\src\Mpu\_Ip\_Cfg.c

    - Rm\_TS\_T40D2M10I1R0\generate\_PB\src\Mpu\_Ip\_PBcfg.c

    - Rm\_TS\_T40D2M10I1R0\generate\_PC\include\CDD\_Rm\_Cfg.h
    - Rm\_TS\_T40D2M10I1R0\generate\_PC\include\CDD\_Rm\_Ipw\_Cfg.h
    - Rm\_TS\_T40D2M10I1R0\generate\_PB\include\CDD\_Rm\_Ipw\_PBcfg.h
    - Rm\_TS\_T40D2M10I1R0\generate\_PB\include\CDD\_Rm\_PBcfg.h
    - Rm\_TS\_T40D2M10I1R0\generate\_PC\include\Mpu\_Ip\_Cfg.h
    - Rm\_TS\_T40D2M10I1R0\generate\_PC\include\Mpu\_Ip\_Cfg\_Defines.h
    - Rm\_TS\_T40D2M10I1R0\generate\_PB\include\Mpu\_Ip\_PBcfg.h

**S32K1 RM Driver**

**3.3.0.0.2   Steps to generate the configuration:**

1. Copy the following module folders into the Tresos plugins folder:

   - Rm_TS_T40D2M10I1R0
   - Base_TS_T40D2M10I1R0
   - Det_TS_T40D2M10I1R0
   - Platform_TS_T40D2M10I1R0
   - Resource_TS_T40D2M10I1R0
   - Rte_TS_T40D2M10I1R0

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

RM CDD driver shall be initialized during STARTUP phase of EcuM initialization. The API member to be called to accomplish this is Rm_Init.

The MCU module should be initialized before RM CDD module is initialized.

## 4.2   Function Calls during Shutdown

None.

## 4.3   Function Calls during Wake-up

None.

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

**RM_EXCLUSIVE_AREA_04** is used in function Rm_Init(), Rm_Mpu_SetRegionConfig() to protect the updates for RGDAAC0, RGDn_WORD0, RGDn_WORD1, RGDn_WORD2, RGDn_WORD3 registers

**RM_EXCLUSIVE_AREA_05** is used in function Rm_Mpu_EnableRegion() to protect the updates for RG↩Dn_WORD3 register

**RM_EXCLUSIVE_AREA_06** is used in function Rm_Mpu_SetAccessMode() to protect the updates for R↩GDAACn register

The critical regions from interrupts are grouped in "Interrupt Service Routines Critical Regions (composed diagram)". If an exclusive area is "exclusive" with the composed "Interrupt Service Routines Critical Regions (composed diagram)" group, it means that it is exclusive with each one of the ISR critical regions.

**Critical Region Exclusive Matrix :**

- Below is the table depicting the exclusivity between different critical region IDs from the Rm driver.

- If there is an "X" in a table, it means that those 2 critical regions cannot interrupt each other.

| Rm__EXCLUSIVE_AREA | AREA_4 | AREA_5 | AREA_6 |
|---|---|---|---|
| AREA_4 | x | X | X |
| AREA_5 | X | x | |
| AREA_6 | 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).

- **RM_EXCLUSIVE_AREA_00**

- **RM_EXCLUSIVE_AREA_01**

- **RM_EXCLUSIVE_AREA_02**

- **RM_EXCLUSIVE_AREA_03**

- **RM_EXCLUSIVE_AREA_07**

- **RM_EXCLUSIVE_AREA_08**

- **RM_EXCLUSIVE_AREA_09**

- **RM_EXCLUSIVE_AREA_10**

- **RM_EXCLUSIVE_AREA_11**

- **RM_EXCLUSIVE_AREA_12**

## 5.3   Peripheral Hardware Requirements

None

## 5.4   ISR to configure within AutosarOS - dependencies

The RM CDD does not have specific interrupt channels. Access violations detected by MPU raise a Hard Fault or Memory Management Core Exceptions.

## 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.

**S32K1 RM Driver**

**Module requirements**

### 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 Data Cache Restrictions

None.

## 5.7 User Mode support

- user_mode_config_in_module

- User Mode configuration in AutosarOS

**S32K1 RM Driver**

### 5.7.1 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_config_in_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.
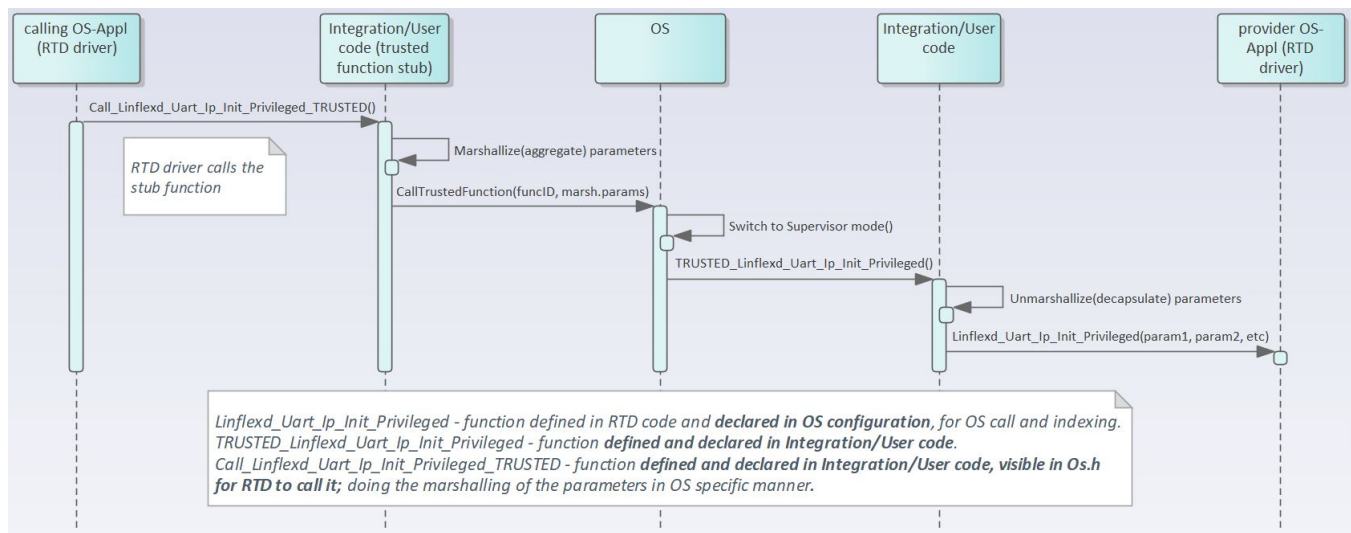


**Figure 5.1 Example sequence chart for calling `Linflexd_Uart_Ip_Init_Privileged` as trusted function**

## 5.8 Multicore support

None.

**Chapter 6**

# Main API Requirements

-

-

-

## 6.1   Main function calls within BSW scheduler

None.

## 6.2   API Requirements

None.

## 6.3   Calls to Notification Functions, Callbacks, Callouts

The RM CDD does not make use of any callbacks or notifications.

# Chapter 7

# Memory allocation

- Sections to be defined in Rm_MemMap.h

- Linker command file

## 7.1   Sections to be defined in Rm_MemMap.h

| Section name | Type of section | Description |
|---|---|---|
| RM_START_SEC_CONFIG_DAT↩A_UNSPECIFIED | Configuration Data | Start of Memory Section for Config Data |
| RM_STOP_SEC_CONFIG_DATA↩_UNSPECIFIED | Configuration Data | End of above section. |
| RM_START_SEC_CODE | Code | Start of memory Section for Code |
| RM_STOP_SEC_CODE | Code | End of above section. |
| RM_START_SEC_VAR_INIT_UN↩SPECIFIED_NO_CACHEABLE | Non-Cacheable 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 initialized by start-up code. |
| RM_STOP_SEC_VAR_INIT_UNS↩PECIFIED_NO_CACHEABLE | Non-Cacheable Variables | End of above section. |
| RM_START_SEC_CONST_UNSPE↩CIFIED | Constant Data | Used for constants. |
| RM_STOP_SEC_CONST_UNSPEC↩IFIED | Constant Data | 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 RM 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_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 |

**S32K1 RM Driver**