Integration Manual

for S32K1 ICU Driver

Document Number: IM2ICU4.4 Rev0000R1.0.1 Rev. 1.0

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

This Integration Manual describes NXP Semiconductor AUTOSAR ICU for S32K1. 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:

- 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

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 Signifigant 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 4.↩ 4.0
2	S32K1xx Series Reference Manual	Rev. 14, 09/2021
3	S32K116_0N96V	Rev. 22/OCT/2021

#	Title	Version
4	S32K118_0N97V	Rev. 22/OCT/2021
5	S32K142_0N33V	Rev. 22/OCT/2021
6	S32K144_0N57U	Rev. 22/OCT/2021
7	S32K144W_0P64A	Rev. 22/OCT/2021
8	S32K146_0N73V	Rev. 22/OCT/2021
9	S32K148_0N20V	Rev. 22/OCT/2021
10	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.

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

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.

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 (if assembler code) and link the **ICU** driver for **S32** 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.

ICU Files

- ../Icu_TS_T40D2M10I1R0/include/Cmp_Ip.h
- ../Icu_TS_T40D2M10I1R0/include/Cmp_Ip_Types.h
- ../Icu_TS_T40D2M10I1R0/include/Ftm_Icu_Ip.h
- ../Icu_TS_T40D2M10I1R0/include/Ftm_Icu_Ip_Irq.h
- ../Icu_TS_T40D2M10I1R0/include/Ftm_Icu_Ip_Types.h
- ../Icu_TS_T40D2M10I1R0/include/Icu.h
- ../Icu_TS_T40D2M10I1R0/include/Icu_EnvCfg.h
- ../Icu_TS_T40D2M10I1R0/include/Icu_Ipw.h
- ../Icu_TS_T40D2M10I1R0/include/Icu_Ipw_Irq.h
- ../Icu_TS_T40D2M10I1R0/include/Icu_Ipw_Types.h
- ../Icu_TS_T40D2M10I1R0/include/Icu_Irq.h
- ../Icu_TS_T40D2M10I1R0/include/Icu_Types.h
- ../Icu_TS_T40D2M10I1R0/include/Lpit_Icu_Ip.h
- ../Icu_TS_T40D2M10I1R0/include/Lpit_Icu_Ip_Irq.h
- ../Icu_TS_T40D2M10I1R0/include/Lpit_Icu_Ip_Types.h
- ../Icu_TS_T40D2M10I1R0/include/Port_Ci_Icu_Ip.h
- ../Icu_TS_T40D2M10I1R0/include/Port_Ci_Icu_Ip_Irq.h
- ../Icu_TS_T40D2M10I1R0/include/Port_Ci_Icu_Ip_Types.h
- ../Icu_TS_T40D2M10I1R0/include/Lptmr_Icu_Ip.h
- ../Icu_TS_T40D2M10I1R0/include/Lptmr_Icu_Ip_Irq.h
- ../Icu_TS_T40D2M10I1R0/include/Lptmr_Icu_Ip_Types.h
- ../Icu_TS_T40D2M10I1R0/src/Cmp_Ip.c
- ../Icu_TS_T40D2M10I1R0/src/Ftm_Icu_Ip.c
- ../Icu_TS_T40D2M10I1R0/src/Ftm_Icu_Ip_Irq.c
- ../Icu_TS_T40D2M10I1R0/src/Icu.c
- ../Icu_TS_T40D2M10I1R0/src/Icu_Ipw.c
- ../Icu_TS_T40D2M10I1R0/src/Lpit_Icu_Ip.c

- ../Icu_TS_T40D2M10I1R0/src/Lpit_Icu_Ip_Irq.c
- ../Icu_TS_T40D2M10I1R0/src/Port_Ci_Icu_Ip.c
- ../Icu_TS_T40D2M10I1R0/src/Port_Ci_Icu_Ip_Irq.c
- ../Icu_TS_T40D2M10I1R0/src/Lptmr_Icu_Ip.c
- ../Icu_TS_T40D2M10I1R0/src/Lptmr_Icu_Ip_Irq.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**

- Cmp_Ip_Cfg.h
- Cmp_Ip_Defines.h
- Cmp_Ip_[VariantName]_PBcfg.h
- Ftm_Icu_Ip_Cfg.h
- Ftm_Icu_Ip_Defines.h
- Ftm_Icu_Ip_[VariantName]_PBcfg.h
- Icu_Cfg.h
- Icu_Ipw_Cfg.h
- Icu_Ipw_[VariantName]_PBcfg.h
- Icu_[VariantName]_PBcfg.h
- Lpit_Icu_Ip_Cfg.h
- Lpit_Icu_Ip_Defines.h
- Lpit_Icu_Ip_[VariantName]_PBcfg.h
- Lptmr_Icu_Ip_Cfg.h
- Lptmr_Icu_Ip_Defines.h
- Lptmr_Icu_Ip_[VariantName]_PBcfg.h
- Port_Ci_Icu_Ip_Cfg.h
- Port_Ci_Icu_Ip_Defines.h
- Port_Ci_Icu_Ip_[VariantName]_PBcfg.h

Other include folders

Building the driver

- /Base_TS_T40D2M10I1R0/include
- /Base_TS_T40D2M10I1R0/header
- /Det_TS_T40D2M10I1R0/include
- /Dem_TS_T40D2M10I1R0/include
- /Rte_TS_T40D2M10I1R0/include
- /Ecum_TS_T40D2M10I1R0/include
- /Mcl_TS_T40D2M10I1R0/include
- /Os_TS_T40D2M10I1R0/include

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.

Steps to generate the configuration:

1. Copy the module folders Icu_TS_T40D2M10I1R0 , Base_TS_T40D2M10I1R0 , Resource_TS_T40D2M10↩ I1R0 , Det_TS_T40D2M10I1R0, Dem_TS_T40D2M10I1R0, Rte_TS_T40D2M10I1R0, EcuC_TS_T40↩ D11M18I0R0, Os_TS_T40D11M18I0R0, Mcl_TS_T40D2M10I1R0 into the Tresos plugins folder.
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.

Dependencies

- **RESOURCE** is required to select processor derivative. Current driver has support for the following derivatives, each one having attached a Resource file: 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, 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 .
- **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 intiate 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 plugins

The *driver* driver was designed to be configured by using the EB Tresos Studio (version EB tresos Studio *27.1.0* or later.)

Chapter 4

Function calls to module

4.1 Function Calls during Startup

This driver does not need OS Support except for ISRs. Hence can be initialized either in STARTUP1 or START↩ UP2 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 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, 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:

- 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 ISR(FTM_0_ISR), ISR(FTM_1_ISR), ISR(FTM_2_I↩SR), ISR(FTM_3_ISR), ISR(FTM_4_ISR), ISR(FTM_5_ISR) to protect the updates for:

- Icu_aBufferIndex
- Icu_aBufferSize
- Icu_aNotifyCount

ICU_EXCLUSIVE_AREA_04 is used in function ISR(FTM_0_ISR), ISR(FTM_1_ISR), ISR(FTM_2_ISR), ISR(FTM_3_ISR), ISR(FTM_4_ISR), ISR(FTM_5_ISR) to protect the updates for:

- chBufferSize
- chBufferNotify
- Icu_aBufferIndex

ICU_EXCLUSIVE_AREA_05 is used in function Icu_GetTimeElapsed to protect the updates for:

- Icu_aPeriod
- Icu_aActivePulseWidth

ICU_EXCLUSIVE_AREA_06 is used in function Icu_GetDutyCycleValues to protect the updates for:

- Icu_aActivePulseWidth
- Icu_aPeriod

ICU_EXCLUSIVE_AREA_07 is used in function ISR(FTM_0_ISR), ISR(FTM_1_ISR), ISR(FTM_2_ISR), ISR(FTM_3_ISR), ISR(FTM_4_ISR), ISR(FTM_5_ISR) to protect the updates for:

- Icu_aActivePulseWidth
- Icu_aPeriod
- Icu_ReportWakeupAndOverflow

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

- Icu_aActivePulseWidth
- Icu_aPeriod

ICU_EXCLUSIVE_AREA_09 is used in function ISR(FTM_0_ISR), ISR(FTM_1_ISR), ISR(FTM_2_ISR), ISR(FTM_3_ISR), ISR(FTM_4_ISR), ISR(FTM_5_ISR) to protect the protect the updates for:

- Icu_aActivePulseWidth
- Icu_aPeriod

ICU_EXCLUSIVE_AREA_15 is used in function Icu_EnableEdgeDetection, Icu_StartTimestamp, Icu_EnableEdgeCount, Icu_StartSignalMeasurement to protect the protect the updates for:

- FTM_CSC_ADDR32 Register
- FTM_SC_ADDR32 Register

ICU_EXCLUSIVE_AREA_16 is used in function Icu_StopTimestamp, Icu_DisableEdgeCount, Icu_Stop↩ SignalMeasurement, to protect the protect the updates for:

- FTM_CSC_ADDR32 Register

ICU_EXCLUSIVE_AREA_17 is used in function Icu_Init, Icu_DeInit to protect the protect the updates for:

- FTM_SC_ADDR32 Register
- FTM_MODE_ADDR32 Register
- FTM_MOD_ADDR32 Register
- FTM_FMS_ADDR32 Register
- FTM_CNTIN_ADDR32 Register

ICU_EXCLUSIVE_AREA_18 is used in function Icu_DeInit to protect the protect the updates for:

- Ftm_Icu_Ip_ChState

ICU_EXCLUSIVE_AREA_19 is used in function Icu_SetActivationCondition to protect the protect the updates for:

- FTM_CSC_ADDR32 Register

ICU_EXCLUSIVE_AREA_20 is used in function Icu_GetTimestampIndex, Icu_GetEdgeNumbers, Icu_↩ GetTimeElapsed, Icu_GetDutyCycleValues, ISR(FTM_0_ISR), ISR(FTM_1_ISR), ISR(FTM_2_ISR), ISR(F↩ TM_3_ISR), ISR(FTM_4_ISR), ISR(FTM_5_ISR) to protect the protect the updates for:

- FTM_SC_ADDR32 Register

ICU_EXCLUSIVE_AREA_21 is used in function Icu_SetMode to protect the protect the updates for:

- FTM_CSC_ADDR32 Register

ICU_EXCLUSIVE_AREA_22 is used in function Icu_SetMode to protect the protect the updates for:

- FTM_CSC_ADDR32 Register

ICU_EXCLUSIVE_AREA_23 is used in function Icu_EnableEdgeDetection to protect the protect the updates for:

- FTM_CSC_ADDR32 Register
- FTM_SC_ADDR32 Register

ICU_EXCLUSIVE_AREA_24 is used in function Icu_DisableEdgeDetection to protect the protect the updates for:

- FTM_CSC_ADDR32 Register

ICU_EXCLUSIVE_AREA_25 is used in function Icu_StartTimestamp to protect the protect the updates for:

- FTM_CSC_ADDR32 Register
- FTM_SC_ADDR32 Register
- Ftm_Icu_Ip_ChState
- Ftm_Icu_Ip_aBufferPtr

ICU_EXCLUSIVE_AREA_26 is used in function Icu_EnableEdgeCount to protect the protect the updates for:

- FTM_CSC_ADDR32 Register
- FTM_SC_ADDR32 Register
- Ftm_Icu_Ip_ChState

ICU_EXCLUSIVE_AREA_27 is used in function Icu_DisableEdgeCount to protect the protect the updates for:

- FTM_CSC_ADDR32 Register
- FTM_SC_ADDR32 Register

ICU_EXCLUSIVE_AREA_28 is used in function Icu_StartSignalMeasurement to protect the protect the updates for:

- FTM_SC_ADDR32 Register
- FTM_COMBINE_ADDR32 Register
- Ftm_Icu_Ip_ChState

ICU_EXCLUSIVE_AREA_29 is used in function Icu_GetInputState to protect the protect the updates for:

- FTM_CSC_ADDR32 Register

ICU_EXCLUSIVE_AREA_30 is used in function Icu_StopSignalMeasurement to protect the protect the updates for:

- FTM_CSC_ADDR32 Register
- FTM_SC_ADDR32 Register

ICU_EXCLUSIVE_AREA_33 is used in function ISR(FTM_0_ISR), ISR(FTM_1_ISR), ISR(FTM_2_ISR), ISR(FTM_3_ISR), ISR(FTM_4_ISR), ISR(FTM_5_ISR) to protect the protect the updates for:

- FTM_CV_ADDR32 Register
- Ftm_Icu_Ip_ChState

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 Matrix																														
Exclusive Area ID	ICU_EXCLUSIVE_AREA_00	ICU_EXCLUSIVE_AREA_01	ICU_EXCLUSIVE_AREA_02	ICU_EXCLUSIVE_AREA_03	ICU_EXCLUSIVE_AREA_04	ICU_EXCLUSIVE_AREA_05	ICU_EXCLUSIVE_AREA_06	ICU_EXCLUSIVE_AREA_07	ICU_EXCLUSIVE_AREA_08	ICU_EXCLUSIVE_AREA_09	ICU_EXCLUSIVE_AREA_15	ICU_EXCLUSIVE_AREA_16	ICU_EXCLUSIVE_AREA_17	ICU_EXCLUSIVE_AREA_18	ICU_EXCLUSIVE_AREA_19	ICU_EXCLUSIVE_AREA_20	ICU_EXCLUSIVE_AREA_21	ICU_EXCLUSIVE_AREA_22	ICU_EXCLUSIVE_AREA_23	ICU_EXCLUSIVE_AREA_24	ICU_EXCLUSIVE_AREA_25	ICU_EXCLUSIVE_AREA_26	ICU_EXCLUSIVE_AREA_27	ICU_EXCLUSIVE_AREA_28	ICU_EXCLUSIVE_AREA_29	ICU_EXCLUSIVE_AREA_30	ICU_EXCLUSIVE_AREA_31	ICU_EXCLUSIVE_AREA_32	ICU_EXCLUSIVE_AREA_33
ICU_EXCLUSIVE_AREA_00	X	X	X	X	X			X		X																			
ICU_EXCLUSIVE_AREA_01	X	X	X	X	X			X		X																			
ICU_EXCLUSIVE_AREA_02	X	X	X	X	X			X		X																			
ICU_EXCLUSIVE_AREA_03	X	X	X	X	X			X		X																			
ICU_EXCLUSIVE_AREA_04	X	X	X	X	X			X		X																			
ICU_EXCLUSIVE_AREA_05						X	X	X	X	X																			
ICU_EXCLUSIVE_AREA_06						X	X	X	X	X																			
ICU_EXCLUSIVE_AREA_07	X	X	X	X	X	X	X	X	X	X																			
ICU_EXCLUSIVE_AREA_08						X	X	X	X	X																			
ICU_EXCLUSIVE_AREA_09	X	X	X	X	X	X	X	X	X	X																			
ICU_EXCLUSIVE_AREA_15											X	X	X		X	X	X	X	X	X	X	X	X	X	X	X	X	X	
ICU_EXCLUSIVE_AREA_16											X	X			X		X	X	X	X	X	X	X			X	X	X	X
ICU_EXCLUSIVE_AREA_17											X		X			X			X		X	X	X	X		X	X	X	
ICU_EXCLUSIVE_AREA_18														X							X	X		X				X	X
ICU_EXCLUSIVE_AREA_19											X	X			X		X	X	X	X	X	X	X		X	X	X		
ICU_EXCLUSIVE_AREA_20											X		X			X			X		X	X	X	X		X	X	X	
ICU_EXCLUSIVE_AREA_21											X	X			X		X	X	X	X	X	X	X		X	X	X		
ICU_EXCLUSIVE_AREA_22											X	X			X		X	X	X	X	X	X	X		X	X	X		
ICU_EXCLUSIVE_AREA_23											X	X	X		X	X	X	X	X	X	X	X	X	X	X	X	X	X	
ICU_EXCLUSIVE_AREA_24											X	X			X		X	X	X	X	X	X	X		X	X	X		
ICU_EXCLUSIVE_AREA_25											X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	
ICU_EXCLUSIVE_AREA_26											X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	
ICU_EXCLUSIVE_AREA_27											X	X	X		X	X	X	X	X	X	X	X	X	X	X	X	X	X	
ICU_EXCLUSIVE_AREA_28											X		X	X		X			X		X	X	X	X		X	X	X	X
ICU_EXCLUSIVE_AREA_29											X				X		X	X	X	X	X	X	X		X	X	X		
ICU_EXCLUSIVE_AREA_30											X	X	X		X	X	X	X	X	X	X	X	X	X	X	X	X	X	
ICU_EXCLUSIVE_AREA_31											X	X	X		X	X	X	X	X	X	X	X	X	X	X	X	X	X	
ICU_EXCLUSIVE_AREA_32											X	X	X	X		X			X		X	X	X	X		X	X	X	X
ICU_EXCLUSIVE_AREA_33												X		X											X				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

ICU driver has 8 modules FlexTimer (from FlexTimer 0 to FlexTimer 7), each module can support 8 channels (FlexTimer channels 0-7). External interrupt channels 0-31 PORT_CI 0 - PORT_CI 4 and 1 channel LPTMR0 are available for ICU driver. ICU driver has 1 module LPIT with 4 channels, LPIT can be used as internal interrupt with TRGMUX configuration

Note

Note:

- Port A, B, C, D, E were renamed PORT_CI_0, PORT_CI_1, PORT_CI_2, PORT_CI_3, PORT_↩ CI_4 in the driver.

Refer Table ICU Hardware Channel availability for S32K1 family in User Manual< >

5.4 ISR to configure within AutosarOS - dependencies

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:

Flextrimer interrupts

FlexTimer Module Interrupts	HW INT Vector	Observations
FTM_0_CH_0_CH_1_ISR	115	None
FTM_0_CH_2_CH_3_ISR	116	None
FTM_0_CH_4_CH_5_ISR	117	None
FTM_0_CH_6_CH_7_ISR	118	None
FTM_0_OVF_ISR	120	None
FTM_1_CH_0_CH_1_ISR	121	None
FTM_1_CH_2_CH_3_ISR	122	None
FTM_1_CH_4_CH_5_ISR	123	None
FTM_1_CH_6_CH_7_ISR	124	None
FTM_1_OVF_ISR	126	None
FTM_2_CH_0_CH_1_ISR	127	None
FTM_2_CH_2_CH_3_ISR	128	None
FTM_2_CH_4_CH_5_ISR	129	None
FTM_2_CH_6_CH_7_ISR	130	None
FTM_2_OVF_ISR	132	None
FTM_3_CH_0_CH_1_ISR	133	None
FTM_3_CH_2_CH_3_ISR	134	None
FTM_3_CH_4_CH_5_ISR	135	None
FTM_3_CH_6_CH_7_ISR	136	None
FTM_3_OVF_ISR	138	None
FTM_4_CH_0_CH_1_ISR	139	None
FTM_4_CH_2_CH_3_ISR	140	None
FTM_4_CH_4_CH_5_ISR	141	None

S32K1 ICU Driver

FlexTimer Module Interrupts	HW INT Vector	Observations
FTM_4_CH_6_CH_7_ISR	142	None
FTM_4_OVF_ISR	144	None
FTM_5_CH_0_CH_1_ISR	145	None
FTM_5_CH_2_CH_3_ISR	146	None
FTM_5_CH_4_CH_5_ISR	147	None
FTM_5_CH_6_CH_7_ISR	148	None
FTM_5_OVF_ISR	150	None
FTM_6_CH_0_CH_1_ISR	151	None
FTM_6_CH_2_CH_3_ISR	152	None
FTM_6_CH_4_CH_5_ISR	153	None
FTM_6_CH_6_CH_7_ISR	154	None
FTM_6_OVF_ISR	156	None
FTM_7_CH_0_CH_1_ISR	157	None
FTM_7_CH_2_CH_3_ISR	158	None
FTM_7_CH_4_CH_5_ISR	159	None
FTM_7_CH_6_CH_7_ISR	160	None
FTM_7_OVF_ISR	162	None

External PORT_CI Interrupts

External PORT_CI Interrupts	HW INT Vector	Observations
ICU_PORT_CI_A_EXT_IRQ_ISR	75	None
ICU_PORT_CI_B_EXT_IRQ_ISR	76	None
ICU_PORT_CI_C_EXT_IRQ_ISR	77	None
ICU_PORT_CI_D_EXT_IRQ_ISR	78	None
ICU_PORT_CI_E_EXT_IRQ_ISR	79	None

External LPTMR (Low power timer) Interrupts

LPTMR Module Interrupts	HW INT Vector	Observations
LPTMR_0_CH_0_ISR	74	None

LPIT Interrupts

LPIT Interrupts	HW INT Vector	Observations
LPIT_0_CH_0_ISR	64	None
LPIT_0_CH_1_ISR	65	None
LPIT_0_CH_2_ISR	66	None
LPIT_0_CH_3_ISR	67	None

5.5 ISR Macro

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

5.5.1 Without an Operating System

The macro *USING_OS_AUTOSAROS* must not be defined.

5.5.1.1 Using Software Vector Mode

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

```
#define ISR(IsrName) void IsrName(void)
```

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

5.5.1.2 Using Hardware Vector Mode

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

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

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

5.5.2 With an Operating System

Please refer to your OS documentation for description of the ISR macro.

5.6 Other AUTOSAR modules - dependencies

- **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.

- **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

5.8.1 User Mode configuration in the module

There is no restriction when running from user mode for all ICU IPs. Therefore no further actions are needed in ICU driver.

5.8.2 User Mode configuration in AutosarOS

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

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

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

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

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

- Ensure `MCAL_ENABLE_USER_MODE_SUPPORT` macro is defined in the build system or somewhere global.
- Define and declare all functions that need to call as trusted functions follow the naming convention above in Integration/User code. They need to visible in `Os.h` for the driver to call them. They will do the marshalling of the parameters and call `CallTrustedFunction()` in OS specific manner.
- `CallTrustedFunction()` will switch to privileged mode and call `TRUSTED_<Function_Name>()`.
- `TRUSTED_<Function_Name>()` function is also defined and declared in Integration/User code. It will unmarshalling of the parameters to call `<Function_Name>`() of driver. The `<Function_Name>`() functions are already defined in driver and declared in `<IpName>_Ip_TrustedFunctions.h`. This header should be included in OS for OS call and indexing these functions.

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

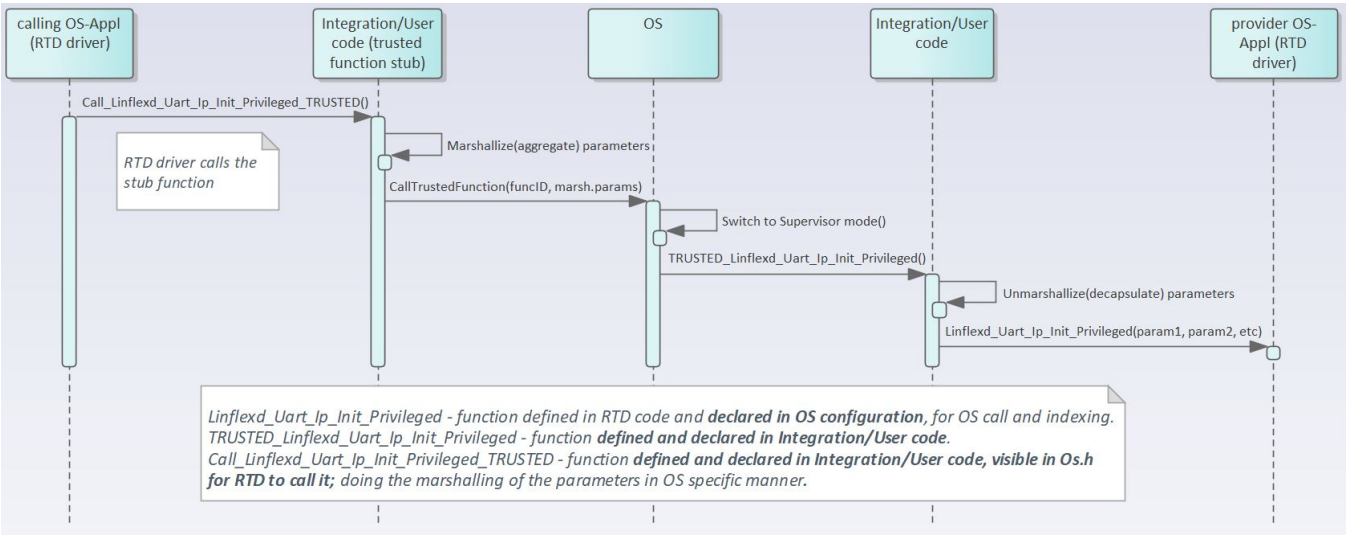

Figure 5.2 Example sequence chart for calling `Linflexd_Uart_Ip_Init_Privileged` as trusted function

5.9 Multicore support

All S32K1 derivatives will be treated as a single-core device, so this platform does not support multicore feature!

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

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_U↩ NSPECIFIED	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_U↩ NSPECIFIED	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↩ NSPECIFIED	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 module's 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 availablity because the check will be supported by ICU driver; see also 00150
SWS_Icu_00276	Module - Header File - Imported Type - EcuM_flex - EcuM.h - EcuM↩ _WakeupSourceType - Std_Types - StandardTypes.h - Std_ReturnType - StandardTypes.h - Std_VersionInfoType - Note: Out of scope sMcal; this is only desctiprion 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 module's 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. SWS_Icu_00022 apply to the function Icu_SetMode. Note: Out of scope sMcal

External assumptions for driver

External Assumption Req ID	External Assumption Text
SWS_Icu_00361	The ICU module's environment shall only use the re-entrant capability of the function Icu_CheckWakeup if the ICU module's 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. Out of scope of sMcal
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 channel's 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 channel's 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 channel's 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_00106	Standalone IP configuration and HL configuration of the same driver shall be done in the same project

External Assumption Req ID	External Assumption Text
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

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.

© 2022 NXP B.V.