# Integration Manual

for MPC574XG ICU Driver

Document Number: IM35ICUASR4.2 Rev0002RTM1.0.0
Rev. 5.0.0

# Contents

**Section number** **Title** **Page**

## Chapter 1
## Revision History

## Chapter 2
## Introduction

## Chapter 3
## Building the Driver

## Chapter 4
## Function calls to module

## Chapter 5
## Module requirements

**Section number** **Title** **Page**

## Chapter 6
## Main API Requirements

## Chapter 7
## Memory Allocation

## Chapter 8
## Configuration parameters considerations

## Chapter 9
## Integration Steps

## Chapter 10
## External Module Assumptions

# Chapter 1
# Revision History

Table 1-1. Revision History

| Revision | Date | Author | Description |
|---|---|---|---|
| 1.0.0 | 22/08/2014 | Son Nguyen | Integration Manual for MPC574XG - 0.9.0 Release |
| 2.0.0 | 24/04/2015 | Lam Tran | Integration Manual for MPC574XG - RTM 1.0.0 Release |
| 3.0.0 | 10/07/2015 | Lam Tran | Integration Manual for CALYPSO - RTM 1.0.1 Release |
| 4.0.0 | 12/08/2016 | My Le | Integration Manual for CALYPSO - RTM 1.0.2 Release |
| 5.0.0 | 17/02/2017 | My Le | Integration Manual for CALYPSO AUTOSAR 4.2.2 - RTM 1.0.0 Release |

# Chapter 2
# Introduction

This integration manual describes the integration requirements for ICU Driver for MPC574XG microcontrollers.

## 2.1 Supported Derivatives

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

**Table 2-1. MPC574XG Derivatives**

| | |
|---|---|
| NXP Semiconductor | MPC5748G_LQFP176,<br>MPC5748G_MAPBGA256,<br>MPC5748G_MAPBGA324,<br>MPC5747G_LQFP176,<br>MPC5747G_MAPBGA256,<br>MPC5747G_MAPBGA324,<br>MPC5746G_LQFP176,<br>MPC5746G_MAPBGA256,<br>MPC5746G_MAPBGA324,<br>MPC5748C_LQFP176,<br>MPC5748C_MAPBGA256,<br>MPC5748C_MAPBGA324,<br>MPC5747C_LQFP176,<br>MPC5747C_MAPBGA256,<br>MPC5747C_MAPBGA324,<br>MPC5746C_LQFP176,<br>MPC5746C_MAPBGA256,<br>MPC5746C_MAPBGA324,<br>MPC5746C_MAPBGA100,<br>MPC5745C_LQFP176,<br>MPC5745C_MAPBGA256,<br>MPC5745C_MAPBGA100,<br>MPC5744C_LQFP176,<br>MPC5744C_MAPBGA256,<br>MPC5744C_MAPBGA100,<br>MPC5746B_LQFP176,<br>MPC5746B_MAPBGA256,<br>MPC5746B_MAPBGA100,<br>MPC5744B_LQFP176,<br>MPC5744B_MAPBGA256, |

**Table 2-1. MPC574XG Derivatives**

| | |
|---|---|
| | MPC5744B_MAPBGA100,<br>MPC5745B_LQFP176,<br>MPC5745B_MAPBGA256,<br>MPC5745B_MAPBGA100 |

All of the above microcontroller devices are collectively named as MPC574XG .

## 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

| Abbreviation and Definitions | Description |
|---|---|
| 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 |
| API | Application Programming Interface |
| SUIL2 | System Integration Unit Lite2 |
| WKPU | Wakeup Unit |
| EMIOS | Enhanced Modular IO Subsystem |

## 2.5 Reference List

Table 2-3. Reference List

| # | Title | Version |
|---|---|---|
| 1 | AUTOSAR 4.2 Rev0002ICU Driver Software Specification Document. | 4.2.2 |
| 2 | MPC5748G Reference Manual | Rev. 5, 12/2016 |
| 3 | MPC5746C Reference Manual | Rev. 4, 12/2016 |
| 4 | MPC5748G_1N81M_Rev.2 (official document) (1N81M) | Jun-16 |
| 5 | MPC5748G_1N81M_0N78S_Comparison_Summary_v2_0 (internal document) (1N81M, 0N78S) | 31.10.2016 |
| 6 | MPC5746C_1N06M_Rev.4 (official document) (1N06M) | Jul-16 |
| 7 | MPC5746C_cut1.1_cut2.0_cut2.1_comparison_v0 (internal document) (1N06M, 0N84S, 1N84S) | 14-Sep-16 |
| 8 | C3M_cut2.1_new_errata_20170113 (internal document) (1N84S) | 13-Jan-17 |

# 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 SemiconductorMPC574XG . It also explains the EB Tresos Studio plugin setup procedure.

## 3.1 Build Options

The ICU driver files are compiled using

- Windriver DIAB DIAB_5_9_6_2
- Green Hills Multi 7.1.4 / Compiler 2015.1.6

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

**Note**

The TS_T2D35M10I0R0 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 = 2 identifies PA architecture and Derivative_Id = 35 identifies the MPC574XG )

### 3.1.1 DIAB Compiler/Linker/Assembler Options

Table 3-1. Compiler Options

| Option | Description |
|---|---|
| -tPPCE200Z4204N3VEN:simple | Sets target processor to PPCE200Z4204N3VEN, generates ELF using EABI conventions, No floating point support (minimizes the required runtime), selects simple environment settings for Startup Module and Libraries |
| -tPPCE200Z210N3VEN:simple | Sets target processor to PPCE200Z210N3VEN, generates ELF using EABI conventions, No floating point support (minimizes the required runtime), selects simple environment settings for Startup Module and Libraries |
| -Xdialect-ansi | Follow the ANSI C standard with some additions |
| -XO | Enables extra optimizations to produce highly optimized code |
| -g3 | Generate symbolic debugger information and do all optimizations. |
| -Xsize-opt | Optimize for size rather than speed when there is a choice |
| -Xsmall-data=0 | Set Size Limit for 'small data' Variables to zero. |
| -Xsmall-const=0 | Set Size Limit for “small const” Variables to zero. |
| -Xaddr-sconst=0x11 | Specify addressing for constant static and global variables with size less than or equal to -Xsmall-const to far-absolute. |
| -Xaddr-sdata=0x11 | Specify addressing for non-constant static and global variables with size less than or equal to -Xsmall-data in size to far-absolute. |
| -Xno-common | Disable use of the 'COMMON' feature so that the compiler or assembler will allocate each uninitialized public variable in the .bss section for the module defining it, and the linker will require exactly one definition of each public variable |
| -Xnested-interrupts | Allow nested interrupts |
| -Xdebug-dwarf2 | Generate symbolic debug information in dwarf2 format |
| -Xdebug-local-all | Force generation of type information for all local variables |
| -Xdebug-local-cie | Create common information entry per module |
| -Xdebug-struct-all | Force generation of type information for all typedefs, struct, union and class types |
| -Xforce-declarations | Generates warnings if a function is used without a previous declaration |
| -ee1481 | Generate an error when the function was used before it has been declared |
| -Xmacro-undefined-warn | Generates a warning when an undefined macro name occurs in a #if preprocessor directive |
| -Xlink-time-lint | Enable the checking of object and function declarations across compilation units, as well as the consistency of compiler options used to compile source files |
| -W:as:,-l | Pass the option '-l' (lower case letter L) to the assembler to get an assembler listing file |
| -Wa,-Xisa-vle | Instruct the assembler to expect and assemble VLE (Variable Length Encoding) instructions rather than BookE instructions. |
| -DAUTOSAR_OS_NOT_USED | -D defines a preprocessor symbol and optionally can set it to a value.<br>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 |
| -DUSE_SW_VECTOR_MODE | -D defines a preprocessor symbol and optionally can set it to a value.<br>USE_SW_VECTOR_MODE: By default in the package, drivers are compiled to be used with interrupt controller configured to be in hardware vector mode. In case of AUTOSAR_OS_NOT_USED, the compiler option "-DUSE_SW_VECTOR_MODE" must be added to the list of compiler options to be used with interrupt controller configured to be in software vector mode. |

*Table continues on the next page...*

Table 3-1. Compiler Options (continued)

| Option | Description |
|---|---|
| -DDIAB | -D defines a preprocessor symbol and optionally can set it to a value. This one defines the DIAB preprocessor symbol. |
| -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. |
| -c | Stop after assembly, produce object file. |

Table 3-2. Assembler Options

| Option | Description |
|---|---|
| -tPPCE200Z4204N3VEN:simple | Sets target processor to PPCE200Z4204N3VEN, generates ELF using EABI conventions, No floating point support (minimizes the required runtime), selects simple environment settings for Startup Module and Libraries |
| -tPPCE200Z210N3VEN:simple | Sets target processor to PPCE200Z210N3VEN, generates ELF using EABI conventions, No floating point support (minimizes the required runtime), selects simple environment settings for Startup Module and Libraries |
| -g | Dump the symbols in the global symbol table in each archive file. |
| -Xisa-vle | Expect and assemble VLE (Variable Length Encoding) instructions rather than Book E instructions. The default code section is named .text_vle instead of .text, and the default code section fill "character" is set to 0x44444444 instead of 0. The .text_vle code section will have ELF section header flags marking it as VLE code, not Book E code. |
| -Xasm-debug-on | Generate debug line and file information |
| -Xdebug-dwarf2 | Generate symbolic debug information in dwarf2 format |
| -Xsemi-is-newline | Treat the semicolon (;) as a statement separator instead of a comment character. |

Table 3-3. Linker Options

| Option | Description |
|---|---|
| -tPPCE200Z4204N3VEN:simple | Sets target processor to tPPCE200Z4204N3VEN, generates ELF using EABI conventions, No floating point support (minimizes the required runtime), selects simple environment settings for Startup Module and Libraries |
| -tPPCE200Z210N3VEN:simple | Sets target processor to tPPCE200Z210N3VEN, generates ELF using EABI conventions, No floating point support (minimizes the required runtime), selects simple environment settings for Startup Module and Libraries |
| -Xelf | Generates ELF object format for output file |
| -m6 | Generates a detailed link map and cross reference table |
| -Xlink-time-lint | Enable the checking of object and function declarations across compilation units, as well as the consistency of compiler options used to compile source files |

### 3.1.2 GHS Compiler/Linker/Assembler Options

Table 3-4. Compiler Options

| Option | Description |
|---|---|
| -cpu=ppc5748gz4204 | Selects target processor: ppc5748gz4204 |
| -cpu=ppc5748gz210 | Selects target processor: ppc5748gz210 |
| -ansi | Specifies ANSI C with extensions. This mode extends the ANSI X3.159-1989 standard with certain useful and compatible constructs. |
| -noSPE | Disables the use of SPE and vector floating point instructions by the compiler. |
| -Ospace | Optimize for size. |
| -sda=0 | Enables the Small Data Area optimization with a threshold of 0. |
| -vle | Enables VLE code generation |
| -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 |
| --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 |
| --diag_error 223 | Sets the specified compiler diagnostic messages to the level of error |
| -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 |
| -DUSE_SW_VECTOR_MODE | -D defines a preprocessor symbol and optionally can set it to a value. USE_SW_VECTOR_MODE: By default in the package, drivers are compiled to be used with interrupt controller configured to be in hardware vector mode. In case of AUTOSAR_OS_NOT_USED, the compiler option "-DUSE_SW_VECTOR_MODE" must be added to the list of compiler options to be used with interrupt controller configured to be in software vector mode. |
| -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. |
| -c | Produces an object file (called input-file.o) for each source file. |

Table 3-5. Assembler Options

| Option | Description |
|---|---|
| -cpu=ppc5748gz4204 | Selects target processor: ppc5748gz4204 |
| -cpu=ppc5748gz210 | Selects target processor: ppc5748gz210 |
| -G | Generates source level debugging information and allows procedure call from debugger's command line. |
| -list | Creates a listing by using the name of the object file with the .lst extension |

Table 3-6. Linker Options

| Option | Description |
|---|---|
| -cpu=ppc5748gz4204 | Selects target processor: ppc5748gz4204 |
| -cpu=ppc5748gz210 | Selects target processor: ppc5748gz210 |
| -nostartfiles | Do not use Start files. |
| -vle | Enables VLE code generation |
| --nocpp | Do not Generate Constructors/Destructors |
| -Mn | sort numerically the MAP file |
| -delete | The -delete option instructs the linker to remove functions that are not referenced in the final executable. |
| -ignore_debug_references | Ignores relocations from DWARF debug sections when using -delete. DWARF debug information will contain references to deleted functions that may break some third-party debuggers. |
| -keepmap | keeps the MAP file in case of link error |

## 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 MPC574XG 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_T2D35M10I0R0 \include\Icu.h
- ..\ ICU_TS_T2D35M10I0R0 \include\Icu_Types.h
- ..\ ICU_TS_T2D35M10I0R0 \include\Icu_EnvCfg.h
- ..\ ICU_TS_T2D35M10I0R0 \include\Icu_eMios.h
- ..\ ICU_TS_T2D35M10I0R0 \include\Icu_eMios_Types.h
- ..\ ICU_TS_T2D35M10I0R0 \include\Icu_eMios_Irq.h
- ..\ ICU_TS_T2D35M10I0R0 \include\Icu_Ipw.h

- ..\ ICU_TS_T2D35M10I0R0 \include\Icu_Ipw_Irq.h
- ..\ ICU_TS_T2D35M10I0R0 \include\Icu_Ipw_Types.h
- ..\ ICU_TS_T2D35M10I0R0 \include\Icu_Irq.h
- ..\ ICU_TS_T2D35M10I0R0 \include\Icu_Reg_eSys_Siul2.h
- ..\ ICU_TS_T2D35M10I0R0 \include\Icu_Siul2.h
- ..\ ICU_TS_T2D35M10I0R0 \include\Icu_Siul2_Types.h
- ..\ ICU_TS_T2D35M10I0R0 \include\Icu_Reg_eSys_Wkpu.h
- ..\ ICU_TS_T2D35M10I0R0 \include\Icu_Wkpu.h
- ..\ ICU_TS_T2D35M10I0R0 \include\Icu_Wkpu_Types.h
- ..\ ICU_TS_T2D35M10I0R0 \src\Icu.c
- ..\ ICU_TS_T2D35M10I0R0 \src\Icu_eMios.c
- ..\ ICU_TS_T2D35M10I0R0 \src\Icu_Ipw.c
- ..\ ICU_TS_T2D35M10I0R0 \src\Icu_Siul2.c
- ..\ ICU_TS_T2D35M10I0R0 \src\Icu_Siul2_Irq.c
- ..\ ICU_TS_T2D35M10I0R0 \src\Icu_Wkpu_Irq.c
- ..\ ICU_TS_T2D35M10I0R0 \src\Icu_Wkpu.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_PBcfg.c (For PB 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

### Files from Base common folder

- ..\Base_TS_T2D35M10I0R0 \include\Compiler.h
- ..\Base_TS_T2D35M10I0R0 \include\Compiler_Cfg.h
- ..\Base_TS_T2D35M10I0R0 \include\ComStack_Types.h
- ..\Base_TS_T2D35M10I0R0 \include\MemMap.h
- ..\Base_TS_T2D35M10I0R0 \include\Mcal.h
- ..\Base_TS_T2D35M10I0R0 \include\Platform_Types.h
- ..\Base_TS_T2D35M10I0R0 \include\Std_Types.h
- ..\Base_TS_T2D35M10I0R0 \include\Reg_eSys.h
- ..\Base_TS_T2D35M10I0R0 \include\Soc_Ips.h
- ..\Base_TS_T2D35M10I0R0 \include\SilRegMacros.h

### Files from Rte folder:

- ..\Rte_TS_T2D35M10I0R0 \include\SchM_Icu.h

### Files from Mcl folder:

- ..\Mcl_TS_T2D35M10I0R0 \include\eMios_Common_Types.h
- ..\Mcl_TS_T2D35M10I0R0 \include\Reg_eSys_eMios.h

- ..\Mcl_TS_T2D35M10I0R0 \include\eMios_Common.h
- ..\Mcl_TS_T2D35M10I0R0 \src\eMios_Common.c
- ..\Mcl_TS_T2D35M10I0R0 \include\CDD_Mcl.h
- ..\Mcl_TS_T2D35M10I0R0 \include\Mcl_Types.h
- ..\Mcl_TS_T2D35M10I0R0 \include\Mcl_EnvCfg.h
- ..\Mcl_TS_T2D35M10I0R0 \include\Mcl_Notif.h
- ..\Mcl_TS_T2D35M10I0R0 \include\Mcl_Dma.h
- ..\Mcl_TS_T2D35M10I0R0 \include\Mcl_Dma_Irq.h
- ..\Mcl_TS_T2D35M10I0R0 \include\Mcl_Dma_Types.h
- ..\Mcl_TS_T2D35M10I0R0 \include\Mcl_DmaMux.h
- ..\Mcl_TS_T2D35M10I0R0 \include\Mcl_DmaMux_Types.h
- ..\Mcl_TS_T2D35M10I0R0 \include\Mcl_Ipw.h
- ..\Mcl_TS_T2D35M10I0R0 \include\Mcl_Ipw_Notif.h
- ..\Mcl_TS_T2D35M10I0R0 \include\Mcl_Ipw_Types.h
- ..\Mcl_TS_T2D35M10I0R0 \include\Reg_eSys_Dma.h
- ..\Mcl_TS_T2D35M10I0R0 \include\Reg_eSys_DmaMux.h
- ..\Mcl_TS_T2D35M10I0R0 \src\CDD_Mcl.c
- ..\Mcl_TS_T2D35M10I0R0 \src\Mcl_Dma.c
- ..\Mcl_TS_T2D35M10I0R0 \src\Mcl_Dma_Irq.c
- ..\Mcl_TS_T2D35M10I0R0 \src\Mcl_DmaMux.c
- ..\Mcl_TS_T2D35M10I0R0 \src\Mcl_IPW.c

**Files from Det folder:**

- ..\Det_TS_T2D35M10I0R0 \include\Det.h

**Files from EcuM folder:**

- ..\EcuMTS_T2D35M10I0R0 \include\EcuM_Cbk.h

## 3.3 Setting up the Plug-ins

All the Autosar MCAL drivers for MPC574XG were designed to be configured using Tresos Studio (version EB tresos Studio 21.0.0 b160607-0933 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_T2D35M10I0R0 \config\Icu.xdm
  - ..\ EcuM_TS_T2D35M10I0R0 \config\EcuM.xdm
  - ..\ Resource_TS_T2D35M10I0R0 \config\Resource.xdm
  - ..\ Mcl_TS_T2D35M10I0R0 \config\Mcl.xdm

- VSMD (Vendor Specific Module Definition) file(s) in AUTOSAR compliant EPD format:
    - ..\ ICU _ TS_T2D35M10I0R0 \autosar\Icu_<subderivative_name>.epd
    - ..\ EcuM_TS_T2D35M10I0R0 \autosar\EcuM.epd
    - ..\ Resource_TS_T2D35M10I0R0 \autosar\Resource_<subderivative_name>.epd
    - ..\Mcl_TS_T2D35M10I0R0 \autosar\Mcl_<subderivative_name>.epd

- Code Generation Templates for Pre-Compile time configuration parameters:
    - ..\ ICU _ TS_T2D35M10I0R0 \output\src\Icu_Cfg.c
    - ..\ ICU _ TS_T2D35M10I0R0 \output\include\Icu_Cfg.h
    - ..\ EcuM_TS_T2D35M10I0R0 \output\include\EcuM_Cfg.h
    - ..\ Mcl_TS_T2D35M10I0R0 \output\include\CDD_Mcl_Cfg.h
    - ..\ Mcl_TS_T2D35M10I0R0 \output\include\Mcl_DmaMux.h
    - ..\ Mcl_TS_T2D35M10I0R0 \output\include\CDD_Mcl_Cfg.c

- Code Generation Templates for Post-Build time configuration parameters:
    - ..\ ICU _ TS_T2D35M10I0R0 \output\src\Icu_PBCfg.c
    - ..\ ICU _ TS_T2D35M10I0R0 \output\include\Icu_Cfg.h
    - ..\ EcuM_TS_T2D35M10I0R0 \output\include\EcuM_Cfg.h
    - ..\ Mcl_TS_T2D35M10I0R0 \output\include\CDD_Mcl_Cfg.h
    - ..\ Mcl_TS_T2D35M10I0R0 \output\include\Mcl_DmaMux.h
    - ..\ Mcl_TS_T2D35M10I0R0 \output\include\CDD_Mcl_PBcfg.c

**Steps to generate the configuration:**

1. Copy the module folders ICU _ TS_T2D35M10I0R0 , Dem_ TS_T2D35M10I0R0 , Base_ TS_T2D35M10I0R0 , Resource_ TS_T2D35M10I0R0 , EcuM_ TS_T2D35M10I0R0 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: MPC5748G_LQFP176, MPC5748G_MAPBGA256, MPC5748G_MAPBGA324, MPC5747G_LQFP176, MPC5747G_MAPBGA256, MPC5747G_MAPBGA324, MPC5746G_LQFP176, MPC5746G_MAPBGA256, MPC5746G_MAPBGA324, MPC5748C_LQFP176, MPC5748C_MAPBGA256, MPC5748C_MAPBGA324, MPC5747C_LQFP176, MPC5747C_MAPBGA256, MPC5747C_MAPBGA324, MPC5746C_LQFP176, MPC5746C_MAPBGA256, MPC5746C_MAPBGA324, MPC5746C_MAPBGA100, MPC5745C_LQFP176, MPC5745C_MAPBGA256, MPC5745C_MAPBGA100, MPC5744C_LQFP176, MPC5744C_MAPBGA256, MPC5744C_MAPBGA100, MPC5746B_LQFP176, MPC5746B_MAPBGA256, MPC5746B_MAPBGA100, MPC5744B_LQFP176, MPC5744B_MAPBGA256, MPC5744B_MAPBGA100, MPC5745B_LQFP176, MPC5745B_MAPBGA256, MPC5745B_MAPBGA100 .

- **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

# 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_WKPU_EnableInterrupt function to protect the updates to:

- IRER register
- WRER register
- WISR register

**ICU_EXCLUSIVE_AREA_10** Used in Icu_WKPU_DisableInterrupt function to protect the updates to:

- IRER register
- WRER register
- WISR register

**ICU_EXCLUSIVE_AREA_11** Used in Icu_Wkpu_SetActivationCondition function to protect the updates to:

- WIREER register
- WIFEER register

**ICU_EXCLUSIVE_AREA_13** Used in Icu_eMios_SetChConfig function to protect the updates to:

- Icu_eMios_aChConfig[]

**ICU_EXCLUSIVE_AREA_14** Used in Icu_eMios_ClearChConfig function to protect the updates to:

- Icu_eMios_aChConfig[]

**ICU_EXCLUSIVE_AREA_15** Used in Icu_eMios_ClearChConfig function to protect the updates to:

- CCR register.

**ICU_EXCLUSIVE_AREA_16** Used in Icu_eMios_DisableInterrupt function to protect the updates to:

- CCR register.

**ICU_EXCLUSIVE_AREA_17** Used in Icu_eMios_UCSetMode function to protect the updates to:

- CCR register.

**ICU_EXCLUSIVE_AREA_18** Used in Icu_eMios_StopSignalMeasurement function to protect the updates to:

- CCR register.

**ICU_EXCLUSIVE_AREA_19** Used in Icu_eMios_SetActivationCondition function to protect the updates to:

- CCR register.

**ICU_EXCLUSIVE_AREA_20** Used in Icu_eMios_Init function to protect the updates to:

- CCR register.

**ICU_EXCLUSIVE_AREA_21** Used in Icu_eMios_StartTimestamp function to protect the updates to:

- CCR register.

**ICU_EXCLUSIVE_AREA_22** Used in Icu_eMios_StartSignalMeasurement function to protect the updates to:

- CCR register.
- MCR register.

**ICU_EXCLUSIVE_AREA_23** Used in Icu_eMios_SignalMeasurement function to protect the updates to:

- Icu_aInt_Counter[].
- Icu_CapturedActivePulseWidth.
- IcuPeriod.

**ICU_EXCLUSIVE_AREA_24** Used in Icu_eMios_SetPrescaler function to protect the updates to:

- Icu_aInt_Counter[].
- Icu_CapturedActivePulseWidth.
- IcuPeriod.

**ICU_EXCLUSIVE_AREA_26** Used in Icu_Siul2_SetActivationCondition function to protect the updates to:

- IFEER0 register.
- IREER0 register.

**ICU_EXCLUSIVE_AREA_27** Used in Icu_Siul2_DisableInterrupt function to protect the updates to:

- DIRER0 register.

**ICU_EXCLUSIVE_AREA_28** Used in Icu_Siul2_EnableInterrupt function to protect the updates to:

- DIRER0 register.

### Critical Region Exclusive Matrix

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

## 5.2 Peripheral Hardware Requirements

EMIOS channels 0-31 for EMIOS 0, EMIOS1, EMIOS2 external interrupts IRQ0–IRQ31 and WKPU channel WKPU0-WKPU31 are available for the ICU driver.

**Refer Table ICU Hardware Channel availability for MPC574XG 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. eMIOS 0 interrupts**

| eMIOS 0 Interrupts | Hardware interrupt vector |
|---|---|
| EMIOS_0_CH_0_CH_1_ISR | 706 |
| EMIOS_0_CH_2_CH_3_ISR | 707 |
| EMIOS_0_CH_4_CH_5_ISR | 708 |
| EMIOS_0_CH_6_CH_7_ISR | 709 |
| EMIOS_0_CH_8_CH_9_ISR | 710 |
| EMIOS_0_CH_10_CH_11_ISR | 711 |

*Table continues on the next page...*

### Table 5-1. eMIOS 0 interrupts (continued)

| eMIOS 0 Interrupts | Hardware interrupt vector |
|---|---|
| EMIOS_0_CH_12_CH_13_ISR | 712 |
| EMIOS_0_CH_14_CH_15_ISR | 713 |
| EMIOS_0_CH_16_CH_17_ISR | 714 |
| EMIOS_0_CH_18_CH_19_ISR | 715 |
| EMIOS_0_CH_20_CH_21_ISR | 716 |
| EMIOS_0_CH_22_CH_23_ISR | 717 |
| EMIOS_0_CH_24_CH_25_ISR | 718 |
| EMIOS_0_CH_26_CH_27_ISR | 719 |
| EMIOS_0_CH_28_CH_29_ISR | 720 |
| EMIOS_0_CH_30_CH_31_ISR | 721 |

### Table 5-2. eMIOS 1 interrupts

| eMIOS 1 Interrupts | Hardware interrupt vector |
|---|---|
| EMIOS_1_CH_0_CH_1_ISR | 722 |
| EMIOS_1_CH_2_CH_3_ISR | 723 |
| EMIOS_1_CH_4_CH_5_ISR | 724 |
| EMIOS_1_CH_6_CH_7_ISR | 725 |
| EMIOS_1_CH_8_CH_9_ISR | 726 |
| EMIOS_1_CH_10_CH_11_ISR | 727 |
| EMIOS_1_CH_12_CH_13_ISR | 728 |
| EMIOS_1_CH_14_CH_15_ISR | 729 |
| EMIOS_1_CH_16_CH_17_ISR | 730 |
| EMIOS_1_CH_18_CH_19_ISR | 731 |
| EMIOS_1_CH_20_CH_21_ISR | 732 |
| EMIOS_1_CH_22_CH_23_ISR | 733 |
| EMIOS_1_CH_24_CH_25_ISR | 734 |
| EMIOS_1_CH_26_CH_27_ISR | 735 |
| EMIOS_1_CH_28_CH_28_ISR | 736 |
| EMIOS_1_CH_30_CH_31_ISR | 737 |

### Table 5-3. eMIOS 2 interrupts

| eMIOS 2 Interrupts | Hardware interrupt vector |
|---|---|
| EMIOS_2_CH_0_CH_1_ISR | 738 |
| EMIOS_2_CH_2_CH_3_ISR | 739 |
| EMIOS_2_CH_4_CH_5_ISR | 740 |
| EMIOS_2_CH_6_CH_7_ISR | 741 |
| EMIOS_2_CH_8_CH_9_ISR | 742 |

*Table continues on the next page...*

Table 5-3. eMIOS 2 interrupts (continued)

| eMIOS 2 Interrupts | Hardware interrupt vector |
|---|---|
| EMIOS_2_CH_10_CH_11_ISR | 743 |
| EMIOS_2_CH_12_CH_13_ISR | 744 |
| EMIOS_2_CH_14_CH_15_ISR | 745 |
| EMIOS_2_CH_16_CH_17_ISR | 746 |
| EMIOS_2_CH_18_CH_19_ISR | 747 |
| EMIOS_2_CH_20_CH_21_ISR | 748 |
| EMIOS_2_CH_22_CH_23_ISR | 749 |
| EMIOS_2_CH_24_CH_25_ISR | 750 |
| EMIOS_2_CH_26_CH_27_ISR | 751 |
| EMIOS_2_CH_28_CH_28_ISR | 752 |
| EMIOS_2_CH_30_CH_31_ISR | 753 |

Table 5-4. External interrupts

| SIUL IRQ Interrupts | Hardware interrupt vector |
|---|---|
| SIUL2_EXT_IRQ_0_7_ISR | 243 |
| SIUL2_EXT_IRQ_8_15_ISR | 244 |
| SIUL2_EXT_IRQ_16_23_ISR | 245 |
| SIUL2_EXT_IRQ_24_31_ISR | 246 |

Table 5-5. Wakeup Unit interrupts

| WKPU IRQ Interrupts | Hardware interrupt vector |
|---|---|
| WKPU_EXT_IRQ_0_7_ISR | 668 |
| WKPU_EXT_IRQ_8_15_ISR | 669 |
| WKPU_EXT_IRQ_16_23_ISR | 670 |
| WKPU_EXT_IRQ_24_31_ISR | 671 |

## 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.

### 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.

### Configuration dependency to other module:

For generating configuration files of ICU, EcuM 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 are 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)

Configuration of MASTER BUS for ICU and dependency with PWM:

If an ICU channel and a PWM channel are configured on the same eMIOS module, the user shall insure that the PWM module configures a different MASTER BUS channel and is not overwriting the MASTER BUS used by the ICU channel. For more information see ICU and PWM User Manuals.

# Chapter 6
# Main API Requirements

## 6.1 Main functions calls within BSW scheduler

None

## 6.2 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 Icu_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_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 |

*Table continues on the next page...*

Table 7-1. Sectionto be define (continued)

| | | |
|---|---|---|
| **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 |
| | IcueMiosChannelRef | VariantPC or VariantPB | Post Build |
| | IcuSiul2ChannelRef | VariantPC or VariantPB | Post Build |
| | IcuWkpuChannelRef | VariantPC or VariantPB | Post Build |
| | IcuDMAChannelEnable | VariantPC or VariantPB | Post Build |
| | IcuDMAChannelReference | VariantPC or VariantPB | Post Build |
| | IcuDefaultStartEdge | VariantPC or VariantPB | Post Build |
| | IcuMeasurementMode | VariantPC or VariantPB | Post Build |
| | IcuOverflowNotification | VariantPC or VariantPB | Post Build |
| | IcuLockableChannel | 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 |
| | | | |
| IcuConfigSet/IcuChannel/IcuTimestampMeasurement | IcuTimestampMeasurementProperty | 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 |
|---|---|---|---|
| | IcuTimestampNotification | VariantPC or VariantPB | Post Build |
| | | | |
| IcuConfigSet/IcuChannel/IcuWakeup | IcuChannelWakeupInfo | VariantPC or VariantPB | Post Build |
| | | | |
| IcuConfigSet/IcueMios | IcueMiosModule | VariantPC or VariantPB | Post Build |
| | | | |
| IcuConfigSet/IcueMios/IcueMiosChannels | IcueMiosChannel | VariantPC or VariantPB | Post Build |
| | IcuEmiosFreeze | VariantPC or VariantPB | Post Build |
| | IcuEmiosPrescaler | VariantPC or VariantPB | Post Build |
| | IcuEmiosPrescaler_Alternate | VariantPC or VariantPB | Post Build |
| | IcuEmiosDigitalFilter | VariantPC or VariantPB | Post Build |
| | IcuEmiosBusSelect | VariantPC or VariantPB | Post Build |
| | IcuUserModeForDutycycle | VariantPC or VariantPB | Post Build |
| | IcuSignalMeasureWithoutInterrupt | VariantPC or VariantPB | Post Build |
| | | | |
| IcuConfigSet/IcueMios/IcueMiosMasterBus | eMiosMasterBus | VariantPC or VariantPB | Post Build |
| | MasterBusPrescaler | VariantPC or VariantPB | Post Build |
| | MasterBusPrescaler_Alternate | VariantPC or VariantPB | Post Build |
| | | | |
| IcuConfigSet/IcuSiul2 | IcuEXT_ISR_InterruptFilterClockPrescaler | VariantPC or VariantPB | Post Build |
| | IcuEXT_ISR_AlternateInterruptFilterClockPrescaler | VariantPC or VariantPB | Post Build |
| | | | |
| IcuConfigSet/IcuSiu/IcuSiul2Channels | IcuSiul2Channel | VariantPC or VariantPB | Post Build |
| | Icu_EXT_ISR_IFERDigitalFilter | VariantPC or VariantPB | Post Build |
| | Icu_EXT_ISR_IFMCDigitalFilter | VariantPC or VariantPB | Post Build |
| | | | |
| IcuConfigSet/IcuWkpu | IcuWkpuChannel | VariantPC or VariantPB | Post Build |
| | Icu_EXT_ISR_WIFERDigitalFilter | VariantPC or VariantPB | Post Build |
| | IcuWKPU_ISR_WIPUER | VariantPC or VariantPB | Post Build |
| | | | |
| IcuGeneral | IcuDevErrorDetect | Pre Compile parameter for all Variants of Configuration | Pre Compile |
| | IcuIndex | VariantPC or VariantPB | Post Build |
| | 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 |

*Table continues on the next page...*

## Table 8-1. Configuration Parameters (continued)

| Configuration Container | Configuration Parameters | Configuration Variant | Current Implementation |
|---|---|---|---|
| | IcuRegisterLockingMode | Pre Compile parameter for all Variants of Configuration | Pre Compile |
| | IcuGetCaptureRegisterValueApi | 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 |
| | 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 |

*Table continues on the next page...*

### Table 8-1. Configuration Parameters (continued)

| Configuration Container | Configuration Parameters | Configuration Variant | Current Implementation |
|---|---|---|---|
| | 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
# External Module Assumptions

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

*[ICU_EXT002]*

<< The external application shall not rely on ICU measurements if ICU related interrupts are disabled. >>

**NOTE**

ICU implementation relies on HW interrupts. If interrupts are disabled, calculations driven by ISRs events will not be performed. See the integration manual for ICU for relevant IRQs.

*[ICU_EXT001]*

<< 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.

*[ICU_EXT003]*

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

*[ICU_EXT004]*

<< 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.

*[ICU_EXT005]*

<< 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.

*[ICU_EXT006]*

<< 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.

*[ICU149]*

<< 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;

*[ICU117]*

<< Values for production code event ID’s are assigned externally by the configuration of Diagnostic Event Manager. >>

*[ICU263]*

<< Values for production code are published in the file Dem_IntErrId.h and included via Dem.h. >>

**NOTE**

Serr.h shall include the Dem.h. All production errors are reported via Serr

*[ICU190]*

<< Dem_EventIdType shall be imported from Dem_Types.h. >>

**NOTE**

Serr.h shall include the Dem.h. All production errors are reported via Serr

*[ICU275]*

<< Std_VersionInfoType shall be imported from Std_Types.h. >>

*[ICU276]*

<< EcuM_WakeupSourceType shall be imported from EcuM_Types.h

---

Module | Imported Type

---

Dem | Dem_EventIdType

| Dem_EventStatusType

EcuM | EcuM_WakeupSourceType

Std_Types | Std_ReturnType

| Std_VersionInfoType >>

***[ICU052]***

<< 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

***[ICU053]***

<< 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

***[ICU128]***

<< 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

***[ICU129]***

<< 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

***[ICU152]***

<< The Icu module’s environment shall not call Icu_DeInit during a running operation (e. g. timestamp measurement or edge counting) >>

***[ICU221]***

<< A re-initialization of the ICU module by executing the Icu_Init() function requires a de-initialization before by executing the Icu_DeInit() function >>

*[ICU133]*

<< 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. ICU022 and ICU048 apply to the function Icu_SetMode.

>>

*[ICU361]*

<< 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

*[ICU348]*

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

*[ICU349]*

<< 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

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 μVision 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.

Document Number IM35ICUASR4.2 Rev0002RTM1.0.0
Revision 5.0.0