Integration Manual

for S32K14X ICU Driver

Document Number: UM2ICUASR4.3 Rev0001R1.0.1
Rev. 1.0

Contents

Chapter 1
Revision History

Table 1-1. Revision History

Revision	Date	Author	Description
1.0	21/06/2019	NXP MCAL Team	Updated version for ASR 4.3.1S32K14XR1.0.1

Chapter 2 Introduction

This integration manual describes the integration requirements for ICU Driver for S32K14X microcontrollers.

2.1 Supported Derivatives

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

Table 2-1. S32K14X Derivatives

NXP Semiconductors	s32k148_lqfp144, s32k148_lqfp176, s32k148_mapbga100, s32k146_lqfp144, s32k146_lqfp100, s32k146_lqfp64, s32k146_mapbga100, s32k144_lqfp100, s32k144_lqfp64, s32k144_mapbga100, s32k142_lqfp100, s32k142_lqfp64, s32k118_lqfp48, s32k118_lqfp64, s32k142_lqfp48, s32k144_lqfp48, s32k148_lqfp100

All of the above microcontroller devices are collectively named as S32K14X .

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 type: Bold is used for important terms, notes and warnings.

Italic font: Italic typeface is 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.

2.4 Acronyms and Definitions

Table 2-2. Acronyms and Definitions

Term	Definition
BSW	Basic Software
DEM	Diagnostic Event Manager
DET	Development Error Tracer
ECU	Electronic Control Unit
ICU	Input Capture Unit
ISR	interrupt Service Routine
OS	Operating System
RAM	Random Access Memory
ROM	Read-only Memory
MCU	Microcontroller Unit
GUI	Graphical User Interface
EcuM	ECU state Manager
FTM	FlexTimer Module

Table continues on the next page...

Table 2-2. Acronyms and Definitions (continued)

Term	Definition
PORT_CI	Port Control and Interrupts
LPTMR	Low-Power Timer
LPIT	Low Power Interrupt Timer
API	Application Programming Interface
PB Variant	Post Build Variant
PC Variant	Pre Compile Variant

2.5 Reference List

Table 2-3. Reference List

#	Title	Version
1	Specification of ICU Driver	AUTOSAR Release 4.3.1
2	S32K14X Reference Manual	Reference Manual, Rev. 9, 9/2018
3	S32K142 Mask Set Errata for Mask 0N33V (0N33V)	30/11/2017
4	S32K144 Mask Set Errata for Mask 0N57U (0N57U)	30/11/2017
5	S32K146 Mask Set Errata for Mask 0N73V (0N73V)	30/11/2017
6	S32K148 Mask Set Errata for Mask 0N20V (0N20V)	25/10/2018
7	S32K118 Mask Set Errata for Mask 0N97V (0N97V)	07/01/2019

Chapter 3
Building the Driver

This section describes the source files and various compilers, linker options used for building the Autosar ICU driver for NXP SemiconductorsS32K14X . It also explains the EB Tresos Studio plugin setup procedure.

3.1 Build Options

The ICU driver files are compiled using

- Green Hills Multi 7.1.4 / Compiler 2017.1.4
- (Linaro GCC 6.3-2017.06~dev) 6.3.1 20170509 (Wed Jan 24 16:21:45 CST 2018 build.sh rev=g27a1317 s=L631 Earmv7 -V release_g27a1317_build_Fed_Earmv7)
- IAR: V8.11.2

The compiler, linker flags used for building the driver are explained below:

Note

The TS_T40D2M10I1R0 plugin name is composed as follow:

TS_T = Target_Id

D = Derivative_Id

M = SW_Version_Major

I = SW_Version_Minor

R = Revision

(i.e. Target_Id = 40 identifies CORTEXM architecture and Derivative_Id = 2 identifies the S32K14X)

3.1.1 GCC Compiler/Linker/Assembler Options

Table 3-1. Compiler Options

Option	Description
-c	Produces an object file (called input-file.o) for each source file.
-Os	Use optimization for size.
-ggdb3	Produce debugging information for use by GDB. Level 3 includes extra information, such as all the macro definitions present in the program.
-mcpu=cortex-m4	Selects target processor: Arm Cortex M4
-mcpu=cortex-m0plus	Selects target processor: Arm Cortex M0+
-mthumb	Selects generating code that executes in Thumb state.
-ansi	Specifies ANSI C with extensions.
-mlittle-endian	Generate code for a processor running in little-endian mode.
-fomit-frame-pointer	Removes the frame pointer for all functions, which might make debugging harder.
-msoft-float	Use software floating-point instructions.
-fno-common	Specifies that the compiler should place uninitialized global variables in the data section of the object file, rather than generating them as common blocks.
-Wall	Enables all the warnings about constructions that some users consider questionable, and that are easy to avoid even in conjunction with macros.
-Wextra	Enables some extra warning flags that are not enabled by '-Wall'.
-Wstrict-prototypes	Warn if a function is declared or defined without specifying the argument types.
-Wno-sign-compare	Do not warn when a comparison between signed and unsigned values could produce an incorrect result when the signed value is converted to unsigned.
-fstack-usage	Geneates an extra file that specifies the maximum amount of stack used, on a per-function basis.
-fdump-ipa-all	Enables all inter-procedural analysis dumps.
-Werror=implicit-function-declaration	Generates an error when the prototype of the function is not defined..
-DAUTOSAR_OS_NOT_USED	-D defines a preprocessor symbol and optionally can set it to a value. AUTOSAR_OS_NOT_USED: By default in the package, the drivers are compiled to be used without Autosar OS. If the drivers are used with Autosar OS, the compiler option '-DAUTOSAR_OS_NOT_USED' must be removed from project options
-DGCC	-D defines a preprocessor symbol and optionally can set it to a value. This one defines the GCC preprocessor symbol.
-std=c99	C programming language standard version c99

Table 3-2. Assembler Options

Option	Description
-mcpu=cortex-m4	Selects target processor: Arm Cortex M4
-mcpu=cortex-m0plus	Selects target processor: Arm Cortex M0+
-c	Produces an object file (called input-file.o) for each source file.
-mthumb	This option specifies that the assembler should start assembling Thumb instructions.
-x assembler-with-cpp	Indicates that the assembly code contains C directives and the C preprocessor must be run.

Table 3-3. Linker Options

Option	Description
-Map=filename	Print a link map to the file mapfile.
-T scriptfile	Use scriptfile as the linker script. This script replaces ld's default linker script(rather than adding to it), so commandfile must specify everything necessary to describe the output file.
--disable-newlib-supplied-syscalls -specs=nosys.specs	These options support for using newlib on core M0+
-u _printf_float -u _scanf_float	These options support generating profile report.
-nostartfiles	Do not use the standard system startup files when linking
-e _start	Specify that the program entry point is _start
-static	The --static flag tells the linker to link a static, not a dynamically linked
-lc	The -lc flag tells the linker to link this binary against the C library, which is newlib in our case.
-lnosys	The -lnosys flag tells the linker to link this binary against the "nosys" library
$(TOOLCHAIN_DIR)/arm-none-eabi/newlib/lib/thumb/v6-m $(TOOLCHAIN_DIR)/lib/gcc/arm-none-eabi/6.3.1/thumb/v6-m	Library for core M0+, added with -L and -B option
$(TOOLCHAIN_DIR)/arm-none-eabi/newlib/lib/thumb $(TOOLCHAIN_DIR)/arm-none-eabi/newlib/lib)	Library for core M4, added with -L and -B option

3.1.2 GHS Compiler/Linker/Assembler Options

Table 3-4. Compiler Options

Option	Description
-cpu=cortexm4	Selects target processor: Arm Cortex M4
-cpu=cortexm0plus	Selects target processor: Arm Cortex M0+
-ansi	Specifies ANSI C with extensions. This mode extends the ANSI X3.159-1989 standard with certain useful and compatible constructs.
-Osize	Optimize for size.
-dual_debug	Enables the generation of DWARF, COFF, or BSD debugging information in the object file
-G	Generates source level debugging information and allows procedure call from debugger's command line.
--no_exceptions	Disables support for exception handling
-Wundef	Generates warnings for undefined symbols in preprocessor expressions
-Wimplicit-int	Issues a warning if the return type of a function is not declared before it is called
-Wshadow	Issues a warning if the declaration of a local variable shadows the declaration of a variable of the same name declared at the global scope, or at an outer scope
-Wtrigraphs	Issues a warning for any use of trigraphs

Table continues on the next page...

Table 3-4. Compiler Options (continued)

Option	Description
-Wall	Enables all the warnings about constructions that some users consider questionable, and that are easy to avoid even in conjunction with macros.
--prototype_errors	Generates errors when functions referenced or called have no prototype
--incorrect_pragma_warnings	Valid #pragma directives with wrong syntax are treated as warnings
-noslashcomment	C++ like comments will generate a compilation error
-preprocess_assembly_files	Preprocesses assembly files
-nostartfile	Do not use Start files
--short_enum	Store enumerations in the smallest possible type
-c	Produces an object file (called input-file.o) for each source file.
--no_commons	Allocates uninitialized global variables to a section and initializes them to zero at program startup.
-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. Produces an object file (called input-file.o) for each source file.
-list	Creates a listing by using the name of the object file with the .lst extension. Assembler option
-DAUTOSAR_OS_NOT_USED	-D defines a preprocessor symbol and optionally can set it to a value. AUTOSAR_OS_NOT_USED: By default in the package, the drivers are compiled to be used without Autosar OS. If the drivers are used with Autosar OS, the compiler option '-DAUTOSAR_OS_NOT_USED' must be removed from project options
-DDISABLE_MCAL_INTERMODULE_ASR_CHECK	-D defines a preprocessor symbol to disable the inter-module version check for AR_RELEASE versions. DISABLE_MCAL_INTERMODULE_ASR_CHECK: By default in the package, drivers are compiled to perform the inter-module version check as per Autosar BSW004. When the inter-module version check needs to be disabled then the DISABLE_MCAL_INTERMODULE_ASR_CHECK global define must be added to the list of compiler options.
-DGHS	-D defines a preprocessor symbol and optionally can set it to a value. This one defines the GHS preprocessor symbol.

Table 3-5. Assembler Options

Option	Description
-cpu=cortexm4	Selects target processor: Arm Cortex M4
-cpu=cortexm0plus	Selects target processor: Arm Cortex M0+
-c	Produces an object file (called input-file.o) for each source file.
-preprocess_assembly_files	Preprocesses assembly files
-asm=list	Creates a listing by using the name of the object file with the .lst extension. Assembler option

Table 3-6. Linker Options

Option	Description
-Mn	Map file numeric ordering
-delete	Removal from the executable of functions that are unused and unreferenced
-v	Display removed unused functions

Table continues on the next page...

Table 3-6. Linker Options (continued)

Option	Description
-ignore_debug_references	Ignores relocations from DWARF debug sections when using -delete.
-map	Creates a detailed map file
-keepmap	Keep the map file in the event of a link error
-lstartup	Link libstartup library -Run-time environment startup routines
-lsys	Link libsys library -Run-time environment system routines
-larch	Link libarch library -Target-specific run-time support. Any file produced by the Green Hills Compiler may depend on symbols in this library.
-lansi	Link libansi library -the standard C library
-L(/lib/thumb2)	Link thumb2 library
-lutf8_s32	Include utf8_s32.a to use the Wide Character Functions

3.1.3 IAR Compiler/Linker/Assembler Options

Table 3-7. Compiler Options

Option	Description
--cpu=Cortex-M4	Selects target processor: Arm Cortex M4
--cpu=Cortex-M0+	Selects target processor: Arm Cortex M0+
--cpu_mode=thumb	Selects generating code that executes in Thumb state.
--endian=little	Specifies the endianess of core: little endian.
-Ohz	Sets the optimization level to High, favoring size.
-c	Produces an object file (called input-file.o) for each source file.
--no_clustering	Disables static clustering optimizations.
--no_mem_idioms	Makes the compiler to not optimize code sequences that clear, set, or copy a memory region.
--no_explicit_zero_opt	Places the zero initialized variables in data section instead of bss.
--debug	Makes the compiler include information in the object modules.
--diag_suppress=Pa050	Suppresses diagnostic messages (warnings) about non-standard line endings.
-DAUTOSAR_OS_NOT_USED	-D defines a preprocessor symbol and optionally can set it to a value. AUTOSAR_OS_NOT_USED: By default in the package, the drivers are compiled to be used without Autosar OS. If the drivers are used with Autosar OS, the compiler option '-DAUTOSAR_OS_NOT_USED' must be removed from project options
-DIAR	-D defines a preprocessor symbol and optionally can set it to a value. This one defines the IAR preprocessor symbol.
--require_prototypes	Forces the compiler to verify that all functions have proper prototypes.
--no_wrap_diagnostics	Disables line wrapping of diagnostic messages issued by compiler.
--no_system_include	Disables the automatic search for system include files.
-e	Enables language extensions. This option is needed by FLS driver which uses _packed structures.

Table 3-8. Assembler Options

Option	Description
--cpu=Cortex-M4	Selects target processor: Arm Cortex M4
--cpu=Cortex-M0+	Selects target processor: Arm Cortex M0+
--cpu_mode=thumb	Selects generating code that executes in Thumb state.
-g	Use this option to disable the automatic search for system include files.

Table 3-9. Linker Options

Option	Description
--cpu=Cortex-M4	Selects target processor: Arm Cortex M4
--cpu=Cortex-M0+	Selects target processor: Arm Cortex M0+
--map filename	Produces a map file.
--no_library_search	Disables automatic runtime library search.
--entry _start	Treats the symbol _start as a root symbol and as the start of the application.
--enable_stack_usage	Enables stack usage analysis.
--skip_dynamic_initialization	Suppress dynamic initialization during system startup.
--no_wrap_diagnostics	Disables line wrapping of diagnostic messages issued by linker.
--config	Specifies the configuration file to be used by the linker.

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 S32K14X 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\Icu.h
- ..\ ICU_TS_T40D2M10I1R0 \include\Icu_Types.h
- ..\ ICU_TS_T40D2M10I1R0 \include\Icu_EnvCfg.h
- ..\ ICU_TS_T40D2M10I1R0 \include\Icu_Ftm.h
- ..\ ICU_TS_T40D2M10I1R0 \include\Icu_Ftm_Irq.h
- ..\ ICU_TS_T40D2M10I1R0 \include\Icu_Ftm_Types.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_Reg_eSys_Port_Ci.h
- ..\ ICU_TS_T40D2M10I1R0 \include\Icu_Port_Ci.h
- ..\ ICU_TS_T40D2M10I1R0 \include\Icu_Port_Ci_Types.h
- ..\ ICU_TS_T40D2M10I1R0 \include\Icu_Lptmr.h
- ..\ ICU_TS_T40D2M10I1R0 \include\Icu_Lptmr_Types.h
- ..\ ICU_TS_T40D2M10I1R0 \include\Icu_LPit.h
- ..\ ICU_TS_T40D2M10I1R0 \include\Icu_LPit_Types.h
- ..\ ICU_TS_T40D2M10I1R0 \src\Icu.c
- ..\ ICU_TS_T40D2M10I1R0 \src\Icu_Ftm.c
- ..\ ICU_TS_T40D2M10I1R0 \src\Icu_Ipw.c
- ..\ ICU_TS_T40D2M10I1R0 \src\Icu_Port_Ci.c
- ..\ ICU_TS_T40D2M10I1R0 \src\Icu_Port_Ci_Irq.c
- ..\ ICU_TS_T40D2M10I1R0 \src\Icu_Lptmr.c
- ..\ ICU_TS_T40D2M10I1R0 \src\Icu_LPit.c

ICU Generated Files

- Icu_Cfg.c (For PC Variant) - For driver compilation, this file should be generated by the user using a configuration tool
- Icu_Cfg.h - For driver compilation, this file should be generated by the user using a configuration tool
- Icu_PBcfg_[variant].c (For PB Variant) - For driver compilation, this file should be generated by the user using a configuration tool
- Icu_PBcfg_[variant].h (For PB Variant) - For driver compilation, this file should be generated by the user using a configuration tool. This is used to export the init configuration pointer of Variant [variant] to be used as parameter for Icu_Init

Note: As a deviation from standard:

- Icu_PBcfg[VariantName].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.

Files from Base common folder

- ..\Base_TS_T40D2M10I1R0 \include\Compiler.h
- ..\Base_TS_T40D2M10I1R0 \include\Compiler_Cfg.h
- ..\Base_TS_T40D2M10I1R0 \include\ComStack_Types.h
- ..\Base_TS_T40D2M10I1R0 \include\MemMap.h

- ..\Base_TS_T40D2M10I1R0 \include\Mcal.h
- ..\Base_TS_T40D2M10I1R0 \include\Platform_Types.h
- ..\Base_TS_T40D2M10I1R0 \include\Std_Types.h
- ..\Base_TS_T40D2M10I1R0 \include\Reg_eSys.h
- ..\Base_TS_T40D2M10I1R0 \include\Soc_Ips.h
- ..\Base_TS_T40D2M10I1R0 \include\SilRegMacros.h

Files from Rte folder:
- ..\Rte_TS_T40D2M10I1R0 \include\SchM_Icu.h

Files from Det folder:
- ..\Det_TS_T40D2M10I1R0 \include\Det.h

Files from EcuM folder:
- ..\EcuMTS_T40D2M10I1R0 \include\EcuM_Cbk.h

Files from Mcl folder:
- ..\Mcl_TS_T40D2M10I1R0 \include\Ftm_Common_Types.h
- ..\Mcl_TS_T40D2M10I1R0 \include\Reg_eSys_Ftm.h
- ..\Mcl_TS_T40D2M10I1R0 \src\Ftm_Common.c
- ..\Mcl_TS_T40D2M10I1R0 \include\Reg_eSys_Lptmr.h
- ..\Mcl_TS_T40D2M10I1R0 \src\Lptmr_Common.c
- ..\Mcl_TS_T40D2M10I1R0 \include\Reg_eSys_LPit.h
- ..\Mcl_TS_T40D2M10I1R0 \src\LPit_Common.c

3.3 Setting up the Plug-ins

All the Autosar MCAL drivers for S32K14X were designed to be configured using Tresos Studio (version EB tresos Studio 24.0.1 b180321-0610 or later).

Location of various files inside the plugin folder is explained below.

- VSMD (Vendor Specific Module Definition) file in EB tresos Studio XDM format:
  - ..\ ICU _ TS_T40D2M10I1R0 \config\Icu.xdm
  - ..\ EcuM_TS_T40D2M10I1R0 \config\EcuM.xdm
  - ..\ Resource_TS_T40D2M10I1R0 \config\Resource.xdm
  - ..\ Mcl_TS_T40D2M10I1R0 \config\Mcl.xdm

- VSMD (Vendor Specific Module Definition) file(s) in AUTOSAR compliant EPD format:
  - ..\ ICU _ TS_T40D2M10I1R0 \autosar\Icu_<subderivative_name>.epd
  - ..\ EcuM_TS_T40D2M10I1R0 \autosar\EcuM.epd

- ..\ Resource_TS_T40D2M10I1R0 \autosar\Resource_<subderivative_name>.epd
- ..\ Mcl_TS_T40D2M10I1R0 \autosar\Mcl_<subderivative_name>.epd

- Code Generation Templates for parameters without variation points:
  - ..\ ICU _ TS_T40D2M10I1R0 \output\src\Icu_Cfg.c
  - ..\ ICU _ TS_T40D2M10I1R0 \output\include\Icu_Cfg.h
  - ..\ EcuM_TS_T40D2M10I1R0 \output\include\EcuM_Cfg.h
  - ..\ Mcl_TS_T40D2M10I1R0 \output\include\CDD_Mcl_Cfg.h
  - ..\ Mcl_TS_T40D2M10I1R0 \output\include\Mcl_DmaMux.h
  - ..\ Mcl_TS_T40D2M10I1R0 \output\include\CDD_Mcl_Cfg.c

- Code Generation Templates for for variant aware parameters:
  - ..\ ICU _ TS_T40D2M10I1R0 \output\src\Icu_PBCfg.c
  - ..\ ICU _ TS_T40D2M10I1R0 \output\include\Icu_Cfg.h
  - ..\ EcuM_TS_T40D2M10I1R0 \output\include\EcuM_Cfg.h
  - ..\ Mcl_TS_T40D2M10I1R0 \output\include\CDD_Mcl_Cfg.h
  - ..\ Mcl_TS_T40D2M10I1R0 \output\include\Mcl_DmaMux.h
  - ..\ Mcl_TS_T40D2M10I1R0 \output\include\CDD_Mcl_PBcfg.c

Steps to generate the configuration:

1. Copy the module folders ICU _ TS_T40D2M10I1R0 , Base_ TS_T40D2M10I1R0 , Resource_ TS_T40D2M10I1R0 , EcuM_ TS_T40D2M10I1R0, EcuC_ 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: s32k148_lqfp144, s32k148_lqfp176, s32k148_mapbga100, s32k146_lqfp144, s32k146_lqfp100, s32k146_lqfp64, s32k146_mapbga100, s32k144_lqfp100, s32k144_lqfp64, s32k144_mapbga100, s32k142_lqfp100, s32k142_lqfp64, s32k118_lqfp48, s32k118_lqfp64, s32k142_lqfp48, s32k144_lqfp48, s32k148_lqfp100 .
- **ECUM** is required for selecting the reference to the wakeup source for every Icu channel configured as a wakeup source.
- **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.
- **ECUC** is required configuring the variant handling in Tresos.

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 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

5.1 Exclusive areas to be defined in BSW scheduler

ICU_EXCLUSIVE_AREA_00 Used in function Icu_SetBitChState to protect the set of the internal channel state

ICU_EXCLUSIVE_AREA_01 Used in function Icu_ClearBitChState to protect the clear internal channel state

ICU_EXCLUSIVE_AREA_02 Used in function Icu_StartTimestamp to protect the updates to:

- Icu_aBuffer[]
- Icu_aBufferSize[]
- Icu_aBufferNotify[]
- Icu_aNotifyCount[]
- Icu_aBufferIndex[]

ICU_EXCLUSIVE_AREA_03 Used in function Icu_TimestampDmaProcessing to protect the updates to:

- Icu_aBufferSize[]
- Icu_aBufferNotify[]
- Icu_aNotifyCount[]
- Icu_aBufferIndex[]

ICU_EXCLUSIVE_AREA_04 Used in interrupt function to protect the updates to:

- Icu_aBuffer[]
- Icu_aBufferSize[]
- Icu_aBufferNotify[]
- Icu_aNotifyCount[]
- Icu_aBufferIndex[]

ICU_EXCLUSIVE_AREA_05 Used in Icu_GetTimeElapsed function to protect the updates to:

- Icu_aPeriod[]
- Icu_aActivePulseWidth[]

ICU_EXCLUSIVE_AREA_06 Used in Icu_GetDutyCycleValues function to protect the updates to:

- Icu_aPeriod[]
- Icu_aActivePulseWidth[]

ICU_EXCLUSIVE_AREA_07 Used in interrupt function to protect the updates to:

- Icu_aPeriod[]
- Icu_aActivePulseWidth[]

ICU_EXCLUSIVE_AREA_08 Used in Icu_StartSignalMeasurement function to protect the updates to:

- Icu_aPeriod[]
- Icu_aActivePulseWidth[]

ICU_EXCLUSIVE_AREA_09 Used in Icu_Ftm_SetPrescaler function to protect the updates to:

- Channel (n) Status And Control Register

ICU_EXCLUSIVE_AREA_10 Used in Icu_Ftm_ProcessTofInterrupt function to protect the updates to:

- Channel (n) Status And Control Register

ICU_EXCLUSIVE_AREA_11 Used in Icu_Ftm_GetOverflow function to protect the updates to:

- Channel (n) Status And Control Register

ICU_EXCLUSIVE_AREA_12 Used in Icu_Ftm_GlobalConfiguration function to protect the updates to:

- Channel (n) Status And Control Register

ICU_EXCLUSIVE_AREA_13, ICU_EXCLUSIVE_AREA_14 Used in Icu_Ftm_StartSignalMeasurement function to protect the updates to:

- Channel (n) Status And Control Register

ICU_EXCLUSIVE_AREA_15 Used in Icu_Ftm_StopSignalMeasurement function to protect the updates to:

- Channel (n) Status And Control Register

ICU_EXCLUSIVE_AREA_16 Used in Icu_Ftm_ClearStatusFlags function to protect the updates to:

- Channel (n) Status And Control Register

ICU_EXCLUSIVE_AREA_17 Used in Icu_Ftm_DisableEdgeCount function to protect the updates to:
- Channel (n) Status And Control Register

ICU_EXCLUSIVE_AREA_18 Used in Icu_Ftm_EnableEdgeCount function to protect the updates to:
- Channel (n) Status And Control Register

ICU_EXCLUSIVE_AREA_19 Used in Icu_Ftm_StopTimestamp function to protect the updates to:
- Channel (n) Status And Control Register

ICU_EXCLUSIVE_AREA_20 Used in Icu_Ftm_StartTimestamp function to protect the updates to:
- Channel (n) Status And Control Register

ICU_EXCLUSIVE_AREA_21 Used in Icu_Ftm_SetChConfig function to protect the updates to:
- Channel (n) Status And Control Register

ICU_EXCLUSIVE_AREA_22 Used in Icu_Ftm_ClearChConfig function to protect the updates to:
- Channel (n) Status And Control Register

ICU_EXCLUSIVE_AREA_23

Used in Icu_Ftm_SignalMeasurement function to protect the updates to:
- Channel (n) Status And Control Register
- Function For Linked Channels

ICU_EXCLUSIVE_AREA_24 Used in Icu_Ftm_DisableEdgeDetection function to protect the updates to:
- Channel (n) Status And Control Register

ICU_EXCLUSIVE_AREA_25 Used in Icu_Ftm_EnableEdgeDetection function to protect the updates to:
- Channel (n) Status And Control Register

ICU_EXCLUSIVE_AREA_26 Used in Icu_Ftm_SetActivationCondition function to protect the updates to:
- Pin Control Register
- Channel (n) Status And Control Register
- Low Power Timer Control Status Register

ICU_EXCLUSIVE_AREA_27 Used in Icu_Ftm_SetNormalMode function to protect the updates to:

- Channel (n) Status And Control Register

ICU_EXCLUSIVE_AREA_28 Used in Icu_Ftm_SetSleepMode function to protect the updates to:

- Channel (n) Status And Control Register

ICU_EXCLUSIVE_AREA_29 Used in Icu_Ftm_GetInputState function to protect the updates to:

- Module Status Register.
- Channel (n) Status And Control Register
- Low Power Timer Control Status Register

ICU_EXCLUSIVE_AREA_30 Used in Icu_Ftm_StartChannel function to protect the updates to:

- Channel (n) Status And Control Register

ICU_EXCLUSIVE_AREA_31 Used in Icu_Ftm_StopChannel function to protect the updates to:

- Channel (n) Status And Control Register

ICU_EXCLUSIVE_AREA_32 Used in Icu_Lptmr_EnableInterrupt function to protect the updates to:

- Channel (n) Status And Control Register

ICU_EXCLUSIVE_AREA_33 Used in Icu_Lptmr_DisableInterrupt function to protect the updates to:

- Channel (n) Status And Control Register

ICU_EXCLUSIVE_AREA_34 Used in Icu_Lptmr_SetChConfig function to protect the updates to:

- Icu_Lptmr_aChConfig[]

ICU_EXCLUSIVE_AREA_35 Used in Icu_Lptmr_ClearChConfig function to protect the updates to:

- Icu_Lptmr_aChConfig[]

ICU_EXCLUSIVE_AREA_36 Used in Icu_Lptmr_SetActivationCondition function to protect the updates to:

- Pin Control Register
- Module Status Register.

ICU_EXCLUSIVE_AREA_37 Used in Icu_Lptmr_ResetEdgeCount function to protect the updates to:

- Channel (n) Status And Control Register

ICU_EXCLUSIVE_AREA_38 Used in Icu_Lptmr_EnableEdgeCount function to protect the updates to:

- Channel (n) Status And Control Register

ICU_EXCLUSIVE_AREA_39 Used in Icu_Lptmr_GetInputState function to protect the updates to:

- Module Status Register.

ICU_EXCLUSIVE_AREA_40 Used in Icu_Lptmr_ProcessInterrupt function to protect the updates to:

- Module Status Register.

ICU_EXCLUSIVE_AREA_41 Used in Icu_Lptmr_EnableInterrupt function to protect the updates to:

- Module Status Register.

ICU_EXCLUSIVE_AREA_60 Used in Icu_Port_Ci_EnableInterrupt function to protect the updates to:

- Channel (n) Status And Control Register

ICU_EXCLUSIVE_AREA_61 Used in Icu_Port_Ci_DisableInterrupt function to protect the updates to:

- Channel (n) Status And Control Register

ICU_EXCLUSIVE_AREA_62 Used in Icu_Port_Ci_SetActivationCondition function to protect the updates to:

- Pin Control Register
- Channel (n) Status And Control Register

ICU_EXCLUSIVE_AREA_63 Used in Icu_LPit_EnableInterrup function to protect the updates to:

- Channel (n) Status And Control Register

ICU_EXCLUSIVE_AREA_64 Used in Icu_LPit_DisableInterrupt function to protect the updates to:

- Channel (n) Status And Control Register

Critical Region Exclusive Matrix

Please see more detail in AUTOSAR_MCAL_ICU_EXCLUSIVE_AREAS.xlsx file that was in design folder.

5.2 Peripheral Hardware Requirements

ICU driver has 4 modules FlexTimer (from FlexTimer 0 to FlexTimer 3), each module can support 8 channels (FlexTimer channels 0-7).

External interrupt channels 0-17 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:

- Port A, B, C, D, D 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 S32K14X family in User Manual

5.3 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:

Table 5-1. FlexTimer Interrupts

FlexTimer Module Interrupts	Hardware interrupt vector
FTM_0_CH_0_CH_1_ISR	115
FTM_0_CH_2_CH_3_ISR	116
FTM_0_CH_4_CH_5_ISR	117
FTM_0_CH_6_CH_7_ISR	118
FTM_0_OVF_ISR	120
FTM_1_CH_0_CH_1_ISR	121
FTM_1_CH_2_CH_3_ISR	122
FTM_1_CH_4_CH_5_ISR	123
FTM_1_CH_6_CH_7_ISR	124
FTM_1_OVF_ISR	126
FTM_2_CH_0_CH_1_ISR	127
FTM_2_CH_2_CH_3_ISR	128

Table continues on the next page...

Table 5-1. FlexTimer Interrupts (continued)

FlexTimer Module Interrupts	Hardware interrupt vector
FTM_2_CH_4_CH_5_ISR	129
FTM_2_CH_6_CH_7_ISR	130
FTM_2_OVF_ISR	132
FTM_3_CH_0_CH_1_ISR	133
FTM_3_CH_2_CH_3_ISR	134
FTM_3_CH_4_CH_5_ISR	135
FTM_3_CH_6_CH_7_ISR	136
FTM_3_OVF_ISR	138

Table 5-2. External PORT_CI Interrupts

PORT_CI Module Interrupts	Hardware interrupt vector
ICU_PORT_CI_A_EXT_IRQ_ISR	75
ICU_PORT_CI_B_EXT_IRQ_ISR	76
ICU_PORT_CI_C_EXT_IRQ_ISR	77
ICU_PORT_CI_D_EXT_IRQ_ISR	78
ICU_PORT_CI_E_EXT_IRQ_ISR	79

Table 5-3. External LPTMR (Low power timer) Interrupts

LPTMR Module Interrupts	Hardware interrupt vector
LPTMR_0_CH_0_ISR	74

Table 5-4. LPIT Interrupts

LPIT Module Interrupts	Hardware interrupt vector
LPIT_0_CH_0_ISR	64
LPIT_0_CH_1_ISR	65
LPIT_0_CH_2_ISR	66
LPIT_0_CH_3_ISR	67

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.

5.4 ISR Macro

MCAL 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:

a. OS is not used - AUTOSAR_OS_NOT_USED is defined:

i. If USE_SW_VECTOR_MODE is defined:

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

In this case, drivers' interrupt handlers are normal C functions and the prolog/epilog handle the context save and restore.

ii. If USE_SW_VECTOR_MODE is not defined:

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

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

Custom OS is used - AUTOSAR_OS_NOT_USED is not defined

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

In this case, OS is handling the execution context when an interrupt occurs. Drivers' interrupt handlers are normal C functions.

Other vendor's OS is used - AUTOSAR_OS_NOT_USED is not defined. Please refer to the OS documentation for description of the ISR macro.

5.5 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.

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.

Hence template files for EcuM is provided at

..\EcuM_<plugin_name>\autosar\EcuM.epd (Module Parameter Definition File – AUTOSAR Format)

..\EcuM_<plugin_name>\config\EcuM.xdm (Module Parameter Definition File – Tresos Format)

5.6 Data cache restriction

None

5.7 User Mode support

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

Chapter 6
Main API Requirements

6.1 Main functions calls within BSW scheduler

None

6.2 Main 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 TimestampNotificationName

(

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 callof 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

7.1 Sections to be defined in MemMap.h

Tables descibe Sections to be defined in MemMap.h:

Table 7-1. Sectionto be define

<Section name>	Tyep 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_INIT_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_INIT_UNSPECIFIED	Variables	End of above section.
ICU_START_SEC_VAR_INIT_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 values after every reset

Table continues on the next page...

Table 7-1. Sectionto be define (continued)

ICU_STOP_SEC_VAR_INIT_8	Variables	End of above section.
ICU_START_SEC_VAR_INIT_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_INIT_16	Variables	End of above section.
ICU_START_SEC_VAR_INIT_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_INIT_32	Variables	End of above section.
ICU_START_SEC_VAR_NO_INIT_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_NO_INIT_UNSPECIFIED	Variables	End of above section.
ICU_START_SEC_VAR_NO_INIT_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 non-cacheable memory section. These variables are never cleared and never initialized by start-up code..
ICU_STOP_SEC_VAR_NO_INIT_32_NO_CACHEABLE	Variables	End of above section.

7.2 Linker command file

Memory shall be allocated for every section defined in ICU_MemMap.h

Chapter 8
Configuration parameters considerations

Configuration parameter class for Autosar ICU driver fall into the following variants as defined below:

8.1 Configuration Parameters

Specifies whether the configuration parameter shall be of configuration class Post Build.

Table 8-1. Configuration Parameters

Configuration Container	Configuration Parameters	Configuration Variant	Current Implementation
Icu	IMPLEMENTATION_CONFIG_VARIANT	Pre Compile parameter for all Variants of Configuration	Pre Compile
IcuConfigSet	IcuMaxChannel	VariantPC or VariantPB	Post Build
IcuConfigSet/IcuChannel	IcuChannelId	VariantPC or VariantPB	Post Build
	IcuHwIP	VariantPC or VariantPB	Post Build
	IcuFtmChannelRef	VariantPC or VariantPB	Post Build
	IcuPortChannelRef	VariantPC or VariantPB	Post Build
	IcuLptmrChannelRef	VariantPC or VariantPB	Post Build
	IcuLpitChannelRef	VariantPC or VariantPB	Post Build
	IcuDMAChannelEnable	VariantPC or VariantPB	Post Build
	IcuDMAChannelRef	VariantPC or VariantPB	Post Build
	IcuDefaultStartEdge	VariantPC or VariantPB	Post Build
	IcuMeasurementMode	VariantPC or VariantPB	Post Build
	IcuOverflowNotification	VariantPC or VariantPB	Post Build
	IcuWakeupCapability	VariantPC or VariantPB	Post Build
IcuConfigSet/IcuChannel/IcuSignalEdgeDetection	IcuSignalNotification	VariantPC or VariantPB	Post Build
IcuConfigSet/IcuChannel/IcuSignalMeasurement	IcuSignalMeasurementProperty	VariantPC or VariantPB	Post Build

Table continues on the next page...

Table 8-1. Configuration Parameters (continued)

Configuration Container	Configuration Parameters	Configuration Variant	Current Implementation
IcuConfigSet/IcuChannel/ IcuTimestampMeasurement	IcuTimestampMeasurementProperty	VariantPC or VariantPB	Post Build
	IcuTimestampNotification	VariantPC or VariantPB	Post Build
IcuConfigSet/IcuChannel/ IcuWakeup	IcuChannelWakeupInfo	VariantPC or VariantPB	Post Build
IcuConfigSet/IcuFtm	IcuFtmModule	VariantPC or VariantPB	Post Build
	Icu_FlexTimer_Prescaler	VariantPC or VariantPB	Post Build
	Icu_FlexTimer_Prescaler_Alternate	VariantPC or VariantPB	Post Build
	Icu_FlexTimer_ClockSource	VariantPC or VariantPB	Post Build
IcuConfigSet/IcuFtm/ IcuFtmChannel	IcuFtmChannel	VariantPC or VariantPB	Post Build
	Icu_FlexTimerFilter	VariantPC or VariantPB	Post Build
	IcuFreezeEnable	VariantPC or VariantPB	Post Build
IcuConfigSet/IcuPort	IcuPortModule	VariantPC or VariantPB	Post Build
IcuConfigSet/IcuPort/ IcuPortChannels	IcuPortChannel	VariantPC or VariantPB	Post Build
IcuConfigSet/IcuLpit	IcuLpitModule	VariantPC or VariantPB	Post Build
	IcuFreezeEnable	VariantPC or VariantPB	Post Build
IcuConfigSet/IcuLpit/ IcuLpitChannels	IcuLpitChannel	VariantPC or VariantPB	Post Build
	IcuLpitTriggerSource	VariantPC or VariantPB	Post Build
	IcuLpitTriggerSelect	VariantPC or VariantPB	Post Build
IcuGeneral	IcuDevErrorDetect	Pre Compile parameter for all Variants of Configuration	Pre Compile
	IcuReportWakeupSource	VariantPC or VariantPB	Post Build
IcuNonAUTOSAR	IcuOverflowNotificationApi	Pre Compile parameter for all Variants of Configuration	Pre Compile
	IcuEnableDualClockMode	Pre Compile parameter for all Variants of Configuration	Pre Compile
	IcuGetInputLevelApi	Pre Compile parameter for all Variants of Configuration	Pre Compile
	IcuGetCaptureRegisterValueApi	Pre Compile parameter for all Variants of Configuration	Pre Compile
	IcuRegisterLockingMode	Pre Compile parameter for all Variants of Configuration	Pre Compile
IcuOptionalApis	IcuDeInitApi	Pre Compile parameter for all Variants of Configuration	Pre Compile
	IcuDisableWakeupApi	Pre Compile parameter for all Variants of Configuration	Pre Compile
	IcuEdgeCountApi	Pre Compile parameter for all Variants of Configuration	Pre Compile

Table continues on the next page...

Table 8-1. Configuration Parameters (continued)

Configuration Container	Configuration Parameters	Configuration Variant	Current Implementation
	IcuEnableWakeupApi	Pre Compile parameter for all Variants of Configuration	Pre Compile
	IcuGetDutyCycleValuesApi	Pre Compile parameter for all Variants of Configuration	Pre Compile
	IcuGetInputStateApi	Pre Compile parameter for all Variants of Configuration	Pre Compile
	IcuGetTimeElapsedApi	Pre Compile parameter for all Variants of Configuration	Pre Compile
	IcuGetVersionInfoApi	Pre Compile parameter for all Variants of Configuration	Pre Compile
	IcuSetModeApi	Pre Compile parameter for all Variants of Configuration	Pre Compile
	IcuSignalMeasurementApi	Pre Compile parameter for all Variants of Configuration	Pre Compile
	IcuTimestampApi	Pre Compile parameter for all Variants of Configuration	Pre Compile
	IcuWakeupFunctionalityApi	Pre Compile parameter for all Variants of Configuration	Pre Compile
	IcuEdgeDetectApi	Pre Compile parameter for all Variants of Configuration	Pre Compile
CommonPublishedInformation	ArReleaseMajorVersion	Pre Compile parameter for all Variants of Configuration	Pre Compile
	ArReleaseMinorVersion	Pre Compile parameter for all Variants of Configuration	Pre Compile
	ArReleaseRevisionVersion	Pre Compile parameter for all Variants of Configuration	Pre Compile
	ModuleId	Pre Compile parameter for all Variants of Configuration	Pre Compile
	SwMajorVersion	Pre Compile parameter for all Variants of Configuration	Pre Compile
	SwMinorVersion	Pre Compile parameter for all Variants of Configuration	Pre Compile
	SwPatchVersion	Pre Compile parameter for all Variants of Configuration	Pre Compile
	VendorApiInfix	Pre Compile parameter for all Variants of Configuration	Pre Compile
	VendorId	Pre Compile parameter for all Variants of Configuration	Pre Compile

Chapter 9
Integration Steps

This section gives a brief overview of the steps needed for integrating Input Capture Unit :

- Generate the required ICU configurations. For more details refer to section Files required for Compilation
- Allocate proper memory sections in ICU_MemMap.h and linker command file. For more details refer to section Sections to be defined in MemMap.h
- Compile & build the ICU with all the dependent modules. For more details refer to section Building the Driver

Chapter 10
ISR Reference

None.

Chapter 11 External Module Assumptions

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

[SMCAL_CPR_EXT46]

<< 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.

[SMCAL_CPR_EXT47]

<< The ICU module's environment shall not call any function of the ICU module before having called Icu_Init. >>

[SMCAL_CPR_EXT48]

<< The application shall call the function that starts a signal measurement (Icu_StartSignalMeasurement()) or a timestamp measurement(Icu_StartTimestamp()) 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.

[SMCAL_CPR_EXT49]

<< 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.

[SMCAL_CPR_EXT50]

<< The application shall stop all running channels before de-initializing the ICU 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.

[SMCAL_CPR_EXT163]

<< If interrupts are locked a centralized function pair to lock and unlock interrupts shall be used. >>

[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 for 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;

[SWS_Icu_00348]

<< Re-entrancy of the Icu_TimestampNotification_<Channel> is not relevant for this module (in general it is in this case not re-entrant). >>

How to Reach Us:

Home Page:
nxp.com

Web Support:
nxp.com/support

Document Number UM2ICUASR4.3 Rev0001R1.0.1
Revision 1.0