


Integration Manual

for S32K3 ICU Driver

Document Number: IM34ICUASRR21-11 Rev0000R3.0.0 Rev. 1.0

1 Revision History	2
2 Introduction	3
2.1 Supported Derivatives	3
2.2 Overview	4
2.3 About This Manual	5
2.4 Acronyms and Definitions	6
2.5 Reference List	6
3 Building the driver	8
3.1 Build Options	8
3.1.1 GCC Compiler/Assembler/Linker Options	9
3.1.2 DIAB Compiler/Assembler/Linker Options	11
3.1.3 GHS Compiler/Assembler/Linker Options	13
3.1.4 IAR Compiler/Assembler/Linker Options	15
3.2 Files required for Compilation	17
3.3 Setting up the Plug-ins	20
4 Function calls to module	21
4.1 Function Calls during Startup	21
4.2 Function Calls during Shutdown	21
4.3 Function Calls during Wakeup	21
5 Module requirements	22
5.1 Exclusive areas to be defined in BSW scheduler	22
5.2 Exclusive areas not available on this platform	27
5.3 Peripheral Hardware Requirements	27
5.4 ISR to Configure Within OS – Dependencies	27
5.5 ISR Macro	29
5.5.1 Without an Operating System	29
5.5.2 With an Operating System	29
5.6 Other AUTOSAR modules - dependencies	29
5.7 Data Cache Restrictions	30
5.8 User Mode support	30
5.8.1 User Mode configuration in the module	30
5.8.2 User Mode configuration in AutosarOS	30
5.9 Multi-core support	32
6 Main API Requirements	33
6.1 Main function calls within BSW scheduler	33
6.2 API Requirements	33
6.3 Calls to Notification Functions, Callbacks, Callouts	33



7 Memory allocation	34
7.1 Sections to be defined in Icu_MemMap.h	34
7.2 Linker command file	35
8 Integration Steps	36
9 External assumptions for driver	37



Chapter 1

Revision History

Revision	Date	Author	Description
1.0	31.03.2023	NXP RTD Team	S32K3 Real-Time Drivers AUTOSAR 4.4 & R21-11 Version 3.0.0

Chapter 2

Introduction

- [Supported Derivatives](#)
- [Overview](#)
- [About This Manual](#)
- [Acronyms and Definitions](#)
- [Reference List](#)

This Integration Manual describes NXP Semiconductor AUTOSAR ICU for S32K3. AUTOSAR ICU driver configuration parameters and deviations from the specification are described in Driver chapter of this document. AUTOSAR ICU driver requirements and APIs are described in the AUTOSAR ICU driver software specification document.

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
LPCMP	Low Power Comparator
CS	Chip Select
CTU	Cross Trigger Unit
DEM	Diagnostic Event Manager
DET	Development Error Tracer
DMA	Direct Memory Access
ECU	Electronic Control Unit
EMIOS	Enhanced Modular IO Subsystem
FIFO	First In First Out
FTM	Flextimer Module
ICU	Input Capture Unit
ISR	Interrupt Service Routine
LSB	Least Significant Bit
MCU	Micro Controller Unit
MIDE	Multi Integrated Development Environment
MSB	Most Significant Bit
N/A	Not Applicable
OS	Operating System
PB Variant	Post Build Variant
PC Variant	Pre Compile Variant
RAM	Random Access Memory
ROM	Read-only Memory
SIUL2	System Integration Unit Lite2
SWS	Software Specification
VLE	Variable Length Encoding
WKPU	Wakeup Unit
XML	Extensible Markup Language

2.5 Reference List

#	Title	Version
1	Specification of ICU Driver	AUTOSAR Release R21-11
2	Specification of Communication Stack Types	AUTOSAR Release R21-11
3	Specification of Compiler Abstraction	AUTOSAR Release R21-11
4	Specification of Platform Types	AUTOSAR Release R21-11

#	Title	Version
5	Specification of Standard Types	AUTOSAR Release R21-11
6	S32K3xx Reference Manual	Rev.6, Draft B, 01/2023
7	S32K39 and S32K37 Reference Manual	Rev. 2 Draft A, 11/2022
8	S32M27x Reference Manual	Rev.2, Draft A, 02/2023
9	S32K3xx Datasheet	Rev. 6, 11/2022
10	S32K396 Datasheet	Rev. 1.1 — 08/2022
11	S32M2xx Datasheet	Rev. 2 RC — 12/2022
11	S32K311 Errata	S32K311_0P98C Mask Set Errata, Rev. 6/March/2023, 3/2023
12	S32K312 Errata	Mask Set Errata for Mask 0P09C, Rev. 25/April/2022
13	S32K342 Errata	Mask Set Errata for Mask 0P97C, Rev. 10, 11/2022
14	S32K3x4 Errata	Mask Set Errata for Mask 0P55A/1P55A, Rev. 14/↔ Oct/2022
15	S32K358 Errata	S32K358_0P14E Mask Set Errata – Rev. 28, 9/2022
16	S32K396 Errata	S32K396_0P40E Mask Set Errata, Rev. DEC2022, 12/2022

Chapter 3

Building the driver

- [Build Options](#)
- [Files required for Compilation](#)
- [Setting up the Plug-ins](#)

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
-mfpv=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 aforementioned -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 initialization 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 initialization in source file system.c under the Platform driver
-DENABLE_FPU	Predefine ENABLE_FPU as a macro, with definition 1. Enables FPU initialization in source file system.c under the Platform driver
-DMCAL_ENABLE_USER_MODE_SUPPORT	Predefine MCAL_ENABLE_USER_MODE_SUPPORT↔ 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
-mfpv=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 <code>#if</code> 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 <code>-std</code> 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 initialization 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 initialization in source file system.c under the Platform driver
-DENABLE_FPU	Predefine ENABLE_FPU as a macro, with definition 1. Enables FPU initialization in source file system.c under the Platform driver
-DMCAL_ENABLE_USER_MODE_SUPPORT	Predefine MCAL_ENABLE_USER_MODE_SUPPORT 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	Bitfields 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 and generate 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 initialization 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 initialization in source file system.c under the Platform driver
-DENABLE_FPU	Predefine ENABLE_FPU as a macro, with definition 1. Enables FPU initialization in source file system.c under the Platform driver
-DMCAL_ENABLE_USER_MODE_SUPPORT	Predefine MCAL_ENABLE_USER_MODE_SUPPORT 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. DWARF 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 the modules 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
-Osz	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 initialization 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 initialization in source file system.c under the Platform driver
-DENABLE_FPU	Predefine ENABLE_FPU as a macro, with definition 1. Enables FPU initialization in source file system.c under the Platform driver
-DMCAL_ENABLE_USER_MODE_SUPPORT	Predefine MCAL_ENABLE_USER_MODE_SUPPORT 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 (if assembler code) and link the INT driver for S32K3XX micro-controllers.

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.

ICU Files

- ../Icu_TS_T40D34M30I0R0/include/Emios_Icu_Ip.h
- ../Icu_TS_T40D34M30I0R0/include/Emios_Icu_Ip_Irq.h
- ../Icu_TS_T40D34M30I0R0/include/Emios_Icu_Ip_Types.h
- ../Icu_TS_T40D34M30I0R0/include/Siul2_Icu_Ip_TrustedFunctions.h
- ../Icu_TS_T40D34M30I0R0/include/Icu.h
- ../Icu_TS_T40D34M30I0R0/include/Icu_EnvCfg.h
- ../Icu_TS_T40D34M30I0R0/include/Icu_Ipw.h
- ../Icu_TS_T40D34M30I0R0/include/Icu_Ipw_Irq.h
- ../Icu_TS_T40D34M30I0R0/include/Icu_Ipw_Types.h
- ../Icu_TS_T40D34M30I0R0/include/Icu_Irq.h
- ../Icu_TS_T40D34M30I0R0/include/Icu_Types.h
- ../Icu_TS_T40D34M30I0R0/include/Siul2_Icu_Ip.h
- ../Icu_TS_T40D34M30I0R0/include/Siul2_Icu_Ip_Irq.h
- ../Icu_TS_T40D34M30I0R0/include/Siul2_Icu_Ip_Types.h
- ../Icu_TS_T40D34M30I0R0/include/Emios_Icu_Ip_TrustedFunctions.h

- ../Icu_TS_T40D34M30I0R0/include/Wkpu_Ip.h
- ../Icu_TS_T40D34M30I0R0/include/Wkpu_Ip_Irq.h
- ../Icu_TS_T40D34M30I0R0/include/Wkpu_Ip_Types.h
- ../Icu_TS_T40D34M30I0R0/include/Cmp_Ip.h
- ../Icu_TS_T40D34M30I0R0/include/Cmp_Ip_Types.h
- ../Icu_TS_T40D34M30I0R0/src/Emios_Icu_Ip.c
- ../Icu_TS_T40D34M30I0R0/src/Emios_Icu_Ip_Irq.c
- ../Icu_TS_T40D34M30I0R0/src/Icu.c
- ../Icu_TS_T40D34M30I0R0/src/Icu_Ipw.c
- ../Icu_TS_T40D34M30I0R0/src/Siul2_Icu_Ip.c
- ../Icu_TS_T40D34M30I0R0/src/Siul2_Icu_Ip_Irq.c
- ../Icu_TS_T40D34M30I0R0/src/Wkpu_Ip.c
- ../Icu_TS_T40D34M30I0R0/src/Wkpu_Ip_Irq.c
- ../Icu_TS_T40D34M30I0R0/src/Cmp_Ip.c

ICU Generated Files

- **<Filename>_Cfg.c (For PC Variant) - For driver compilation, this file should be generated by the user using a configuration tool**
- **<Filename>_PBcfg.c (For PB Variant) - For driver compilation, this file should be generated by the user using a configuration tool**
- **<Filename>_Cfg.h - For driver compilation, this file should be generated by the user using a configuration tool**
- Emios_Icu_Ip_Cfg.h
- Emios_Icu_Ip_Defines.h
- Emios_Icu_Ip_[VariantName]_PBcfg.h
- Icu_Cfg.h
- Icu_Ipw_Cfg.h
- Icu_Ipw_[VariantName]_PBcfg.h
- Icu_[VariantName]_PBcfg.h
- Siul2_Icu_Ip_Cfg.h
- Siul2_Icu_Ip_Defines.h
- Siul2_Icu_Ip_[VariantName]_PBcfg.h
- Wkpu_Ip_Cfg.h

- Wkpu_Ip_Defines.h
- Wkpu_Ip_[VariantName]_PBcfg.h
- Cmp_Ip_Cfg.h
- Cmp_Ip_[VariantName]_PBcfg.h

Note: As a deviation from standard:

- Icu_[VariantName]_PBcfg.c files will contain the definition for all parameters that are variant aware, independent of the configuration class that will be selected (PC, LT, PB).
- Icu_Cfg.c file will contain the definition for all configuration structures containing only variables that are not variant aware, configured and generated only once. This file alone does not contain the whole structure needed by Icu_Init function to configure the driver. Based on the number of variants configured in the EcuC, there can be more than one configuration structure for one module even for PreCompile variant.

Other include folders

- /BaseNXP_TS_T40D34M30I0R0/include
- /BaseNXP_TS_T40D34M30I0R0/header
- /Det_TS_T40D34M30I0R0/include
- /Dem_TS_T40D34M30I0R0/include
- /Rte_TS_T40D34M30I0R0/include
- /Ecum_TS_T40D34M30I0R0/include
- /Mcl_TS_T40D34M30I0R0/include
- /Os_TS_T40D34M30I0R0/include

Dependencies

- **RESOURCE** is required to select processor derivative. Current driver has support for the following derivatives, each one having attached a Resource file:
 - s32k388_mapbga289
 - s32m276_lqfp64
- **ECUM** is required for selecting the reference to the wakeup source for every Icu channel configured as a wakeup source.
- **DEM** is required to manage the runtime erros.
- **BASE** is required to select the OsIf type.
- **DET** is required for signaling the development error detection (parameters out of range, null pointers, etc).
- **RTE** is required for critical sections
- **MCL** is required for support for ICU measurements with DMA and initiate EMIOS IP.
- **ECUC** is required configuring the variant handling in Tresos.
- **OS** is required for support for ICU driver run with multi-core or single-core.

3.3 Setting up the Plug-ins

The Icu driver was designed to be configured by using the EB Tresos (version EB tresos Studio 29.0.0 or later)

All the Autosar RTD drivers for S32K3XX were designed to be configured using Tresos Studio configuration and code generation tool from EB tresos Studio 29.0.0 b220329-0119.

Chapter 4

Function calls to module

- [Function Calls during Startup](#)
- [Function Calls during Shutdown](#)
- [Function Calls during Wakeup](#)

4.1 Function Calls during Startup

This driver does not need OS Support except for ISRs. Hence can be initialized either in STARTUP1 or STARTUP2 phase of EcuM initialization. This depends on the implementation, desired duration for STARTUP1 & target hardware design. The API to be called is Icu_Init(ConfigPtr).

NOTE

For proper driver usage, prior MCU and PORT modules initialization should be done.

4.2 Function Calls during Shutdown

Icu_SetMode(ICU_MODE_SLEEP) API shall be called during GO SLEEP phase of EcuM to configure the hardware for Sleep mode.

4.3 Function Calls during Wakeup

The ICU shall report the wakeup event to EcuM through EcuM_CheckWakeupEvent (event) upon a wakeup event.

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 OS – Dependencies
- ISR Macro
- Other AUTOSAR modules - dependencies
- Data Cache Restrictions
- User Mode support
- Multi-core support

5.1 Exclusive areas to be defined in BSW scheduler

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

ICU_EXCLUSIVE_AREA_00 is used in function `Icu_EnableWakeup` to protect the updates for:

- `Icu_aChannelState`

ICU_EXCLUSIVE_AREA_00 is used in function `Icu_EnableNotification` to protect the updates for:

- `Icu_aChannelState`

ICU_EXCLUSIVE_AREA_00 is used in function `Icu_StartTimestamp` to protect the updates for:

- `Icu_aChannelState`

ICU_EXCLUSIVE_AREA_00 is used in function `Icu_EnableEdgeCount` to protect the updates for:

- `Icu_aChannelState`

ICU_EXCLUSIVE_AREA_00 is used in function `Icu_EnableEdgeDetection` to protect the updates for:

- `Icu_aChannelState`

ICU_EXCLUSIVE_AREA_00 is used in function `Icu_StartSignalMeasurement` to protect the updates for:

- `Icu_aChannelState`

ICU_EXCLUSIVE_AREA_01 is used in function `Icu_DisableWakeup` to protect the updates for:

- `Icu_aChannelState`

ICU_EXCLUSIVE_AREA_01 is used in function `Icu_CheckWakeup` to protect the updates for:

- `Icu_aChannelState`

ICU_EXCLUSIVE_AREA_01 is used in function `Icu_SetActivationCondition` to protect the updates for:

- `Icu_aChannelState`

ICU_EXCLUSIVE_AREA_01 is used in function `Icu_DisableNotification` to protect the updates for:

- `Icu_aChannelState`

ICU_EXCLUSIVE_AREA_01 is used in function `Icu_GetInputState` to protect the updates for:

- `Icu_aChannelState`

ICU_EXCLUSIVE_AREA_01 is used in function `Icu_StartTimestamp` to protect the updates for:

- `Icu_aChannelState`

ICU_EXCLUSIVE_AREA_01 is used in function `Icu_StopTimestamp` to protect the updates for:

- `Icu_aChannelState`

ICU_EXCLUSIVE_AREA_01 is used in function `Icu_ResetEdgeCount` to protect the updates for:

Module requirements

- Icu_aChannelState

ICU_EXCLUSIVE_AREA_01 is used in function Icu_EnableEdgeCount to protect the updates for:

- Icu_aChannelState

ICU_EXCLUSIVE_AREA_01 is used in function Icu_DisableEdgeCount to protect the updates for:

- Icu_aChannelState

ICU_EXCLUSIVE_AREA_01 is used in function Icu_DisableEdgeDetection to protect the updates for:

- Icu_aChannelState

ICU_EXCLUSIVE_AREA_01 is used in function Icu_StartSignalMeasurement to protect the updates for:

- Icu_aChannelState

ICU_EXCLUSIVE_AREA_01 is used in function Icu_StopSignalMeasurement to protect the updates for:

- Icu_aChannelState

ICU_EXCLUSIVE_AREA_01 is used in function Icu_GetTimeElapsed to protect the updates for:

- Icu_aChannelState

ICU_EXCLUSIVE_AREA_02 is used in function Icu_StartTimestamp to protect the updates for:

- Icu_aBuffer
- Icu_aBufferSize
- Icu_aBufferNotify
- Icu_aNotifyCount
- Icu_aBufferIndex

ICU_EXCLUSIVE_AREA_03 is used in function Icu_TimestampDmaProcessing to protect the updates for:

- Icu_aBufferIndex
- Icu_aNotifyCount

ICU_EXCLUSIVE_AREA_08 is used in function Icu_StartSignalMeasurement to protect the updates for:

- Icu_aDmaBuffer

ICU_EXCLUSIVE_AREA_09 is used in function Icu_SignalMeasurementDmaProcessing to protect the updates for:

- Icu_aActivePulseWidth
- Icu_aPeriod

ICU_EXCLUSIVE_AREA_46 is used in function Icu_SetMode, Icu_EnableEdgeDetection, Icu_StartTimestamp, Icu_ResetEdgeCount, Icu_EnableEdgeCount, Icu_StartSignalMeasurement to protect the updates for:

- EMIOS_Sn register
- EMIOS_Cn register

ICU_EXCLUSIVE_AREA_47 is used in function Icu_Init, Icu_SetMode, Icu_DisableEdgeDetection, Icu_StopTimestamp, Icu_ResetEdgeCount, Icu_DisableEdgeCount, Icu_StartSignalMeasurement, Icu_StopSignalMeasurement to protect the updates for:

- EMIOS_Cn register

ICU_EXCLUSIVE_AREA_48 is used in function Icu_EnableEdgeDetection, Icu_DisableEdgeDetection, Icu_StartTimestamp, Icu_StopTimestamp, Icu_ResetEdgeCount, Icu_EnableEdgeCount, Icu_DisableEdgeCount, Icu_StartSignalMeasurement, Icu_StopSignalMeasurement to protect the updates for:

- EMIOS_Cn register

ICU_EXCLUSIVE_AREA_49 is used in function Icu_StopSignalMeasurement to protect the updates for:

- EMIOS_Cn register
- EMIOS_Sn register
- eMios_Icu_Ip_ChState

ICU_EXCLUSIVE_AREA_50 is used in function Icu_SetActivationCondition, Icu_StartSignalMeasurement, ISR(EMIOS0_0_IRQ), ISR(EMIOS0_1_IRQ), ISR(EMIOS0_2_IRQ), ISR(EMIOS0_3_IRQ), ISR(EMIOS0_4_IRQ), ISR(EMIOS0_5_IRQ), ISR(EMIOS1_0_IRQ), ISR(EMIOS1_1_IRQ), ISR(EMIOS1_2_IRQ), ISR(EMIOS1_3_IRQ), ISR(EMIOS1_4_IRQ), ISR(EMIOS1_5_IRQ), ISR(EMIOS2_0_IRQ), ISR(EMIOS2_1_IRQ), ISR(EMIOS2_2_IRQ), ISR(EMIOS2_3_IRQ), ISR(EMIOS2_4_IRQ), ISR(EMIOS2_5_IRQ) to protect the updates for:

- EMIOS_Cn register

ICU_EXCLUSIVE_AREA_51 is used in function Icu_StartTimestamp to protect the updates for:

Module requirements

- EMIOS_Cn register
- eMios_Icu_Ip_ChState

ICU_EXCLUSIVE_AREA_52 is used in function Icu_StartSignalMeasurement to protect the updates for:

- EMIOS_Cn register

ICU_EXCLUSIVE_AREA_53 is used in function Icu_SetClockMode to protect the updates for:

- EMIOS_Cn register
- EMIOS_C2n register

ICU_EXCLUSIVE_AREA_57 is used in function Icu_SetMode, Icu_EnableEdgeDetection to protect the updates for:

- WKPU_IRER register
- WKPU_WRER register
- WKPU_WISR register

ICU_EXCLUSIVE_AREA_58 is used in function Icu_Init, Icu_Deinit, Icu_SetMode, Icu_EnableEdgeDetection to protect the updates for:

- WKPU_IRER register
- WKPU_WRER register
- WKPU_WISR 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.

Please see more detail in the following table.

Exclusive Area ID	ICU_EXCLUSIVE_AREA_00	ICU_EXCLUSIVE_AREA_01	ICU_EXCLUSIVE_AREA_02	ICU_EXCLUSIVE_AREA_03	ICU_EXCLUSIVE_AREA_08	ICU_EXCLUSIVE_AREA_09	ICU_EXCLUSIVE_AREA_46	ICU_EXCLUSIVE_AREA_47	ICU_EXCLUSIVE_AREA_48	ICU_EXCLUSIVE_AREA_50	ICU_EXCLUSIVE_AREA_51	ICU_EXCLUSIVE_AREA_52	ICU_EXCLUSIVE_AREA_49	ICU_EXCLUSIVE_AREA_53	ICU_EXCLUSIVE_AREA_57	ICU_EXCLUSIVE_AREA_58
ICU_EXCLUSIVE_AREA_00	X	X														
ICU_EXCLUSIVE_AREA_01	X	X														
ICU_EXCLUSIVE_AREA_02			X	X												
ICU_EXCLUSIVE_AREA_03			X	X												
ICU_EXCLUSIVE_AREA_08					X											
ICU_EXCLUSIVE_AREA_09						X										
ICU_EXCLUSIVE_AREA_46							X	X	X	X	X	X	X	X		
ICU_EXCLUSIVE_AREA_47							X	X	X	X	X	X	X	X		
ICU_EXCLUSIVE_AREA_48							X	X	X	X	X	X	X	X		
ICU_EXCLUSIVE_AREA_49							X	X	X	X	X	X	X	X		
ICU_EXCLUSIVE_AREA_50							X	X	X	X	X	X	X	X		
ICU_EXCLUSIVE_AREA_51							X	X	X	X	X	X	X	X		
ICU_EXCLUSIVE_AREA_52							X	X	X	X	X	X	X	X		
ICU_EXCLUSIVE_AREA_53							X	X	X	X	X	X	X	X		
ICU_EXCLUSIVE_AREA_57															X	X
ICU_EXCLUSIVE_AREA_58															X	X

Figure 5.1 Exclusive Areas

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).

- None

5.3 Peripheral Hardware Requirements

Refer Table ICU Hardware Channel availability for S32K3XX family in User Manual

5.4 ISR to Configure Within OS – Dependencies

The following ISR's are used by the ICU driver:

The ISR table is presented below. Depending on the derivative used, some of the ISRs may not be available. For complete details please consult the Reference Manual:

System Integration Unit interrupts

System Integration Unit Lite2	Hardware interrupt vector
SIUL2_EXT_IRQ_0_7_ISR	53
SIUL2_EXT_IRQ_8_15_ISR	54
SIUL2_EXT_IRQ_16_23_ISR	55
SIUL2_EXT_IRQ_24_31_ISR	56

Wakeup Unit interrupts

Wakeup Unit interrupts	Hardware interrupt vector
WKPU_EXT_IRQ_SINGLE_ISR	83

Enhanced Modular IO Subsystem

Enhanced Modular IO Subsystem	Hardware interrupt vector
EMIOS0_0_IRQ	61
EMIOS0_1_IRQ	62
EMIOS0_2_IRQ	63
EMIOS0_3_IRQ	64
EMIOS0_4_IRQ	65
EMIOS0_5_IRQ	66
EMIOS1_0_IRQ	69
EMIOS1_1_IRQ	70
EMIOS1_2_IRQ	71
EMIOS1_3_IRQ	72
EMIOS1_4_IRQ	73
EMIOS1_5_IRQ	74
EMIOS2_0_IRQ	77
EMIOS2_1_IRQ	78
EMIOS2_2_IRQ	79
EMIOS2_3_IRQ	80
EMIOS2_4_IRQ	81
EMIOS2_5_IRQ	82

LPCMP	Hardware interrupt vector
LPCMP0	183
LPCMP1	184
LPCMP2	185

= Note: In case of AUTOSAR_OS_NOT_USED, the compiler option "-DUSE_HW_VECTOR_MODE" must be added to the list of compiler options to be used with interrupt controller configured to be in hardware vector mode.

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 be 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

- **Development Error Tracer:** This module is necessary for enabling Development error detection. The API function used is `Det_ReportError()`. The activation / deactivation of Development error detection is configurable using the 'IcuDevErrorDetect' configuration parameter.
- **Diagnostic Event Manager:** This module is necessary for enabling reporting of production relevant error status. Since there are no production relevant error codes in ICU this is not used.
- **ECU State Manager:** This module is used for processing the Wakeup notifications of ICU. Whenever the module is in 'Sleep' mode and a wakeup event occurs on a wakeup capable channel, it is reported to EcuM through the `EcuM_CheckWakeupEvent ()` API. This is configurable using the 'IcuChannelWakeupInfo' configuration parameter.

Module requirements

- **MCL** : This module is used to obtain the common interrupts sources. Optionally, if the DMA API is enabled, this modules provides the DMA channels over which DMA transfer is done.
- **ECUC** : This module is required for configuring the variant handling in Tresos.
- **OS** : This module is required for support for ICU driver run with multi-core or single-core.
- **Configuration dependency to other module:** For generating configuration files of ICU and EcuM also is required as ICU refers to EcuM parameter. EcuM need to be configure first before generating configuration files of ICU.

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 Icu_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

The ICU can be run in user mode if the following steps are performed:

- Enable **IcuEnableUserModeSupport** from the configuration
- Call the following functions as trusted functions:

Function syntax	Available via	Description
void Siul2_Icu_Ip_SetUser↔ AccessAllowed(uint32 Siul2Base↔ Addr)	Siul2_Icu_Ip_TrustedFunctions.h	For setting the user access allowed for ICU registers protected by RE↔ G_PROT
void Emios_Icu_Ip_SetUser↔ AccessAllowed(uint32 Siul2Base↔ Addr)	Emios_Icu_Ip_Trusted↔ Functions.h	For setting the user access allowed for ICU registers protected by RE↔ G_PROT

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 be 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 un-marshalling 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.

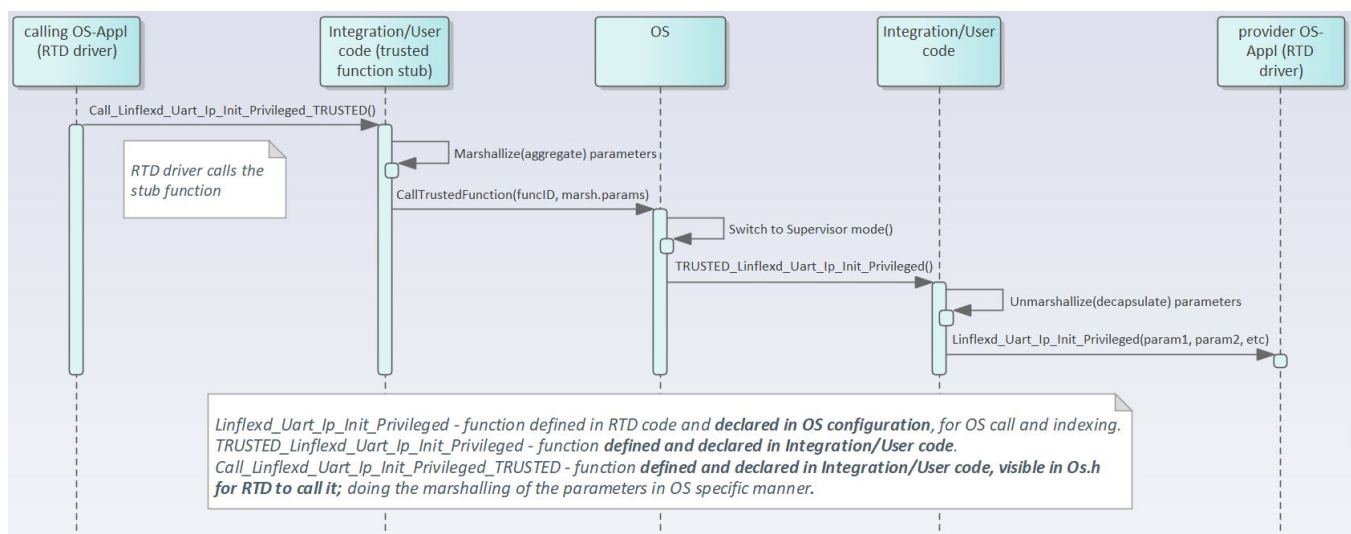


Figure 5.2 Example sequence chart for calling `Linflexd_Uart_Ip_Init_Privileged` as trusted function

5.9 Multi-core support

The Icu driver implements the "Autosar R21-11 MCAL Multi-core Distribution" according to type II, in which the mappable element is set to logical channel. For additional details, please refer to AUTOSAR_EXP_BSW↔DistributionGuide. The Icu driver and the mappable elements can be allocated to zero, one or several ECUC partitions, by means of "IcuEcucPartionRef". If the Icu is mapped to zero ECUC partitions, the Icu behavior reverts to single-core implementation, similar to previous Autosar versions (e.g. R21-11). If the Icu is mapped to one or more ECUC partitions, the Icu enforces the following multi-core assumptions:

- The Icu driver 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 Icu driver 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 Icu internal variables and the EcucCoreIDs are used for indexing those variables. (AR-86601 Zero based and dense IDs for OS-Cores and OSApplications)
- The Icu driver assumes that initialization is performed on each core, Icu_Init() is called separately for each core, using a different configuration structure. (Type II)
- The Icu driver 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 4; Icu_Init function will be called with Icu_Config_EcucPartition_1 configuration structure, on Core 4.
- The Icu driver 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 Icu driver requires that all variables in NonCacheable MemMap sections be allocated accordingly, to avoid data corruption in multi-core context.
- The Icu driver assumes that RTE module implements the EXCLUSIVE AREAS to be core-aware 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 multi-core 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 Icu use case, to be detailed/removed if some modules require such kind of functionality for critical features which cannot be atomically shared among cores)
- The Icu driver assumes that each interrupt is routed by the system only to the core on which is supposed to be serviced.

Module specific limitation:

- In Icu configuration, a submodule cannot be contained by upper two ECUC partitions, so if a channel is mapping with ECUC partition A, then other channels belong to submodule contains that channel will not be allowed to map other ECUC partitions, except ECUC partition A.
- The multi-core mappable element on FlexTimer is the entire HW_UNIT not the channel. This is due to the fact that parallel accesses are done for FlexTimer registers from different cores and those are not synchronized. There is no check implemented in driver for this map action – user configuration must consider this

Chapter 6

Main API Requirements

- Main function calls within BSW scheduler
- API Requirements
- Calls to Notification Functions, Callbacks, Callouts

6.1 Main function calls within BSW scheduler

None.

6.2 API Requirements

None.

6.3 Calls to Notification Functions, Callbacks, Callouts

Call-back Notifications: None.

User Notification: The ICU Driver provides a notification per channel. The ISR 's shall be responsible for resetting the interrupt flags (if needed by hardware) and calling the corresponding notification functions. The notifications can be configured as pointers to user defined functions. If notification is not desired, 'NULL_PTR' shall be configured.

Icu_SignalNotification_<Channel> The syntax of this function is as follows: void NotificationName (void) According to the last call of Icu_EnableNotification, this notification function shall be called if the requested signal edge (rising / falling / both edges) occurs (once per edge).

Icu_TimestampNotification_<Channel> The syntax of this function is as follows: void Timestamp← NotificationName (void) This notification shall be called if the number of requested timestamps (Notification interval > 0) are acquired and if the notification has been enabled by the call of Icu_EnableNotification().

After a call of Icu_DisableNotification() this function must not be called. An extern declaration of these functions is available in Icu_PBcfg.c. The functions shall be implemented by the user.

Chapter 7

Memory allocation

- [Sections to be defined in Icu_MemMap.h](#)
- [Linker command file](#)

7.1 Sections to be defined in Icu_MemMap.h

Section name	Type of section	Description
ICU_START_SEC_CONFIG_DATA_↔ UNSPECIFIED	Configuration Data	Start of Memory Section for Config Data.
ICU_STOP_SEC_CONFIG_DATA_↔ UNSPECIFIED	Configuration Data	End of Memory Section for Config Data.
ICU_START_SEC_CODE	Code	Start of memory Section for Code.
ICU_STOP_SEC_CODE	Code	Stop of memory Section for Code.
ICU_START_SEC_VAR_CLEARED_↔ 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 initialized with values after every reset.
ICU_STOP_SEC_VAR_CLEARED_↔ UNSPECIFIED	Variables	End of above section.
ICU_START_SEC_VAR_CLEARED_8	Variables	Used for variables which have to be aligned to 8 bit. For instance used for variables of size 8 bit or used for composite data types: arrays, structs containing elements of maximum 8 bits. These variables are initialized with
ICU_STOP_SEC_VAR_CLEARED_8	Variables	End of above section.
ICU_START_SEC_VAR_CLEARED_↔ 16	Variables	Used for variables which have to be aligned to 16 bit. For instance used for variables of size 16 bit or used for composite data types: arrays, structs containing elements of maximum 16 bits. These variables are initialized with values after every reset
ICU_STOP_SEC_VAR_CLEARED_16	Variables	End of above section.

Section name	Type of section	Description
ICU_START_SEC_VAR_CLEARED_↔ 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, structs containing elements of maximum 32 bits. These variables are initialized with values after every reset
ICU_STOP_SEC_VAR_CLEARED_32	Variables	End of above section.
ICU_START_SEC_VAR_CLEARED_↔ 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 (BBS).
ICU_STOP_SEC_VAR_CLEARED_U↔ UNSPECIFIED	Variables	End of above section.
ICU_START_SEC_VAR_CLEARED_↔ 32_NO_CACHEABLE	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, structs containing elements of maximum 32 bits and that have to be stored in a noncacheable memory section. These variables are never cleared and never initialized by start-up code..
ICU_STOP_SEC_VAR_CLEARED_↔ 32_NO_CACHEABLE	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 ICU driver into the application.

External Assumption Req ID	External Assumption Text
SWS_Icu_00149	The Icu modules environment shall check the integrity if several calls for the same ICU channel are used during runtime in different tasks or ISRs. Note: The ICU149 is a safety integrity assumption on external environment, which shall be implemented for FTE; For GTE and NTE ICU149 has a role to increase availability because the check will be supported by ICU driver; see also 00150
SWS_Icu_00276	Module: Header File: Imported Type EcuM: EcuM.h: EcuM_Wakeup↔ SourceType Std: Std_Types.h: Std_ReturnType Std_Types.h: Std_↔ VersionInfoType Note: Out of scope sMcal; this is only description of types; in former 4.2.2 this was in same id with actual SWS_Icu_00383
SWS_Icu_00052	If the register can affect several hardware modules and if it is an I/O register it shall be initialized by the PORT driver. Note: Generic assumption not specific to ICU; it is implicitly resolved by PORT requirements
SWS_Icu_00053	If the register can affect several hardware modules and if it is not an I/O register it shall be initialized by the MCU driver. Note: Generic assumption not specific to ICU; it is implicitly resolved by MCU & MCL requirements
SWS_Icu_00128	One-time writable registers that require initialization directly after reset shall be initialized by the start-up code. Note: Generic assumption not specific to ICU; it shall be documented in the SM at the transversal level
SWS_Icu_00129	All other registers shall be initialized by the startup code. Note: Generic assumption not specific to ICU; it shall be documented in the SM at the transversal level
SWS_Icu_00152	The Icu modules environment shall not call Icu_DeInit during a running operation (e. g. timestamp measurement or edge counting) Note: Out of scope sMcal
SWS_Icu_00221	A re-initialization of the ICU module by executing the Icu_Init() function requires a de-initialization before by executing the Icu_DeInit() function. Note: Out of scope sMcal
SWS_Icu_00133	This service can be called during running operations. If so, an ongoing operation that generates interrupts on a wakeup capable channel like e.g. time stamping or edge counting might lead to the ICU module not being able to properly enter sleep mode. This is then a system or ECU configuration issue not a problem of this specification. Note: Out of scope sMcal

External Assumption Req ID	External Assumption Text
SWS_Icu_00361	The ICU modules environment shall only use the re-entrant capability of the function Icu_CheckWakeup if the ICU modules environment takes care that there is no simultaneous usage of the same channel. Note: The wakeup functionality is not considered to take part in a safety functionality
SWS_Icu_00348	Re-entrancy of operation Icu_SignalNotification_<Channel> is not relevant for this module (In general it is in this case not re-entrant). Note: Out of scope sMcal - from text seems nothing to do. Similar with SWS_Icu_↔00214 description.
SWS_Icu_00349	Re-entrancy of the Icu_TimestampNotification_<Channel> is not relevant for this module (in general it is in this case not re-entrant). Note: Out of scope sMcal. Similar with SWS_Icu_00214 description.
EA_RTD_00037	The external application shall invoke Icu_EnableWakeup() and Icu_↔DisableWakeup() only when ICU driver is in ICU_MODE_NORMAL mode. Note: It is assumed that the wakeup channel configuration is established before entering in sleep mode.
EA_RTD_00038	The ICU module's environment shall not call any function of the ICU module before having called Icu_Init.
EA_RTD_00039	The application shall call the function that starts a signal measurement (Icu_StartSignalMeasurement()) or a timestamp measurement(Icu_Start↔Timestamp()) only on channels that are not running. If this rule cannot be fulfilled, the application shall ensure that ICU HW channels interrupt routine will not be pre-empted by tasks invoking these functions. Note↔: Rationale: If channel ICU ISR is preempted by a function that starts a signal measurement or timestamp, the first set of values reported may be incorrect.
EA_RTD_00040	For the situations when notification disablement is requested on running channel, the application shall ensure that ICU HW channels interrupt routine will not be pre-empted by Icu_DisableNotification() calls. Note↔: Rationale: If channel ISR is preempted by the task which disables the notifications, an unexpected notification report might still occur, after the notifications disablement.
EA_RTD_00041	The application shall stop all running channels before de-initializing the I↔CU driver through Icu_DeInit(). Otherwise, it shall ensure that ICU HW channels interrupt routine will not be pre-empted by the task calling Icu_↔DeInit(). Note: Rationale: If a HW channel interrupt is preempted by Icu_Deinit() function erroneous memory access may occur.
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 Icu is allocated shall be exclusively mapped to a core. Note↔: Internally, the Icu 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.

External Assumption Req ID	External Assumption Text
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 Icu_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 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>TRUSTED(parameter1,parameter2,...) in Integration/User code. They need to be visible in Os.h for the driver to call them. They will call RTD <Function_Name>() as trusted functions in OS specific manner.

How to Reach Us:

Home Page:

nxp.com

Web Support:

nxp.com/support

Information in this document is provided solely to enable system and software implementers to use NXP products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits based on the information in this document. NXP reserves the right to make changes without further notice to any products herein.

NXP makes no warranty, representation, or guarantee regarding the suitability of its products for any particular purpose, nor does NXP assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. "Typical" parameters that may be provided in NXP data sheets and/or specifications can and do vary in different applications, and actual performance may vary over time. All operating parameters, including "typicals," must be validated for each customer application by customer's technical experts. NXP does not convey any license under its patent rights nor the rights of others. NXP sells products pursuant to standard terms and conditions of sale, which can be found at the following address: nxp.com/SalesTermsandConditions.

NXP, the NXP logo, NXP SECURE CONNECTIONS FOR A SMARTER WORLD, COOLFLUX, EMBRACE, GREENCHIP, HITAG, I2C BUS, ICODE, JCOP, LIFE VIBES, MIFARE, MIFARE CLASSIC, MIFARE DESFire, MIFARE PLUS, MIFARE FLEX, MANTIS, MIFARE ULTRALIGHT, MIFARE4MOBILE, MIGLO, NTAG, ROADLINK, SMARTLX, SMARTMX, STARPLUG, TOPFET, TRENCHMOS, UCODE, Freescale, the Freescale logo, Altivec, C-5, CodeTEST, CodeWarrior, ColdFire, ColdFire+, C-Ware, the Energy Efficient Solutions logo, Kinetis, Layerscape, MagniV, mobileGT, PEG, PowerQUICC, Processor Expert, QorIQ, QorIQ Qonverge, Ready Play, SafeAssure, the SafeAssure logo, StarCore, Symphony, VortiQa, Vybrid, Airfast, BeeKit, BeeStack, CoreNet, Flexis, MXC, Platform in a Package, QUICC Engine, SMARTMOS, Tower, TurboLink, and UMEMS are trademarks of NXP B.V. All other product or service names are the property of their respective owners. ARM, AMBA, ARM Powered, Artisan, Cortex, Jazelle, Keil, SecurCore, Thumb, TrustZone, and Vision are registered trademarks of ARM Limited (or its subsidiaries) in the EU and/or elsewhere. ARM7, ARM9, ARM11, big.LITTLE, CoreLink, CoreSight, DesignStart, Mali, mbed, NEON, POP, Sensinode, Socrates, ULINK and Versatile are trademarks of ARM Limited (or its subsidiaries) in the EU and/or elsewhere. All rights reserved. Oracle and Java are registered trademarks of Oracle and/or its affiliates. The Power Architecture and Power.org word marks and the Power and Power.org logos and related marks are trademarks and service marks licensed by Power.org.

© 2023 NXP B.V.

