

ICU driver user guide

TRAVEO™ T2G family

About this document

Scope and purpose

This guide describes the architecture, configuration, and use of the input capture unit (ICU) driver. This document explains the functionality of the driver and provides a reference for the driver's API.

The installation, build process, and general information to use EB tresos Studio software are not within the scope of this document.

Intended audience

This document is intended for anyone who uses the ICU driver of the TRAVEO™ T2G family.

Document structure

Chapter [1 General overview](#) gives a brief introduction to the ICU driver, explains the embedding in the AUTOSAR environment, and describes the supporting hardware and development environment.

Chapter [2 Using the ICU driver](#) details the steps on how to use the ICU driver in your application.

Chapter [3 Structure and dependencies](#) describes the file structure and the dependencies for the ICU driver.

Chapter [4 EB tresos Studio configuration interface](#) describes the driver's configuration.

Chapter [5 Functional description](#) gives a functional description of the services offered by the ICU driver.

Chapter [6 Hardware resources](#) gives a description of the hardware resources.

The [Appendix A](#) and [Appendix B](#) provides a complete API reference and access register table.

Abbreviations and definitions

Abbreviation	Definition
API	Application Programming Interface
ASIL	Automotive Safety Integrity Level
AUTOSAR	Automotive Open System Architecture
BSW	Basic Software. Standardized part of software which does not fulfill a vehicle functional job.
DEM	Diagnostic Event Manager
DET	Default Error Tracer
DMA	Direct Memory Access
DW	Data Wire, a CPU feature. DW is used for peripheral-to-memory and memory-to-peripheral data transfers. DW is also called Peripheral-DMA (P-DMA) controller. Generically, this feature is called "DMA".
EcuM	ECU State Manager
EB tresos Studio	Elektrobit Automotive configuration framework

About this document

Abbreviation	Definition
GCE	Generic Configuration Editor
ICU	Input Capture Unit
IRQ	Interrupt Request
ISR	Interrupt Service Routine
MCU	Micro Controller Unit
MCAL	Microcontroller Abstraction Layer
MPU	Memory Protection Unit
OS	Operating System
TCPWM	Timer Counter Pulse Width Modulation
Tick	Defines the timer resolution, the duration of a timer increment

Related documents**AUTOSAR requirements and specifications****Bibliography**

- [1] General specification of basic software modules, AUTOSAR release 4.2.2.
- [2] Specification of ICU driver, AUTOSAR release 4.2.2.
- [3] Specification of standard types, AUTOSAR release 4.2.2.
- [4] Specification of ECU configuration parameters, AUTOSAR release 4.2.2.
- [5] Specification of default error tracer, AUTOSAR release 4.2.2.
- [6] Specification of diagnostic event manager, AUTOSAR release 4.2.2.

Elektrobit automotive documentation**Bibliography**

- [7] EB tresos Studio for ACG8 user's guide.

Hardware documentation**Bibliography**

The hardware documents are listed in the delivery notes.

Related standards and norms**Bibliography**

- [8] Layered software architecture, AUTOSAR release 4.2.2.

Table of contents

About this document.....	1
Table of contents.....	3
1 General overview	6
1.1 Introduction to the ICU driver.....	6
1.2 User profile	6
1.3 Embedding in the AUTOSAR environment.....	6
1.4 Supported hardware	7
1.5 Development environment.....	7
1.6 Character set and encoding.....	7
2 Using the ICU driver.....	8
2.1 Installation and prerequisites.....	8
2.2 Configuring the ICU driver	8
2.2.1 Architecture details.....	9
2.3 Adapting an application.....	10
2.4 Starting the build process.....	12
2.5 Measuring stack consumption.....	12
2.6 Memory mapping	12
2.6.1 Memory allocation keyword	12
2.6.2 Restriction on memory allocation.....	13
3 Structure and dependencies.....	14
3.1 Static files	14
3.2 Configuration files	14
3.3 Generated files	14
3.4 Dependencies	15
3.4.1 MCU driver	15
3.4.2 PORT driver	15
3.4.3 ECU state manager.....	15
3.4.4 AUTOSAR OS.....	15
3.4.5 DET.....	15
3.4.6 DEM.....	15
3.4.7 BSW scheduler.....	15
3.4.8 Error callout handler	16
4 EB tresos Studio configuration interface.....	17
4.1 General configuration	17
4.2 ICU DEM event parameter references.....	17
4.3 Configuration of optional API services	17
4.4 ICU configuration set.....	18
4.4.1 ICU channel configuration	18
4.4.1.1 ICU signal edge detection configuration.....	21
4.4.1.2 ICU signal measurement property configuration	21
4.4.1.3 ICU time stamp measurement configuration	21
4.4.1.4 ICU wakeup configuration	23
4.4.2 ICU channel group configuration	23
5 Functional description.....	25
5.1 Inclusion	25
5.2 Initialization.....	25
5.3 Runtime reconfiguration.....	25

Table of contents

5.4	ICU channel.....	25
5.5	Notification	25
5.6	Overflow notification	26
5.7	Wakeup	26
5.8	ICU mode	26
5.9	ICU input state.....	27
5.10	ICU results.....	27
5.11	Prescaler setting function	28
5.12	ICU channel group synchronous start	28
5.13	ICU channel group synchronous stop	29
5.14	DMA transfer	29
5.15	API parameter checking	30
5.16	Reentrancy.....	32
5.17	Configuration checking.....	32
5.18	Debugging support.....	32
5.19	Execution time dependencies	32
6	Hardware resources	33
6.1	Ports and pins.....	33
6.2	Timer	33
6.3	Interrupts.....	33
6.4	Triggers	34
7	Appendix A – API reference	36
7.1	Include files.....	36
7.2	Data types.....	36
7.2.1	Icu_ChannelType	36
7.2.2	Icu_ChannelStatusType.....	36
7.2.3	Icu_IndexType	36
7.2.4	Icu_GroupType.....	36
7.2.5	Icu_EdgeNumberType	36
7.2.6	Icu_ValueType.....	37
7.2.7	Icu_ClkFrequencyType	37
7.2.8	Icu_LevelType	37
7.2.9	Icu_ModeType.....	37
7.2.10	Icu_InputStateType	37
7.2.11	Icu_ActivationType	38
7.2.12	Icu_DutyCycleType	38
7.2.13	Icu_ConfigType	38
7.2.14	Icu_DriverStatusType	38
7.2.15	Icu_CheckChannelStatusType	39
7.3	Constants.....	39
7.3.1	Error codes	39
7.3.2	Version information	40
7.3.3	Module information	41
7.3.4	API service IDs	41
7.3.5	Channel status.....	42
7.3.6	Symbolic names	42
7.4	Functions	42
7.4.1	Icu_Init.....	42
7.4.2	Icu_DeInit	43
7.4.3	Icu_SetMode.....	44

Table of contents

7.4.4	Icu_DisableWakeup.....	45
7.4.5	Icu_EnableWakeup	45
7.4.6	Icu_SetActivationCondition.....	46
7.4.7	Icu_DisableNotification	47
7.4.8	Icu_EnableNotification	48
7.4.9	Icu_GetInputState	48
7.4.10	Icu_StartTimestamp	49
7.4.11	Icu_StopTimestamp.....	50
7.4.12	Icu_GetTimestampIndex	51
7.4.13	Icu_ResetEdgeCount.....	51
7.4.14	Icu_EnableEdgeCount	52
7.4.15	Icu_DisableEdgeCount.....	53
7.4.16	Icu_GetEdgeNumbers	53
7.4.16.1	Icu_GetTimeElapsed.....	54
7.4.17	Icu_GetDutyCycleValues	55
7.4.18	Icu_GetVersionInfo.....	56
7.4.18.1	Icu_StartSignalMeasurement.....	56
7.4.19	Icu_StopSignalMeasurement	57
7.4.20	Icu_CheckWakeup.....	58
7.4.21	Icu_EnableEdgeDetection	58
7.4.22	Icu_DisableEdgeDetection.....	59
7.4.23	Icu_DisableOverflowNotification	60
7.4.24	Icu_EnableOverflowNotification	60
7.4.25	Icu_SetPrescaler	61
7.4.26	Icu_StopGroupTrigger	62
7.4.27	Icu_StartGroupTrigger.....	63
7.4.28	Icu_GetInputLevel.....	63
7.4.29	Icu_CheckChannelStatus.....	64
7.5	Required callback functions	65
7.5.1	DET	65
7.5.1.1	Det_ReportError.....	65
7.5.2	DEM.....	66
7.5.2.1	Dem_ReportErrorStatus	66
7.5.3	Callout functions.....	66
7.5.3.1	Error callout API	66
8	Appendix B – Access register table.....	68
8.1	TCPWM.....	68
8.2	GPIO	72
8.3	DW_CH_STRUCT/DW_DESCR_STRUCT	73
	Revision history.....	75

1 General overview

1.1 Introduction to the ICU driver

The input capture unit (ICU) driver is a set of software routines, which enables you to capture input signals on special input pins of the TRAVEO™ T2G family microcontroller.

Therefore, the ICU driver provides services to count or measure external events. Additionally, it provides services to initialize and de-initialize the ICU driver and to enable or disable a notification callback function for an interrupt. Furthermore, the wakeup capability of a channel can be enabled or disabled. It also provides a service for returning a channel status.

The ICU driver is not responsible for initializing or configuring hardware ports. This is done by the PORT driver.

The driver conforms to the AUTOSAR standard and is implemented according to the AUTOSAR *specification of ICU driver* [2].

The ICU driver is delivered with a plugin for the EB tresos Studio software, which allows you to statically configure the driver. The driver provides an interface to enable ICU channels and to configure the parameters.

1.2 User profile

This guide is intended for users with a basic knowledge of the following:

- Embedded systems
- C programming language
- AUTOSAR standard
- Target hardware architecture

1.3 Embedding in the AUTOSAR environment

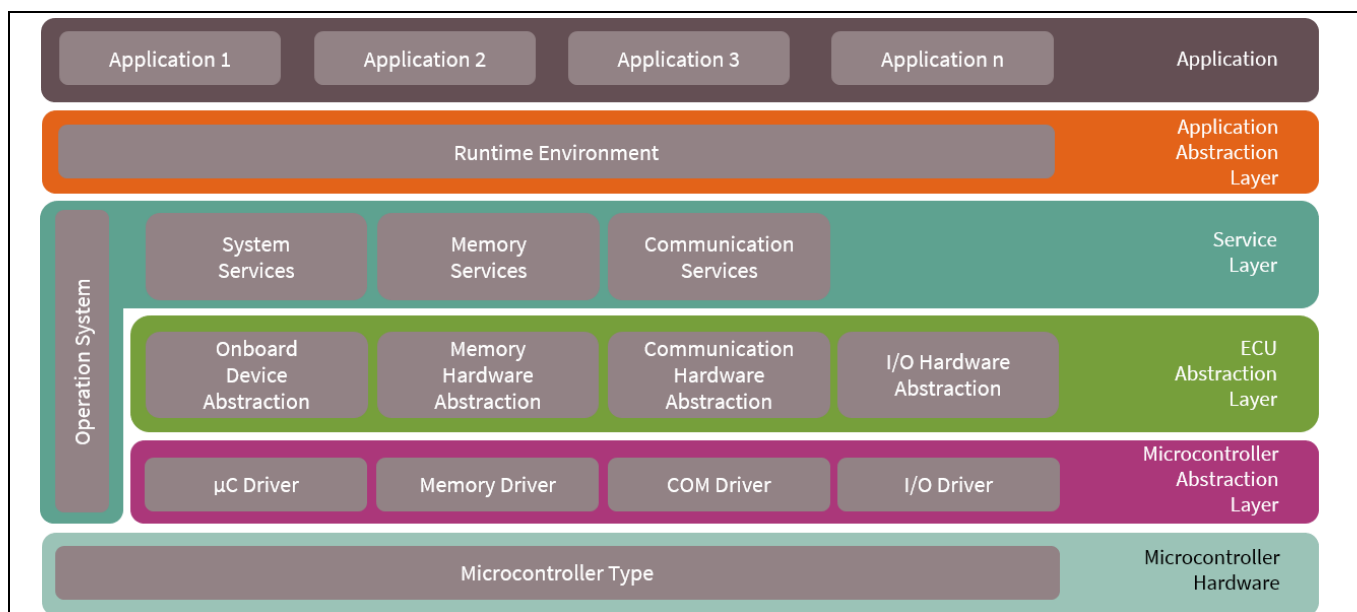


Figure 1 Overview of AUTOSAR software layers

Figure 1 shows the layered AUTOSAR software architecture. The ICU driver (**Figure 2**) is part of the microcontroller abstraction layer (MCAL), the lowest layer of basic software in the AUTOSAR environment.

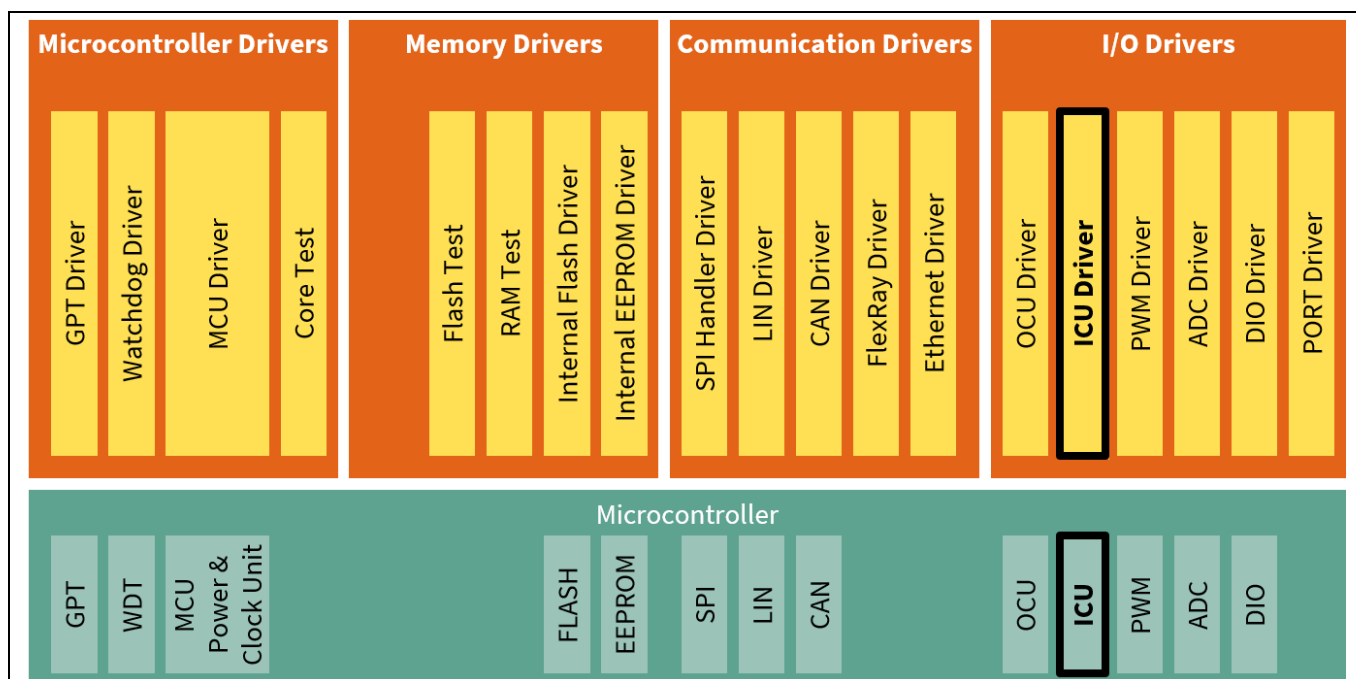


Figure 2 ICU driver in MCAL layer

1.4 Supported hardware

This version of the ICU driver supports TRAVEO™ T2G family microcontroller. The supported derivatives are listed in the release notes. No further special external hardware devices are required.

The ICU captures the counter values of the timer counter pulse width modulation (TCPWM). The 16-bit TCPWM includes 16-bit timer channels using a peripheral clock as clock source. The maximum measurable time is $2^{16} - 1 = 0xFFFF$. The 32-bit TCPWM includes 32-bit timer channels using a peripheral clock as clock source. The maximum measurable time is $2^{32} - 1 = 0xFFFFFFFF$.

External interrupts are also supported to be used for edge counting and as wakeup source. The `IcuSignalMeasurement` and `IcuTimestamp` modes are not supported by the external interrupt channels.

1.5 Development environment

The development environment corresponds to AUTOSAR release 4.2.2 [2]. The BASE, MAKE, MCU, PORT, and RESOURCE modules are required for the proper functionality of the ICU driver.

1.6 Character set and encoding

All source code files of the ICU driver are restricted to the ASCII character set. The files are encoded in UTF-8 format, with 7-bit subset (values 0x00 ... 0x7F).

2 Using the ICU driver

This chapter describes all the necessary steps to incorporate the ICU driver into your application.

2.1 Installation and prerequisites

Note: Before you start, see the *EB tresos Studio for ACG8 user's guide* [7] for the following information.

1. The installation procedure of EB tresos ECU AUTOSAR components
2. The usage of the EB tresos Studio
3. The usage of the EB tresos ECU AUTOSAR build environment (It includes the steps to setup and integrate the own application within the EB tresos ECU AUTOSAR build environment)

The installation of the ICU driver complies with the general installation procedure for EB tresos ECU AUTOSAR components given in the document mentioned above. If the driver has successfully installed, the driver will appear in the module list of the EB tresos Studio (see *EB tresos Studio for ACG8 user's guide* [7]).

This document assumes that the project is properly set up and is using the application template as described in the *EB tresos Studio for ACG8 user's guide* [7]. This template provides the necessary folder structure, project and makefiles needed to configure and compile your application within the build environment. You must be familiar with the use of the command shell.

2.2 Configuring the ICU driver

This section gives a brief overview about the configuration structure defined by AUTOSAR to use the ICU driver.

The following basic containers are used to configure common behavior:

- `IcuOptionalApis`: This container is mainly used to restrict or extend the API of the ICU driver.
- `IcuGeneral`: This container is mainly used to enable or disable default error tracer (DET) and to enable or disable wakeup reporting.

For detailed information and description, see chapter 4 **EB tresos Studio configuration interface**.

- `IcuGeneral`: This container is mainly used to enable or disable DET and to enable or disable wakeup reporting.

The `IcuChannel` container groups configured channels. Each `IcuChannel` has a parameter:

- `IcuChannelId`: Assigns the logical number to the ICU channel.
- `IcuDefaultStartEdge`: Defines the start condition (pin level change) of the external event to start the measurement (can be falling, rising, or both edges).
- `IcuMeasurementMode`: Selects the channel's used mode (edge count, signal edge detection, signal measurement, and timestamp).
- `IcuWakeupCapability`: Enables or disables the wakeup capability of the channel.
- `IcuWakeup`: Specifies the reference to the `EcuM` wakeup reason if the ICU channel is configured. It will be reported to `EcuM`.

The `IcuChannel` container can be configured for the different ICU channels.

Each measurement mode needs different additional configuration parameters. The driver gets the information based on the selected `IcuMeasurementMode`:

- `IcuEdgeCounterMeasurement` has no additional parameter. The channel default start edge is used to detect ICU port pin signal changes.
- `IcuSignalEdgeDetection` detects a single level change on the port pin and calls a user-specific callback notification function specified by the `IcuSignalNotification` parameter.
- `IcuSignalMeasurement` has a property to select the time measurement behavior of the signal. It can be duty cycle, high time, low time, active time, or period time.
- `IcuTimestampMeasurement` has two parameters. The first `IcuTimestampMeasurementProperty` describes the handling of the timestamp buffer. The buffer and the size will be given at runtime by the appropriate API call. The buffer can be used as circular or linear storage. After the number of requested timestamps are acquired a notification function `IcuTimestampNotification` will be called.

For each channel that is configured, an interrupt service routine (ISR) is needed. The ISR must be called `Icu_Isr_Vector_<IRQ No>_Cat1` for Category-1 ISR or `Icu_Isr_Vector_<IRQ No>_Cat2` for Category-2 ISR. The priority of this ISR determines the interrupt priority of the ICU channel.

2.2.1 Architecture details

- `IcuErrorCalloutFunction`: Specifies the error callout handler which is called when errors are detected during runtime.
- `IcuIncludeFile`: Specifies the filename that is used to include some definitions (such as the declaration for the error callout handler).
- `IcuChannelBufferName`: Specifies the name of data array used for the channel's timestamp buffer. When `Icu_StartTimestamp()` is called, it confirms whether `BUFFERPTR` has a value in the range specified by `IcuChannelBufferName` and `IcuChannelBufferSize`. If `IcuChannelBufferName` is set to "NULL", this confirmation is not carried out.
- `IcuChannelBufferSize`: Specifies the length of data array used for the channel's timestamp buffer. When `Icu_StartTimestamp()` is called, it confirms whether `BufferPtr` has a value in the range specified by `IcuChannelBufferName` and `IcuChannelBufferSize`.
- `IcuUseDma`: Enables or disables the DMA function for `IcuTimestamp` mode.
- `IcuDmaChannel`: Specifies the input trigger from TCPWM to the DMA channel to initiate a timestamp data transfer.
- `IcuDmaErrorNotification`: Specifies the callback function name for DMA error.
- `IcuSafetyFunctionApi`: Adds or removes the `Icu_CheckChannelStatus()` service to or from the code.
- `IcuSetPrescalerApi`: Adds or removes the service `Icu_SetPrescaler()` to or from the code.
- `IcuGetInputLevelApi`: Adds or removes the service `Icu_GetInputLevel()` to or from the code.
- `IcuChannelGroupApi`: Adds or removes the service `Icu_StartGroupTrigger()` and `Icu_StopGroupTrigger()` to or from the code.
- `IcuEnableNotiApiCapableWakeup`: Specifies whether `Icu_EnableNotification()` will enable the associated notification for both interrupt and wakeup display AUTOSAR standard behavior.
- `IcuDisableNotiApiCapableWakeup`: Specifies whether `Icu_DisableNotification()` will disable the associated notification for both interrupt and wakeup or display AUTOSAR standard behavior.
- `IcuWakeupAcceptanceInSetMode`: Specifies the acceptance of wakeup signal during Sleep transition processing with `Icu_SetMode()`.
- `IcuResource`: Selects the hardware resource to be used for the logical channel.
- `IcuNoiseFilterEnable`: If GPIO resource is used, it enables or disables the noise filter function.
- `IcuOverflowNotification`: Specifies the notification function when the related timer overflows.

- `IcuChannelClkSrcRef`: Specifies the reference to `McuClockReferencePoint` from which the channel clock is derived.
- `IcuChannelTickFrequency`: Specifies the tick frequency of the timer in hertz that is used for `IcuSignalMeasurement` and `IcuTimestamp` modes.
- `IcuInputTriggerSelection`: Specifies the input trigger only when TCPWM resource is selected. The input trigger is used as the input signal bound to one or more TCPWM resources.
- `IcuEnableDebug`: Enables or disables the debug capability to stop a timer channel when processor is in debug mode.
- `IcuDisableEcumWakeupNotification`: Specifies whether to call `Ecum_CheckWakeup()` from the ICU interrupt function for this channel.
- `IcuChannelGroupId`: Specifies the group Id of the ICU channel group.
- `IcuChannelGroupStartTrigger`: Specifies the input trigger to start the ICU channel group synchronously.
- `IcuChannelGroupStopTrigger`: Specifies the input trigger to stop the ICU channel group synchronously.
- `IcuChannelRef`: Specifies the assignment of ICU channels to ICU channel group.
- `IcuDemEventParameterRefs`: Configures DEM event notification settings.
- `ICU_E_HARDWARE_ERROR`: Specifies a reference to report hardware failures by DEM events.

2.3 Adapting an application

To use the ICU driver in your application, include ICU, MCU, and PORT driver header files by adding the following lines of code in your source file:

```
#include "Mcu.h"    /* AUTOSAR MCU Driver */
#include "Port.h"   /* AUTOSAR PORT Driver */
#include "Icu.h"    /* AUTOSAR ICU Driver */
```

These publish all required function and data prototypes and symbolic names of the configuration into the application. In addition, you must implement the error callout function for ASIL safety extension.

Declare the error callout function in the specified file with the `IcuIncludeFile` parameter and implement it in your application (see the Error Callout API in the [\(7.5 Required callback functions\)](#) section).

The error callout function name can be configured by the `IcuErrorCalloutFunction` parameter.

To use the ICU driver, the appropriate port pins and ICU channel interrupts must be configured in the PORT driver and OS. For detailed information see chapter [6 Hardware resources](#).

Initialization of MCU, PORT, and ICU driver needs to be done in the following order:

```
Mcu_Init(&Mcu_Config[0]);
Port_Init(&Port_Config[0]);
Icu_Init(&Icu_Config[0]);
```

The function `Mcu_Init()`, `Port_Init()`, and `Icu_Init()` are called with the configuration data pointer. The `Init` functions are called, the ICU functions can be used.

The following is a short sample for a channel with name `MY_TIMESTAMP_CHANNEL` configured as `IcuTimestamp` mode:

```
Icu_ValueType myTimeStampBuffer[10]; /* buffer for 10 timestamp values */
volatile boolean notifyCalled = FALSE; /* has to be set in the notify function */
uint8 i;

/* notify interval > 0 configures the notify generation after 5 timestamps */
Icu_StartTimestamp(IcuConf_IcuChannel_MY_TIMESTAMP_CHANNEL, myTimeStampBuffer, 10,
5);

/* notification must be enabled explicit */
Icu_EnableNotification(IcuConf_IcuChannel_MY_TIMESTAMP_CHANNEL);

/* wait until notify was called */
while(!notifyCalled);

/* results are valid (can be used) after stopping the channel */
Icu_StopTimestamp(IcuConf_IcuChannel_MY_TIMESTAMP_CHANNEL);

/* do something with the timestamps */
for(i=0; i<5; i++)
{
    myTimeStampBuffer[i] = myTimeStampBuffer[i];
}
```

Example

The usage as linear/circular buffer is statically configured and cannot be changed at runtime. For more information on API functions and data types, see section [Appendix A – API reference](#).

Note: *The `IcuTimestamp` and `IcuSignalMeasurement` modes require a running counter to measure the time between two signal edges. It is needed to initialize and start a GPT timer for that purpose.*

Your application must provide the notification functions and its declarations that you configured. The file containing the declarations must be included using the `IcuGeneral/IcuIncludeFile` parameter. The notification function takes no parameters and have void return type:

```
void MyNotificationFunction(void)
{
    /* Insert your code here */
}
```

The notification function is called from an interrupt context.

2.4 Starting the build process

Do the following to build your application:

Note: For a clean build, use the build command with target `clean_all`. before (`make clean_all`).

1. On the command shell, type the following command to generate the necessary configuration-dependent files. See [3.3 Generated files](#).

```
> make generate
```

2. Type the following command to resolve required file dependencies:

```
> make depend
```

3. Type the following command to compile and link the application:

```
> make (optional target: all)
```

The application is now built. All files are compiled and linked to a binary file which can be downloaded to the target hardware.

2.5 Measuring stack consumption

Do the following to measure stack consumption. It requires the Base module for proper measurement.

Note: All files (including library files) should be rebuilt with the dedicated compiler option. The executable file built in this step must be used only to measure stack consumption.

1. Add the following compiler option to the Makefile to enable stack consumption measurement.

```
-DSTACK_ANALYSIS_ENABLE
```

2. Type the following command to clean library files.

```
> make clean_lib
```

3. Follow the build process described in [2.4 Starting the build process](#).
4. Follow the instructions in the release notes and measure the stack consumption.

2.6 Memory mapping

The `Icu_MemMap.h` file in the `$(TRESOS_BASE)/plugins/MemMap_TS_T40D13M0I0R0/include` directory is a sample. This file is replaced by the file generated by MEMMAP module. Input to MEMMAP module is generated as `Icu_Bswmd.arxml` in the `$(PROJECT_ROOT)/output/generated/swcd` directory of your project folder.

2.6.1 Memory allocation keyword

- `ICU_START_SEC_CODE_ASIL_B / ICU_STOP_SEC_CODE_ASIL_B`

The memory section type is CODE. All executable code is allocated in this section.

- `ICU_START_SEC_CONST_ASIL_B_UNSPECIFIED / ICU_STOP_SEC_CONST_ASIL_B_UNSPECIFIED`

The memory section type is CONST. ICU whole configuration setting is allocated in this section:

- ICU configuration setting
- `ICU_START_SEC_VAR_INIT_ASIL_B_UNSPECIFIED / ICU_STOP_SEC_VAR_INIT_ASIL_B_UNSPECIFIED`

The memory section type is VAR. The following variables are allocated in this section:

- Pointer to the whole configuration setting
- Information for ICU driver state
- ICU_START_SEC_VAR_NO_INIT_ASIL_B_UNSPECIFIED /
ICU_STOP_SEC_VAR_NO_INIT_ASIL_B_UNSPECIFIED

The memory section type is VAR. The following variables are allocated in this section:

- Information for target ICU channel and driver state of `IcuTimestamp` mode
- Information for target ICU channel and driver state of `IcuSignalEdgeDetect` mode
- Information for target ICU channel and driver state of `IcuEdgeCounter` mode.
- Information for target ICU channel and driver state of `IcuSignalMeasurement` mode.
- Information for target ICU channel and driver state of common state data
- Information for target ICU channel and driver state of DMA.
- ICU DMA descriptors.

2.6.2 Restriction on memory allocation

The CPU has an individual cache that is not shared with the DMA bus master. Therefore, it must be ensured that data related to DMA are in specific region where can be accessible by the DMA. Besides some sections need to be allocated in specific memory region. This driver does not support the use of data related to DMA placed in CPU's tightly coupled memories (TCMs).

- The section that contains timestamp buffers
 - When using DMA:

The section shall be allocated to a user-specific memory region configured by the CPU's memory protection unit (MPU) as non-cache-able.

- When not using DMA:

There is no restriction.

- The section surrounded by ICU_START_SEC_VAR_NO_INIT_ASIL_B_UNSPECIFIED /
ICU_STOP_SEC_VAR_NO_INIT_ASIL_B_UNSPECIFIED
 - When using DMA:

The section shall be allocated to a user-specific memory region configured by the CPU's memory protection unit (MPU) as write-through or non-cache-able.

- When not using DMA:

There is no restriction on memory allocation.

Note: *This restriction is only applied to Arm® Cortex®-M7 devices because they include TCM and inner cache. There is no restriction in case of using Arm® Cortex®-M4 devices.*

3 Structure and dependencies

ICU driver consists of static, configuration, and generated files.

3.1 Static files

- $\$(PLUGIN_PATH)=\$(TRESOS_BASE)/plugins/Icu_TS_*$ is the path to the ICU driver plugin.
- $\$(PLUGIN_PATH)/lib_src$ contains all static source files of the ICU driver. These files contain the functionality of the driver that does not depend on the current configuration. The files are grouped into a static library.
- $\$(PLUGIN_PATH)/lib_include$ contains all internal header files for the ICU driver.
- $\$(PLUGIN_PATH)/src$ comprises configuration dependent source files or special derivate files. Each file will be built again when the configuration is changed.

All necessary source files will be automatically compiled and linked during the build process and all include paths will be set if the ICU driver is enabled.

- $\$(PLUGIN_PATH)/include$ is the basic public include directory that is required and should include *Icu.h*.
- $\$(PLUGIN_PATH)/autosar$ directory contains the AUTOSAR ECU parameter definition with vendor, architecture and derivate specific adaptations to create a correct matching parameter configuration for the ICU driver.

3.2 Configuration files

The configuration of the ICU driver is done via EB tresos Studio. The file containing the ICU driver's configuration is named *Icu.xdm* and is in the directory $\$(PROJECT_ROOT)/config$. This file serves as input to generate the configuration dependent source and header files during the build process.

3.3 Generated files

During the build process, the following files are generated based on the current configuration. They are in the *output/generated* sub folder of your project folder.

- *include/Icu_Cfg.h* provides settings of the configurations with pre-compile attribute, for example, provides all symbolic names of the configuration. It will be included in *Icu.h*.
- *include/Icu_Cfg_Include.h* includes the header files specified by *IcuIncludeFile*.
- *include/Icu_Irq.h* declares interrupt service routine (ISR) function.
- *include/Icu_PBcfg.h* provides settings of configurations with post-build attribute, for example, symbolic names of module configurations. It will be included in *Icu.h*.
- *src/Icu_Irq.c* contains the ICU channel ISR implementation.
- *src/Icu_PBcfg.c* contains the constant structure for the ICU configuration.

Note: *Generated source files need not be added to your application make file. They will be compiled and linked automatically during the build process.*

- *swcd/Icu_Bswmd.arxml* contains BSW module description.

Note: *Additional steps are required for the generation of BSW module description. In EB tresos Studio, follow the menu path **Project > Build Project** and click **generate_swcd**.*

3.4 Dependencies

3.4.1 MCU driver

The MCU driver needs to be initialized and all MCU clock reference points referenced by the ICU driver channels via configuration parameter `IcuChannelClkSrcRef` must have been activated (via calls of MCU API functions) before initializing the ICU driver. See the MCU driver's user guide for details.

3.4.2 PORT driver

Although the ICU driver can successfully be compiled and linked without an AUTOSAR compliant PORT driver, the latter is required to configure and initialize all ports. Also, the configuration of triggers is required and it is necessary to call `Port_ActTrigger()` to issue a trigger signal after calling `Icu_StartGroupTrigger()/Icu_StopGroupTrigger()`, if parameter `IcuChannelGroupStartTrigger/IcuChannelGroupStopTrigger` is configured. If `Port_ActTrigger()` is not called, channels in the group are not started/stopped except for channels in `IcuSignalMeasurement` mode. The PORT driver needs to be initialized before the ICU driver is initialized. See the PORT driver's user guide for details.

3.4.3 ECU state manager

The ICU driver can be compiled and linked without an EcuM when `IcuReportWakeupSource` is disabled. The EcuM is needed if `IcuReportWakeupSource` is enabled or the ISR of a wakeup source services the wakeup event.

3.4.4 AUTOSAR OS

The OS must be used to configure and create the ISR vector table entries for the ICU used channels. See the hardware/derivate specific [6.3 Interrupts](#).

3.4.5 DET

If the development error detection is enabled in the ICU driver, the DET needs to be installed, configured, and integrated into the application.

This driver reports DET error codes as 'instance 0'.

3.4.6 DEM

If the DEM is enabled in the ICU module configuration, the DEM needs to be installed, configured, and integrated into the application.

To enable DEM to support the ICU driver, the `ICU_E_HARDWARE_ERROR` needs to be defined in the DEM configuration in the container `IcuDemEventParameterRefs`.

3.4.7 BSW scheduler

The ICU driver uses the following services of the BSW scheduler to enter and leave critical sections:

- `SchM_Enter_Icu_ICU_EXCLUSIVE_AREA_0(void)`
- `SchM_Exit_Icu_ICU_EXCLUSIVE_AREA_0(void)`

You must ensure that the BSW scheduler is properly configured and initialized before using the ICU driver.

3.4.8 Error callout handler

The error callout handler is called on every error that is detected, regardless of whether development error detection is enabled. The error callout handler is an ASIL safety extension that is not specified by AUTOSAR. It is configured via configuration parameter `IcuErrorCalloutFunction`.

4 EB tresos Studio configuration interface

The GUI is not part of the current delivery; see *EB tresos Studio for ACG8 user's guide* [7].

4.1 General configuration

The module comes preconfigured with default settings. These settings should be adapted when necessary.

- `IcuDevErrorDetect` enables or disables the development error notification for the ICU driver.

Setting this parameter to `FALSE` will disable the notification of development errors via DET. However, in contrast to AUTOSAR specification, detection of development errors is still enabled as safety mechanisms (fault detection).

- `IcuIndex` represents the ICU driver's ID so that it can be referenced by the upper layer.

Note: The logical number must be " ≥ 0 " and " ≤ 255 ".

- `IcuReportWakeupSource` enables or disables the wakeup source reporting.
- `IcuErrorCalloutFunction` specifies the error callout function name. The function is called on every error. The ASIL level of this function limits the ASIL level of the ICU driver.

Note: `IcuErrorCalloutFunction` must be a valid C function name; otherwise an error would occur in the configuration phase.

- `IcuIncludeFile` lists the filenames that will be included within the driver. Any application-specific symbol that is used by the ICU configuration (such as error callout function) should be included by configuring this parameter.

Note: `IcuIncludeFile` must be a unique filename with an extension `.h`; otherwise some errors would occur in the configuration phase.

4.2 ICU DEM event parameter references

This container has the following parameter to configure the DEM event notification settings.

- `ICU_E_HARDWARE_ERROR` is the reference to the configured DEM event to report hardware failure. If the reference is not configured the error will not be reported.

4.3 Configuration of optional API services

- `IcuDeInitApi` adds or removes the service `Icu_DeInit()` to or from the code.
- `IcuEnableWakeupApi` adds or removes the service `Icu_EnableWakeup()` to or from the code.
- `IcuDisableWakeupApi` adds or removes the service `Icu_DisableWakeup()` to or from the code.
- `IcuEdgeCountApi` adds or removes the services `Icu_ResetEdgeCount()`, `Icu_EnableEdgeCount()`, `Icu_DisableEdgeCount()`, and `Icu_GetEdgeNumbers()` to or from the code.
- `IcuEdgeDetectApi` adds or removes the services `Icu_EnableEdgeDetection()` and `Icu_DisableEdgeDetection()` to or from the code.
- `IcuGetDutyCycleValuesApi` adds or removes the service `Icu_GetDutyCycleValues()` to or from the code.
- `IcuGetInputStateApi` adds or removes the service `Icu_GetInputState()` to or from the code.
- `IcuGetTimeElapsedApi` adds or removes the service `Icu_GetTimeElapsed()` to or from the code.

- `IcuGetVersionInfoApi` adds or removes the service `Icu_GetVersionInfo()` to or from the code.
- `IcuSetModeApi` adds or removes the service `Icu_SetMode()` to or from the code.
- `IcuSignalMeasurementApi` adds or removes the services `Icu_StartSignalMeasurement()` and `Icu_StopSignalMeasurement()` to or from the code.
- `IcuTimestampApi` adds or removes the services `Icu_StartTimestamp()`, `Icu_StopTimestamp()`, and `Icu_GetTimestampIndex()` to or from the code.
- `IcuWakeupFunctionalityApi` adds or removes the service `Icu_CheckWakeup()` to or from the code.
- `IcuSafetyFunctionApi` adds or removes the service `Icu_CheckChannelStatus()` to or from the code.
- `IcuSetPrescalerApi` adds or removes the service `Icu_SetPrescaler()` to or from the code.
- `IcuGetInputLevelApi` adds or removes the service `Icu_GetInputLevel()` to or from the code.
- `IcuChannelGroupApi` adds or removes the services `Icu_StartGroupTrigger()` and `Icu_StopGroupTrigger()` to/from the code.
- `IcuEnableNotiApiCapableWakeup` specifies whether `Icu_EnableNotification()` will enable the associated notification for both interrupt and wakeup or display AUTOSAR standard behavior (only associated notification interrupt).
- `IcuDisableNotiApiCapableWakeup` specifies whether `Icu_DisableNotification()` will disable the associated notification for both interrupt and wakeup or display AUTOSAR standard behavior (only associated notification interrupt).
- `IcuWakeupAcceptanceInSetMode` specifies the acceptance of wakeup signal during Sleep transition processing with `Icu_SetMode()`:
 - TRUE: Accepts wakeup signal during `Icu_SetMode()`
 - FALSE: Does not accept wakeup signal during `Icu_SetMode()`

4.4 ICU configuration set

- `IcuMaxChannel` specifies the number of channels configured.

Note: *`IcuMaxChannel` must be equal to the configured channels.*

Note: *As the number of ICU channel increases, the duration of the critical section in the `Icu_Init()`, `Icu_SetMode()`, and `Icu_DeInit()` will be longer.*

4.4.1 ICU channel configuration

- `IcuChannelId` specifies the group Id of the ICU channel. This value will be assigned to a symbolic name:
 - The symbolic name is prefixed with `IcuConf_IcuChannel`.

Note: *The logical number must be unique, zero-based, and consecutive.*

- `IcuResource` specifies the physical hardware timer that is assigned to this logical channel. The following resources can be selected:
 - TCPWM resource is used for ICU channel and supports all ICU measurement modes.
 - GPIO resource is used to detect external interrupts and supports only `IcuEdgeCounter` and `IcuSignalEdgeDetection` modes.
 - TCPWM_0_0: TCPWM instance 0 Channel 0
 - TCPWM_0_1: TCPWM instance 0 Channel 1
 - ...
 - TCPWM_1_0: TCPWM instance 1 Channel 0
 - TCPWM_1_1: TCPWM instance 1 Channel 1

- ...
- TCPWM_m_n: TCPWM instance m Channel n (m: TCPWM instance number, n: TCPWM channel number)
- GPIO_0_0: GPIO port 0 Channel 0
- GPIO_0_1: GPIO port 0 Channel 1
- ...
- GPIO_1_0: GPIO port 1 Channel 0
- GPIO_1_1: GPIO port 1 Channel 1
- ...
- GPIO_m_n: GPIO port m Channel n (m: port number, n: pin number)

Note: *Selectable resource depends on the subderivatives.*

Note: *GPT, OCU, PWM drivers and OS use TCPWM channels. The ICU driver must not use TCPWM channel that is used by the other modules.*

Note: *IcuResource shows all TCPWM resources and GPIO resources on the device. See [Hardware documentation](#) for the resources connected to the pin for IcuResource.*

- IcuDefaultStartEdge specifies the default-activation-edge which will be used for this channel if there was no activation-edge configured by the call of service Icu_SetActivationCondition().
 - ICU_FALLING_EDGE: Falling edge is the used.
 - ICU_RISING_EDGE: Rising edge is the used.
 - ICU_BOTH_EDGES: Both edges are used.
- IcuMeasurementMode specifies the measurement mode of this channel.
 - ICU_MODE_EDGE_COUNTER: This mode is used to count the edges which are configured by the call to the service Icu_SetActivationCondition().
 - ICU_MODE_SIGNAL_EDGE_DETECT: This mode is used for detecting the edges which are configured by the call to the service Icu_SetActivationCondition().
 - ICU_MODE_SIGNAL_MEASUREMENT: This mode is used to measure time between various configurable edges. The period start edges are configured and cannot be changed during runtime.
 - ICU_MODE_TIMESTAMP: This mode is used to capture timer values on the edges which are configured by the call to the service Icu_SetActivationCondition().

Note: *GPIO does not support IcuSignalMeasurement and IcuTimestamp modes.*

- IcuWakeupCapability specifies the wakeup capability of this channel.
 - TRUE: Channel is capable to wakeup.
 - FALSE: Channel is incapable to wakeup.
- IcuNoiseFilterEnable enables or disables the noise filter functionality of this channel.

Note: *The noise filter function can be used only by GPIO.*

Note: *Only one IcuNoiseFilterEnable can be enabled in the same GPIO port.*

- IcuOverflowNotification specifies the notification function when the related timer overflows.

Note: *This function is available in IcuSignalMeasurement and IcuTimestamp modes.*

Note: *IcuOverflowNotification, IcuTimestampNotification, IcuSignalNotification, and IcuDmaErrorNotification must be unique across all channels.*

Note: *Overflow is notified periodically when the measurement mode is timestamp.*

- `IcuChannelClkSrcRef` specifies the reference to `McuClockReferencePoint` from which the channel clock is derived.

Note: *IcuChannelClkSrcRef can be selected only when the TCPWM resource is used.*

Note: *Peripheral clock must be set in the MCU driver.*

Note: *IcuChannelClkSrcRef can select the clock of this IcuResource.*

- `IcuChannelTickFrequency` specifies the tick frequency of the timer, in Hz, used for `IcuSignalMeasurement` and `IcuTimestamp` modes.

Note: *IcuChannelTickFrequency can be selected only by IcuSignalMeasurement and IcuTimestamp modes.*

Note: *This parameter is used for calculating the prescaler value. If the calculated value is not supported in hardware, an error message is reported.*

Note: *Possible prescalers are 1, 2, 4, 8, 16, 32, 64, and 128 for TCPWM resource.*

- `IcuInputTriggerSelection` specifies the input trigger only when a TCPWM resource is used. The input trigger is used as the input signal bound to one or more TCPWM resources.
 - `TCPWM_0_TR_ONE_CNT_IN_0`: one-to-one trigger signal 0 of TCPWM instance 0
 - `TCPWM_0_TR_ONE_CNT_IN_1`: one-to-one trigger signal 1 of TCPWM instance 0
 - ...
 - `TCPWM_1_TR_ONE_CNT_IN_0`: one-to-one trigger signal 0 of TCPWM instance 1
 - `TCPWM_1_TR_ONE_CNT_IN_1`: one-to-one trigger signal 1 of TCPWM instance 1
 - ...
 - `TCPWM_m_TR_ONE_CNT_IN_n`: one-to-one trigger signal n of TCPWM instance m
 - `TCPWM_0_TR_ALL_CNT_IN_0`: multiplexer trigger signal 0 of TCPWM instance 0
 - `TCPWM_0_TR_ALL_CNT_IN_1`: multiplexer trigger signal 1 of TCPWM instance 0
 - ...
 - `TCPWM_1_TR_ALL_CNT_IN_0`: multiplexer trigger signal 0 of TCPWM instance 1
 - `TCPWM_1_TR_ALL_CNT_IN_1`: multiplexer trigger signal 1 of TCPWM instance 1
 - ...
 - `TCPWM_m_TR_ALL_CNT_IN_n`: multiplexer trigger signal n of TCPWM instance m

Note: *Trigger (Multiplexer-based trigger) group or trigger One-to-One group configuration between IO input and TCPWM is also required in the PORT module. IcuInputTriggerSelection must select the same instance number as the TCPWM channel selected by IcuResource.*

Note: *It is not possible to select the same trigger (Multiplexer-based trigger) as IcuChannelGroupStartTrigger or IcuChannelGroupStopTrigger are in same configuration set. GPT, OCU and PWM drivers also use input triggers of TCPWM. In case, the same input trigger is configured:*

- `GptInputTriggerSelection`: a warning occurs.
- `IcuChannelGroupStartTrigger`, `IcuChannelGroupStopTriger` or other modules except for `GptInputTriggerSelection`: an error occurs.
- `IcuEnableDebug` enables or disables the debug capability to stop a timer channel when the processor is in the debug mode.

Note: *`IcuEnableDebug` can be selected only by `IcuSignalMeasurement` and `IcuTimestamp` modes.*

4.4.1.1 ICU signal edge detection configuration

The `IcuSignalEdgeDetection` container contains the configuration (parameters) only if the measurement mode is `IcuSignalEdgeDetection`.

- `IcuSignalNotification` specifies the callback function name.

Note: *If `IcuSignalNotification` is “NULL” or “BLANK”, the callback function is not called. Notifications must be declared and defined outside the ICU module. The file containing the declarations must be included using the `IcuGeneral/IcuIncludeFile` parameter.*

- `IcuOverflowNotification`, `IcuTimestampNotification`, `IcuSignalNotification`, and `IcuDmaErrorNotification` must be unique across all channels.

4.4.1.2 ICU signal measurement property configuration

The `IcuSignalMeasurement` container contains the configuration (parameters), if the measurement mode is `IcuSignalMeasurement`.

- `IcuSignalMeasurementProperty` specifies the property that could be measured only if the mode is `IcuSignalMeasurement`.
 - `ICU_DUTY_CYCLE`: Duty cycle (coherent Active and period time)
 - `ICU_HIGH_TIME`: Elapsed signal high time
 - `ICU_LOW_TIME`: Elapsed signal low time
 - `ICU_PERIOD_TIME`: Elapsed signal period time

4.4.1.3 ICU time stamp measurement configuration

The `IcuTimestampMeasurement` container contains the configuration (parameters) only if the measurement mode is `IcuTimestamp`.

- `IcuTimestampMeasurementProperty` specifies the handling type of the buffer only if the mode is `IcuTimestamp`.
 - `ICU_CIRCULAR_BUFFER`: Circular type
After reaching the end of the buffer, the driver starts from the beginning of the buffer.
 - `ICU_LINEAR_BUFFER`: Linear type
The buffer will be filled once.
- `IcuTimestampNotification` specifies the callback function name.

Note: *If `IcuTimestampNotification` is “NULL” or “BLANK”, the callback function is not called. Notifications must be declared and defined outside the ICU module. The file containing the declarations must be included using the `IcuGeneral/IcuIncludeFile` parameter.*

Note: *IcuOverflowNotification, IcuTimestampNotification, IcuSignalNotification, and IcuDmaErrorNotification must be unique across all channels.*

- `IcuChannelBufferName` specifies the name of the data array used for the channel's timestamp buffer. When `Icu_StartTimestamp()` is called, it confirms if `BufferPtr` has a value in the range specified by `IcuChannelBufferName` and `IcuChannelBufferSize`.

Note: *If `IcuChannelBufferName` is set to "NULL", range check of `BufferPtr` is not carried out.*

Note: *`IcuChannelBufferName` must be unique across all channels.*

- `IcuChannelBufferSize` specifies the length of the data array which is used for the channel's timestamp buffer. When `Icu_StartTimestamp()` is called, it confirms if `BufferPtr` has a value in the range specified by `IcuChannelBufferName` and `IcuChannelBufferSize`.

Note: *The logical number must be ">=1" and "<=65535".*

- `IcuUseDma` enables or disables the DMA function for the `IcuTimestamp` mode.
- `IcuDmaChannel` specifies the input trigger from TCPWM to DMA channel to initiate a timestamp data transfer.

Trigger (Multiplexer-based trigger) group configuration between trigger signal and TCPWM is required in the PORT module.

- `CPUSS_DW0_TR_IN_0`: Trigger signal 0 of DW instance 0 for transfer data between memory and peripherals
- `CPUSS_DW0_TR_IN_1`: Trigger signal 1 of DW instance 0 for transfer data between memory and peripherals
- ...
- `CPUSS_DW1_TR_IN_0`: Trigger signal 0 of DW instance 1 for transfer data between memory and peripherals
- `CPUSS_DW1_TR_IN_1`: Trigger signal 1 of DW instance 1 for transfer data between memory and peripherals
- ...
- `CPUSS_DWm_TR_IN_n`: Trigger signal n of DW instance m for transfer data between memory and peripherals

Note: *`IcuDmaChannel` and `IcuResource` must be of the same combination across all channels.*

Note: *If `IcuUseDma` is disabled, `IcuDmaChannel` is not used.*

- `IcuDmaErrorNotification` specifies the callback function name for the DMA error.

Note: *If `IcuDmaErrorNotification` is "NULL" or "BLANK" the callback function is not called. Notifications must be declared and defined outside the ICU module. The file containing the declarations must be included using the `IcuGeneral/IcuIncludeFile` parameter.*

- `IcuOverflowNotification, IcuTimestampNotification, IcuSignalNotification, and IcuDmaErrorNotification` must be unique across all channels.
- If `IcuUseDma` is disabled, `IcuDmaErrorNotification` is not used.

4.4.1.4 ICU wakeup configuration

- `IcuChannelWakeupInfo` specifies the reference to the wakeup source of the ECU state manager (`EcuM`) only if `IcuWakeupCapability` is `TRUE`.

Note: *If `IcuChannelWakeupInfo` is blank, the wakeup source value is set as “0” by the ICU driver.*

- `IcuDisableEcuMWakeUpNotification` specifies the calling of `EcuM_CheckWakeUp()` from the ICU interrupt function for this channel.
 - `TRUE`: ICU interrupt function does not call `EcuM_CheckWakeUp()`
 - `FALSE` (AUTOSAR standard behavior): ICU interrupt function calls the `EcuM_CheckWakeUp()`

4.4.2 ICU channel group configuration

- `IcuChannelGroupId` specifies the logical number of the ICU channel group.
 - The symbolic name is prefixed with `IcuConf_IcuChannelGroup_`

Note: *The logical number must be unique, zero-based, and consecutive.*

- `IcuChannelGroupStartTrigger` specifies the input trigger to start the ICU channel group synchronously.
 - `TCPWM_0_TR_ALL_CNT_IN_0`: multiplexer trigger signal 0 of TCPWM instance 0
 - `TCPWM_0_TR_ALL_CNT_IN_1`: multiplexer trigger signal 1 of TCPWM instance 0
 - ...
 - `TCPWM_1_TR_ALL_CNT_IN_0`: multiplexer trigger signal 0 of TCPWM instance 1
 - `TCPWM_1_TR_ALL_CNT_IN_1`: multiplexer trigger signal 1 of TCPWM instance 1
 - ...
 - `TCPWM_m_TR_ALL_CNT_IN_n`: multiplexer trigger signal n of TCPWM instance m

Note: *`IcuChannelGroupStartTrigger` must select the same instance as all channels in the ICU channel group.*

Note: *When this parameter is configured, a trigger signal is required to start all channels in the group. It is necessary to call `Port_ActTrigger()` to issue a trigger signal after calling `Icu_StartGroupTrigger()`. In this case, the group trigger configuration is also required in `PORT` module. If `Port_ActTrigger()` is not called, channels in the group are not started except for channels in `IcuSignalMeasurement` mode.*

Note: *When this parameter is not configured, the channels in the group are started sequentially by `Icu_StartGroupTrigger()`. In the following cases, the channels will started sequentially by `Icu_StartGroupTrigger()` even if this parameter is configured:*

- `IcuSignalMeasurement` mode
- `IcuResource` is `GPIO`

Note: *`IcuChannelGroupStartTrigger` must be unique.*

Note: *It is not possible to select the same trigger (Multiplexer-based trigger) as `IcuInputTriggerSelection` or `IcuChannelGroupStopTrigger` in same configuration set. If you select, an error occurs.*

Note: *In case if the same input trigger is configured by the other group of `IcuChannelGroupStartTrigger` in same configuration set, a warning occurs. Also GPT, OCU and PWM drivers also use input triggers of TCPWM. In case, the same input trigger is configured:*

- `PwmChannelGroupStartTrigger`: a warning occurs.
- Other modules except for `PwmChannelGroupStartTrigger`: an error occurs.
- `IcuChannelGroupStopTrigger` specifies the input trigger to stop the ICU channel group synchronously.
 - `TCPWM_0_TR_ALL_CNT_IN_0`: multiplexer trigger signal 0 of TCPWM instance 0
 - `TCPWM_0_TR_ALL_CNT_IN_1`: multiplexer trigger signal 1 of TCPWM instance 0
 - ...
 - `TCPWM_1_TR_ALL_CNT_IN_0`: multiplexer trigger signal 0 of TCPWM instance 1
 - `TCPWM_1_TR_ALL_CNT_IN_1`: multiplexer trigger signal 1 of TCPWM instance 1
 - ...
 - `TCPWM_m_TR_ALL_CNT_IN_n`: multiplexer trigger signal n of TCPWM instance m

Note: *`IcuChannelGroupStopTrigger` must select the same instance as all channels in the ICU channel group.*

Note: *When this parameter is configured, a trigger signal is required to stop all channels in the group. It is necessary to call `Port_ActTrigger()` to issue a trigger signal after calling `Icu_StopGroupTrigger()`. In this case, the group trigger configuration is also required in PORT module. If `Port_ActTrigger()` is not called, channels in the group are not stopped except for channels in `IcuSignalMeasurement` mode.*

Note: *When this parameter is not configured, the channels in the group are stopped sequentially by `Icu_StopGroupTrigger()`. In the following cases, the channels will be stopped sequentially by `Icu_StopGroupTrigger()` even if this parameter is configured:*

- `IcuSignalMeasurement` mode
- `IcuResource` is GPIO

Note: *`IcuChannelGroupStopTrigger` must be unique. It is not possible to select the same trigger (Multiplexer-based trigger) as `IcuInputTriggerSelection` or `IcuChannelGroupStartTrigger` in same configuration set. If you select, an error occurs.*

Note: *In case if the same input trigger is configured by the other group of `IcuChannelGroupStopTrigger` in same configuration set, a warning occurs. GPT, OCU and PWM drivers also use input triggers of TCPWM.*

Note: *In case if the same input trigger is configured in:*

- `PwmChannelGroupStopTrigger`: a warning occurs.
- Other modules except for `PwmChannelGroupStopTrigger`: an error occurs.
- `IcuChannelRef` specifies assignment of ICU channel to an ICU channel group.

Note: *`IcuChannelRef` must be unique in a channel group. Also, if `IcuChannelGroupStopTrigger` is configured, it must be unique across all groups.*

Note: *If `IcuChannelGroupStartTrigger` or `IcuChannelGroupStopTrigger` are configured, all channels in the ICU group must be the same instance.*

5 Functional description

5.1 Inclusion

The *Icu.h* file includes all necessary external identifiers. Thus, the application only needs to include *Icu.h* to make all API functions and data types available.

5.2 Initialization

The ICU driver does not provide functions for PORT, EcuM, SchM, and MCU configuration and initialization. This must be done by the PORT driver, EcuM, and MCU driver prior to using the ICU driver.

Note: This ICU driver supports post-build-time configuration, thus different configuration set pointer can be passed to the function *Icu_Init()*.

5.3 Runtime reconfiguration

The ICU driver is not reconfigurable at runtime. The only way to change the driver's configuration is to stop all channels, de-initialize the driver, and reinitialize with a different configuration set.

5.4 ICU channel

The *IcuChannelId* defines the channel number to be used. It specifies the ICU channel for which the service is done. A user-given symbolic name is available or generated for each configured channel.

5.5 Notification

The notification capability is available for each channel in *IcuSignalEdgeDetection* or *IcuTimestamp* modes. The API functions *Icu_EnableNotification()* and *Icu_DisableNotification()* have to be used to enable or disable the notification for the channels at runtime. For channels in *IcuTimestamp* mode, a notification interval can be set when the channel is started. If the *NotifyInterval* parameter is 0 and *Icu_EnableNotification()* is called, the channel's notification capability will remain disabled. If the number of timestamp reaches *NotifyInterval*, then the notification function is called. In case of linear buffer, if the *NotifyInterval* parameter is lower than the *BufferSize* parameter, the notification function may be called more than once.

The notification capability for signal edge detect channel is also available in SLEEP mode. So, you will be informed via notification call on wakeup after the ECU state manager (EcuM) report has been done.

Note: No runtime callback will be generated, if the function name for the notification in the module configuration for the channel is NULL ("").

Note: After *Icu_Init()*, notification is not enabled.

Note: The notification can be disabled by calling *Icu_DisableNotification* as mentioned in this chapter. However, it would not work with notifications that have already been handled, if *Icu_DisableNotification* is called from a high priority interruption during a few cycles before the user-defined notification function is called.

5.6 Overflow notification

The overflow notification capability is available for each channel in `IcuSignalMeasurement` or `IcuTimestamp` mode. The API functions `Icu_EnableOverflowNotification()` and `Icu_DisableOverflowNotification()` has to be used to enable or disable the overflow notification for the channels at runtime. For channels in `IcuTimestamp` mode, it is notified each time the counter value overflows. For channels in `IcuSignalMeasurement` mode, it is notified when the counter value overflows until the edge to be measured is detected.

The overflow notification capability is not available in the SLEEP mode.

Note: *No runtime callback will be generated if the function name for the overflow notification in the module configuration for the channel is NULL ("").*

Note: *After `Icu_Init()`, overflow notification is not enabled.*

Note: *The overflow notification can be disabled by calling `Icu_DisableOverflowNotification` as mentioned in this chapter. However, it would not work for the overflow notification that have already been handled, if `Icu_DisableOverflowNotification` is called from a high priority interruption during a few cycles before the user-defined overflow notification function is called.*

5.7 Wakeup

The `IcuWakeupCapability` defines if an ICU channel is capable to wakeup from SLEEP mode. To allow wakeup for a channel it has to be enabled by `Icu_EnableWakeup()`; the wakeup source will be reported to the EcuM when an interrupt is triggered. This is only supported in `ICU_MODE_SLEEP` mode. When wakeup is disabled by `Icu_DisableWakeup()`, no report is done on an interrupt.

The wakeup is disabled for all channels after `Icu_Init()`. Also, runtime enable or disable of wakeup is only successful if the static configuration of the channel `IcuWakeupCapability` is TRUE. Even in this case, the channel wakeup configuration will not be changed while the ICU driver is in the Sleep mode.

Note: *The wakeup functionality is not available while the MCU is in DeepSleep or Hibernate mode when the TCPWM resource is used as the ICU channel. When the GPIO resource that supports DeepSleep is used, the wakeup functionality is available in DeepSleep mode.*

Note: *All TCPWM counters must be stopped if the TCPWM resource is used for the ICU channel before entering DeepSleep mode with the following APIs.*

`Icu_DisableEdgeDetection()`, `Icu_DisableEdgeCount()`, `Icu_StopTimestamp()`, `Icu_StopSignalMeasurement()`, or `Icu_StopGroupTrigger()` if the channel group is configured.

5.8 ICU mode

The mode of ICU driver can be changed to normal mode and SLEEP mode. This can be done by calling `Icu_SetMode()` with `ICU_MODE_NORMAL` or `ICU_MODE_SLEEP` of the type `Icu_ModeType`.

Channel wakeup must be enabled explicitly at runtime. You must stop all channels with disabled wakeup (using `Icu_StopTimestamp()`, `Icu_StopSignalMeasurement()`, and `Icu_DisableEdgeCount()`) before going from normal to SLEEP mode. The ICU driver cannot go into SLEEP mode if a channel is running.

All wakeup channels will be configured to recognize an activation condition signal edge when the driver goes into sleep. A signal edge on the channel matching the activation condition results in leaving SLEEP mode of the CPU. The appropriate interrupt service routine will be executed and an EcuM report will be done.

5.9 ICU input state

`Icu_GetInputState()` returns the status of a single channel. The state can be `ICU_ACTIVE` if an external input signal edge has been detected or `ICU_IDLE` if no external input signal edge has been detected.

When an interrupt is triggered for a channel, the status for the channel will be set to active. After the call of `Icu_GetInputState()` and the return of `ICU_ACTIVE`, the status will be set to idle.

This function is available for each channel in `IcuSignalEdgeDetection` or `IcuSignalMeasurement` mode.

5.10 ICU results

In the `IcuEdgeCounter` mode, the number of detected input signal edges is counted. The behavior can be modified, and the results can be retrieved by using:

- `Icu_ResetEdgeCount()`
- `Icu_EnableEdgeCount()`
- `Icu_DisableEdgeCount()`
- `Icu_GetEdgeNumbers()`

In the `IcuSignalEdgeDetection` mode, the notification function is called, if it is enabled when detecting an external pin level change. The activation condition can be modified (both edges, rising, or falling) at runtime. The behavior can be modified by using:

- `Icu_SetActivationCondition()`
- `Icu_EnableEdgeDetection()`
- `Icu_DisableEdgeDetection()`

In the `IcuSignalMeasurement` mode, the signal itself is measured. Therefore, it provides period time or duty cycle time as results. The following functions can be used to get the resulting values:

- `Icu_GetDutyCycleValues()`
- `Icu_GetTimeElapsed()`

The following API functions can start and stop signal measurement:

- `Icu_StartSignalMeasurement()`
- `Icu_StopSignalMeasurement()`

In `IcuTimestamp` mode, the ticks of a ticking timer are captured when an external input signal edge is detected. The behavior can be modified, and the results can be retrieved by using:

- `Icu_StartTimestamp()`
- `Icu_StopTimestamp()`
- `Icu_GetTimestampIndex()`

`Icu_GetTimestampIndex()` always returns the last index after the channel was stopped. The index will be reset when the channel is started again.

5.11 Prescaler setting function

The ICU driver provides the prescaler setting function for the ICU channel during runtime. This function is to maintain the same tick frequency of a TCPWM channel by changing the prescaler setting without initialization, when the input clock frequency for a TCPWM channel is changed. This function is available only for the channel of `IcuSignalMeasurement` and `IcuTimestamp` modes.

`Icu_SetPrescaler()` changes the TCPWM prescaler setting of the selected channel according to the specified input clock frequency.

The ICU channel will be stopped by `Icu_StopSignalMeasurement()`, `Icu_StopTimestamp()`, or `Icu_StopGroupTrigger()` before the calling of `Icu_SetPrescaler()`. If the ICU channel is in the running state, `Icu_SetPrescaler()` rises the `ICU_E_BUSY_OPERATION`.

IcuSignalMeasurement mode:

Input clock frequency of `IcuConf_IcuChannel_MY_SIGNALMEASUREMENT_CHANNEL` is changed by the MCU driver.

```
Icu_StopSignalMeasurement(IcuConf_IcuChannel_MY_SIGNALMEASUREMENT_CHANNEL);  
Icu_SetPrescaler(IcuConf_IcuChannel_MY_SIGNALMEASUREMENT_CHANNEL,  
MY_CLOCK_FREQUENCY);  
Icu_StartSignalMeasurement(IcuConf_IcuChannel_MY_SIGNALMEASUREMENT_CHANNEL);
```

IcuTimestamp mode:

Input clock frequency of `IcuConf_IcuChannel_MY_TIMESTAMP_CHANNEL` is changed by the MCU driver.

```
Icu_StopTimestamp(IcuConf_IcuChannel_MY_TIMESTAMP_CHANNEL);  
Icu_SetPrescaler(IcuConf_IcuChannel_MY_TIMESTAMP_CHANNEL, MY_CLOCK_FREQUENCY);  
Icu_StartTimestamp(IcuConf_IcuChannel_MY_TIMESTAMP_CHANNEL, myTimeStampBuffer, 10,  
5);
```

Note: *`Icu_SetPrescaler()` calculates the prescaler value based on the value of the input clock frequency and `IcuChannelTickFrequency`.*

Note: *Calculating formula: `MY_CLOCK_FREQUENCY` divided by `IcuChannelTickFrequency` (Round down decimals)*

Note: *The following cases cause a poor accuracy of tick duration. In that case, the MCU driver must adjust the frequency of input clock to meet the prescaler value (1, 2, 4, 8, 16, 32, 64, 128).*

- The calculation result of `ClockFrequency` divided by `TickFrequency` does not meet the prescaler value (1, 2, 4, 8, 16, 32, 64, 128).
- If the input clock frequency is close to the `TickFrequency`, the appropriate prescaler value may not be set.

5.12 ICU channel group synchronous start

The ICU driver starts the ICU channels in a group synchronously by configuring `IcuChannelGroupStartTrigger`. In this case, the trigger signal should be configured by the PORT driver.

When this parameter is configured, `Icu_StartGroupTrigger()` does not start ICU channels in a group. The calling of `Port_ActTrigger()` is necessary to generate a trigger signal for synchronous start. When

`IcuChannelGroupStartTrigger` is not configured, ICU channels start sequentially by the calling `Icu_StartGroupTrigger()`.

In the following cases, the channels will be started sequentially by `Icu_StartGroupTrigger()` even if `IcuChannelGroupStartTrigger` is configured.

- `IcuSignalMeasurement` mode
- `IcuResource` is GPIO

```
Icu_StartGroupTrigger(IcuConf_IcuChannelGroup_GROUP_0);
Port_ActTrigger(PortConf_PortTrGroupContainer_MY_TRIGGER_GROUP,
                PortConf_PortOutputTrigger_MY_START_TRIGGER,
                PORT_TR_ACTIVATION_OUTPUT,
                PORT_TR_SENSITIVE_EDGE);
```

5.13 ICU channel group synchronous stop

The ICU driver stops the ICU channels in a group synchronously by configuring `IcuChannelGroupStopTrigger`. In this case, the trigger signal should be configured by the PORT driver.

When this parameter is configured, `Icu_StopGroupTrigger()` does not stop ICU channels in a group. The calling of `Port_ActTrigger()` is necessary to generate a trigger signal for synchronous stop. When `IcuChannelGroupStopTrigger` is not configured, ICU channels stop sequentially by calling the `Icu_StopGroupTrigger()`.

In the following cases, the channels will be stopped sequentially by `Icu_StopGroupTrigger()` even if `IcuChannelGroupStopTrigger` is configured.

- `IcuSignalMeasurement` mode
- `IcuResource` is GPIO

```
Icu_StopGroupTrigger(IcuConf_IcuChannelGroup_GROUP_0);
Port_ActTrigger(PortConf_PortTrGroupContainer_MY_TRIGGER_GROUP,
                PortConf_PortOutputTrigger_MY_STOP_TRIGGER,
                PORT_TR_ACTIVATION_OUTPUT,
                PORT_TR_SENSITIVE_EDGE);
```

5.14 DMA transfer

DMA can be used to copy the timestamp from the register to the result buffer if a configuration parameter `IcuUseDma` is enabled in an ICU channel. It is only `IcuTimestamp` mode.

DW trigger connects to a TCPWM channel to start DMA transfer. This connection needs to be established in advance. DMA transfer will be started immediately after the edge for timestamp is detected. This connection is supported by multiplexer-based trigger group.

If `IcuUseDma` is enabled, a DW channel should be specified by using a configuration parameter `IcuDmaChannel`.

If `IcuDmaErrorNotification` is enabled, the notification function is called after a DMA error is detected, and DEM will be reported (if configured). `Icu_StopGroupTrigger()` or `Icu_StopTimestamp()` should be called before restarting the ICU channel in which the error is detected.

The parameter `BufferSize` of `Icu_StartTimestamp()` must be a multiple of `NotifyInterval` when both the DMA function and the notification function are enabled.

- Note:* Trigger group needs to be configured by the PORT driver but not the ICU driver.
- Note:* The ICU driver's environment must guarantee that DMA is enabled (`DW:CTL:ENABLED = 1`) when DMA is used. The ICU driver cannot access the global register (i.e., `DW:CTL:ENABLED` bit) directly because this setting affects other modules that access to the same register.
- Note:* If DMA feature is used, the ICU driver's environment will guarantee that the result buffer is aligned at memory address, which is some multiple of 4 bytes regardless of whether DMA is enabled.
- Note:* For sub-derivatives which support the cache feature, the CPU has an individual cache that is not shared with the DMA bus master. Therefore, you must ensure that timestamp buffers used by ICU channels for timestamp in which DMA is enabled reside in a non-cacheable memory area. This can be achieved by placing the buffer in a user-specific memory region configured by the CPU's memory protection unit (MPU) as non-cacheable. The ICU driver does not support use of DMA for the result buffer placed in CPUs tightly coupled memories (TCMs). If used, the ICU driver reports to DEM error by DMA transfer.

5.15 API parameter checking

The driver services perform regular error checks. When an error occurs, the error hook routine (configured via `IcuErrorCalloutFunction`) is called and the error code, service ID, module ID and instance ID are passed as parameters.

If development error detection is enabled, all errors are also reported to the DET, a central error hook function within in the AUTOSAR environment. The checking itself cannot be deactivated for safety reasons.

The following development error checks are performed by the services of the ICU driver:

- The function `Icu_Init()` checks whether the configuration set is valid.

In case of an invalid configuration set, the error code `ICU_E_INIT_FAILED` is returned. The function `Icu_Init()` also checks whether the driver's status is initialized. If `Icu_Init()` is called when the driver's status is already initialized, the error code `ICU_E_ALREADY_INITIALIZED` is reported.

If API functions except `Icu_GetVersionInfo()` are called before `Icu_Init()`, the error code `ICU_E_UNINIT` is reported.
- If the functions performing actions on single channel (for example `Icu_GetInputState()`, `Icu_EnableEdgeCount()`, `Icu_StopTimestamp()`, ...) are called with wrong parameter channel, the error code `ICU_E_PARAM_CHANNEL` is reported.
- If the function `Icu_GetDutyCycleValues()` is called with a buffer that is a NULL pointer, the error code `ICU_E_PARAM_POINTER` is reported.
- If the function `Icu_StartTimestamp()` is called with a buffer that is a NULL pointer or is not in the range configured by `IcuChannelBufferName` and `IcuChannelBufferSize`, the error code `ICU_E_PARAM_POINTER` is reported.
- If the function `Icu_StartTimestamp()` is called with an invalid range of `NotifyInterval`, the error code `ICU_E_PARAM_NOTIFY_INTERVAL` is reported.
- If the function `Icu_StartTimestamp()` is called with an invalid `NotifyInterval` that does not match the `BufferSize`, the error code `ICU_E_PARAM_NOTIFY_INTERVAL` is reported. `BufferSize` must be a multiple of `NotifyInterval` when both the DMA and the notification functions are enabled.
- If the function `Icu_StartTimestamp()` is called with a buffer size 0 or with the value that exceeds the configured range, the error code `ICU_E_PARAM_BUFFER_SIZE` is reported.

- If DMA is enabled and `Icu_StartTimestamp()` is called with `NotifyInterval` exceeding 256, the error code `ICU_E_PARAM_NOTIFY_INTERVAL` is reported.
- If DMA is enabled and `Icu_StartTimestamp()` is called under the following conditions, the error code `ICU_E_PARAM_BUFFER_SIZE` is reported:
 - The notification function is configured
The `BufferSize` divided by `NotifyInterval` exceeds 256.
 - The notification function is not configured
The `BufferSize` is a prime number that exceeds 256.
- If the functions `Icu_DisableWakeup()` and `Icu_EnableWakeup()` are called with a parameter channel which is not wakeup capable, the error code `ICU_E_PARAM_CHANNEL` is reported.
- If the function `Icu_SetMode()` is called with a parameter other than `ICU_MODE_NORMAL` or `ICU_MODE_SLEEP`, the error code `ICU_E_PARAM_MODE` is reported.
- If the function `Icu_SetMode()` and `Icu_DeInit()` are called while one or more channels are started, the error code `ICU_E_BUSY_OPERATION` is reported.
- If the function `Icu_SetPrescaler()` is called when the current channel is running, the error code `ICU_E_BUSY_OPERATION` is reported.
- If the function `Icu_SetActivationCondition()` is called with wrong parameter activation, the error code `ICU_E_PARAM_ACTIVATION` is reported.
- If the function `Icu_StopTimestamp()` is called a channel that was not started with `Icu_StartTimestamp`, the error code `ICU_E_NOT_STARTED` is reported.
- If the function `Icu_GetVersionInfo()` is called with `versioninfo` parameter that is a NULL pointer, the error code `ICU_E_PARAM_VINFO` is reported.
- If the function `Icu_CheckChannelStatus()` is called with `CheckChannelStatusPtr` parameter that is a NULL pointer, the error code `ICU_E_PARAM_CHECK_STATUS_POINTER` is reported.
- If the function `Icu_StartGroupTrigger()` and `Icu_StopGroupTrigger()` are called with an invalid channel group, the error code `ICU_E_PARAM_CHANNEL_GROUP` is reported.
- If the function `Icu_StartGroupTrigger()` is called before the buffer information is not set previously when the measurement mode is `ICU_MODE_TIMESTAMP`, the error code `ICU_E_CHANNEL_GROUP_CONDITION` is reported.
- If the function `Icu_SetPrescaler()` is called with an invalid parameter `ClockFrequency`, the error code `ICU_E_PARAM_CLOCK` is reported.
- If one of the following functions (*1) is called by following steps (*2), the error code `ICU_E_WAITING_TRIGGER` is reported.
 - The following functions (*1):

`Icu_SetMode()`, `Icu_SetActivationCondition()`, `Icu_StartTimestamp()`, `Icu_StopTimestamp()`,
`Icu_EnableEdgeCount()`, `Icu_DisableEdgeCount()`, `Icu_EnableEdgeDetection()`,
`Icu_DisableEdgeDetection()`, `Icu_StartGroupTrigger()` or `Icu_SetPrescaler()`.

- The following steps (*2):

1. `Icu_StartGroupTrigger()` or `Icu_StopGroupTrigger()` is called.
 2. API service is called.
 3. The trigger signal by `Port_ActTrigger()` is received.
- If one of the following functions is called during SLEEP mode, the error code `ICU_E_DURING_SLEEP` is reported.

`Icu_EnableWakeup()`, `Icu_DisableWakeup()`, `Icu_SetActivationCondition()`,
`Icu_StartTimestamp()`, `Icu_EnableEdgeCount()`, `Icu_DisableEdgeCount()`,
`Icu_EnableEdgeDetection()`, `Icu_DisableEdgeDetection()`, `Icu_StartSignalMeasurement()`,


```
Icu_StopSignalMeasurement(), Icu_StartGroupTrigger(), Icu_StopGroupTrigger() or  
Icu_SetPrescaler().
```

When an error is detected, the service returns without any action. The function `Icu_GetInputState()` returns `ICU_IDLE`. All other functions return nothing.

5.16 Reentrancy

All functions except `Icu_Init()`, `Icu_DeInit()`, and `Icu_SetMode()` are reentrants.

Note: The ICU module user shall establish a mutual exclusion mechanism, if several function calls are made during run time in different tasks or ISRs targeting the same ICU channel.

5.17 Configuration checking

The channel ID is used to assign the configuration to the underlying HW. The channel ID cannot be used more than once. If channel wakeup capability is enabled, you must configure the wakeup information. Channel selected measurement mode must be enabled, and the additional appropriate mode container must be configured.

5.18 Debugging support

The ICU driver does not support debugging.

5.19 Execution time dependencies

The execution of the API function is dependent on certain factors. [Table 1](#) lists these dependencies.

Table 1 Execution time dependencies

Affected function	Dependency
<code>Icu_Init()</code> <code>Icu_DeInit()</code> <code>Icu_SetMode()</code> <code>Icu_StartGroupTrigger()</code> <code>Icu_StopGroupTrigger()</code>	Runtime depends on the number of configured channels and groups because all configured channels and groups are processed in these functions.

6 Hardware resources

6.1 Ports and pins

To use the ICU driver, you should configure the pins listed in [Table 2](#) within the PORT driver first.

Table 2 Pin configuration

IcuResource	PortPinInitialMode	Description
TCPWM_m_n	Case of one-to-one trigger group: Same as IcuInputTriggerSelection Case of multiplexer-based trigger group: Same as PortTrInputName	Use TCPWM resource (m is instance number, n is channel number)
GPIO_m_n	GPIO	Use GPIO resource (m is port number, n is pin number.)

The PORT driver must be configured with the appropriate port pin:

- PortPinName = Pm_n for TCPWM pin or GPIO pin
- PortPinDirection = PORT_PIN_IN
- PortPinInitialMode = See [Table 2](#)

The associated port pin for each ICU channel is subderivative-dependent. See the hardware manual.

6.2 Timer

The ICU driver uses the counter/timers of TCPWM that are configured. For each configured ICU channel, one hardware timer of the TRAVERO™ T2G family is reserved exclusively. This is done via configuration parameter IcuResource.

TCPWM timer is not used when GPIO is used.

6.3 Interrupts

The ICU driver uses the interrupts associated with the configured hardware resource. The ISR must be declared in the AUTOSAR OS as Category 1 Interrupt or Category 2 Interrupt.

Note: Vector numbers depend on the subderivative. See the hardware manual.

Some channels may share the IRQ. Those channels will configure one ISR for IRQ.

You can define the ISR, which can be specified as:

The IRQ-Name of each ICU channel must be Icu_Isr_Vector_<IRQ No>_Cat1 for Category-1 ISR or Icu_Isr_Vector_<IRQ No>_Cat2 for Category-2 ISR.

The IRQ-Name of each DW channel must be Icu_DwIsr_Vector_<IRQ No>_Cat1 for Category-1 ISR or Icu_DwIsr_Vector_<IRQ No>_Cat2 for Category-2 ISR.

Note: Icu_Isr_Vector_<IRQ No>_Cat2 and Icu_DwIsr_Vector_<IRQ No>_Cat2 must be called from the (OS) interrupt service routine.

In case of category1 usage, the address of `Icu_Isr_Vector_<IRQ No>_Cat1` and `Icu_DwIsr_Vector_<IRQ No>_Cat1` must be the entry in the (OS) interrupt vector table.

Example: Category-1 ISR for the TCPWM resource located in the generated file `src/Icu_Irq.c`:

```
ISR_NATIVE(Icu_Isr_Vector_273_Cat1)
{
    ...
}
```

Example: Category-2 ISR for the GPIO resource located in the generated file `src/Icu_Irq.c`:

```
ISR(Icu_Isr_Vector_21_Cat2)
{
    ...
}
```

Example: Category-1 ISR for using the DMA resource located in the generated file `src/Icu_Irq.c`:

```
ISR_NATIVE(Icu_DwIsr_Vector_151_Cat1)
{
    ...
}
```

Note: *On the Arm® Cortex®-M4 CPU, priority inversion of interrupts may occur under specific timing conditions in the integrated system with TRAVERO™ T2G MCAL. For more details, see the following errata notice.*

*Arm® Cortex®-M4 Software Developers Errata Notice - 838869:
“Store immediate overlapping exception return operation might vector to incorrect interrupt”*

If the user application cannot tolerate the priority inversion, a DSB instruction should be added at the end of the interrupt function to avoid the priority inversion.

TRAVERO™ T2G MCAL interrupts are handled by an ISR wrapper (handler) in the integrated system. Thus, if necessary, the DSB instruction should be added just before the end of the handler by the integrator.

6.4 Triggers

In general, a trigger input signal indicates the completion of a peripheral action or a peripheral event. A trigger output signal initiates a peripheral action. There are two kinds of trigger groups; Trigger (Multiplexer-based trigger) group and trigger One-to-One group. Trigger group is a multiplexer-based connectivity group. This type connects a peripheral input trigger to multiple peripheral output triggers. Trigger One-to-One group is a One-to-One-based connectivity group. This type connects a peripheral input trigger to one specific peripheral output trigger.

For more detail about triggers, see hardware documentation.

The ICU driver uses trigger input signal in the following cases:

Triggering from port pin to TCPWM channel

One-to-One trigger or multiplexer based trigger from port pin to TCPWM channel should be configured by the PORT driver.

- Case of One-to-One trigger:
 - PortTr1To1GroupName = PASS to TCPWM trigger name
 - PortTr1To1OutputName = Select trigger signal to a TCPWM (e.g. TCPWM_0_TR_ONE_CNT_IN_2)
 - PortTr1To1InputType = PORT_TR_1TO1_IN_CONST0
 - PortTr1To1InvertEnable = Disable
 - PortTr1To1SensitiveType = PORT_TR_SENSITIVE_LEVEL
- Case of multiplexer based trigger:
 - PortTrGroupName = TCPWM input trigger group name
 - PortTrOutputName = Select output trigger signal to all TCPWM (e.g. TCPWM_0_TR_ALL_CNT_IN_8)
 - PortTrInputName = Select input trigger signal to all TCPWM (e.g. PERI_TR_IO_INPUT_0)
 - PortTrInvertEnable = Disable
 - PortTrSensitiveType = PORT_TR_SENSITIVE_LEVEL

To start or stop the ICU channel group synchronously

A trigger multiplexer from CPUSS_ZERO to TCPWM channels should be configured in the PORT driver:

- PortTrGroupName = TCPWM input trigger group name
- PortOutputTrigger Configuration setting:
 - PortTrOutputName: Select output trigger signal to all TCPWM (e.g. TCPWM_0_TR_ALL_CNT_IN_8)
 - PortTrInputName: CPUSS_ZERO
 - PortTrInvertEnable: Disable
 - PortTrSensitiveType: PORT_TR_SENSITIVE_EDGE

Debug capability to stop timer channels

A trigger multiplexer from CTI to TCPWM should be configured in PORT driver.

- PortTrGroupName = Debug input trigger group name
- PortOutputTrigger Configuration setting:
 - PortTrOutputName: Select debug input trigger (e.g. TR_GROUP_8_INPUT_1)
 - PortTrInputName: Select CTI signal (e.g. CPUSS_CTI_TR_OUT_0)
 - PortTrInvertEnable: Disable
 - PortTrSensitiveType: PORT_TR_SENSITIVE_LEVEL
- PortTrGroupName = Debug output trigger group name
- PortOutputTrigger Configuration setting:
 - PortTrOutputName: TCPWM_n_TR_DEBUG_FREEZE (n: TCPWM instance number)
 - PortTrInputName: Select debug input trigger (e.g. TR_GROUP_9_OUTPUT_1)
 - PortTrInvertEnable: Disable
 - PortTrSensitiveType: PORT_TR_SENSITIVE_LEVEL

7 Appendix A – API reference

7.1 Include files

The *Icu.h* file includes the necessary external identifiers. Thus, your application only needs to include *Icu.h* to make all API functions and data types available.

7.2 Data types

7.2.1 Icu_ChannelType

Type

uint16

Description

Defines the channel ID of the ICU channel.

7.2.2 Icu_ChannelStatusType

Type

uint8

Description

Defines the channel status of an ICU channel. For valid values, see [Table 7](#).

7.2.3 Icu_IndexType

Type

uint16

Description

Type to abstract the return value of the service `Icu_GetTimestampIndex()`.

7.2.4 Icu_GroupType

Type

uint16

Description

Numeric identifier of a channel group.

7.2.5 Icu_EdgeNumberType

Type

uint16

Description

Type to abstract the return value of the service `Icu_GetEdgeNumbers()`.

7.2.6 Icu_ValueType

Type

```
uint32
```

Description

Type to abstract the measurement value of the `IcuTimestamp` and the `IcuSignalMeasurement` modes.

7.2.7 Icu_ClkFrequencyType

Type

```
uint32
```

Description

Defines the clock frequency.

7.2.8 Icu_LevelType

Type

```
typedef enum
{
    ICU_LOW = 0,
    ICU_HIGH
} Icu_LevelType;
```

Description

The state of the input pin related to an ICU channel.

7.2.9 Icu_ModeType

Type

```
typedef enum
{
    ICU_MODE_NORMAL = 1,
    ICU_MODE_SLEEP
} Icu_ModeType;
```

Description

The mode of the ICU driver.

7.2.10 Icu_InputStateType

Type

```
typedef enum
{
    ICU_IDLE = 0,
    ICU_ACTIVE
} Icu_InputStateType;
```

Description

The input state of an ICU channel.

7.2.11 Icu_ActivationType

Type

```
typedef enum
{
    ICU_INIT_EDGE = 0,
    ICU_RISING_EDGE,
    ICU_FALLING_EDGE,
    ICU_BOTH_EDGES
} Icu_ActivationType;
```

Description

Defines the activation type. Note that `ICU_INIT_EDGE` cannot be used by APIs because it is defined for internal function.

7.2.12 Icu_DutyCycleType

Type

```
typedef struct
{
    Icu_ValueType ActiveTime;
    Icu_ValueType PeriodTime;
} Icu_DutyCycleType;
```

Description

Type which contains the values needed for calculating duty cycles.

7.2.13 Icu_ConfigType

Type

```
typedef struct
```

Description

Defines the structure of the configuration data for the ICU driver.

7.2.14 Icu_DriverStatusType

Type

```
typedef enum
{
    ICU_S_ILLEGAL = 0,
    ICU_S_UNINITIALIZED,
    ICU_S_INITIALIZED,
    ICU_S_LOCKED
} Icu_DriverStatusType;
```

Description

Defines the ICU driver status.

7.2.15 Icu_CheckChannelStatusType

Type

```
typedef struct
{
    Icu_DriverStatusType    DriverStatus;
    Icu_ModeType            DriverWakeupMode;
    Icu_ActivationType       ChannelActivationEdge;
    Icu_ChannelStatusType    ChannelStatus;
    uint8                    Prescaler;

    boolean                  ChannelWakeupEnable;
    boolean                  ChannelNotifyEnabled;
    boolean                  ChannelOverflowNotifyEnabled;
} Icu_CheckChannelStatusType;
```

Description

Structure of a channel's state.

7.3 Constants

7.3.1 Error codes

The services might return the error codes listed in [Table 3](#) if development error detection is enabled:

Table 3 Error codes

Name	Value	Description
ICU_E_PARAM_POINTER	10	API service called with an invalid pointer.
ICU_E_PARAM_CHANNEL	11	API service used with an invalid channel identifier or channel was not configured for the functionality of the calling API.
ICU_E_PARAM_ACTIVATION	12	API service used with an invalid or not feasible activation.
ICU_E_INIT_FAILED	13	API service <code>Icu_Init</code> function failed with an invalid configuration pointer.
ICU_E_PARAM_BUFFER_SIZE	14	API service used with an invalid buffer size.
ICU_E_PARAM_MODE	15	API service <code>Icu_SetMode</code> used with an invalid mode.
ICU_E_UNINIT	20	API service used prior to module initialization.
ICU_E_NOT_STARTED	21	API service <code>Icu_StopTimestamp</code> called on a channel which was not started or already stopped.
ICU_E_BUSY_OPERATION	22	API service <code>Icu_SetMode</code> is called during a running operation.
ICU_E_ALREADY_INITIALIZED	23	API service <code>Icu_Init</code> is called when driver's state is already initialized.
ICU_E_PARAM_NOTIFY_INTERVAL	24	API service <code>Icu_Init</code> is called when <code>Icu_StartTimestamp</code> is called and the <code>NotifyInterval</code> parameter is invalid.

Name	Value	Description
ICU_E_PARAM_VINFO	25	API service Icu_Init is called when Icu_GetVersionInfo is called and the versioninfo parameter is invalid.
ICU_E_PARAM_CHECK_STATUS_POINTER	32	API service Icu_CheckChannelStatus is called with an invalid CheckChannelStatusPtr pointer.
ICU_E_HW_ERROR	64	Hardware error identifier for callout.
ICU_E_CHANNEL_GROUP_CONDITION	65	API service used with group start condition error.
ICU_E_PARAM_CHANNEL_GROUP	66	API service used with an invalid channel group identifier or channel group was not configured as a functionality of the calling API.
ICU_E_PARAM_CLOCK	67	API service used with prescaler parameter error.
ICU_E_WAITING_TRIGGER	68	API service is called by the following steps. 1. Icu_StartGroupTrigger() or Icu_StopGroupTrigger() is called. 2. API service is called. 3. The trigger signal by Port_ActTrigger() is received.
ICU_E_DURING_SLEEP	69	API service is called in Sleep mode.

7.3.2 Version information

The version information listed in [Table 4](#) is published in the driver's header file.

Table 4 Version information

Name	Value	Description
ICU_AR_RELEASE_MAJOR_VERSION	4	AUTOSAR specification major version
ICU_AR_RELEASE_MINOR_VERSION	2	AUTOSAR specification minor version
ICU_AR_RELEASE_PATCH_VERSION	2	AUTOSAR specification patch version
ICU_SW_MAJOR_VERSION	See release notes	Software major version number
ICU_SW_MINOR_VERSION	See release notes	Software minor version number
ICU_SW_PATCH_VERSION	See release notes	Software patch version number

7.3.3 Module information

Table 5 Module information

Name	Value	Description
ICU_MODULE_ID	122	Module ID
ICU_VENDOR_ID	66	Vendor ID

7.3.4 API service IDs

The API service IDs listed in [Table 6](#) are published in the driver's header file.

Table 6 API service IDs

Name	Value	API service
ICU_API_INIT	0x0	Icu_Init
ICU_API_DE_INIT	0x1	Icu_DeInit
ICU_API_SET_MODE	0x2	Icu_SetMode
ICU_API_DISABLE_WAKEUP	0x3	Icu_DisableWakeup
ICU_API_ENABLE_WAKEUP	0x4	Icu_EnableWakeup
ICU_API_SET_ACTIVATION_CONDITION	0x5	Icu_SetActivationCondition
ICU_API_DISABLE_NOTIFICATION	0x6	Icu_DisableNotification
ICU_API_ENABLE_NOTIFICATION	0x7	Icu_EnableNotification
ICU_API_GET_INPUT_STATE	0x8	Icu_GetInputState
ICU_API_START_TIMESTAMP	0x9	Icu_StartTimestamp
ICU_API_STOP_TIMESTAMP	0xA	Icu_StopTimestamp
ICU_API_GET_TIMESTAMP_INDEX	0xB	Icu_GetTimestampIndex
ICU_API_RESET_EDGE_COUNT	0xC	Icu_ResetEdgeCount
ICU_API_ENABLE_EDGE_COUNT	0xD	Icu_EnableEdgeCount
ICU_API_DISABLE_EDGE_COUNT	0xE	Icu_DisableEdgeCount
ICU_API_GET_EDGE_NUMBERS	0xF	Icu_GetEdgeNumbers
ICU_API_GET_TIME_ELAPSED	0x10	Icu_GetTimeElapsed
ICU_API_GET_DUTY_CYCLE_VALUES	0x11	Icu_GetDutyCycleValues
ICU_API_GET_VERSION_INFO	0x12	Icu_GetVersionInfo
ICU_API_START_SIGNAL_MEAS	0x13	Icu_StartSignalMeasurement
ICU_API_STOP_SIGNAL_MEAS	0x14	Icu_StopSignalMeasurement
ICU_API_CHECK_WAKEUP	0x15	Icu_CheckWakeup
ICU_API_ENABLE_EDGE_DETECTION	0x16	Icu_EnableEdgeDetection
ICU_API_DISABLE_EDGE_DETECTION	0x17	Icu_DisableEdgeDetection
ICU_API_DW_INTERRUPT_EVENT	0xF7	Icu_DwIsr_Vector_Internal
ICU_API_INTERRUPT_EVENT	0xF8	Icu_Isr_Vector_Internal
ICU_API_DISABLE_OVERFLOW_NOTIFICATION	0xF9	Icu_DisableOverflowNotification
ICU_API_ENABLE_OVERFLOW_NOTIFICATION	0xFA	Icu_EnableOverflowNotification
ICU_API_SET_PRESCALER	0xFB	Icu_SetPrescaler

Name	Value	API service
ICU_API_STOP_GROUP_TRIGGER	0xFC	Icu_StopGroupTrigger
ICU_API_START_GROUP_TRIGGER	0xFD	Icu_StartGroupTrigger
ICU_API_GET_INPUT_LEVEL	0xFE	Icu_GetInputLevel
ICU_API_CHECK_CHANNEL_STATUS	0xFF	Icu_CheckChannelStatus

7.3.5 Channel status

Table 7 Channel status

Name	Value	Description
ICU_NOT_STARTED	0x00	The channel has not been started or enabled yet.
ICU_RUNNING	0x01	The channel has been started or enabled and is running.
ICU_STOPPED	0x02	The channel has been stopped or disabled and is not running.
ICU_WAITING_START_TRIGGER	0x03	The channel has been waiting for a start trigger.
ICU_WAITING_STOP_TRIGGER	0x04	The channel has been waiting for a stop trigger.

7.3.6 Symbolic names

Table 8 Symbolic names

Name	Description
IcuConf_IcuChannel_<n>	Symbolic name for ICU channel <i>n</i> (<i>n</i> is channel's short name).
IcuConf_IcuChannelGroup_<n>	Symbolic name for ICU group <i>n</i> (<i>n</i> is channel group's short name).
IcuConf_IcuConfigSet_<n>	Symbolic name for ICU configuration setting <i>n</i> (<i>n</i> is configuration set's short name).

7.4 Functions

7.4.1 Icu_Init

Syntax

```
void Icu_Init
(
    const Icu_ConfigType* ConfigPtr
)
```

Service ID

0x0

Parameters (in)

ConfigPtr – Pointer to the configuration set.

Parameters (out)

None

Return value

None

DET errors

- `ICU_E_ALREADY_INITIALIZED` - The driver's state is already initialized at the time this function is called.
- `ICU_E_INIT_FAILED` - The configuration pointer is incorrect.

DEM errors

None

Description

`Icu_Init()` is a service for ICU initialization. This function will initialize all internal variables and the used ICU structure of the microcontroller according to a selected configuration set. This function will be called with a pointer to a selected configuration structure.

7.4.2 Icu_DeInit

Syntax

```
void Icu_DeInit  
(  
    void  
)
```

Service ID

0x1

Parameters (in)

None

Parameters (out)

None

Return value

None

DET errors

- `ICU_E_UNINIT` - `Icu_Init()` will be called before calling any other ICU services.
- `ICU_E_BUSY_OPERATION` - API service is called during a running operation.

DEM errors

None

Description

`Icu_DeInit()` is a service for ICU deinitialization.

7.4.3 Icu_SetMode

Syntax

```
void Icu_SetMode  
(  
    Icu_ModeType Mode  
)
```

Service ID

0x2

Parameters (in)

Mode - Operation mode of the ICU driver.

Parameters (out)

None

Return value

None

DET errors

- ICU_E_UNINIT - Icu_Init() will be called before calling any other ICU services.
- ICU_E_PARAM_MODE - Mode is invalid.
- ICU_E_BUSY_OPERATION - API service is called during a running operation.
- ICU_E_WAITING_TRIGGER - API service is called by the following steps:
 - Icu_StartGroupTrigger() or Icu_StopGroupTrigger() is called.
 - API service is called.
 - The trigger signal by Port_ActTrigger() is received.

DEM errors

None

Description

Icu_SetMode() is a service for changing the operation mode of the ICU driver.

If IcuWakeupAcceptanceInSetMode is TRUE, wakeup signal during Sleep transition processing with Icu_SetMode() is accepted.

If IcuWakeupAcceptanceInSetMode is FALSE, wakeup signal during Sleep transition processing with Icu_SetMode() is not accepted.

Note: *If an error of ICU_E_WAITING_TRIGGER is detected, it is possible to change the operation mode by calling Icu_SetMode() again after calling Icu_StopGroupTrigger().*

7.4.4 Icu_DisableWakeup

Syntax

```
void Icu_DisableWakeup  
(  
    Icu_ChannelType Channel  
)
```

Service ID

0x3

Parameters (in)

Channel - Numeric identifier of the ICU channel.

Parameters (out)

None

Return value

None

DET errors

- ICU_E_UNINIT - Icu_Init() will be called before calling any other ICU services.
- ICU_E_PARAM_CHANNEL - Channel is invalid or it is not capable to wakeup.
- ICU_E_DURING_SLEEP - API service is called in SLEEP mode.

DEM errors

None

Description

Icu_DisableWakeup() is a service for disabling the wakeup capability of a channel.

7.4.5 Icu_EnableWakeup

Syntax

```
void Icu_EnableWakeup  
(  
    Icu_ChannelType Channel  
)
```

Service ID

0x4

Parameters (in)

Channel - Numeric identifier of the ICU channel.

Parameters (out)

None

Return value

None

DET errors

- ICU_E_UNINIT - Icu_Init() will be called before calling any other ICU services.
- ICU_E_PARAM_CHANNEL - Channel is invalid or it is not capable to wakeup.
- ICU_E_DURING_SLEEP - API service is called in SLEEP mode.

DEM errors

None

Description

Icu_EnableWakeup() is a service for enabling the wakeup capability of a channel.

7.4.6 Icu_SetActivationCondition

Syntax

```
void Icu_SetActivationCondition
(
    Icu_ChannelType Channel,
    Icu_ActivationType Activation
)
```

Service ID

0x5

Parameters (in)

- Channel - Numeric identifier of the ICU channel.
- Activation - Type of input signal edge to detect, count, or measure.

Parameters (out)

None

Return value

None

DET errors

- ICU_E_UNINIT - Icu_Init() will be called before calling any other ICU services.
- ICU_E_PARAM_CHANNEL - Channel is invalid or it is not capable to wakeup.
- ICU_E_PARAM_ACTIVATION - Invalid/unknown/non-support activation condition.
- ICU_E_WAITING_TRIGGER - API service is called by the following steps:
 - Icu_StartGroupTrigger() or Icu_StopGroupTrigger() is called.
 - API service is called.
 - The trigger signal by Port_ActTrigger() is received.
- ICU_E_DURING_SLEEP - API service is called in SLEEP mode.

DEM errors

None

Description

`Icu_SetActivationCondition()` is a service to set the activation-edge according to activation parameter for the given channel. `ICU_LOW_LEVEL` and `ICU_HIGH_LEVEL` are supported only by external interrupts.

Note that even after ISR is returned, if level detection (`ICU_LOW_LEVEL` or `ICU_HIGH_LEVEL`) is specified as an activation edge. If the input pin it is at the active level, the interrupt is detected again

7.4.7 Icu_DisableNotification

Syntax

```
void Icu_DisableNotification
(
    Icu_ChannelType Channel
)
```

Service ID

0x6

Parameters (in)

Channel - Numeric identifier of the ICU channel.

Parameters (out)

None

Return value

None

DET errors

- `ICU_E_UNINIT` - `Icu_Init()` will be called before calling any other ICU services.
- `ICU_E_PARAM_CHANNEL` - Channel is invalid.

DEM errors

None

Description

- `Icu_DisableNotification()` is a service for disabling the notification for a channel.
- If `IcuDisableNotiApiCapableWakeup` is `TRUE`, `Icu_DisableNotification()` will enable the associated notification for both interrupt and wakeup.

If `IcuDisableNotiApiCapableWakeup` is `FALSE`, `Icu_DisableNotification()` will display AUTOSAR standard behavior.

7.4.8 Icu_EnableNotification

Syntax

```
void Icu_EnableNotification
(
    Icu_ChannelType Channel
)
```

Service ID

0x7

Parameters (in)

Channel - Numeric identifier of the ICU channel.

Parameters (out)

None

Return value

None

DET errors

- ICU_E_UNINIT - Icu_Init() will be called first before calling any other ICU services.
- ICU_E_PARAM_CHANNEL - Channel is invalid.

DEM errors

None

Description

- Icu_EnableNotification() is a service to enable the notification for a channel.
- If IcuEnableNotiApiCapableWakeup is TRUE, Icu_EnableNotification() will enable associated the notification for both interrupt and wakeup regardless of CPU mode.
- If IcuEnableNotiApiCapableWakeup is FALSE, Icu_EnableNotification() will display AUTOSAR standard behavior.

7.4.9 Icu_GetInputState

Syntax

```
Icu_InputStateType Icu_GetInputState
(
    Icu_ChannelType Channel
)
```

Service ID

0x8

Parameters (in)

Channel - Numeric identifier of the ICU channel.

Parameters (out)

None

Return value

Icu_InputStateType - The state of a channel.

DET errors

- ICU_E_UNINIT - Icu_Init() will be called before calling any other ICU services.
- ICU_E_PARAM_CHANNEL - Channel is invalid.

DEM errors

None

Description

Icu_GetInputState() is a service for scanning the state of a channel.

Note: If overflow occurs in the IcuSignalMeasurement mode, the correct state may not be returned.

7.4.10 Icu_StartTimestamp

Syntax

```
void Icu_StartTimestamp
(
    Icu_ChannelType Channel,
    Icu_ValueType* BufferPtr,
    uint16 BufferSize,
    uint16 NotifyInterval
)
```

Service ID

0x9

Parameters (in)

- Channel - Numeric identifier of the ICU channel.
- BufferPtr - Pointer to the buffer where timestamps are saved.
- BufferSize - Size of the buffer (number of entries).
- NotifyInterval - Number of saved timestamps after which a notification is sent.

Parameters (out)

None

Return value

None

DET errors

- ICU_E_UNINIT - Icu_Init() will be called before calling any other ICU services.
- ICU_E_PARAM_CHANNEL - Channel is invalid.
- ICU_E_PARAM_BUFFER_SIZE - BufferSize is invalid (see [5.15 API parameter checking](#)).

- ICU_E_PARAM_NOTIFY_INTERVAL - NotifyInterval is invalid or NotifyInterval does not match BufferSize (see [5.15 API parameter checking](#)).
- ICU_E_PARAM_POINTER - BufferPtr is invalid.
- ICU_E_WAITING_TRIGGER - API service is called by the following steps:
 - Icu_StartGroupTrigger() or Icu_StopGroupTrigger() is called.
 - API service is called.
 - The trigger signal by Port_ActTrigger() is received.
- ICU_E_DURING_SLEEP - API service is called in SLEEP mode.

DEM errors

None

Description

Icu_StartTimestamp() is a service to start the capture of timer tick values into the given buffer on detection of input signal edges.

7.4.11 Icu_StopTimestamp**Syntax**

```
void Icu_StopTimestamp
(
    Icu_ChannelType Channel
)
```

Service ID

0xA

Parameters (in)

Channel – Numeric identifier of the ICU channel.

Parameters (out)

None

Return value

None

DET errors

- ICU_E_UNINIT - Icu_Init() will be called before calling any other ICU services.
- ICU_E_PARAM_CHANNEL - Channel is invalid.
- ICU_E_NOT_STARTED - Channel is not started.
- ICU_E_WAITING_TRIGGER - API service is called by the following steps:
 - Icu_StartGroupTrigger() or Icu_StopGroupTrigger() is called.
 - API service is called.
 - The trigger signal by Port_ActTrigger() is received.

DEM errors

None

Description

`Icu_StopTimestamp()` is a service to stop the capturing on the given channel that was started by `Icu_StartTimestamp()`.

7.4.12 Icu_GetTimestampIndex

Syntax

```
Icu_IndexType Icu_GetTimestampIndex  
(  
    Icu_ChannelType Channel  
)
```

Service ID

0xB

Parameters (in)

Channel - Numeric identifier of the ICU channel.

Parameters (out)

None

Return value

`Icu_IndexType` - The timestamp index of the given channel, which is to be written next.

DET errors

- `ICU_E_UNINIT` - `Icu_Init()` will be called before calling any other ICU services.
- `ICU_E_PARAM_CHANNEL` - Channel is invalid.

DEM errors

None

Description

This service reads the timestamp index of the given channel, which is to be written next.

7.4.13 Icu_ResetEdgeCount

Syntax

```
void Icu_ResetEdgeCount  
(  
    Icu_ChannelType Channel  
)
```

Service ID

0xC

Parameters (in)

Channel - Numeric identifier of the ICU channel.

Parameters (out)

None

Return value

None

DET errors

- ICU_E_UNINIT - Icu_Init() will be called before calling any other ICU services.
- ICU_E_PARAM_CHANNEL - Channel is invalid.

DEM errors

None

Description

The value of the counted edges is reset to zero.

7.4.14 Icu_EnableEdgeCount

Syntax

```
void Icu_EnableEdgeCount
(
    Icu_ChannelType Channel
)
```

Service ID

0xD

Parameters (in)

Channel - Numeric identifier of the ICU channel.

Parameters (out)

None

Return value

None

DET errors

- ICU_E_UNINIT - Icu_Init() will be called before calling any other ICU services.
- ICU_E_PARAM_CHANNEL - Channel is invalid.
- ICU_E_WAITING_TRIGGER - API service is called by the following steps:
 - Icu_StartGroupTrigger() or Icu_StopGroupTrigger() is called.
 - API service is called.
 - The trigger signal by Port_ActTrigger() is received.
- ICU_E_DURING_SLEEP - API service is called in SLEEP mode.

DEM errors

None

Description

This service enables the counting of edges of the given channel.

7.4.15 Icu_DisableEdgeCount**Syntax**

```
void Icu_DisableEdgeCount
(
    Icu_ChannelType Channel
)
```

Service ID

0xE

Parameters (in)

Channel - Numeric identifier of the ICU channel.

Parameters (out)

None

Return value

None

DET errors

- ICU_E_UNINIT - Icu_Init() will be called before calling any other ICU services.
- ICU_E_PARAM_CHANNEL - Channel is invalid.
- ICU_E_WAITING_TRIGGER - API service is called by the following steps:
 - Icu_StartGroupTrigger() or Icu_StopGroupTrigger() is called.
 - API service is called.
 - The trigger signal by Port_ActTrigger() is received.
- ICU_E_DURING_SLEEP - API service is called in SLEEP mode.

DEM errors

None

Description

This service disables the counting of edges of the given channel.

7.4.16 Icu_GetEdgeNumbers**Syntax**

```
Icu_EdgeNumberType Icu_GetEdgeNumbers
(
    Icu_ChannelType Channel
)
```

Service ID

0xF

Parameters (in)

Channel - Numeric identifier of the ICU channel.

Parameters (out)

None

Return value

`Icu_EdgeNumberType` - The number of counted edges.

DET errors

- `ICU_E_UNINIT` - `Icu_Init()` will be called before calling any other ICU services.
- `ICU_E_PARAM_CHANNEL` - Channel is invalid.

DEM errors

None

Description

This service reads the number of counted edges after the last call of `Icu_ResetEdgeCount()`.

7.4.16.1 `Icu_GetTimeElapsed`

Syntax

```
Icu_ValueType Icu_GetTimeElapsed  
(  
    Icu_ChannelType Channel  
)
```

Service ID

0x10

Parameters (in)

Channel - Numeric identifier of the ICU channel.

Parameters (out)

None

Return value

`Icu_ValueType` - The elapsed signal low, high, or period time in ticks.

DET errors

- `ICU_E_UNINIT` - `Icu_Init()` will be called before calling any other ICU services.
- `ICU_E_PARAM_CHANNEL` - Channel is invalid.

DEM errors

None

Description

This service reads the elapsed low, high, or period time for the given channel if configured for measurement property `ICU_LOW_TIME`, `ICU_HIGH_TIME`, `ICU_PERIOD_TIME`, or `ICU_ACTIVE_TIME`.

7.4.17 Icu_GetDutyCycleValues

Syntax

```
void Icu_GetDutyCycleValues
(
    Icu_ChannelType Channel,
    Icu_DutyCycleType* DutyCycleValues
)
```

Service ID

0x11

Parameters (in)

Channel - Numeric identifier of the ICU channel.

Parameters (out)

DutyCycleValues - Pointer to a buffer where the results (active time and period time) will be stored.

Return value

None

DET errors

- `ICU_E_UNINIT` - `Icu_Init()` will be called before calling any other ICU services.
- `ICU_E_PARAM_CHANNEL` - Channel is invalid.
- `ICU_E_PARAM_POINTER` - `DutyCycleValues` is invalid.

DEM errors

None

Description

This service reads the elapsed period and active time for the given channel if configured for the measurement property `ICU_DUTY_CYCLE`.

Note: *If the edge interval is shorter than the execution time of `Icu_GetDutyCycleValues()`, correct `DutyCycleValues` cannot be returned.*

7.4.18 Icu_GetVersionInfo

Syntax

```
void Icu_GetVersionInfo  
(  
    Std_VersionInfoType* versioninfo  
)
```

Service ID

0x12

Parameters (in)

None

Parameters (out)

`versioninfo` - Pointer to the location where the version information of this module is stored.

Return value

None

DET errors

ICU_E_PARAM_VINFO - `versioninfo` is invalid.

DEM errors

None

Description

This service returns the version information of this module.

7.4.18.1 Icu_StartSignalMeasurement

Syntax

```
void Icu_StartSignalMeasurement  
(  
    Icu_ChannelType Channel  
)
```

Service ID

0x13

Parameters (in)

Channel - Numeric identifier of the ICU channel.

Parameters (out)

None

Return value

None

DET errors

- `ICU_E_UNINIT` - `Icu_Init()` will be called before calling any other ICU services.
- `ICU_E_PARAM_CHANNEL` - `Channel` is invalid or channel has an incorrect mode.
- `ICU_E_DURING_SLEEP` - API service is called in SLEEP mode.

DEM errors

None

Description

This service starts a signal measurement configured channel.

7.4.19 `Icu_StopSignalMeasurement`

Syntax

```
void Icu_StopSignalMeasurement  
(  
    Icu_ChannelType Channel  
)
```

Service ID

0x14

Parameters (in)

Channel - Numeric identifier of the ICU channel.

Parameters (out)

None

Return value

None

DET errors

- `ICU_E_UNINIT` - `Icu_Init()` will be called before calling any other ICU services.
- `ICU_E_PARAM_CHANNEL` - `Channel` is invalid or channel has an incorrect mode.
- `ICU_E_DURING_SLEEP` - API service is called in SLEEP mode.

DEM errors

None

Description

This service stops a signal measurement configured channel.

7.4.20 Icu_CheckWakeup

Syntax

```
void Icu_CheckWakeup  
(  
    Ecum_WakeupSourceType WakeupSource  
)
```

Service ID

0x15

Parameters (in)

WakeupSource - Wakeup source information that needs to be checked. The associated ICU channel can be determined from the configuration data.

Parameters (out)

None

Return value

None

DET errors

ICU_E_UNINIT - Icu_Init() will be called before calling any other ICU services.

DEM errors

None

Description

Icu_CheckWakeup() is a service for checking if a wakeup capable ICU channel is the source for a wakeup event and calls the Ecum service Ecum_SetWakeupEvent() in case of a valid ICU channel wakeup event.

7.4.21 Icu_EnableEdgeDetection

Syntax

```
void Icu_EnableEdgeDetection  
(  
    Icu_ChannelType Channel  
)
```

Service ID

0x16

Parameters (in)

Channel - Numeric identifier of the ICU channel.

Parameters (out)

None

Return value

None

DET errors

- ICU_E_UNINIT - Icu_Init() will be called before calling any other ICU services.
- ICU_E_PARAM_CHANNEL - Channel is invalid.
- ICU_E_WAITING_TRIGGER - API service is called by the following steps:
 - Icu_StartGroupTrigger() or Icu_StopGroupTrigger() is called.
 - API service is called.
 - The trigger signal by Port_ActTrigger() is received.
- ICU_E_DURING_SLEEP - API service is called in SLEEP mode.

DEM errors

None

Description

This service enables or re-enables the detection of edges of the given channel.

7.4.22 Icu_DisableEdgeDetection

Syntax

```
void Icu_DisableEdgeDetection
(
    Icu_ChannelType Channel
)
```

Service ID

0x17

Parameters (in)

Channel – Numeric identifier of the ICU channel.

Parameters (out)

None

Return value

None

DET errors

- ICU_E_UNINIT - Icu_Init() will be called before calling any other ICU services.
- ICU_E_PARAM_CHANNEL - Channel is invalid.
- ICU_E_WAITING_TRIGGER - API service is called by the following steps:
 - Icu_StartGroupTrigger() or Icu_StopGroupTrigger() is called.
 - API service is called.
 - The trigger signal by Port_ActTrigger() is received.
- ICU_E_DURING_SLEEP - API service is called in SLEEP mode.

DEM errors

None

Description

This service disables the detection of edges of the given channel.

7.4.23 Icu_DisableOverflowNotification**Syntax**

```
void Icu_DisableOverflowNotification
(
    Icu_ChannelType Channel
)
```

Service ID

0xF9

Parameters (in)

Channel - Numeric identifier of the ICU channel.

Parameters (out)

None

Return value

None

DET errors

- ICU_E_UNINIT - Icu_Init() will be called before calling any other ICU services.
- ICU_E_PARAM_CHANNEL - Channel is invalid.

DEM errors

None

Description

This service disables the overflow notification on the given channel.

7.4.24 Icu_EnableOverflowNotification**Syntax**

```
void Icu_EnableOverflowNotification
(
    Icu_ChannelType Channel
)
```

Service ID

0xFA

Parameters (in)

Channel - Numeric identifier of the ICU channel.

Parameters (out)

None

Return value

None

DET errors

- ICU_E_UNINIT - Icu_Init() will be called before calling any other ICU services.
- ICU_E_PARAM_CHANNEL - Channel is invalid.

DEM errors

None

Description

This service enables the overflow notification on the given channel.

7.4.25 Icu_SetPrescaler

Syntax

```
void Icu_SetPrescaler
(
    Icu_ChannelType Channel,
    Icu_ClkFrequencyType ClockFrequency
)
```

Service ID

0xFB

Parameters (in)

- Channel - Numeric identifier of the ICU channel.
- ClockFrequency - Clock frequency.

Parameters (out)

None

Return value

None

DET errors

- ICU_E_UNINIT - Icu_Init() will be called before calling any other ICU services.
- ICU_E_PARAM_CHANNEL - Channel is invalid.
- ICU_E_PARAM_CLOCK - ClockFrequency is invalid.
- ICU_E_BUSY_OPERATION - API service is called during a running operation.
- ICU_E_WAITING_TRIGGER - API service is called by the following steps:
 - Icu_StartGroupTrigger() or Icu_StopGroupTrigger() is called.
 - API service is called.
 - The trigger signal by Port_ActTrigger() is received.

- `ICU_E_DURING_SLEEP` - API service is called in SLEEP mode.

DEM errors

None

Description

This service sets the prescaler.

7.4.26 Icu_StopGroupTrigger

Syntax

```
void Icu_StopGroupTrigger  
(  
    Icu_GroupType Group  
)
```

Service ID

0xFC

Parameters (in)

Group - Numeric identifier of the ICU channel group.

Parameters (out)

None

Return value

None

DET errors

- `ICU_E_UNINIT` - `Icu_Init()` will be called before calling any other ICU services.
- `ICU_E_PARAM_CHANNEL_GROUP` - Channel group is invalid.
- `ICU_E_DURING_SLEEP` - API service is called in SLEEP mode.

DEM errors

None

Description

This service stops the trigger to all ICU channels provided in the channel group.

Note: If `Icu_StopGroupTrigger()` is called again before the trigger signal is received after `Icu_StartGroupTrigger()` or `Icu_StopGroupTrigger()` is called, the channels in the group are stopped sequentially.

7.4.27 Icu_StartGroupTrigger

Syntax

```
void Icu_StartGroupTrigger  
(  
    Icu_GroupType Group  
)
```

Service ID

0xFD

Parameters (in)

Group - Numeric identifier of the ICU channel group.

Parameters (out)

None

Return value

None

DET errors

- ICU_E_UNINIT - Icu_Init() will be called before calling any other ICU services.
- ICU_E_PARAM_CHANNEL_GROUP - Channel group is invalid.
- ICU_E_CHANNEL_GROUP_CONDITION - The measurement mode is ICU_MODE_TIMESTAMP and the buffer information is not set previously.
- ICU_E_WAITING_TRIGGER - API service is called by the following steps:
 - Icu_StartGroupTrigger() or Icu_StopGroupTrigger() is called.
 - API service is called.
 - The trigger signal by Port_ActTrigger() is received.
- ICU_E_DURING_SLEEP - API service is called in SLEEP mode.

DEM errors

None

Description

This service starts the trigger all ICU channels in provided channel group.

7.4.28 Icu_GetInputLevel

Syntax

```
Icu_LevelType Icu_GetInputLevel  
(  
    Icu_ChannelType Channel  
)
```

Service ID

0xFE

Parameters (in)

Channel - Numeric identifier of the ICU channel.

Parameters (out)

None

Return value

Icu_LevelType - The level obtained from the channel.

DET errors

- ICU_E_UNINIT - Icu_Init() will be called before calling any other ICU services.
- ICU_E_PARAM_CHANNEL - Channel is invalid.

DEM errors

None

Description

This service returns the state of the input pin related to an ICU channel.

7.4.29 Icu_CheckChannelStatus

Syntax

```
Std_ReturnType Icu_CheckChannelStatus  
(  
    Icu_ChannelType Channel,  
    Icu_CheckChannelStatusType* CheckChannelStatusPtr  
)
```

Service ID

0xFF

Parameters (in)

Channel - Numeric identifier of the ICU channel.

Parameters (out)

CheckChannelStatusPtr - Pointer to the location where the status information is stored.

Return value

E_OK or E_NOT_OK

DET errors

- ICU_E_UNINIT - Icu_Init() will be called before calling any other ICU services.
- ICU_E_PARAM_CHANNEL - Channel is invalid
- ICU_E_PARAM_CHECK_STATUS_POINTER - CheckChannelStatusPtr is invalid.

DEM errors

None

Description

This service returns internal status and detects a mismatch between status and register values.

Note: Icu_CheckChannelStatus() may return E_NOT_OK, if the hardware status has not yet changed to the running after Icu_EnableEdgeDetection(), Icu_EnableEdgeCount(), Icu_StartTimestamp(), Icu_StartSignalMeasurement(), or Icu_StartGroupTrigger(). It may occur when the tick frequency of the ICU channel is very slow.

7.5 Required callback functions

7.5.1 DET

If development error detection is enabled, the ICU driver uses the following callback function provided by DET. If you do not use DET, you must implement this function within your application.

7.5.1.1 Det_ReportError

Syntax

```
Std_ReturnType Det_ReportError
(
    uint16 ModuleId,
    uint8 InstanceId,
    uint8 ApiId,
    uint8 ErrorId
)
```

Reentrancy

Reentrant

Parameters (in)

- `ModuleId` - Module ID of the calling module.
- `InstanceId` - Instance ID of the calling module.
- `ApiId` - ID of the API service that calls this function.
- `ErrorId` - ID of the detected development error.

Return value

Always returns `E_OK` (is required for services).

Description

Service for reporting development errors.

7.5.2 DEM

If DEM notifications are enabled, the ICU driver uses the following callback function provided by DEM. If you do not use DEM, you must implement this function within your application.

7.5.2.1 Dem_ReportErrorStatus

Syntax

```
void Dem_ReportErrorStatus
(
    Dem_EventIdType EventId,
    Dem_EventStatusType EventStatus
)
```

Reentrancy

Reentrant

Parameters (in)

- `EventId` - Identification of an event by assigned event ID.
- `EventStatus` – Monitor test result of a given event.

Return value

None

Description

Service for reporting diagnostic events.

7.5.3 Callout functions

7.5.3.1 Error callout API

The AUTOSAR ICU module requires an error callout handler. Each error is reported to this handler; error checking cannot be switched OFF. The name of the function to be called can be configured by `IcuErrorCalloutFunction` parameter.

Syntax

```
void Error_Handler_Name
(
    uint16 ModuleId,
    uint8 InstanceId,
    uint8 ApiId,
    uint8 ErrorId
)
```

Reentrancy

Reentrant

Parameters (in)

- `ModuleId` - Module ID of the calling module.
- `InstanceId` - Instance ID of the calling module.
- `ApiId` - ID of the API service that calls this function.

- `ErrorId` - ID of the detected error.

Return value

None

Description

Service for reporting errors.

Appendix B – Access register table

8

8.1

TCPWM

Register	Bit No.	Access size	Value	Description	Timing	Mask value	Monitoring value
CTRL	31:0	Word (32 bits)	Depends on configuration value or API.	Counter control register	Icu_Init Icu_DeInit Icu_SetMode Icu_SetPrescaler Icu_StartGroupTrigger Icu_StopGroupTrigger Icu_SetActivationCondition Icu_StartSignalMeasurement Icu_StopSignalMeasurement Icu_StopTimestamp Icu_DisableEdgeCount Icu_DisableEdgeDetection ISR	0x473707FF	0x*20000F0 (After Icu_Init, Digit * depends on configuration value.) 0x000000F0 (After Icu_DeInit)
STATUS	31:0	Word (32 bits)	-	Counter status register	Read only	0x00000000 (Monitoring is not needed.)	0x00000000 (Monitoring is not needed.)
COUNTER	31:0	Word (32 bits)	Counter value	Counter count register	Icu_Init Icu_DeInit	0x00000000 (Monitoring is not needed.)	0x00000000 (Monitoring is not needed.)
CC0	31:0	Word (32 bits)	CC0 value	Counter compare/capture 0 register	Icu_StartSignalMeasurement Icu_StartGroupTrigger Icu_GetTimeElapsed Icu_GetDutyCycleValues Icu_GetInputState	0x00000000 (Monitoring is not needed.)	0x00000000 (Monitoring is not needed.)
CC0_BUFF	31:0	Word (32 bits)	CC0_BUFF value	Counter buffered compare/capture 0 register	Icu_StartSignalMeasurement Icu_StartGroupTrigger Icu_GetTimeElapsed Icu_GetDutyCycleValues Icu_GetInputState	0x00000000 (Monitoring is not needed.)	0x00000000 (Monitoring is not needed.)

Appendix B – Access register table

Register	Bit No.	Access size	Value	Description	Timing	Mask value	Monitoring value
CC1	31:0	Word (32 bits)	CC1 value	Counter compare/capture 1 register	Icu_StartSignalMeasurement Icu_StartGroupTrigger Icu_GetTimeElapsed Icu_GetDutyCycleValues Icu_GetInputState	0x00000000 (Monitoring is not needed.)	0x00000000 (Monitoring is not needed.)
CC1_BUFF	31:0	Word (32 bits)	CC1_BUFF value	Counter buffered compare/capture 1 register	Icu_StartSignalMeasurement Icu_StartGroupTrigger Icu_GetTimeElapsed Icu_GetDutyCycleValues Icu_GetInputState	0x00000000 (Monitoring is not needed.)	0x00000000 (Monitoring is not needed.)
PERIOD	31:0	Word (32 bits)	Period value	Counter period register	Icu_Init Icu_DeInit	0xFFFFFFFF (for 32 bit counter) 0xFFFF (for 16 bit counter)	0xFFFFFFFF (for 32 bit counter) 0xFFFF (for 16 bit counter)
PERIOD_BUFF	31:0	Word (32 bits)	-	Counter buffered period register	Not used.	0x00000000 (Monitoring is not needed.)	0x00000000 (Monitoring is not needed.)
LINE_SEL	31:0	Word (32 bits)	-	Counter line selection register	Not used.	0x00000000 (Monitoring is not needed.)	0x00000000 (Monitoring is not needed.)
LINE_SEL_BUFF	31:0	Word (32 bits)	-	Counter buffered line selection register	Not used.	0x00000000 (Monitoring is not needed.)	0x00000000 (Monitoring is not needed.)
DT	31:0	Word (32 bits)	Divider value	Counter PWM dead time register	Icu_Init Icu_DeInit Icu_SetPrescaler	0x000000FF	0x000000** (Digit * depends on configuration value and API.) 0x00000000 (After Gpt_DeInit.)

Appendix B – Access register table

Register	Bit No.	Access size	Value	Description	Timing	Mask value	Monitoring value
TR_CMD	31:0	Word (32 bits)	Depends on configuration value or API.	Counter trigger command register	Icu_EnableEdgeCount Icu_StartTimestamp Icu_StartGroupTrigger Icu_EnableEdgeDetection Icu_SetMode Icu_SetActivationCondition	0x00000029	0x00000000
TR_IN_SEL0	31:0	Word (32 bits)	Depends on configuration value or API.	Counter input trigger selection register 0	Icu_Init Icu_DeInit Icu_StartSignalMeasurement Icu_StopSignalMeasurement Icu_StartGroupTrigger Icu_StopGroupTrigger Icu_DisableEdgeDetection Icu_StopTimestamp Icu_DisableEdgeCount ISR	0xFF00FFFF	0x00000100 (After Icu_DeInit.) 0x**0001** (After Icu_Init, Digit * depends on configuration value.)
TR_IN_SEL1	31:0	Word (32 bits)	0x00000000 Capture1 trigger select value << 8.	Counter input trigger selection register 1	Icu_Init Icu_DeInit	0x0000FFFF	0x00000000 (After Icu_DeInit.) 0x0000**00 (After Icu_Init, Digit * depends on configuration value.)
TR_IN_EDGE_SEL	31:0	Word (32 bits)	Depends on configuration value or API.	Counter input trigger edge selection register	Icu_Init Icu_DeInit Icu_SetActivationCondition Icu_StartSignalMeasurement Icu_StopSignalMeasurement Icu_StartGroupTrigger Icu_StopGroupTrigger Icu_DisableEdgeCount Icu_StopTimestamp Icu_DisableEdgeDetection ISR	0x00000FCC	0x00000FFF (After Icu_DeInit.) 0x00000F*C (After Icu_Init, Digit * depends on configuration value.)

Appendix B – Access register table

Register	Bit No.	Access size	Value	Description	Timing	Mask value	Monitoring value
TR_PWM_CTRL	31:0	Word (32 bits)	-	Counter trigger PWM control register	Not used.	0x00000000 (Monitoring is not needed.)	0x00000000 (Monitoring is not needed.)
TR_OUT_SEL	31:0	Word (32 bits)	0x00000032, 0x00000077, 0x00000073	Counter output trigger selection register	Icu_Init Icu_DeInit	0x00000077	0x00000032 (After Icu_DeInit.) 0x0000007* (After Icu_Init, Digit * depends on configuration value.)
INTR	31:0	Word (32 bits)	0x00000001, 0x00000002, 0x00000003, 0x00000007	Interrupt request register	Icu_EnableEdgeCount Icu_EnableEdgeDetection Icu_EnableOverflowNotificat ion Icu_StartTimestamp Icu_SetMode Icu_GetInputState Icu_GetTimeElapsed Icu_GetDutyCycleValues Icu_StartGroupTrigger ISR	0x00000000 (Monitoring is not needed.)	0x00000000 (Monitoring is not needed.)
INTR_SET	31:0	Word (32 bits)	-	Interrupt set request register	Not used.	0x00000000 (Monitoring is not needed.)	0x00000000 (Monitoring is not needed.)
INTR_MASK	31:0	Word (32 bits)	0x00000000, 0x00000001, 0x00000002, 0x00000003.	Interrupt mask register	Icu_Init Icu_DeInit Icu_StartTimestamp Icu_StopTimestamp Icu_StartGroupTrigger Icu_StopGroupTrigger Icu_SetMode Icu_EnableEdgeCount Icu_DisableEdgeCount Icu_EnableEdgeDetection Icu_DisableEdgeDetection	0x00000000 (Monitoring is not needed.)	0x00000000 (Monitoring is not needed.)

Register	Bit No.	Access size	Value	Description	Timing	Mask value	Monitoring value
					Icu_DisableOverflowNotification Icu_EnableOverflowNotification ISR		
INTR_MASKED	31:0	Word (32 bits)	-	Interrupt masked request register	Read only.	0x00000000 (Monitoring is not needed.)	0x00000000 (Monitoring is not needed.)

8.2 GPIO

Register	Bit No.	Access size	Value	Description	Timing	Mask value	Monitoring value
INTR	31:0	Word (32 bits)	Depends on configuration value or API.	Interrupt request register	Icu_EnableEdgeCount Icu_EnableEdgeDetection Icu_SetActivationCondition Icu_SetMode Icu_StartGroupTrigger ISR	0x00000000 (monitoring is not needed.)	0x00000000 (monitoring is not needed.)
INTR_MASK	31:0	Word (32 bits)	Depends on configuration value or API.	Interrupt set request register	Icu_Init Icu_DeInit Icu_StartGroupTrigger Icu_StopGroupTrigger Icu_SetMode Icu_EnableEdgeCount Icu_DisableEdgeCount Icu_EnableEdgeDetection Icu_DisableEdgeDetection Icu_SetActivationCondition	0x00000000 (Monitoring is not needed.)	0x00000000 (Monitoring is not needed.)
INTR_MASKED	31:0	Word (32 bits)	-	Interrupt mask register	Read only.	0x00000000 (Monitoring is not needed.)	0x00000000 (Monitoring is not needed.)
INTR_CFG	31:0	Word (32 bits)	Depends on configuration value or API.	Interrupt masked request register	Icu_SetActivationCondition Icu_Init Icu_DeInit	0x001C0000	0x00**0000 (After Icu_Init, Digit * depends on configuration value.)

8.3 DW_CH_STRUCT/DW_DESCR_STRUCT

Register	Bit No.	Access size	Value	Description	Timing	Mask value	Monitoring value
INTR	31:0	Word (32 bits)	0x00000001	Clear DW interrupt.	Icu_StartTimestamp Icu_StartGroupTrigger ISR	0x00000000 (monitoring is not needed.)	0x00000000 (monitoring is not needed.)
INTR_MASK	31:0	Word (32 bits)	0x00000000, 0x00000001	DW interrupt mask register	Icu_Init Icu_DeInit Icu_StopTimestamp Icu_StartTimestamp Icu_StartGroupTrigger Icu_StopGroupTrigger ISR	0x00000000 (Monitoring is not needed.)	0x00000000 (Monitoring is not needed.)
INTR_MASKED	31:0	Word (32 bits)	-	DW interrupt masked register	Read only.	0x00000000 (Monitoring is not needed.)	0x00000000 (Monitoring is not needed.)
CH_CTL	31:0	Word (32 bits)	0x00000002, 0x80000002	DW channel control register	Icu_Init Icu_DeInit Icu_StopTimestamp Icu_StartTimestamp Icu_StartGroupTrigger Icu_StopGroupTrigger ISR	0x00000BF7	0x00000002
CH_STATUS	31:0	Word (32 bits)	-	DW channel status register	Read only.	0x00000000 (Monitoring is not needed.)	0x00000000 (Monitoring is not needed.)
CH_IDX	31:0	Word (32 bits)	Channel indices	DW channel current indices register	Icu_StartTimestamp Icu_StartGroupTrigger	0x00000000 (Monitoring is not needed.)	0x00000000 (Monitoring is not needed.)
CH_CURR_PTR	31:0	Word (32 bits)	Address	Channel current descriptor pointer	Icu_StartTimestamp Icu_StartGroupTrigger	0x00000000 (Monitoring is not needed.)	0x00000000 (Monitoring is not needed.)
DESCR_CTL	31:0	Word (32 bits)	Depends on configuration value or API.	Descriptor control	Icu_StartTimestamp Icu_StartGroupTrigger	0x00000000 (Monitoring is not needed.)	0x00000000 (Monitoring is not needed.)

Appendix B – Access register table

Register	Bit No.	Access size	Value	Description	Timing	Mask value	Monitoring value
DESCR_SRC	31:0	Word (32 bits)	Source address.	Base address of source location.	Icu_StartTimestamp Icu_StartGroupTrigger	0x00000000 (Monitoring is not needed.)	0x00000000 (Monitoring is not needed.)
DESCR_DST	31:0	Word (32 bits)	Destination address.	Base address of destination location.	Icu_StartTimestamp Icu_StartGroupTrigger	0x00000000 (Monitoring is not needed.)	0x00000000 (Monitoring is not needed.)
DESCR_X_CTL	31:0	Word (32 bits)	Depends on configuration value or API.	Descriptor X loop control	Icu_StartTimestamp Icu_StartGroupTrigger	0x00000000 (Monitoring is not needed.)	0x00000000 (Monitoring is not needed.)
DESCR_Y_CTL	31:0	Word (32 bits)	Depends on configuration value or API.	Descriptor Y loop control	Icu_StartTimestamp Icu_StartGroupTrigger	0x00000000 (Monitoring is not needed.)	0x00000000 (Monitoring is not needed.)
DESCR_NEXT_PTR	31:0	Word (32 bits)	Next address.	Address of next descriptor in descriptor list.	Icu_StartTimestamp Icu_StartGroupTrigger	0x00000000 (Monitoring is not needed.)	0x00000000 (Monitoring is not needed.)

Revision history

Revision	Issue date	Description of change
**	2018-06-15	New spec.
*A	2018-12-12	<p>Changed referenced hardware documentation.</p> <p>Changed description of PORT Driver in chapter “3.4.2 PORT Driver”.</p> <p>Changed notes of IcuResource in chapter “4.4.1 ICU Channel Configuration”.</p> <p>Added notes of IcuInputTriggerSelection in chapter “4.4.1 ICU Channel Configuration”.</p> <p>Added notes of IcuChannelGroupStartTrigger and IcuChannelGroupStopTrigger in chapter “4.4.2 ICU Channel Group Configuration”.</p> <p>Added new error code ICU_E_WAITING_TRIGGER and ICU_E_DURING_SLEEP in chapter “A.1.3 Constants”.</p> <p>Added description of ICU_E_WAITING_TRIGGER and ICU_E_DURING_SLEEP in chapter “5.15 API Parameter Checking” and “A.1.4 Functions”.</p> <p>Changed description of group trigger setting in chapter “6.4 Triggers”.</p> <p>Changed resource name of TCPWM and Trigger.</p> <p>Added notes of 5.16 Reentrancy.</p> <p>Changed notes of Icu_CheckChannelStatus in chapter “A.1.4 Functions”.</p> <p>Updated B.1.1 TCPWM table.</p>
*B	2019-06-06	<p>Updated AUTOSAR Requirements and Specifications.</p> <p>Updated hardware documentation information.</p>
*C	2019-08-07	<p>Added MPU to glossary.</p> <p>Added notes of chapter “5.14 DMA Transfer”</p>
*D	2020-04-09	<p>Added a chapter “2.6.2 Restriction of Memory Allocation”.</p> <p>Added notes of chapter “5.14 DMA Transfer”.</p> <p>Added description of ICU_E_PARAM_NOTIFY_INTERVAL in chapter “5.15 API Parameter Checking” and “A.1.4 Functions”.</p>
*E	2020-09-04	<p>Changed a memmap file include folder in chapter 2.6.</p> <p>Updated Notes of IcuResource in chapter 4.4.1.</p> <p>Added description of ICU_E_PARAM_NOTIFY_INTERVAL and ICU_E_PARAM_BUFFER_SIZE in chapter “5.15 API Parameter Checking” and “A.1.4 Functions”.</p>
*F	2020-11-19	MOVED TO INFINEON TEMPLATE.
*G	2021-02-10	Added notes in chapters “5.5 Notification” and “5.6 Overflow Notification”.
*H	2021-05-20	Added notes in chapter “5.7 Wakeup”.
*I	2021-06-14	Changed description of IcuDisableEcumWakeupNotification in chapter “2.2.1 Architecture Details” and “4.4.1.4 ICU Wakeup Configuration”.

Revision history

Revision	Issue date	Description of change
*J	2021-08-18	Added a note on DeepSleep mode in 5.7 Wakeup Added a note on Arm® errata in 6.3 Interrupts
*K	2021-09-07	Changed the location of common state data in 2.6.1 Memory allocation keyword .
*L	2021-12-22	Updated to the latest branding guidelines.

Trademarks

All referenced product or service names and trademarks are the property of their respective owners.

Edition 2021-12-22

Published by

**Infineon Technologies AG
81726 Munich, Germany**

**© 2021 Infineon Technologies AG.
All Rights Reserved.**

Do you have a question about this document?

Go to www.cypress.com/support

Document reference

002-23348 Rev. *L

IMPORTANT NOTICE

The information given in this document shall in no event be regarded as a guarantee of conditions or characteristics ("Beschaffheitsgarantie").

With respect to any examples, hints or any typical values stated herein and/or any information regarding the application of the product, Infineon Technologies hereby disclaims any and all warranties and liabilities of any kind, including without limitation warranties of non-infringement of intellectual property rights of any third party.

In addition, any information given in this document is subject to customer's compliance with its obligations stated in this document and any applicable legal requirements, norms and standards concerning customer's products and any use of the product of Infineon Technologies in customer's applications.

The data contained in this document is exclusively intended for technically trained staff. It is the responsibility of customer's technical departments to evaluate the suitability of the product for the intended application and the completeness of the product information given in this document with respect to such application.

For further information on the product, technology, delivery terms and conditions and prices please contact your nearest Infineon Technologies office (www.infineon.com).

WARNINGS

Due to technical requirements products may contain dangerous substances. For information on the types in question please contact your nearest Infineon Technologies office.

Except as otherwise explicitly approved by Infineon Technologies in a written document signed by authorized representatives of Infineon Technologies, Infineon Technologies' products may not be used in any applications where a failure of the product or any consequences of the use thereof can reasonably be expected to result in personal injury.