Document Title	Specification of ICU Driver
Document Category	Software specification (SWS)
Document UID	23
Document Status	published
Part of AUTOSAR Standard	Classic Platform
Part of Standard Release	R26-11

Document Change History			
Date	**Release**	**Changed by**	**Description**
2026-11-26	R26-11	AUTOSAR Release Management	• Removed the `Icu_GetVersionInfo` API and the `IcuGetVersionInfoApi` configuration parameter. • Updated the error classification and API chapter structure. • Clarified ECU configuration table information and updated diagram formatting. • Corrected requirement traceability and non-applicable upstream references.
2025-11-27	R25-11	AUTOSAR Release Management	• No content changes
2024-11-27	R24-11	AUTOSAR Release Management	• Editorial Changes
2023-11-23	R23-11	AUTOSAR Release Management	• No content changes
2022-11-24	R22-11	AUTOSAR Release Management	• Changed [SWS_Icu_00380] to [SWS_Icu_NA_00380]
2021-11-25	R21-11	AUTOSAR Release Management	• Clean up of `Icu_ConfigType` related requirements regarding the data structure. • Correct sequence diagrams in chapter 9.4

▽

△

2020-11-30	R20-11	AUTOSAR Release Management	• Error table cleanup in Error classification • Removed "7.y Error Detection" • Moved [SWS_Icu_00022] to new requirement number in "8.3 Function definitions"
2019-11-28	R19-11	AUTOSAR Release Management	• Incorporation of validation results • Changed Document Status from Final to published
2018-11-31	4.4.0	AUTOSAR Release Management	• MCAL Multicore Distribution (Draft) • Header File Cleanup
2017-12-08	4.3.1	AUTOSAR Release Management	• Removed [SWS_Icu_00116] and [SWS_Icu_00190] • Added [SRS_BSW_00450] to the list of non applicable requirements • Renamed "default error" to "development error" • [SWS_Icu_00201]: Icu_StartTimestamp: Parameter (IN): Icu_ValueType* BufferPtr changed to Parameters (OUT) type • Changed `ICU_E_NOT_STARTED` from development error to runtime error • Editorial Changes
2016-11-30	4.3.0	AUTOSAR Release Management	• Removed chapter "10.2.1 Variants" • Changed upper multiplicity of the ICU_EcuModuleDef to 1 in fugure of section 10.2.2 • Removed config parameter IcuIndex ([ECUC_Icu_00221]) from IcuGeneral section 10.2.3 and in figure of section 10.2.3 • Requirement ID [SWS_Icu_00383] given to additional test "EcuM_WakeupSourceType shal be imported from EcuM_Types.h" • Removed requirement [SWS_Icu_00346] • Editorial changes

▽

△

2015-07-31	4.2.2	AUTOSAR Release Management	• Editorial changes • DET renamed from "Development Error Tracer" to "Default Error Tracer" • All references to obsolete [SWS_Icu_00048] removed from the document
2014-10-31	4.2.1	AUTOSAR Release Management	• IcuChannelId: postBuildVariantValue set to false • SWS IDs with respect to NULL_PTR check for Icu_Init removed • ICU_E_PARAM_POINTER and ICU_E_PARAM_BUFFER_PTR removed from Error classification
2013-10-31	4.1.2	AUTOSAR Release Management	• ICU00354 - Check for a valid notification interval rephrased • ICU078 - Removed the sentence "This is done by the hardware." from the note • ICU295 - Removed ICU_ACTIVE_TIME from the range of enumeration Icu_SignalMeasurementPropertyType • Editorial changes • Removed chapter(s) on change documentation
2013-03-15	4.1.1	AUTOSAR Administration	• Modified the scope of the parameters from ECU/Module to local • Reworked according to the new SWS_BSWGeneral • Changed MemMap.h to Icu_MemMap.h
2011-12-22	4.0.3	AUTOSAR Administration	• Corrected Type errors • Updated description of Icu_IndexType

▽

△

2010-09-30	3.1.5	AUTOSAR Administration	• Services 'Icu_DisableEdgeDetection' and 'Icu_EnableEdgeDetection' were added. • Configuration parameters 'IcuEdgeDetectApi' and 'IcuWakeupFunctionalityApi' has been added • Definition of 'duty cycle' has been corrected • Corrected values of the parameter 'Icu_SignalMeasurementPropertyType'
2008-08-13	3.1.1	AUTOSAR Administration	• Legal disclaimer revised
2008-02-01	3.0.2	AUTOSAR Administration	• The code file structure of the module was completely reworked • The following requirements were added: [SWS_Icu_00088], [SWS_Icu_00220], [SWS_Icu_00221], [SWS_Icu_00228], [SWS_Icu_00229] • The flow charts related to the ECU Wake-Up moved to the • SWS document of the ECU State Manager • Document meta information extended • Small layout adaptations made
2007-12-21	3.0.1	AUTOSAR Administration	• Default start edge is now used for edge configuration • Enable and Disable Notification can now be used for Timestamp functionality • Edge detection functionality is now pre complie time configurable On/Off • Legal disclaimer revised • Release notes added • "Advice for users" revised • "Revision Information" added

▽

△

2006-05-16	2.0	AUTOSAR Administration	• Added the following services – Icu_SetActivationCondition – Icu_StartTimestamp – Icu_StopTimestamp – Icu_GetTimestampIndex – Icu_ResetEdgeCount – Icu_EnableEdgeCount – Icu_DisableEdgeCount – Icu_GetEdgeNumbers – Icu_GetTimeElapsed – Icu_GetDutyCycleValues – Icu_GetVersionInfo
2005-05-31	1.0	AUTOSAR Administration	• Initial Release

Disclaimer

This work (specification and/or software implementation) and the material contained in it, as released by AUTOSAR, is for the purpose of information only. AUTOSAR and the companies that have contributed to it shall not be liable for any use of the work.

The material contained in this work is protected by copyright and other types of intellectual property rights. The commercial exploitation of the material contained in this work requires a license to such intellectual property rights.

This work may be utilized or reproduced without any modification, in any form or by any means, for informational purposes only. For any other purpose, no part of the work may be utilized or reproduced, in any form or by any means, without permission in writing from the publisher.

The work has been developed for automotive applications only. It has neither been developed, nor tested for non-automotive applications.

The word AUTOSAR and the AUTOSAR logo are registered trademarks.

Table of Contents

1 Introduction and functional overview 10
2 Acronyms and Abbreviations 11
3 Related documentation 12
3.1 Input documents & related standards and norms . . . 12
3.2 Related specification . . . 12
4 Constraints and assumptions 13
4.1 Limitations . . . 13
4.2 Applicability to car domains . . . 13
5 Dependencies to other modules 14
5.1 Module DET (Default Error Tracer) . . . 14
5.2 Module MCU . . . 14
5.3 OS (Operating System) . . . 14
5.4 Module PORT . . . 14
5.5 Module EcuM . . . 14
6 Requirements Tracing 15
7 Functional specification 19
7.1 General behavior . . . 19
7.1.1 Background & Rationale . . . 19
7.1.2 Requirements . . . 19
7.1.3 Time Unit Ticks . . . 20
7.1.3.1 Background & Rationale . . . 20
7.1.3.2 Requirements . . . 20
7.2 Error Classification . . . 20
7.2.1 Development Errors . . . 21
7.2.2 Runtime Errors . . . 21
7.2.3 Production Errors . . . 21
8 API specification 22
8.1 Imported types . . . 22
8.2 Type definitions . . . 22
8.2.1 Icu_ModeType . . . 22
8.2.2 Icu_ChannelType . . . 22
8.2.3 Icu_InputStateType . . . 23
8.2.4 Icu_ConfigType . . . 23
8.2.5 Icu_ActivationType . . . 24
8.2.6 Icu_ValueType . . . 24
8.2.7 Icu_DutyCycleType . . . 24
8.2.8 Icu_IndexType . . . 25
8.2.9 Icu_EdgeNumberType . . . 25

8.2.10 Icu_MeasurementModeType . . . 26
8.2.11 Icu_SignalMeasurementPropertyType . . . 26
8.2.12 Icu_TimestampBufferType . . . 26
8.3 Function definitions . . . 27
8.3.1 Icu_Init . . . 27
8.3.2 Icu_DeInit . . . 29
8.3.3 Icu_SetMode . . . 30
8.3.4 Icu_DisableWakeup . . . 32
8.3.5 Icu_EnableWakeup . . . 33
8.3.6 Icu_CheckWakeup . . . 34
8.3.7 Icu_SetActivationCondition . . . 35
8.3.8 Icu_DisableNotification . . . 36
8.3.9 Icu_EnableNotification . . . 37
8.3.10 Icu_GetInputState . . . 38
8.3.11 Icu_StartTimestamp . . . 40
8.3.12 Icu_StopTimestamp . . . 42
8.3.13 Icu_GetTimestampIndex . . . 43
8.3.14 Icu_ResetEdgeCount . . . 44
8.3.15 Icu_EnableEdgeCount . . . 45
8.3.16 Icu_EnableEdgeDetection . . . 46
8.3.17 Icu_DisableEdgeDetection . . . 47
8.3.18 Icu_DisableEdgeCount . . . 48
8.3.19 Icu_GetEdgeNumbers . . . 49
8.3.20 Icu_StartSignalMeasurement . . . 50
8.3.21 Icu_StopSignalMeasurement . . . 51
8.3.22 Icu_GetTimeElapsed . . . 52
8.3.23 Icu_GetDutyCycleValues . . . 54
8.3.24 Icu_DisableNotificationAsync . . . 56
8.3.25 Icu_EnableNotificationAsync . . . 56
8.4 Scheduled functions . . . 56
8.5 Expected interfaces . . . 56
8.5.1 Mandatory Interfaces . . . 57
8.5.2 Optional Interfaces . . . 57
8.5.3 Configurable interfaces . . . 58
9 Sequence diagrams 61
9.1 Icu_Init . . . 61
9.2 Icu_DeInit . . . 61
9.3 Check Wakeup Events . . . 61
9.4 Icu_SetMode . . . 62
9.5 Icu_DisableWakeup . . . 66
9.6 Icu_EnableWakeup . . . 67
9.7 Icu_SetActivationCondition . . . 68
9.8 Icu_DisableNotification . . . 69

9.9 Icu_EnableNotification . . . 70
9.10 Icu_GetInputState . . . 72
9.11 Icu Timestamping . . . 72
9.12 Icu Edge Counting . . . 75
9.13 Icu_GetTimeElapsed . . . 76
9.14 Icu_GetDutyCycleValues . . . 79
9.15 Icu_DisableNotificationAsync . . . 80
9.16 Icu_SignalNotification and Icu_GetInputState . . . 81
9.17 Icu_EnableNotificationAsync . . . 82

10 Configuration specification 83
10.1 How to read this chapter . . . 83
10.2 Containers and configuration parameters . . . 83
10.2.1 Icu . . . 83
10.2.2 IcuGeneral . . . 84
10.2.3 IcuOptionalApis . . . 86
10.2.4 IcuChannel . . . 91
10.2.5 IcuSignalEdgeDetection . . . 95
10.3 Published Information . . . 100

A Not applicable requirements 101

B Change history of AUTOSAR traceable items 102
B.1 Traceable item history of this document according to AUTOSAR Release R26-11 . . . 102
B.1.1 Added Specification Items in R26-11 . . . 102
B.1.2 Changed Specification Items in R26-11 . . . 102
B.1.3 Deleted Specification Items in R26-11 . . . 102
B.1.4 Added Constraints in R26-11 . . . 103
B.1.5 Changed Constraints in R26-11 . . . 103
B.1.6 Deleted Constraints in R26-11 . . . 103
B.2 Traceable item history of this document according to AUTOSAR Release R25-11 . . . 103
B.2.1 Added Specification Items in R25-11 . . . 103
B.2.2 Changed Specification Items in R25-11 . . . 103
B.2.3 Deleted Specification Items in R25-11 . . . 103
B.2.4 Added Constraints in R25-11 . . . 103
B.2.5 Changed Constraints in R25-11 . . . 103
B.2.6 Deleted Constraints in R25-11 . . . 103

1 Introduction and functional overview

This specification specifies the functionality, API and configuration of the AUTOSAR Basic Software module ICU driver.

The ICU driver is a module using the input capture unit (ICU) for demodulation of a PWM signal, counting pulses, measuring of frequency and duty cycle, generating simple interrupts and also wakeup interrupts.

The ICU driver provides services for

- Signal edge notification
- Controlling wakeup interrupts
- Periodic signal time measurement
- Edge time stamping, usable for the acquisition of non-periodic signals
- Edge counting

2 Acronyms and Abbreviations

The glossary below includes acronyms and abbreviations relevant to the ICU driver module that are not included in the [1, AUTOSAR glossary].

Abbreviation / Acronym:	Description:
Active Time	This depends on the starting edge of the signal to be captured. • Start edge = falling edge => Active Time = Low Time • Start edge = rising edge => Active Time = High Time • Start edge = both edges => Active Time = High Time (if rising edge occurs initially) • Start edge = both edges => Active Time = Low Time (if falling edge occurs initially)
DEM	Diagnostic Event Manager [2]
DET	Default Error Tracer [3]
EcuM	ECU State Manager [4]
Enumeration	This can be in "C" programming language an enum or a #define.
ICU	Input Capture Unit (not Intensive Care Unit)
ICU Channel	Represents a logical ICU entity bound to one input signal and the hardware resources for the configured measurement mode.
ICU State	Logical input state of an ICU Channel. It can be ICU_ACTIVE or ICU_IDLE.
ICU_ACTIVE	Input state of an ICU Channel, an activation edge has been detected.
ICU_IDLE	Input state of an ICU Channel, no activation edge has been detected since the last call of Icu_GetInputState() or Icu_Init().
Symbolic name for a channel	A symbolic name is a substitution of a handle with a name. With this handle each channel and its related properties can be found within the configuration structure. In "C" programming language this can be realized e.g. by #defines and enums.
Wakeup event	A wakeup event is understood as a pattern of edges, which will lead to the wake up of this driver. Nevertheless the decision whether a pattern is valid or not isn't done by this driver. This shall be done by an upper layer.

Table 2.1: Acronyms and abbreviations used in the scope of this Document

3 Related documentation

3.1 Input documents & related standards and norms

[1] Glossary
AUTOSAR_FO_TR_Glossary

[2] Specification of Diagnostic Event Manager
AUTOSAR_CP_SWS_DiagnosticEventManager

[3] Specification of Default Error Tracer
AUTOSAR_CP_SWS_DefaultErrorTracer

[4] Specification of ECU State Manager
AUTOSAR_CP_SWS_ECUStateManager

[5] General Specification of Basic Software Modules
AUTOSAR_CP_SWS_BSWGeneral

[6] Specification of Operating System
AUTOSAR_CP_SWS_OS

[7] Specification of Port Driver
AUTOSAR_CP_SWS_PortDriver

[8] Requirements on ICU Driver
AUTOSAR_CP_RS_ICUDriver

[9] General Requirements on Basic Software Modules
AUTOSAR_CP_RS_BSWGeneral

[10] General Requirements on SPAL
AUTOSAR_CP_RS_SPALGeneral

3.2 Related specification

AUTOSAR provides a General Specification on Basic Software modules [5, SWS BSW General], which is also valid for ICU driver.

Thus, the specification SWS BSW General shall be considered as additional and required specification for ICU driver.

4 Constraints and assumptions

4.1 Limitations

No limitations.

4.2 Applicability to car domains

No restrictions.

5 Dependencies to other modules

5.1 Module DET (Default Error Tracer)

The detailed description of the detected errors can be found in chapter 7.2 and chapter 8.

5.2 Module MCU

The ICU driver depends on the system clock, prescaler(s) and PLL. Hence the length of an ICU timer tick depends on the clock settings made in the module MCU.

The ICU driver will not take care of setting the registers which configure the global clock, global prescaler(s) and PLL in its Init function. This has to be done by the MCU module. The ICU driver only configures local (ICU peripheral specific) clocks, prescalers and so on.

5.3 OS (Operating System)

The ICU driver uses interrupts and therefore there is a dependency on the OS [6] which configures the interrupt sources. It will provide the call-back functions only.

The ICU driver will not take care of setting the registers for interrupt association in its Init function. The overall assignment and activation of the interrupt system is done by the Operating System.

5.4 Module PORT

The configuration of port pins used for the ICU as inputs is done by the PORT driver [7]. Hence the PORT driver has to be initialized prior to the use of ICU functions. Otherwise ICU functions will exhibit undefined behaviour.

5.5 Module EcuM

[SWS_Icu_00244] ⌈The ICU driver will do the reporting of wakeup interrupts to the EcuM.⌋

6 Requirements Tracing

The following tables reference the requirements specified in [8], [9], [10] and links to the fulfillment of these. Please note that if column “Satisfied by” is empty for a specific requirement this means that this requirement is not fulfilled by this document.

Requirement	Description	Satisfied by
[SRS_BSW_00101]	The Basic Software Module shall be able to initialize variables and hardware in a separate initialization function	[SWS_Icu_00006]
[SRS_BSW_00171]	Optional functionality of a Basic-SW component that is not required in the ECU shall be configurable at pre-compile-time	[SWS_Icu_00092] [SWS_Icu_00095] [SWS_Icu_00096] [SWS_Icu_00097] [SWS_Icu_00098] [SWS_Icu_00099] [SWS_Icu_00100] [SWS_Icu_00101] [SWS_Icu_00102] [SWS_Icu_00103] [SWS_Icu_00104] [SWS_Icu_00105] [SWS_Icu_00106] [SWS_Icu_00122]
[SRS_BSW_00323]	All AUTOSAR Basic Software Modules shall check passed API parameters for validity	[SWS_Icu_00024] [SWS_Icu_00043] [SWS_Icu_00125] [SWS_Icu_00385] [SWS_Icu_00386] [SWS_Icu_00387] [SWS_Icu_00388] [SWS_Icu_00389] [SWS_Icu_00390] [SWS_Icu_00391] [SWS_Icu_00392] [SWS_Icu_00393] [SWS_Icu_00394] [SWS_Icu_00395] [SWS_Icu_00396] [SWS_Icu_00397] [SWS_Icu_00398] [SWS_Icu_00399] [SWS_Icu_00400] [SWS_Icu_00401] [SWS_Icu_00402] [SWS_Icu_00403] [SWS_Icu_00404]
[SRS_BSW_00336]	Basic SW module shall be able to shutdown	[SWS_Icu_00037]
[SRS_BSW_00344]	BSW Modules shall support link-time configuration	[SWS_Icu_00006]
[SRS_BSW_00359]	Callback Function Return Types for AUTOSAR BSW	[SWS_Icu_00187]
[SRS_BSW_00369]	All AUTOSAR Basic Software Modules shall not return specific development error codes via the API	[SWS_Icu_00049]
[SRS_BSW_00375]	Basic Software Modules shall report wake-up reasons	[SWS_Icu_00359]
[SRS_BSW_00384]	The Basic Software Module specifications shall specify at least in the description which other modules they require	[SWS_Icu_00131]
[SRS_BSW_00386]	The BSW shall specify the configuration and conditions for detecting an error	[SWS_Icu_00220]
[SRS_BSW_00404]	BSW Modules shall support post-build configuration	[SWS_Icu_00006]
[SRS_BSW_00405]	BSW Modules shall support multiple configuration sets	[SWS_Icu_00006]



△

Requirement	Description	Satisfied by
[SRS_BSW_00406]	API handling in uninitialized state	[SWS_Icu_00385] [SWS_Icu_00386] [SWS_Icu_00387] [SWS_Icu_00388] [SWS_Icu_00389] [SWS_Icu_00390] [SWS_Icu_00391] [SWS_Icu_00392] [SWS_Icu_00393] [SWS_Icu_00394] [SWS_Icu_00395] [SWS_Icu_00396] [SWS_Icu_00397] [SWS_Icu_00398] [SWS_Icu_00399] [SWS_Icu_00400] [SWS_Icu_00401] [SWS_Icu_00402] [SWS_Icu_00403] [SWS_Icu_00404]
[SRS_BSW_00410]	Compiler switches shall have defined values	[SWS_Icu_00055] [SWS_Icu_00063] [SWS_Icu_00090] [SWS_Icu_00092] [SWS_Icu_00095] [SWS_Icu_00096] [SWS_Icu_00097] [SWS_Icu_00099] [SWS_Icu_00100] [SWS_Icu_00101] [SWS_Icu_00102] [SWS_Icu_00103] [SWS_Icu_00104] [SWS_Icu_00105] [SWS_Icu_00106] [SWS_Icu_00122]
[SRS_Icu_12305]	The ICU driver shall allow to enable/ disable the notification for an ICU channel at runtime	[SWS_Icu_00009] [SWS_Icu_00010] [SWS_Icu_00042] [SWS_Icu_00044]
[SRS_Icu_12369]	The ICU driver shall provide notification for an ICU Channel at the configured signal edge	[SWS_Icu_00021]
[SRS_Icu_12370]	The ICU driver shall provide a service for selecting the sleep mode	[SWS_Icu_00008]
[SRS_Icu_12371]	The ICU driver shall provide a synchronous service that returns the status of the ICU input	[SWS_Icu_00030] [SWS_Icu_00031] [SWS_Icu_00032]
[SRS_Icu_12407]	After initialization of the ICU driver all notifications shall be disabled	[SWS_Icu_00040] [SWS_Icu_00061]
[SRS_Icu_12408]	The ICU driver shall provide a service for enabling / disabling the wake-up capability of single ICU channels	[SWS_Icu_00013] [SWS_Icu_00014]
[SRS_Icu_12425]	For each ICU Channel the 'property' that could be measured shall be configurable	[SWS_Icu_00088]
[SRS_Icu_12429]	The ICU Driver shall provide the functionality to deinitialize ICU channels to their power on reset state	[SWS_Icu_00036]
[SRS_Icu_12430]	The ICU driver shall provide an asynchronous service for starting the timestamp measurement on an ICU channel	[SWS_Icu_00063] [SWS_Icu_00066]
[SRS_Icu_12431]	The ICU driver shall provide a synchronous service for canceling the timestamp measurement on an ICU channel	[SWS_Icu_00067]
[SRS_Icu_12432]	Edge counting service shall be available on an ICU channel	[SWS_Icu_00078]
[SRS_Icu_12433]	Edge counting service on a ICU channel shall be disabled	[SWS_Icu_00079]
[SRS_Icu_12434]	Edge counting read service shall be available	[SWS_Icu_00080]
[SRS_Icu_12435]	The elapsed Signal High Time for each ICU Channel shall be provided	[SWS_Icu_00082]

▽

△

Requirement	Description	Satisfied by
[SRS_Icu_12436]	The High time and Period Time of an ICU Channel shall be provided	[SWS_Icu_00084]
[SRS_Icu_12438]	The ICU driver shall provide the functionality to capture timer values on configurable edges to an external buffer	[SWS_Icu_00063]
[SRS_Icu_12439]	Edges of a signal shall be counted by the ICU	[SWS_Icu_00072] [SWS_Icu_00073] [SWS_Icu_00074]
[SRS_Icu_12442]	The elapsed Signal Low Time for each ICU Channel shall be provided	[SWS_Icu_00081]
[SRS_Icu_12443]	The elapsed Period Time for an ICU Channel shall be provided	[SWS_Icu_00083]
[SRS_Icu_12444]	The ICU driver shall provide a notification if the number of requested timestamps are acquired	[SWS_Icu_00215]
[SRS_Icu_12453]	The Timestamp index service shall be provided by ICU	[SWS_Icu_00071]
[SRS_Icu_12456]	If linear buffer handling is configured, the driver shall stop capturing timer values, when the end of the buffer is reached	[SWS_Icu_00065]
[SRS_Icu_13100]	Reseting the value of counted edges of an ICU channel shall be available	[SWS_Icu_00072]
[SRS_SPAL_00157]	All drivers and handlers of the AUTOSAR Basic Software shall implement notification mechanisms of drivers and handlers	[SWS_Icu_00021] [SWS_Icu_00030]
[SRS_SPAL_12056]	All driver modules shall allow the static configuration of notification mechanism	[SWS_Icu_00018]
[SRS_SPAL_12057]	All driver modules shall implement an interface for initialization	[SWS_Icu_00006] [SWS_Icu_00040] [SWS_Icu_00060] [SWS_Icu_00061]
[SRS_SPAL_12063]	All driver modules shall only support raw value mode	[SWS_Icu_00063] [SWS_Icu_00081] [SWS_Icu_00082] [SWS_Icu_00083]
[SRS_SPAL_12064]	All driver modules shall raise an error if the change of the operation mode leads to degradation of running operations	[SWS_Icu_00133]
[SRS_SPAL_12067]	All driver modules shall set their wake-up conditions depending on the selected operation mode	[SWS_Icu_00008] [SWS_Icu_00011] [SWS_Icu_00012]
[SRS_SPAL_12069]	All drivers of the SPAL that wake up from a wake-up interrupt shall report the wake-up reason	[SWS_Icu_00055] [SWS_Icu_00056] [SWS_Icu_00057]
[SRS_SPAL_12075]	All drivers with random streaming capabilities shall use application buffers	[SWS_Icu_00063]
[SRS_SPAL_12125]	All driver modules shall only initialize the configured resources	[SWS_Icu_00054]
[SRS_SPAL_12129]	The ISRs shall be responsible for resetting the interrupt flags and calling the according notification function	[SWS_Icu_00119]
[SRS_SPAL_12163]	All driver modules shall implement an interface for de-initialization	[SWS_Icu_00036] [SWS_Icu_00037]

▽

△

Requirement	Description	Satisfied by
[SRS_SPAL_12169]	All driver modules that provide different operation modes shall provide a service for mode selection	[SWS_Icu_00008]
[SRS_SPAL_12448]	All driver modules shall have a specific behavior after a development error detection	[SWS_Icu_00049] [SWS_Icu_00107] [SWS_Icu_00108]
[SRS_SPAL_12461]	Specific rules regarding initialization of controller registers shall apply to all driver implementations	[SWS_Icu_00006] [SWS_Icu_00051] [SWS_Icu_00052] [SWS_Icu_00053] [SWS_Icu_00128] [SWS_Icu_00129]

Table 6.1: Requirements Tracing

7 Functional specification

7.1 General behavior

7.1.1 Background & Rationale

To ensure data consistency re-entrant code shall be provided.

7.1.2 Requirements

[SWS_Icu_00050] ⌈The Icu module functions for different channel numbers shall be re-entrant, except for:

- `Icu_Init`
- `Icu_DeInit`
- `Icu_SetMode`

⌋

[SWS_Icu_00149] ⌈The Icu module's environment shall check the integrity if several calls for the same ICU channel are used during runtime in different tasks or ISRs.⌋

[SWS_Icu_00150] ⌈The Icu module shall not check the integrity if several calls for the same ICU channel are used during runtime in different tasks or ISRs.⌋

[SWS_Icu_00258] ⌈The Icu module has 2 modes:

- `ICU_MODE_NORMAL`
- `ICU_MODE_SLEEP`

⌋

[SWS_Icu_00011]

Upstream requirements: SRS_SPAL_12067

⌈In `ICU_MODE_NORMAL` mode all notifications are available as configured by service `Icu_SetActivationCondition` or IcuDefaultStartEdge.⌋

[SWS_Icu_00259] ⌈In `ICU_MODE_NORMAL` mode all notifications are available as selected by the `Icu_DisableNotification` and `Icu_EnableNotification` services before or after the call of `Icu_SetMode`.⌋

[SWS_Icu_00012]

Upstream requirements: SRS_SPAL_12067

⌈In `ICU_MODE_SLEEP` mode only those wakeup events are available which are configured as wakeup capable, enabled via `Icu_EnableWakeup` after `Icu_Init` and which are not disabled via service `Icu_DisableWakeup`⌋

[SWS_Icu_00260] ⌈In `ICU_MODE_SLEEP` mode all other interrupts handled by this module are disabled and must not lead to an exit from the reduced power mode state (e.g. idle, halt) of the MCU if the event occurs.⌋

[SWS_Icu_00261] ⌈All channels are stopped except those channels

- which have been configured as wakeup capable and
- which were explicitly enabled by the call of `Icu_EnableWakeup`.

⌋

[SWS_Icu_00088]

Upstream requirements: SRS_Icu_12425

⌈The module Icu shall allow the configuration per channel of the definition on which edge the period starts.⌋

7.1.3 Time Unit Ticks

7.1.3.1 Background & Rationale

To get times out of register values it is necessary to know the oscillator frequency, prescalers and so on. Since these settings are made in the MCU module and/or in other modules it is not possible to calculate such times.

Hence the conversions between time and ticks shall be part of an upper layer.

7.1.3.2 Requirements

All time units used within the API services of the ICU driver are unit ticks.

7.2 Error Classification

Chapter [5, General Specification of Basic Software Modules] 7.2 *"Error Handling"* describes the error handling of the Basic Software in detail. Above all, it constitutes a classification scheme consisting of five error types which may occur in BSW modules.

Based on this foundation, the following section specifies particular errors arranged in the respective subsections below.

7.2.1 Development Errors

[SWS_Icu_00382] Definition of development errors in module Icu ⌈

Type of error	*Related error code*	*Error value*
API IS called with invalid pointer.	ICU_E_PARAM_POINTER	0x0A
API service used with an invalid channel identifier or channel was not configured for the functionality of the calling API.	ICU_E_PARAM_CHANNEL	0x0B
API service used with an invalid or not feasible activation.	ICU_E_PARAM_ACTIVATION	0x0C
Init function failed.	ICU_E_INIT_FAILED	0x0D
API service used with an invalid buffer size.	ICU_E_PARAM_BUFFER_SIZE	0x0E
API service Icu_SetMode used with an invalid mode.	ICU_E_PARAM_MODE	0x0F
API service used without module initialization.	ICU_E_UNINIT	0x14
API service Icu_SetMode is called while a running operation.	ICU_E_BUSY_OPERATION	0x16
API Icu_Init service is called and when the ICU driver and the Hardware are already initialized.	ICU_E_ALREADY_INITIALIZED	0x17
API Icu_StartTimeStamp is called and the parameter NotifyInterval is invalid (e.g.”0”, Notify Interval < 1)	ICU_E_PARAM_NOTIFY_INTERVAL	0x18

⌋

7.2.2 Runtime Errors

[SWS_Icu_91004] Definition of runtime errors in module Icu ⌈

Type of error	*Related error code*	*Error value*
API service Icu_StopTimestamp called on a channel which was not started or already stopped	ICU_E_NOT_STARTED	0x15

⌋

7.2.3 Production Errors

There are no production errors.

8 API specification

8.1 Imported types

In this chapter all types included from the following modules are listed:

[SWS_Icu_00276] Definition of imported datatypes of module Icu ⌈

Module	*Header File*	*Imported Type*
EcuM	EcuM.h	EcuM_WakeupSourceType
Std	Std_Types.h	Std_ReturnType

⌋

8.2 Type definitions

8.2.1 Icu_ModeType

[SWS_Icu_00277] Definition of datatype Icu_ModeType ⌈

Name	Icu_ModeType		
Kind	Enumeration		
Range	ICU_MODE_NORMAL	0x00	Normal operation, all used interrupts are enabled according to the notification requests.
	ICU_MODE_SLEEP	0x01	Reduced power operation. In sleep mode only those notifications are available which are configured as wakeup capable.
Description	Allow enabling / disabling of all interrupts which are not required for the ECU wakeup.		
Available via	Icu.h		

⌋

8.2.2 Icu_ChannelType

[SWS_Icu_00278] Definition of datatype Icu_ChannelType ⌈

Name	Icu_ChannelType		
Kind	Type		
Derived from	uint		
Range	--	–	This is implementation specific but not all values may be valid within the type. This type shall be chosen in order to have the most efficient implementation on a specific microcontroller platform.
Description	Numeric identifier of an ICU channel		

▽

△

Available via	Icu.h

⌋

8.2.3 Icu_InputStateType

[SWS_Icu_00279] Definition of datatype Icu_InputStateType ⌈

Name	Icu_InputStateType		
Kind	Enumeration		
Range	ICU_ACTIVE	0x00	An activation edge has been detected
	ICU_IDLE	0x01	No activation edge has been detected since the last call of Icu_GetInputState() or Icu_Init().
Description	Input state of an ICU channel		
Available via	Icu.h		

⌋

8.2.4 Icu_ConfigType

[SWS_Icu_00280] Definition of datatype Icu_ConfigType ⌈

Name	Icu_ConfigType	
Kind	Structure	
Elements	--	
	Type	–
	Comment	Hardware and implementation dependent structure. The contents of the initialization data structure are microcontroller specific.
Description	This type contains initialization data.	
Available via	Icu.h	

⌋

[SWS_Icu_00287] ⌈If in the definition for each Channel within the `Icu_ConfigType` the channel is configured as wakeup capable then the function called for validation of wakeup reason shall be EcuM_CheckWakeup.⌋

8.2.5 Icu_ActivationType

[SWS_Icu_00289] Definition of datatype Icu_ActivationType ⌈

Name	Icu_ActivationType		
Kind	Enumeration		
Range	ICU_RISING_EDGE	0x00	An appropriate action shall be executed when a rising edge occurs on the ICU input signal.
	ICU_FALLING_EDGE	0x01	An appropriate action shall be executed when a falling edge occurs on the ICU input signal.
	ICU_BOTH_EDGES	0x02	An appropriate action shall be executed when either a rising or falling edge occur on the ICU input signal.
Description	Definition of the type of activation of an ICU channel.		
Available via	Icu.h		

⌋

8.2.6 Icu_ValueType

[SWS_Icu_00290] Definition of datatype Icu_ValueType ⌈

Name	Icu_ValueType		
Kind	Type		
Derived from	uint		
Range	0 ... <width of the timer register>	–	Implementation specific. This type shall be chosen in order to have the most efficient implementation on a specific microcontroller platform.
Description	Width of the buffer for timestamp ticks and measured elapsed timeticks.		
Available via	Icu.h		

⌋

8.2.7 Icu_DutyCycleType

[SWS_Icu_00291] Definition of datatype Icu_DutyCycleType ⌈

Name	Icu_DutyCycleType	
Kind	Structure	
Elements	ActiveTime	
	Type	Icu_ValueType
	Comment	This shall be the coherent active-time measured on a channel
	PeriodTime	
	Type	Icu_ValueType
	Comment	This shall be the coherent period-time measured on a channel
Description	Type which shall contain the values, needed for calculating duty cycles.	

▽

△

Available via	Icu.h

⌋

8.2.8 Icu_IndexType

[SWS_Icu_00292] Definition of datatype Icu_IndexType ⌈

Name	Icu_IndexType		
Kind	Type		
Derived from	uint		
Range	--	–	Implementation specific. This type shall be chosen in order to have the most efficient implementation on a specific microcontroller platform.
Description	Type, to abstract the return value of the service Icu_GetTimestampIndex().Since circular buffer handling is supported and Icu_GetTimestampIndex can return '0' as a legally true value (not as an error according to ICU107 and ICU135), Icu_IndexType may be implemented to have values 1..xyz.		
Available via	Icu.h		

⌋

8.2.9 Icu_EdgeNumberType

[SWS_Icu_00293] Definition of datatype Icu_EdgeNumberType ⌈

Name	Icu_EdgeNumberType		
Kind	Type		
Derived from	uint		
Range	--	–	Implementation specific. This type shall be chosen in order to have the most efficient implementation on a specific microcontroller platform.
Description	Type, to abstract the return value of the service Icu_GetEdgeNumbers().		
Available via	Icu.h		

⌋

8.2.10 Icu_MeasurementModeType

[SWS_Icu_00294] Definition of datatype Icu_MeasurementModeType ⌈

Name	Icu_MeasurementModeType		
Kind	Enumeration		
Range	ICU_MODE_SIGNAL_EDGE_DETECT	0x00	Mode for detecting edges
	ICU_MODE_SIGNAL_MEASUREMENT	0x01	Mode for measuring different times between various configurable edges
	ICU_MODE_TIMESTAMP	0x02	Mode for capturing timer values on configurable edges
	ICU_MODE_EDGE_COUNTER	0x03	Mode for counting edges on configurable edges
Description	Definition of the measurement mode type		
Available via	Icu.h		

⌋

8.2.11 Icu_SignalMeasurementPropertyType

[SWS_Icu_00295] Definition of datatype Icu_SignalMeasurementPropertyType ⌈

Name	Icu_SignalMeasurementPropertyType		
Kind	Enumeration		
Range	ICU_LOW_TIME	0x00	The channel is configured for reading the elapsed Signal Low Time
	ICU_HIGH_TIME	0x01	The channel is configured for reading the elapsed Signal High Time
	ICU_PERIOD_TIME	0x02	The channel is configured for reading the elapsed Signal Period Time
	ICU_DUTY_CYCLE	0x03	The channel is configured to read values which are needed for calculating the duty cycle (coherent Active and Period Time).
Description	Definition of the measurement property type		
Available via	Icu.h		

⌋

8.2.12 Icu_TimestampBufferType

[SWS_Icu_00296] Definition of datatype Icu_TimestampBufferType ⌈

Name	Icu_TimestampBufferType		
Kind	Enumeration		
Range	ICU_LINEAR_BUFFER	0x00	The buffer will just be filled once
	ICU_CIRCULAR_BUFFER	0x01	After reaching the end of the buffer, the driver restarts at the beginning of the buffer

▽

△

Description	Definition of the timestamp measurement property type
Available via	Icu.h

⌋

8.3 Function definitions

This chapter contains all APIs provided by the module.

Since the ICU is a driver module, it doesn't provide any callback functions for lower layer modules.

8.3.1 Icu_Init

[SWS_Icu_00191] Definition of API function Icu_Init ⌈

Service Name	Icu_Init	
Syntax	`void Icu_Init ( const Icu_ConfigType* ConfigPtr )`	
Service ID [hex]	0x00	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	ConfigPtr	Pointer to a selected configuration structure
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	This function initializes the driver.	
Available via	Icu.h	

⌋

[SWS_Icu_00297] ⌈The function `Icu_Init` shall be non re-entrant.⌋

[SWS_Icu_00298] ⌈The function `Icu_Init` initializes the driver.⌋

[SWS_Icu_00006]

Upstream requirements: SRS_BSW_00344, SRS_BSW_00404, SRS_BSW_00405, SRS_BSW_00101, SRS_SPAL_12057, SRS_SPAL_12461

⌈The function `Icu_Init` shall initialize all relevant registers of the configured hardware with the values of the structure referenced by the parameter `ConfigPtr`.⌋

The following rules regarding initialization of controller registers shall apply to this driver implementation:

- **[SWS_Icu_00051]**

  Upstream requirements: SRS_SPAL_12461

  ⌈If the hardware allows for only one usage of the register, the driver module implementing that functionality is responsible for initializing the register.⌋

- **[SWS_Icu_00052]**

  Upstream requirements: SRS_SPAL_12461

  ⌈If the register can affect several hardware modules and if it is an I/O register it shall be initialized by the PORT driver.⌋

- **[SWS_Icu_00053]**

  Upstream requirements: SRS_SPAL_12461

  ⌈If the register can affect several hardware modules and if it is not an I/O register it shall be initialized by the MCU driver.⌋

- **[SWS_Icu_00128]**

  Upstream requirements: SRS_SPAL_12461

  ⌈One-time writable registers that require initialization directly after reset shall be initialized by the start-up code.⌋

- **[SWS_Icu_00129]**

  Upstream requirements: SRS_SPAL_12461

  ⌈All other registers shall be initialized by the startup code.⌋

[SWS_Icu_00061]

Upstream requirements: SRS_SPAL_12057, SRS_Icu_12407

⌈The function `Icu_Init` shall disable all notifications.⌋

[SWS_Icu_00121] ⌈The function `Icu_Init` shall disable the wakeup-capability of all channels.⌋

[SWS_Icu_00040]

Upstream requirements: SRS_SPAL_12057, SRS_Icu_12407

⌈The function `Icu_Init` shall set all used ICU channels to status `ICU_IDLE`.⌋

[SWS_Icu_00060]

Upstream requirements: SRS_SPAL_12057

⌈The function `Icu_Init` shall set the module mode to `ICU_MODE_NORMAL`.⌋

[SWS_Icu_00054]

Upstream requirements: SRS_SPAL_12125

⌈The function `Icu_Init` shall only set the resources that are configured in the configuration file (including clearing of pending interrupt flags).

The Icu module's environment shall not call `Icu_Init` during a running operation (e.g. timestamp measurement or edge counting).⌋

[SWS_Icu_00220]

Upstream requirements: SRS_BSW_00386

⌈If development error detection for the ICU module is enabled and the function `Icu_Init` is called when the ICU driver and hardware are already initialized, the function `Icu_Init` shall raise development error `ICU_E_ALREADY_INITIALIZED` and return without any action.⌋

[SWS_Icu_00138] ⌈The initialization function of this module shall always have a pointer as a parameter, even though for Variant PC no configuration set shall be given. Instead a NULL pointer shall be passed to the initialization function.⌋

Note: Parameter checking for the initialization function is specified within BSW General [5].

8.3.2 Icu_DeInit

[SWS_Icu_00193] Definition of API function Icu_DeInit ⌈

Service Name	Icu_DeInit
Syntax	`void Icu_DeInit ( void )`
Service ID [hex]	0x01
Sync/Async	Synchronous
Reentrancy	Non Reentrant
Parameters (in)	None
Parameters (inout)	None
Parameters (out)	None
Return value	None
Description	This function de-initializes the ICU module.
Available via	Icu.h

⌋

[SWS_Icu_00036]

Upstream requirements: SRS_SPAL_12163, SRS_Icu_12429

⌈The function `Icu_DeInit` shall set the state of the peripherals used by configuration as the same after power on reset.⌋

[SWS_Icu_00300] ⌈Values of registers which are not writeable are excluded from setting the state by the function `Icu_DeInit`.⌋

[SWS_Icu_00091] ⌈The function `Icu_DeInit` shall influence only the peripherals which are allocated by static configuration and/or the runtime configuration set passed by the previous call of `Icu_Init`.⌋

[SWS_Icu_00037]

Upstream requirements: SRS_BSW_00336, SRS_SPAL_12163

⌈The function `Icu_DeInit` shall disable all used interrupts and notifications.⌋

[SWS_Icu_00152] ⌈The Icu module's environment shall not call `Icu_DeInit` during a running operation (e.g. timestamp measurement or edge counting)⌋

[SWS_Icu_00092]

Upstream requirements: SRS_BSW_00410, SRS_BSW_00171

⌈The function `Icu_DeInit` shall be pre compile time configurable by configuration parameter IcuDeInitApi.⌋

[SWS_Icu_00301] ⌈The function `Icu_DeInit` shall be configurable ON/OFF by configuration parameter IcuDeInitApi.⌋

[SWS_Icu_00221] ⌈A re-initialization of the ICU module by executing the `Icu_Init` function requires a de-initialization before by executing the `Icu_DeInit` function.⌋

[SWS_Icu_00299] ⌈`Icu_DeInit` operation is non re-entrant.⌋

[SWS_Icu_00385]

Upstream requirements: SRS_BSW_00323, SRS_BSW_00406

⌈If development error detection for the Icu module is enabled: This function shall raise development error `ICU_E_UNINIT` when the function `Icu_Init` has not been called.⌋

8.3.3 Icu_SetMode

[SWS_Icu_00194] Definition of API function Icu_SetMode ⌈

Service Name	Icu_SetMode
Syntax	`void Icu_SetMode ( Icu_ModeType Mode )`
Service ID [hex]	0x02
Sync/Async	Synchronous
Reentrancy	Non Reentrant

▽

△

Parameters (in)	Mode	ICU_MODE_NORMAL: Normal operation, all used interrupts are enabled according to the notification requests. ICU_MODE_SLEEP: Reduced power mode. In sleep mode only those notifications are available which are configured as wakeup capable.
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	This function sets the ICU mode.	
Available via	Icu.h	

⌋

[SWS_Icu_00008]

Upstream requirements: SRS_SPAL_12067, SRS_SPAL_12169, SRS_Icu_12370

⌈The function `Icu_SetMode` shall set the operation mode to the given mode parameter. The function `Icu_SetMode` shall set the operation mode to the given mode parameter. This function influences the functionality of the ICU channels. Therefore the mode switching of the module shall be compatible to the overall state of the ECU.⌋

[SWS_Icu_00302] ⌈The function `Icu_SetMode` shall be non re-entrant.

This function influences the functionality of the ICU channels. Therefore the mode switching of the module shall be compatible to the overall state of the ECU.⌋

[SWS_Icu_00095]

Upstream requirements: SRS_BSW_00410, SRS_BSW_00171

⌈The function `Icu_SetMode` shall be pre-compile time configurable by the configuration parameter IcuSetModeApi.⌋

[SWS_Icu_00303] ⌈The function `Icu_SetMode` shall be configurable ON/OFF by the configuration parameter IcuSetModeApi.⌋

[SWS_Icu_00125]

Upstream requirements: SRS_BSW_00323

⌈If development error detection is enabled for the module Icu the function `Icu_SetMode` shall check the parameter Mode and shall raise the error `ICU_E_PARAM_MODE` if the parameter Mode is not within the allowed range set in the configuration.⌋

[SWS_Icu_00133]

Upstream requirements: SRS_SPAL_12064

⌈This service can be called during running operations. If so, an ongoing operation that generates interrupts on a wakeup capable channel like e.g. time stamping or edge counting might lead to the ICU module not being able to properly enter sleep mode. This is then a system or ECU configuration issue not a problem of this specification.⌋

[SWS_Icu_00386]

Upstream requirements: SRS_BSW_00323, SRS_BSW_00406

⌈If development error detection for the Icu module is enabled: This function shall raise development error `ICU_E_UNINIT` when the function `Icu_Init` has not been called.⌋

8.3.4 Icu_DisableWakeup

[SWS_Icu_00195] Definition of API function Icu_DisableWakeup ⌈

Service Name	Icu_DisableWakeup	
Syntax	`void Icu_DisableWakeup ( Icu_ChannelType Channel )`	
Service ID [hex]	0x03	
Sync/Async	Synchronous	
Reentrancy	Reentrant (limited according to ICU050)	
Parameters (in)	Channel	Numeric identifier of the ICU channel
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	This function disables the wakeup capability of a single ICU channel.	
Available via	Icu.h	

⌋

[SWS_Icu_00013]

Upstream requirements: SRS_Icu_12408

⌈The function `Icu_DisableWakeup` shall disable the wakeup capability of a single ICU channel.⌋

[SWS_Icu_00305] ⌈The function `Icu_DisableWakeup` shall disable the wakeup capability of a single ICU channel only for ICU channels configured statically as wakeup capable true.⌋

[SWS_Icu_00304] ⌈The function `Icu_DisableWakeup` shall be re-entrant.⌋

[SWS_Icu_00096]

Upstream requirements: SRS_BSW_00410, SRS_BSW_00171

⌈The function `Icu_DisableWakeup` shall be pre compile time configurable by the configuration parameter IcuDisableWakeupApi.⌋

[SWS_Icu_00306] ⌈The function `Icu_DisableWakeup` shall be configurable ON/OFF by the configuration parameter IcuDisableWakeupApi.

The settings done by this function are only relevant after the ICU_MODE_SLEEP is set.⌋

[SWS_Icu_00024]

Upstream requirements: SRS_BSW_00323

⌈If development error detection is enabled: The function `Icu_DisableWakeup` shall check the parameter Channel and shall raise development error `ICU_E_PARAM_CHANNEL` if Channel is not within the allowed range set in the configuration.⌋

[SWS_Icu_00059] ⌈If development error detection is enabled: The function `Icu_DisableWakeup` shall check the parameter Channel. The function `Icu_DisableWakeup` shall raise development error `ICU_E_PARAM_CHANNEL` if Channel is indexing an ICU channel statically not configured as wakeup capable.⌋

[SWS_Icu_00387]

Upstream requirements: SRS_BSW_00323, SRS_BSW_00406

⌈If development error detection for the Icu module is enabled: This function shall raise development error `ICU_E_UNINIT` when the function `Icu_Init` has not been called.⌋

8.3.5 Icu_EnableWakeup

[SWS_Icu_00196] Definition of API function Icu_EnableWakeup ⌈

Service Name	Icu_EnableWakeup	
Syntax	`void Icu_EnableWakeup ( Icu_ChannelType Channel )`	
Service ID [hex]	0x04	
Sync/Async	Synchronous	
Reentrancy	Reentrant (limited according to ICU050)	
Parameters (in)	Channel	Numeric identifier of the ICU channel
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	This function (re-)enables the wakeup capability of the given ICU channel.	
Available via	Icu.h	

⌋

[SWS_Icu_00307] ⌈The function `Icu_EnableWakeup` shall be re-entrant.⌋

[SWS_Icu_00014]

Upstream requirements: SRS_Icu_12408

⌈The function `Icu_EnableWakeup` shall re-enable the wakeup capability of a single ICU channel for the following ICU mode selection(s). This service is only feasible for ICU channels configured as wakeup capable true.

To make the selection effective a call of the function `Icu_SetMode`, requesting the mode ICU_MODE_SLEEP is required.⌋

[SWS_Icu_00097]

Upstream requirements: SRS_BSW_00410, SRS_BSW_00171

⌈The function `Icu_EnableWakeup` shall be pre compile time configurable by configuration parameter IcuEnableWakeupApi.⌋

[SWS_Icu_00308] ⌈The function `Icu_EnableWakeup` shall be configurable ON/OFF by configuration parameter IcuEnableWakeupApi.⌋

[SWS_Icu_00155] ⌈If development error detection is enabled: The function `Icu_EnableWakeup` shall check the parameter Channel and shall raise the error `ICU_E_PARAM_CHANNEL` if Channel is invalid.⌋

[SWS_Icu_00156] ⌈If development error detection is enabled: The function `Icu_EnableWakeup` shall check the parameter Channel. The function `Icu_EnableWakeup` shall raise the error `ICU_E_PARAM_CHANNEL` if Channel is indexing an ICU channel statically not configured as wakeup capable.⌋

[SWS_Icu_00388]

Upstream requirements: SRS_BSW_00323, SRS_BSW_00406

⌈If development error detection for the Icu module is enabled: This function shall raise development error `ICU_E_UNINIT` when the function `Icu_Init` has not been called.⌋

8.3.6 Icu_CheckWakeup

[SWS_Icu_00358] Definition of API function Icu_CheckWakeup ⌈

Service Name	Icu_CheckWakeup	
Syntax	`void Icu_CheckWakeup ( EcuM_WakeupSourceType WakeupSource )`	
Service ID [hex]	0x15	
Sync/Async	Synchronous	
Reentrancy	Reentrant (limited according to ICU050)	
Parameters (in)	WakeupSource	Informatin on wakeup source to be checked. The associated ICU channel can be determined from configuration data.
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	Checks if a wakeup capable ICU channel is the source for a wakeup event and calls the ECU state manager service EcuM_SetWakeupEvent in case of a valid ICU channel wakeup event.	
Available via	Icu.h	

⌋

[SWS_Icu_00359]

Upstream requirements: SRS_BSW_00375

⌈The function `Icu_CheckWakeup` shall check if a wakeup capable ICU channel is the source for a wakeup event and call EcuM_SetWakeupEvent to indicate a valid timer wakeup event to the ECU State Manager.⌋

[SWS_Icu_00360] ⌈The function `Icu_CheckWakeup` is only feasible, if IcuReportWakeupSource is statically configured available.⌋

[SWS_Icu_00361] ⌈The ICU module's environment shall only use the re-entrant capability of the function `Icu_CheckWakeup` if the ICU module's environment takes care that there is no simultaneous usage of the same channel.⌋

[SWS_Icu_00362] ⌈The function `Icu_CheckWakeup` shall be pre compile time configurable On/Off by the configuration parameter: IcuWakeupFunctionalityApi⌋

[SWS_Icu_00363] ⌈If development error detection for the ICU module is enabled: if the function `Icu_CheckWakeup` is called before the ICU module was initialized, the function `Icu_CheckWakeup` shall raise the development error `ICU_E_UNINIT`.⌋

8.3.7 Icu_SetActivationCondition

[SWS_Icu_00197] Definition of API function Icu_SetActivationCondition ⌈

Service Name	Icu_SetActivationCondition	
Syntax	`void Icu_SetActivationCondition ( Icu_ChannelType Channel, Icu_ActivationType Activation )`	
Service ID [hex]	0x05	
Sync/Async	Synchronous	
Reentrancy	Reentrant (limited according to ICU050)	
Parameters (in)	Channel	Numeric identifier of the ICU channel
	Activation	Type of activation (if supported by hardware) • ICU_RISING_EDGE • ICU_FALLING_EDGE • ICU_BOTH_EDGES
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	This function sets the activation-edge for the given channel.	
Available via	Icu.h	

⌋

[SWS_Icu_00090]

Upstream requirements: SRS_BSW_00410

⌈The function `Icu_SetActivationCondition` shall set the activation-edge according to Activation parameter for the given channel. This service shall support channels which are configured for the following `Icu_MeasurementModeType`:

- `ICU_MODE_SIGNAL_EDGE_DETECT`
- `ICU_MODE_TIMESTAMP`
- `ICU_MODE_EDGE_COUNTER`

⌋

[SWS_Icu_00139] ⌈The function `Icu_SetActivationCondition` shall reset the state for the given channel to `ICU_IDLE`.⌋

[SWS_Icu_00309] ⌈The function `Icu_SetActivationCondition` shall be reentrant.⌋

[SWS_Icu_00159] ⌈If development error detection is enabled the function `Icu_SetActivationCondition` shall check the parameter Channel and shall raise the error `ICU_E_PARAM_CHANNEL` if Channel is not within the range set in the configuration.⌋

[SWS_Icu_00043]

Upstream requirements: SRS_BSW_00323

⌈If development error detection is enabled the function `Icu_SetActivationCondition` shall check the parameter Activation. The function `Icu_SetActivationCondition` shall raise the error `ICU_E_PARAM_ACTIVATION` if Activation is invalid but only for the requested ICU channel.⌋

8.3.8 Icu_DisableNotification

[SWS_Icu_00198] Definition of API function Icu_DisableNotification ⌈

Service Name	Icu_DisableNotification	
Syntax	`void Icu_DisableNotification ( Icu_ChannelType Channel )`	
Service ID [hex]	0x06	
Sync/Async	Synchronous	
Reentrancy	Reentrant (limited according to ICU050)	
Parameters (in)	Channel	Numeric identifier of the ICU channel
Parameters (inout)	None	
Parameters (out)	None	

▽

△

Return value	None
Description	This function disables the notification of a channel.
Available via	Icu.h

⌋

[SWS_Icu_00009]

Upstream requirements: SRS_Icu_12305

⌈The function `Icu_DisableNotification` shall disable the notification on the given channel.⌋

[SWS_Icu_00310] ⌈The function `Icu_DisableNotification` shall be re-entrant.⌋

[SWS_Icu_00160] ⌈If development error detection is enabled the function `Icu_DisableNotification` shall check the parameter `Channel` and shall raise the error `ICU_E_PARAM_CHANNEL` if `Channel` is invalid (invalid identifier).⌋

[SWS_Icu_00389]

Upstream requirements: SRS_BSW_00323, SRS_BSW_00406

⌈If development error detection for the Icu module is enabled: This function shall raise development error `ICU_E_UNINIT` when the function `Icu_Init` has not been called.⌋

8.3.9 Icu_EnableNotification

[SWS_Icu_00199] Definition of API function Icu_EnableNotification ⌈

Service Name	Icu_EnableNotification	
Syntax	`void Icu_EnableNotification ( Icu_ChannelType Channel )`	
Service ID [hex]	0x07	
Sync/Async	Synchronous	
Reentrancy	Reentrant (limited according to ICU050)	
Parameters (in)	Channel	Numeric identifier of the ICU channel
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	This function enables the notification on the given channel.	
Available via	Icu.h	

⌋

[SWS_Icu_00010]

Upstream requirements: SRS_Icu_12305

⌈The function `Icu_EnableNotification` shall enable the notification on the given channel.⌋

[SWS_Icu_00311] ⌈The function `Icu_EnableNotification` shall be re-entrant.⌋

[SWS_Icu_00161] ⌈If development error detection is enabled the function `Icu_EnableNotification` shall check the parameter `Channel` and shall raise the error `ICU_E_PARAM_CHANNEL` if `Channel` is invalid (invalid identifier).⌋

[SWS_Icu_00390]

Upstream requirements: SRS_BSW_00323, SRS_BSW_00406

⌈If development error detection for the Icu module is enabled: This function shall raise development error `ICU_E_UNINIT` when the function `Icu_Init` has not been called.⌋

8.3.10 Icu_GetInputState

[SWS_Icu_00200] Definition of API function Icu_GetInputState ⌈

Service Name	Icu_GetInputState	
Syntax	`Icu_InputStateType Icu_GetInputState ( Icu_ChannelType Channel )`	
Service ID [hex]	0x08	
Sync/Async	Synchronous	
Reentrancy	Reentrant (limited according to ICU050)	
Parameters (in)	Channel	Numeric identifier of the ICU channel
Parameters (inout)	None	
Parameters (out)	None	
Return value	Icu_InputStateType	`ICU_ACTIVE`: An activation edge has been detected `ICU_IDLE`: No activation edge has been detected since the last call of Icu_GetInputState() or Icu_Init().
Description	This function returns the status of the ICU input.	
Available via	Icu.h	

⌋

[SWS_Icu_00313] ⌈`Icu_GetInputState` shall return `Icu_InputStateType` which will have value `ICU_IDLE` when no activation edge has been detected since the last call of `Icu_GetInputState` or `Icu_Init`.⌋

[SWS_Icu_00030]

Upstream requirements: SRS_SPAL_00157, SRS_Icu_12371

⌈The function `Icu_GetInputState` shall return the status of the ICU input. Only channels which are configured for the following IcuMeasurementMode shall be supported:

- ICU_MODE_SIGNAL_EDGE_DETECT
- ICU_MODE_SIGNAL_MEASUREMENT

⌋

[SWS_Icu_00312] ⌈The function `Icu_GetInputState` shall be re-entrant.⌋

[SWS_Icu_00031]

Upstream requirements: SRS_Icu_12371

⌈If an activation edge has been detected the function `Icu_GetInputState` shall return `ICU_ACTIVE` for Edge Detection channels.⌋

[SWS_Icu_00314] ⌈For Signal Measurement a channel should be set to `ICU_ACTIVE` not until this measurement has completed and the driver is able to provide useful information on the input signal.⌋

[SWS_Icu_00032]

Upstream requirements: SRS_Icu_12371

⌈Once the function `Icu_GetInputState` has returned the status `ICU_ACTIVE`, the function `Icu_GetInputState` shall set the stored status to `ICU_IDLE` until the next edge is detected.⌋

[SWS_Icu_00122]

Upstream requirements: SRS_BSW_00410, SRS_BSW_00171

⌈The function `Icu_GetInputState` shall be pre compile time configurable by the configuration parameter IcuGetInputStateApi.⌋

[SWS_Icu_00315] ⌈The function `Icu_GetInputState` shall be configurable ON/OFF by the configuration parameter IcuGetInputStateApi.⌋

[SWS_Icu_00162] ⌈If development error detection is enabled the function `Icu_GetInputState` shall check the parameter Channel and shall raise the error `ICU_E_PARAM_CHANNEL` if Channel is invalid (invalid identifier or channel not configured for modes ICU_MODE_SIGNAL_EDGE_DETECT or ICU_MODE_SIGNAL_MEASUREMENT)⌋

[SWS_Icu_00049]

Upstream requirements: SRS_SPAL_12448, SRS_BSW_00369

⌈If development error detection is enabled the function `Icu_GetInputState` shall return `ICU_IDLE` if an error is detected.⌋

[SWS_Icu_00391]

Upstream requirements: SRS_BSW_00323, SRS_BSW_00406

⌈If development error detection for the Icu module is enabled: This function shall raise development error `ICU_E_UNINIT` when the function `Icu_Init` has not been called.⌋

8.3.11 Icu_StartTimestamp

[SWS_Icu_00201] Definition of API function Icu_StartTimestamp ⌈

Service Name	Icu_StartTimestamp	
Syntax	`void Icu_StartTimestamp ( Icu_ChannelType Channel, Icu_ValueType* BufferPtr, uint16 BufferSize, uint16 NotifyInterval )`	
Service ID [hex]	0x09	
Sync/Async	Asynchronous	
Reentrancy	Reentrant (limited according to ICU050)	
Parameters (in)	Channel	Numeric identifier of the ICU channel
	BufferSize	Size of the external buffer (number of entries)
	NotifyInterval	Notification interval (number of events). This parameter can not be checked in a reasonable way.
Parameters (inout)	None	
Parameters (out)	BufferPtr	Pointer to the buffer-array where the timestamp values shall be placed.
Return value	None	
Description	This function starts the capturing of timer values on the edges.	
Available via	Icu.h	

⌋

[SWS_Icu_00317] ⌈The function `Icu_StartTimestamp` shall start the capturing of timer values on the edges to an external buffer, at the beginning of the buffer.⌋

[SWS_Icu_00063]

Upstream requirements: SRS_BSW_00410, SRS_SPAL_12063, SRS_SPAL_12075, SRS_Icu_12430, SRS_Icu_12438

⌈The function `Icu_StartTimestamp` shall start the capturing of timer values on the edges activated by the service `Icu_SetActivationCondition` (rising / falling / both edges)⌋

[SWS_Icu_00316] ⌈The function `Icu_StartTimestamp` shall be re-entrant.⌋

[SWS_Icu_00064] ⌈If circular buffer handling is configured (for the given channel), when the capture functionality reaches the end of the buffer, the Icu module shall start at the beginning of the buffer.⌋

[SWS_Icu_00065]

Upstream requirements: SRS_Icu_12456

⌈If linear buffer handling is configured, when the capture functionality reaches the end of the buffer, the Icu module shall stop capturing timer values.⌋

[SWS_Icu_00134] ⌈The Icu module shall only call a notification function if a notification function is configured.⌋

[SWS_Icu_00318] ⌈The Icu module shall only call a notification function if the notification has been enabled by the call of `Icu_EnableNotification`.⌋

[SWS_Icu_00319] ⌈The Icu module shall only call a notification function if NotifyInterval is greater than "0".⌋

[SWS_Icu_00320] ⌈The Icu module shall only call a notification function if the number of events specified by NotifyInterval has been captured.⌋

[SWS_Icu_00066]

Upstream requirements: SRS_Icu_12430

⌈The function `Icu_StartTimestamp` shall only be available in Measurement Mode "ICU_MODE_TIMESTAMP".⌋

[SWS_Icu_00098]

Upstream requirements: SRS_BSW_00171

⌈The function `Icu_StartTimestamp` shall be pre-compile time configurable by the configuration parameter: ICU_TIMESTAMP_API.⌋

[SWS_Icu_00321] ⌈The function `Icu_StartTimestamp` shall be configurable ON/OFF by the configuration parameter: ICU_TIMESTAMP_API.⌋

[SWS_Icu_00163] ⌈If development error detection is enabled the function `Icu_StartTimestamp` shall check the parameter Channel and shall raise the error `ICU_E_PARAM_CHANNEL` if Channel is invalid (invalid identifier or channel not configured for mode ICU_MODE_TIMESTAMP).⌋

[SWS_Icu_00354] ⌈If development error detection is enabled and a notification function has been configured for the addressed channel, the function `Icu_StartTimestamp` shall check the parameter NotifyInterval for validity and raise the error `ICU_E_PARAM_NOTIFY_INTERVAL` if the parameter NotifyInterval is "0".⌋

[SWS_Icu_00108]

Upstream requirements: SRS_SPAL_12448

⌈If development error detection is enabled the function `Icu_StartTimestamp` shall check the parameter BufferSize (check that size > 0). The function `Icu_StartTimestamp` shall raise the error `ICU_E_PARAM_BUFFER_SIZE` if BufferSize is invalid (e.g. "0").⌋

[SWS_Icu_00392]

Upstream requirements: SRS_BSW_00323, SRS_BSW_00406

⌈If development error detection for the Icu module is enabled: This function shall raise development error `ICU_E_UNINIT` when the function `Icu_Init` has not been called.⌋

8.3.12 Icu_StopTimestamp

[SWS_Icu_00202] Definition of API function Icu_StopTimestamp ⌈

Service Name	Icu_StopTimestamp	
Syntax	`void Icu_StopTimestamp ( Icu_ChannelType Channel )`	
Service ID [hex]	0x0a	
Sync/Async	Synchronous	
Reentrancy	Reentrant (limited according to ICU050)	
Parameters (in)	Channel	Numeric identifier of the ICU channel
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	This function stops the timestamp measurement of the given channel.	
Available via	Icu.h	

⌋

[SWS_Icu_00067]

Upstream requirements: SRS_Icu_12431

⌈The function `Icu_StopTimestamp` shall stop the timestamp measurement of the given channel.⌋

[SWS_Icu_00322] ⌈`Icu_StopTimestamp` operation is Re-entrant.

In production mode the function `Icu_StopTimestamp` shall not return an error when the `Channel` is not active (has not started or has already stopped).⌋

[SWS_Icu_00165] ⌈The function `Icu_StopTimestamp` shall only be available in Measurement Mode: ICU_MODE_TIMESTAMP.⌋

[SWS_Icu_00099]

Upstream requirements: SRS_BSW_00410, SRS_BSW_00171

⌈The function `Icu_StopTimestamp` shall be pre-compile time configurable by the configuration parameter: IcuTimestampApi.⌋

[SWS_Icu_00164] ⌈If development error detection is enabled the function `Icu_StopTimestamp` shall check the parameter `Channel` and shall raise development error `ICU_E_PARAM_CHANNEL` if `Channel` is invalid (invalid identifier or channel not configured for mode ICU_MODE_TIMESTAMP).⌋

[SWS_Icu_00323] ⌈The function `Icu_StopTimestamp` shall be configurable ON/OFF by the configuration parameter: IcuTimestampApi.⌋

[SWS_Icu_00166] ⌈The function `Icu_StopTimestamp` shall raise runtime error `ICU_E_NOT_STARTED` if Channel is not active (has not started or is already stopped).⌋

[SWS_Icu_00393]

Upstream requirements: SRS_BSW_00323, SRS_BSW_00406

⌈If development error detection for the Icu module is enabled: This function shall raise development error `ICU_E_UNINIT` when the function `Icu_Init` has not been called.⌋

8.3.13 Icu_GetTimestampIndex

[SWS_Icu_00203] Definition of API function Icu_GetTimestampIndex ⌈

Service Name	Icu_GetTimestampIndex	
Syntax	`Icu_IndexType Icu_GetTimestampIndex ( Icu_ChannelType Channel )`	
Service ID [hex]	0x0b	
Sync/Async	Synchronous	
Reentrancy	Reentrant (limited according to ICU050)	
Parameters (in)	Channel	Numeric identifier of the ICU channel
Parameters (inout)	None	
Parameters (out)	None	
Return value	Icu_IndexType	Abstract return type to cover different microcontrollers.
Description	This function reads the timestamp index of the given channel.	
Available via	Icu.h	

⌋

[SWS_Icu_00071]

Upstream requirements: SRS_Icu_12453

⌈The function `Icu_GetTimestampIndex` shall read the timestamp index of the given channel, which is the next to be written.⌋

[SWS_Icu_00324] ⌈The function `Icu_GetTimestampIndex` shall be re-entrant.⌋

[SWS_Icu_00135] ⌈The function `Icu_GetTimestampIndex` shall return "0" in case the service is called before `Icu_StartTimestamp` (no buffer is defined in this case).⌋

[SWS_Icu_00170] ⌈The function `Icu_GetTimestampIndex` shall only be available in Measurement Mode ICU_MODE_TIMESTAMP.⌋

[SWS_Icu_00100]

Upstream requirements: SRS_BSW_00410, SRS_BSW_00171

⌈The function `Icu_GetTimestampIndex` shall be pre compile time configurable by the configuration parameter: IcuTimestampApi.⌋

[SWS_Icu_00325] ⌈The function `Icu_GetTimestampIndex` shall be configurable ON/OFF by the configuration parameter: IcuTimestampApi.⌋

[SWS_Icu_00169] ⌈If development error detection is enabled the function `Icu_GetTimestampIndex` shall check the parameter Channel. If Channel is invalid (invalid identifier or channel not configured for mode ICU_MODE_TIMESTAMP), the function `Icu_GetTimestampIndex` shall raise development error `ICU_E_PARAM_CHANNEL`.⌋

[SWS_Icu_00107]

Upstream requirements: SRS_SPAL_12448

⌈If development error detection is enabled the function `Icu_GetTimestampIndex` shall return "0" if an error is detected.⌋

[SWS_Icu_00394]

Upstream requirements: SRS_BSW_00323, SRS_BSW_00406

⌈If development error detection for the Icu module is enabled: This function shall raise development error `ICU_E_UNINIT` when the function `Icu_Init` has not been called.⌋

8.3.14 Icu_ResetEdgeCount

[SWS_Icu_00204] Definition of API function Icu_ResetEdgeCount ⌈

Service Name	Icu_ResetEdgeCount	
Syntax	`void Icu_ResetEdgeCount ( Icu_ChannelType Channel )`	
Service ID [hex]	0x0c	
Sync/Async	Synchronous	
Reentrancy	Reentrant (limited according to ICU050)	
Parameters (in)	Channel	Numeric identifier of the ICU channel
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	This function resets the value of the counted edges to zero.	
Available via	Icu.h	

⌋

[SWS_Icu_00072]

Upstream requirements: SRS_Icu_12439, SRS_Icu_13100

⌈The function `Icu_ResetEdgeCount` shall reset the value of the counted edges to zero.⌋

[SWS_Icu_00326] ⌈The function `Icu_ResetEdgeCount` shall be re-entrant.⌋

[SWS_Icu_00101]

Upstream requirements: SRS_BSW_00410, SRS_BSW_00171

⌈The function `Icu_ResetEdgeCount` shall be pre-compile time configurable by the configuration parameter Icu_EDGE_COUNT_API.⌋

[SWS_Icu_00327] ⌈The function `Icu_ResetEdgeCount` shall be configurable ON/OFF by the configuration parameter: ICU_EDGE_COUNT_API.⌋

[SWS_Icu_00171] ⌈If development error detection is enabled the function `Icu_ResetEdgeCount` shall check the parameter `Channel`. If `Channel` is invalid (invalid identifier or channel not configured for mode ICU_MODE_EDGE_COUNTER), then `Icu_ResetEdgeCount` shall raise development error `ICU_E_PARAM_CHANNEL`.⌋

[SWS_Icu_00395]

Upstream requirements: SRS_BSW_00323, SRS_BSW_00406

⌈If development error detection for the Icu module is enabled: This function shall raise development error `ICU_E_UNINIT` when the function `Icu_Init` has not been called.⌋

8.3.15 Icu_EnableEdgeCount

[SWS_Icu_00205] Definition of API function Icu_EnableEdgeCount ⌈

Service Name	Icu_EnableEdgeCount	
Syntax	`void Icu_EnableEdgeCount ( Icu_ChannelType Channel )`	
Service ID [hex]	0x0d	
Sync/Async	Synchronous	
Reentrancy	Reentrant (limited according to ICU050)	
Parameters (in)	Channel	Numeric identifier of the ICU channel
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	This function enables the counting of edges of the given channel.	
Available via	Icu.h	

⌋

[SWS_Icu_00078]

Upstream requirements: SRS_Icu_12432

⌈The function `Icu_EnableEdgeCount` shall enable the counting of edges of the given channel.⌋

Note: This service does not do the real counting itself.

[SWS_Icu_00073]

Upstream requirements: SRS_Icu_12439

⌈The function `Icu_EnableEdgeCount` shall only count the configured[1] edges (rising edge / falling edge / both edges).⌋

[1] Configured edge after the call of `Icu_Init` (default-edge) or `Icu_SetActivationCondition`.

[SWS_Icu_00074]

Upstream requirements: SRS_Icu_12439

⌈The function `Icu_EnableEdgeCount` shall be available for each ICU channel in Measurement Mode "Edge Counter".⌋

[SWS_Icu_00328] ⌈The function `Icu_EnableEdgeCount` shall be re-entrant.⌋

[SWS_Icu_00102]

Upstream requirements: SRS_BSW_00410, SRS_BSW_00171

⌈The function `Icu_EnableEdgeCount` shall be pre-compile time configurable by the configuration parameter Icu_EDGE_COUNT_API.⌋

[SWS_Icu_00329] ⌈The function `Icu_EnableEdgeCount` shall be configurable On/Off by the configuration parameter: ICU_EDGE_COUNT_API.⌋

[SWS_Icu_00172] ⌈If development error detection is enabled, the function `Icu_EnableEdgeCount` shall check the parameter Channel. If Channel is invalid (invalid identifier or channel not configured for mode ICU_MODE_EDGE_COUNTER), then the function `Icu_EnableEdgeCount` shall raise development error `ICU_E_PARAM_CHANNEL`.⌋

[SWS_Icu_00396]

Upstream requirements: SRS_BSW_00323, SRS_BSW_00406

⌈If development error detection for the Icu module is enabled: This function shall raise development error `ICU_E_UNINIT` when the function `Icu_Init` has not been called.⌋

8.3.16 Icu_EnableEdgeDetection

[SWS_Icu_00364] Definition of API function Icu_EnableEdgeDetection ⌈

Service Name	Icu_EnableEdgeDetection	
Syntax	`void Icu_EnableEdgeDetection ( Icu_ChannelType Channel )`	
Service ID [hex]	0x16	
Sync/Async	Synchronous	
Reentrancy	Reentrant (limited according to ICU050)	
Parameters (in)	Channel	Numeric identifier of the ICU channel
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	This function enables / re-enables the detection of edges of the given channel.	
Available via	Icu.h	

⌋

[SWS_Icu_00365] ⌈The function `Icu_EnableEdgeDetection` shall enable the detection of edges for the given channel.⌋

[SWS_Icu_00366] ⌈The function `Icu_EnableEdgeDetection` shall only detect the configured edges (rising edge / falling edge / both edges).⌋

[SWS_Icu_00367] ⌈The function `Icu_EnableEdgeDetection` shall be available for each ICU Channel in Measurement Mode "Edge Detection".⌋

[SWS_Icu_00368] ⌈The function `Icu_EnableEdgeDetection` shall be re-entrant.⌋

[SWS_Icu_00369] ⌈The function `Icu_EnableEdgeDetection` shall be pre-compile time configurable by the configuration parameter IcuEdgeDetectApi.⌋

[SWS_Icu_00370] ⌈The function `Icu_EnableEdgeDetection` shall be configurable ON/OFF by the configuration parameter: IcuEdgeDetectApi.⌋

[SWS_Icu_00371] ⌈If development error detection is enabled; the function `Icu_EnableEdgeDetection` shall check the parameter `Channel`. If `Channel` is invalid (invalid identifier or channel not configured for mode ICU_MODE_SIGNAL_EDGE_DETECT), then the function `Icu_EnableEdgeDetection` shall raise development error `ICU_E_PARAM_CHANNEL`.⌋

[SWS_Icu_00397]

Upstream requirements: SRS_BSW_00323, SRS_BSW_00406

⌈If development error detection for the Icu module is enabled: This function shall raise development error `ICU_E_UNINIT` when the function `Icu_Init` has not been called.⌋

8.3.17 Icu_DisableEdgeDetection

[SWS_Icu_00377] Definition of API function Icu_DisableEdgeDetection ⌈

Service Name	Icu_DisableEdgeDetection	
Syntax	`void Icu_DisableEdgeDetection ( Icu_ChannelType Channel )`	
Service ID [hex]	0x17	
Sync/Async	Synchronous	
Reentrancy	Reentrant (limited according to ICU050)	
Parameters (in)	Channel	Numeric identifier of the ICU channel
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	This function disables the detection of edges of the given channel.	
Available via	Icu.h	

⌋

[SWS_Icu_00372] ⌈The function Icu_DisableEdgeDetection shall disable the detection of edges of the given channel⌋

[SWS_Icu_00373] ⌈The function Icu_DisableEdgeDetection shall be re-entrant.⌋

[SWS_Icu_00374] ⌈The function Icu_DisableEdgeDetection shall be pre-compile time configurable by the configuration parameter IcuEdgeDetectApi.⌋

[SWS_Icu_00375] ⌈The function Icu_DisableEdgeDetection shall be configurable ON/OFF by the configuration parameter IcuEdgeDetectApi.⌋

[SWS_Icu_00376] ⌈If development error detection is enabled the function Icu_DisableEdgeDetection shall check the parameter Channel. If Channel is invalid (invalid identifier or channel not configured for mode ICU_MODE_SIGNAL_EDGE_DETECT), the function Icu_DisableEdgeDetection shall raise development error ICU_E_PARAM_CHANNEL.⌋

[SWS_Icu_00398]

Upstream requirements: SRS_BSW_00323, SRS_BSW_00406

⌈If development error detection for the Icu module is enabled: This function shall raise development error ICU_E_UNINIT when the function Icu_Init has not been called.⌋

8.3.18 Icu_DisableEdgeCount

[SWS_Icu_00206] Definition of API function Icu_DisableEdgeCount ⌈

Service Name	Icu_DisableEdgeCount	
Syntax	`void Icu_DisableEdgeCount ( Icu_ChannelType Channel )`	
Service ID [hex]	0x0e	
Sync/Async	Synchronous	
Reentrancy	Reentrant (limited according to ICU050)	
Parameters (in)	Channel	Numeric identifier of the ICU channel
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	This function disables the counting of edges of the given channel.	
Available via	Icu.h	

⌋

[SWS_Icu_00079]

Upstream requirements: SRS_Icu_12433

⌈The function Icu_DisableEdgeCount shall disable the counting of edges of the given channel.⌋

[SWS_Icu_00330] ⌈The function `Icu_DisableEdgeCount` shall be re-entrant.

To reset the edge counter, the service `Icu_ResetEdgeCount` is available.⌋

[SWS_Icu_00103]

Upstream requirements: SRS_BSW_00410, SRS_BSW_00171

⌈The function `Icu_DisableEdgeCount` shall be pre-compile time configurable by the configuration parameter IcuEdgeCountApi.⌋

[SWS_Icu_00331] ⌈The function `Icu_DisableEdgeCount` shall be configurable ON/OFF by the configuration parameter IcuEdgeCountApi.⌋

[SWS_Icu_00173] ⌈If development error detection is enabled the function `Icu_DisableEdgeCount` shall check the parameter `Channel`. If `Channel` is invalid (invalid identifier or channel not configured for mode ICU_MODE_EDGE_COUNTER), the function `Icu_DisableEdgeCount` shall raise development error `ICU_E_PARAM_CHANNEL`.⌋

[SWS_Icu_00399]

Upstream requirements: SRS_BSW_00323, SRS_BSW_00406

⌈If development error detection for the Icu module is enabled: This function shall raise development error `ICU_E_UNINIT` when the function `Icu_Init` has not been called.⌋

8.3.19 Icu_GetEdgeNumbers

[SWS_Icu_00207] Definition of API function Icu_GetEdgeNumbers ⌈

Service Name	Icu_GetEdgeNumbers	
Syntax	`Icu_EdgeNumberType Icu_GetEdgeNumbers ( Icu_ChannelType Channel )`	
Service ID [hex]	0x0f	
Sync/Async	Synchronous	
Reentrancy	Reentrant (limited according to ICU050)	
Parameters (in)	Channel	Numeric identifier of the ICU channel
Parameters (inout)	None	
Parameters (out)	None	
Return value	Icu_EdgeNumberType	Abstract return type to cover different microcontrollers.
Description	This function reads the number of counted edges.	
Available via	Icu.h	

⌋

[SWS_Icu_00080]

Upstream requirements: SRS_Icu_12434

⌈The function `Icu_GetEdgeNumbers` shall read the number of counted edges after the last call of `Icu_ResetEdgeCount`.⌋

[SWS_Icu_00332] ⌈The function `Icu_GetEdgeNumbers` shall be re-entrant.⌋

[SWS_Icu_00104]

Upstream requirements: SRS_BSW_00410, SRS_BSW_00171

⌈The function `Icu_GetEdgeNumbers` shall be pre compile time configurable by the configuration parameter: Icu_EDGE_COUNT_API.⌋

[SWS_Icu_00333] ⌈The function `Icu_GetEdgeNumbers` shall be configurable ON/OFF by the configuration parameter: ICU_EDGE_COUNT_API.⌋

[SWS_Icu_00174] ⌈If development error detection is enabled, the function `Icu_GetEdgeNumbers` shall check the parameter `Channel`. If `Channel` is invalid (invalid identifier or channel not configured for mode ICU_MODE_EDGE_COUNTER), the function `Icu_GetEdgeNumbers` shall raise development error `ICU_E_PARAM_CHANNEL`.⌋

[SWS_Icu_00175] ⌈If development error detection is enabled the function `Icu_GetEdgeNumbers` shall return "0" if an error is detected.⌋

[SWS_Icu_00400]

Upstream requirements: SRS_BSW_00323, SRS_BSW_00406

⌈If development error detection for the Icu module is enabled: This function shall raise development error `ICU_E_UNINIT` when the function `Icu_Init` has not been called.⌋

8.3.20 Icu_StartSignalMeasurement

[SWS_Icu_00208] Definition of API function Icu_StartSignalMeasurement ⌈

Service Name	Icu_StartSignalMeasurement	
Syntax	`void Icu_StartSignalMeasurement ( Icu_ChannelType Channel )`	
Service ID [hex]	0x13	
Sync/Async	Asynchronous	
Reentrancy	Reentrant (limited according to ICU050)	
Parameters (in)	Channel	Numeric identifier of the ICU channel
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	This function starts the measurement of signals.	
Available via	Icu.h	

⌋

[SWS_Icu_00334] ⌈The function `Icu_StartSignalMeasurement` shall be re-entrant.⌋

[SWS_Icu_00140] ⌈The function Icu_StartSignalMeasurement shall start the measurement of signals beginning with the configured default start edge which occurs first after the call of this service.⌋

[SWS_Icu_00141] ⌈The function Icu_StartSignalMeasurement shall only be available in Measurement Mode "ICU_MODE_SIGNAL_MEASUREMENT".⌋

[SWS_Icu_00146] ⌈The function Icu_StartSignalMeasurement shall reset the state for the given channel to ICU_IDLE.⌋

[SWS_Icu_00142] ⌈The function Icu_StartSignalMeasurement shall be pre-compile time configurable by the configuration parameter IcuSignalMeasurementApi.⌋

[SWS_Icu_00335] ⌈The function Icu_StartSignalMeasurement shall be configurable ON/OFF by the configuration parameter IcuSignalMeasurementApi.⌋

[SWS_Icu_00176] ⌈If development error detection is enabled, the function Icu_StartSignalMeasurement shall check the parameter Channel. If Channel is invalid (invalid identifier or channel not configured for mode ICU_MODE_SIGNAL_MEASUREMENT), the function Icu_StartSignalMeasurement shall raise development error ICU_E_PARAM_CHANNEL.⌋

[SWS_Icu_00401]

Upstream requirements: SRS_BSW_00323, SRS_BSW_00406

⌈If development error detection for the Icu module is enabled: This function shall raise development error ICU_E_UNINIT when the function Icu_Init has not been called.⌋

8.3.21 Icu_StopSignalMeasurement

[SWS_Icu_00209] Definition of API function Icu_StopSignalMeasurement ⌈

Service Name	Icu_StopSignalMeasurement	
Syntax	`void Icu_StopSignalMeasurement ( Icu_ChannelType Channel )`	
Service ID [hex]	0x14	
Sync/Async	Synchronous	
Reentrancy	Reentrant (limited according to ICU050)	
Parameters (in)	Channel	Numeric identifier of the ICU channel
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	This function stops the measurement of signals of the given channel.	
Available via	Icu.h	

⌋

[SWS_Icu_00336] ⌈The function `Icu_StopSignalMeasurement` shall be re-entrant.⌋

[SWS_Icu_00143] ⌈The function `Icu_StopSignalMeasurement` shall stop the measurement of signals of the given channel.⌋

[SWS_Icu_00144] ⌈The function `Icu_StopSignalMeasurement` shall only be available in Measurement Mode: "ICU_MODE_SIGNAL_MEASUREMENT".⌋

[SWS_Icu_00145] ⌈The function `Icu_StopSignalMeasurement` shall be pre compile time configurable by the configuration parameter IcuSignalMeasurementApi.⌋

[SWS_Icu_00337] ⌈The function `Icu_StopSignalMeasurement` shall be configurable ON/OFF by the configuration parameter IcuSignalMeasurementApi.⌋

[SWS_Icu_00177] ⌈If development error detection is enabled the function `Icu_StopSignalMeasurement` shall check the parameter `Channel`. If `Channel` is invalid (invalid identifier or channel not configured for mode ICU_MODE_SIGNAL_MEASUREMENT), the function `Icu_StopSignalMeasurement` shall raise development error `ICU_E_PARAM_CHANNEL`.⌋

[SWS_Icu_00402]

Upstream requirements: SRS_BSW_00323, SRS_BSW_00406

⌈If development error detection for the Icu module is enabled: This function shall raise development error `ICU_E_UNINIT` when the function `Icu_Init` has not been called.⌋

8.3.22 Icu_GetTimeElapsed

[SWS_Icu_00210] Definition of API function Icu_GetTimeElapsed ⌈

Service Name	Icu_GetTimeElapsed	
Syntax	`Icu_ValueType Icu_GetTimeElapsed ( Icu_ChannelType Channel )`	
Service ID [hex]	0x10	
Sync/Async	Synchronous	
Reentrancy	Reentrant (limited according to ICU050)	
Parameters (in)	Channel	Numeric identifier of the ICU channel
Parameters (inout)	None	
Parameters (out)	None	
Return value	Icu_ValueType	see Description
Description	This function reads the elapsed Signal Low Time for the given channel.	
Available via	Icu.h	

⌋

[SWS_Icu_00338] ⌈The function `Icu_GetTimeElapsed` shall be re-entrant.⌋

[SWS_Icu_00081]

Upstream requirements: SRS_SPAL_12063, SRS_Icu_12442

⌈The function `Icu_GetTimeElapsed` shall read the elapsed Signal Low Time for the given channel that is configured in Measurement Mode "Signal Measurement, Signal Low Time". The elapsed time is measured between a falling edge and the consecutive rising edge of the channel.⌋

[SWS_Icu_00082]

Upstream requirements: SRS_SPAL_12063, SRS_Icu_12435

⌈The function `Icu_GetTimeElapsed` shall read the elapsed Signal High Time for the given channel that is configured in Measurement Mode "Signal Measurement, Signal High Time". The elapsed time is measured between a rising edge and the consecutive falling edge of the channel.⌋

[SWS_Icu_00083]

Upstream requirements: SRS_SPAL_12063, SRS_Icu_12443

⌈The function `Icu_GetTimeElapsed` shall read the elapsed Signal Period Time for the given channel that is configured in Measurement Mode "Signal Measurement, Signal Period Time". The elapsed time is measured between consecutive rising (or falling) edges of the channel. The period start edge is configurable.⌋

[SWS_Icu_00136] ⌈The function `Icu_GetTimeElapsed` shall return "0" in case no requested time has been captured.⌋

Hint: See Figure 9.19, Letter "A" for more details.

[SWS_Icu_00339] ⌈The function `Icu_GetTimeElapsed` shall return "0" in case the capturing of a requested time is ongoing and not finished.⌋

Hint: See Figure 9.19, Letter "B" for more details.

[SWS_Icu_00340] ⌈The function `Icu_GetTimeElapsed` shall return "0" in case a captured time was already returned once by this service and this service is called again.⌋

Hint: See Figure 9.19, Letter "D" for more details.

[SWS_Icu_00105]

Upstream requirements: SRS_BSW_00410, SRS_BSW_00171

⌈The function `Icu_GetTimeElapsed` shall be pre compile time configurable by the configuration parameter IcuGetTimeElapsedApi.⌋

[SWS_Icu_00341] ⌈The function `Icu_GetTimeElapsed` shall be configurable ON/OFF by the configuration parameter IcuGetTimeElapsedApi.⌋

[SWS_Icu_00178] ⌈If development error detection is enabled, the parameter `Channel` shall be checked by this service. If `Channel` is invalid (invalid identifier or channel not configured for mode ICU_MODE_SIGNAL_MEASUREMENT), then the error `ICU_E_PARAM_CHANNEL` shall be reported to the Default Error Tracer.⌋

[SWS_Icu_00179] ⌈If development error detection is enabled and an error is detected this service shall return "0".⌋

[SWS_Icu_00403]

Upstream requirements: SRS_BSW_00323, SRS_BSW_00406

⌈If development error detection for the Icu module is enabled: This function shall raise development error `ICU_E_UNINIT` when the function `Icu_Init` has not been called.⌋

8.3.23 Icu_GetDutyCycleValues

[SWS_Icu_00211] Definition of API function Icu_GetDutyCycleValues ⌈

Service Name	Icu_GetDutyCycleValues	
Syntax	`void Icu_GetDutyCycleValues ( Icu_ChannelType Channel, Icu_DutyCycleType* DutyCycleValues )`	
Service ID [hex]	0x11	
Sync/Async	Synchronous	
Reentrancy	Reentrant (limited according to ICU050)	
Parameters (in)	Channel	Numeric identifier of the ICU channel
Parameters (inout)	None	
Parameters (out)	DutyCycleValues	Pointer to a buffer where the results (high time and period time) shall be placed.
Return value	None	
Description	This function reads the coherent active time and period time for the given ICU Channel.	
Available via	Icu.h	

⌋

[SWS_Icu_00342] ⌈The function `Icu_GetDutyCycleValues` shall be re-entrant.⌋

[SWS_Icu_00084]

Upstream requirements: SRS_Icu_12436

⌈The function `Icu_GetDutyCycleValues` shall read the coherent active time and period time for the given ICU Channel, if it is configured in Measurement Mode "Signal Measurement, Duty Cycle Values".⌋

[SWS_Icu_00137] ⌈The function `Icu_GetDutyCycleValues` shall return "0" in case no coherent active- and period time has been captured.⌋

Hint: See Figure 9.19, Letter "A" for more details.

[SWS_Icu_00343] ⌈The function `Icu_GetDutyCycleValues` shall return "0" in case the capturing of a requested high- and period time is ongoing and not finished (meant: the function shall return "0" until the first valid value has been captured and the captured value shall be stored until a new value is captured).⌋

Hint: See Figure 9.19, Letter "B" for more details.

[SWS_Icu_00344] ⌈The function `Icu_GetDutyCycleValues` shall return "0" in case captured duty cycle values were already returned once by this service and this service is called again.⌋

Hint: See Figure 9.19, Letter "D" for more details.

[SWS_Icu_00106]

Upstream requirements: SRS_BSW_00410, SRS_BSW_00171

⌈The function `Icu_GetDutyCycleValues` shall be pre compile time configurable by the configuration parameter IcuGetDutyCycleValuesApi.⌋

[SWS_Icu_00345] ⌈The function `Icu_GetDutyCycleValues` shall be configurable ON/OFF by the configuration parameter IcuGetDutyCycleValuesApi.⌋

[SWS_Icu_00180] ⌈If development error detection is enabled: the function `Icu_GetDutyCycleValues` shall check the parameter `Channel`. If `Channel` is invalid (invalid identifier or channel not configured for mode ICU_MODE_SIGNAL_MEASUREMENT, Duty Cycle Values), the function `Icu_GetDutyCycleValues` shall raise development error `ICU_E_PARAM_CHANNEL`.⌋

[SWS_Icu_00181] ⌈If development error detection is enabled, the function `Icu_GetDutyCycleValues` shall check the parameter `DutyCycleValues`. If `DutyCycleValues` is invalid, the function `Icu_GetDutyCycleValues` shall raise development error `ICU_E_PARAM_POINTER`.⌋

[SWS_Icu_00404]

Upstream requirements: SRS_BSW_00323, SRS_BSW_00406

⌈If development error detection for the Icu module is enabled: This function shall raise development error `ICU_E_UNINIT` when the function `Icu_Init` has not been called.⌋

8.3.24 Icu_DisableNotificationAsync

[SWS_Icu_91002] Definition of API function Icu_DisableNotificationAsync ⌈

Service Name	Icu_DisableNotificationAsync	
Syntax	`void Icu_DisableNotificationAsync ( Icu_ChannelType Channel )`	
Service ID [hex]	0x18	
Sync/Async	Asynchronous	
Reentrancy	Reentrant (limited according to ICU050)	
Parameters (in)	Channel	Numeric identifier of the ICU channel.
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	This function disables the notification of a channel.	
Available via	Icu.h	

⌋

8.3.25 Icu_EnableNotificationAsync

[SWS_Icu_91003] Definition of API function Icu_EnableNotificationAsync ⌈

Service Name	Icu_EnableNotificationAsync	
Syntax	`void Icu_EnableNotificationAsync ( Icu_ChannelType Channel )`	
Service ID [hex]	0x19	
Sync/Async	Asynchronous	
Reentrancy	Non Reentrant Reentrant (limited according to ICU050)	
Parameters (in)	Channel	Numeric identifier of the ICU channel.
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	This function enables the notification on the given channel.	
Available via	Icu.h	

⌋

8.4 Scheduled functions

None.

8.5 Expected interfaces

In this chapter, all interfaces required from other modules are listed.

8.5.1 Mandatory Interfaces

This chapter defines all interfaces, which are required, in order to fulfill the core functionality of the module.

[SWS_Icu_91001] Definition of mandatory interfaces required by module Icu ⌈

API Function	*Header File*	*Description*
Det_ReportRuntimeError	Det.h	Service to report runtime errors. If a callout has been configured then this callout shall be called.

⌋

8.5.2 Optional Interfaces

This chapter defines all interfaces which are required to fulfil an optional functionality of the module.

[SWS_Icu_00213] Definition of optional interfaces requested by module Icu ⌈

API Function	*Header File*	*Description*
Det_ReportError	Det.h	Service to report development errors.
EcuM_CheckWakeup	EcuM.h	This function can be called to check the given wakeup sources. It will pass the argument to the integrator function EcuM_CheckWakeupHook. It can also be called by the ISR of a wakeup source to set up the PLL and check other wakeup sources that may be connected to the same interrupt.
EcuM_SetWakeupEvent	EcuM.h	Sets the wakeup event.

⌋

The service EcuM_CheckWakeup will be called if all of the following are true:

- **[SWS_Icu_00055]**
  Upstream requirements: SRS_SPAL_12069, SRS_BSW_00410
  ⌈The static configuration parameter IcuReportWakeupSource is set to "ON"⌋
- **[SWS_Icu_00056]**
  Upstream requirements: SRS_SPAL_12069
  ⌈The module is in mode ICU_MODE_SLEEP⌋
- **[SWS_Icu_00057]**
  Upstream requirements: SRS_SPAL_12069
  ⌈A wakeup event occurs on a wakeup capable ICU channel.⌋

[SWS_Icu_00228] ⌈EcuM_CheckWakeup shall be called within the Interrupt Service Routine servicing the ICU channel wakeup event on wakeup-capable channel.⌋

[SWS_Icu_00229] ⌈The ISR's, providing the wakeup events, shall be responsible for resetting the interrupt flags if required by hardware.⌋

8.5.3 Configurable interfaces

In this chapter all interfaces are listed where the target function could be configured. The target function is usually a call-back function. The names of these kinds of interfaces are not fixed because they are configurable.

[SWS_Icu_00119]

Upstream requirements: SRS_SPAL_12129

⌈The ISRs shall reset the interrupt flags (if needed by hardware) and call the corresponding notification functions.⌋

[SWS_Icu_00018]

Upstream requirements: SRS_SPAL_12056

⌈The Icu notification functions shall be configurable as function pointers within the initialization data structure (`Icu_ConfigType`).⌋

[SWS_Icu_00187]

Upstream requirements: SRS_BSW_00359

⌈The Icu module's notification functions shall have no parameters and no return value.⌋

[SWS_Icu_00214] Definition of configurable interface Icu_SignalNotification_<Channel> ⌈

Service Name	Icu_SignalNotification_<Channel>
Syntax	`void Icu_SignalNotification_<Channel> ( void )`
Sync/Async	Synchronous
Reentrancy	Reentrancy of interface not relevant for this module. (in general it is in this case not reentrant).
Parameters (in)	None
Parameters (inout)	None
Parameters (out)	None
Return value	None
Description	According to the last call of Icu_EnableNotification, this notification function to be called if the requested signal edge (rising / falling / both edges) occurs (once per edge).
Available via	Icu_Externals.h

⌋

[SWS_Icu_00348] ⌈Re-entrancy of operation Icu_SignalNotification_<Channel> is not relevant for this module (In general it is in this case not re-entrant).⌋

[SWS_Icu_00021]

Upstream requirements: SRS_SPAL_00157, SRS_Icu_12369

⌈According to the last call of `Icu_EnableNotification`, the Icu module shall call the notification function Icu_SignalNotification_<Channel> if the requested signal edge (rising / falling / both edges) occurs (once per edge).⌋

[SWS_Icu_00044]

Upstream requirements: SRS_Icu_12305

⌈Only those edge notifications shall be provided, which are supported by hardware.⌋

[SWS_Icu_00042]

Upstream requirements: SRS_Icu_12305

⌈After a call of `Icu_DisableNotification`, the Icu module shall not call the notification function Icu_SignalNotification_<Channel>.⌋

[SWS_Icu_00215] Definition of configurable interface Icu_TimestampNotification_<Channel>

Upstream requirements: SRS_Icu_12444

⌈

Service Name	Icu_TimestampNotification_<Channel>
Syntax	`void Icu_TimestampNotification_<Channel> ( void )`
Sync/Async	Synchronous
Reentrancy	Reentrancy of interface not relevant for this module. (in general it is in this case not reentrant).
Parameters (in)	None
Parameters (inout)	None
Parameters (out)	None
Return value	None
Description	This notification to be called if the number of requested timestamps (Notification interval > 0) are acquired and if the notification has been enabled by the call of Icu_EnableNotification().
Available via	Icu_Externals.h

⌋

[SWS_Icu_00349] ⌈Re-entrancy of the Icu_TimestampNotification_<Channel> is not relevant for this module (in general it is in this case not re-entrant).⌋

[SWS_Icu_00216] ⌈The Icu module shall call the notification Icu_TimestampNotification_<Channel> if the number of requested timestamps (Notification interval > 0) are acquired and if the notification has been enabled by the call of `Icu_EnableNotification`.⌋

[SWS_Icu_00217] ⌈After a call of `Icu_DisableNotification` the Icu module shall NOT call the notification Icu_TimestampNotification_<Channel>.⌋

[SWS_Icu_00218] ⌈The Icu module's notification Icu_TimestampNotification_<Channel> depends on pre-processor switch IcuTimestampApi.⌋

9 Sequence diagrams

9.1 Icu_Init

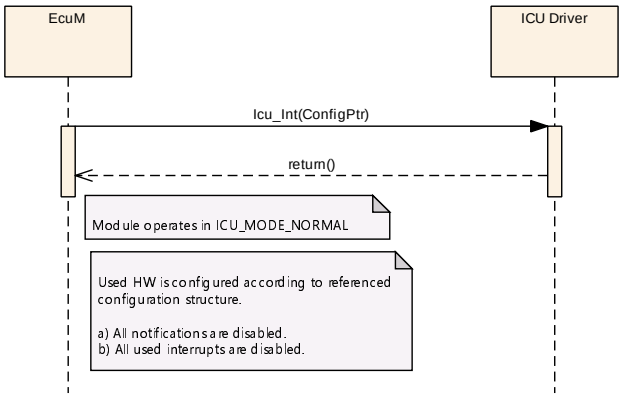


Figure 9.1: Initialization of the ICU driver

9.2 Icu_DeInit

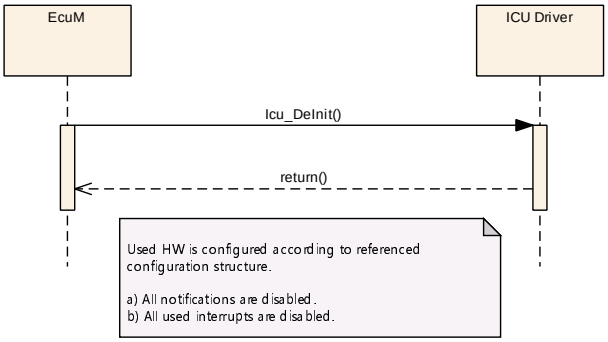


Figure 9.2: De-Initialization of the ICU driver

9.3 Check Wakeup Events

Note: The Sequence charts for the ICU can be found in the ECU State Manager specification [4].

9.4 Icu_SetMode

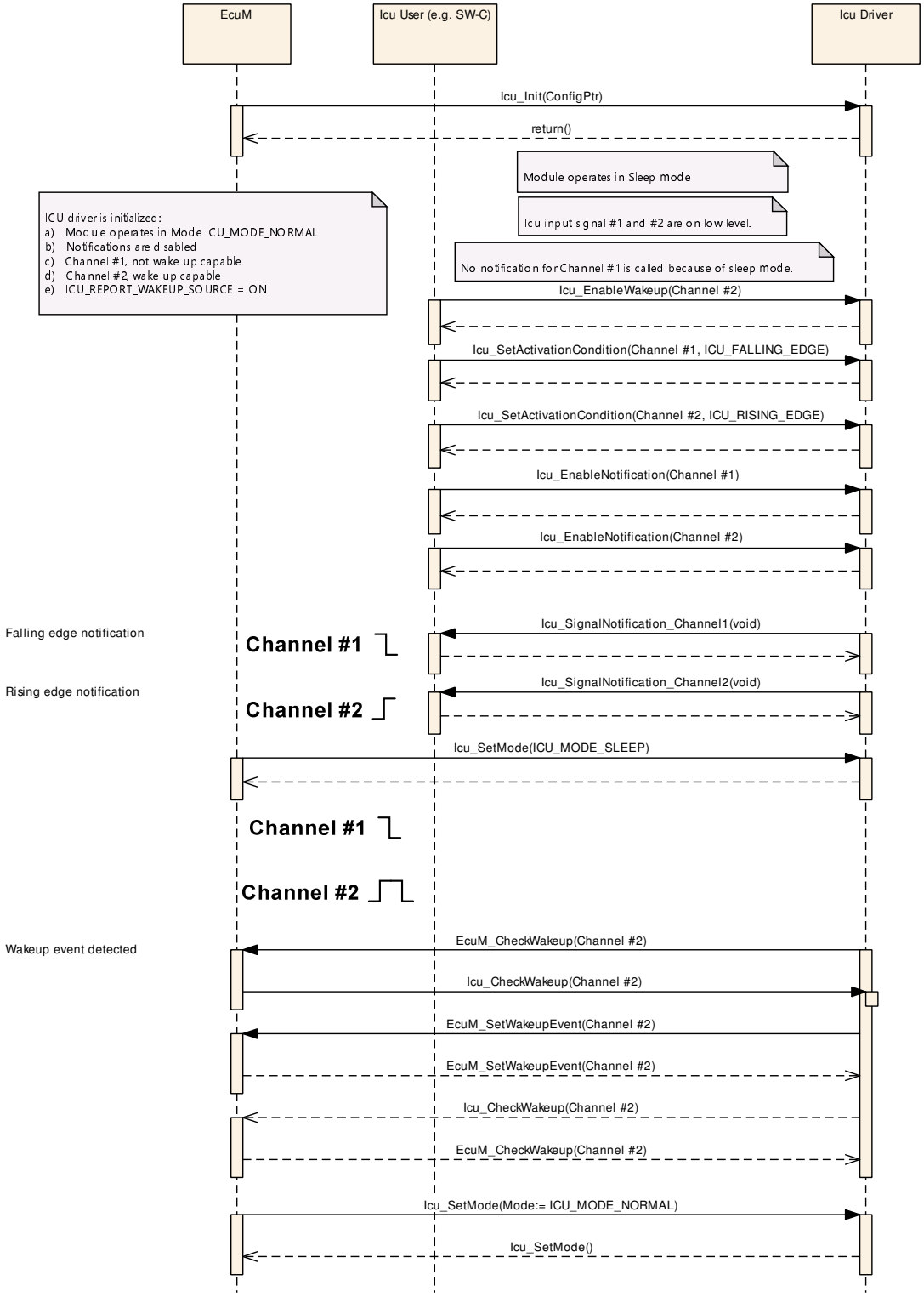


Figure 9.3: Enabled notifications in SLEEP mode

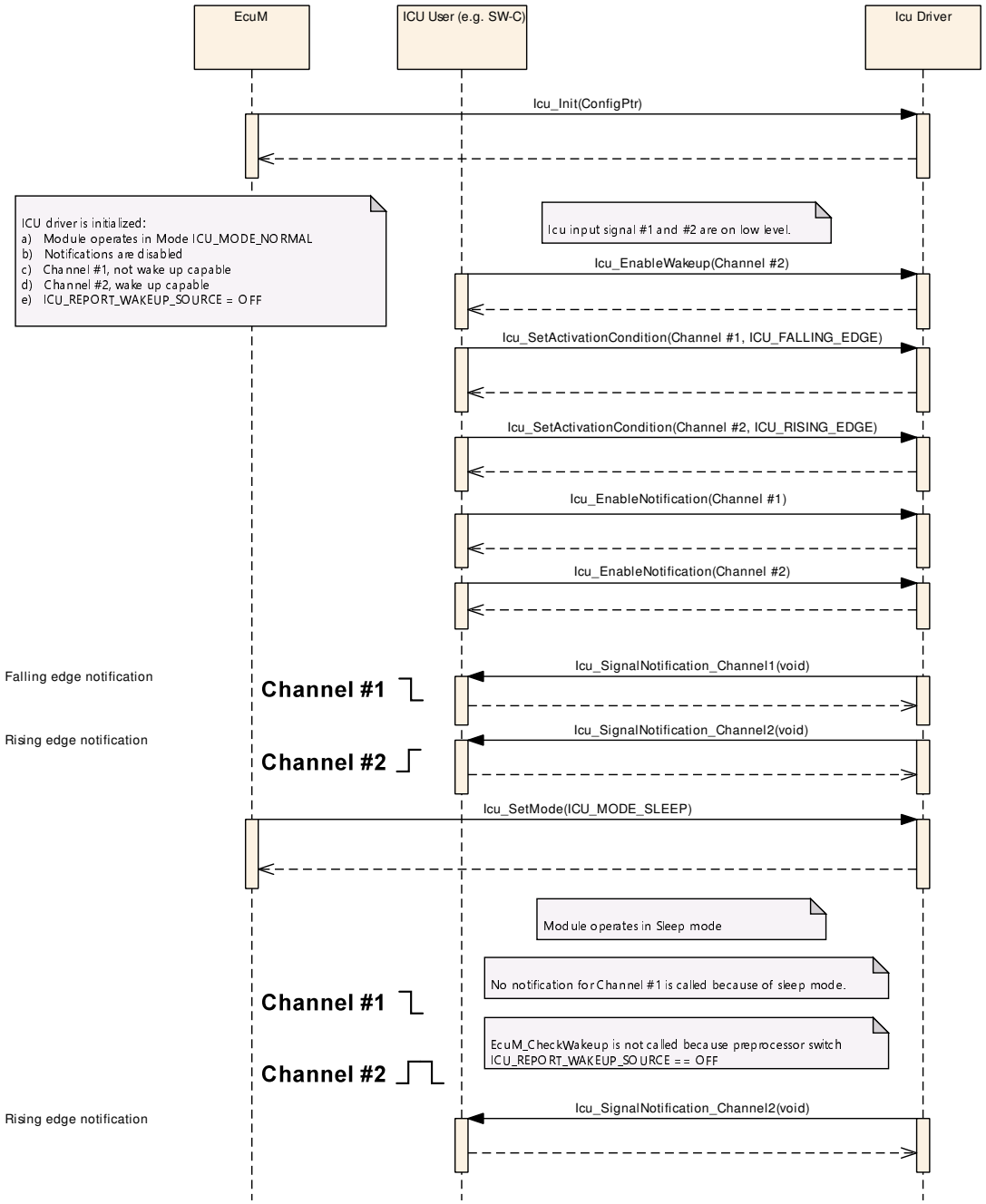


Figure 9.4: Disabled reporting of wakeup sources in SLEEP mode

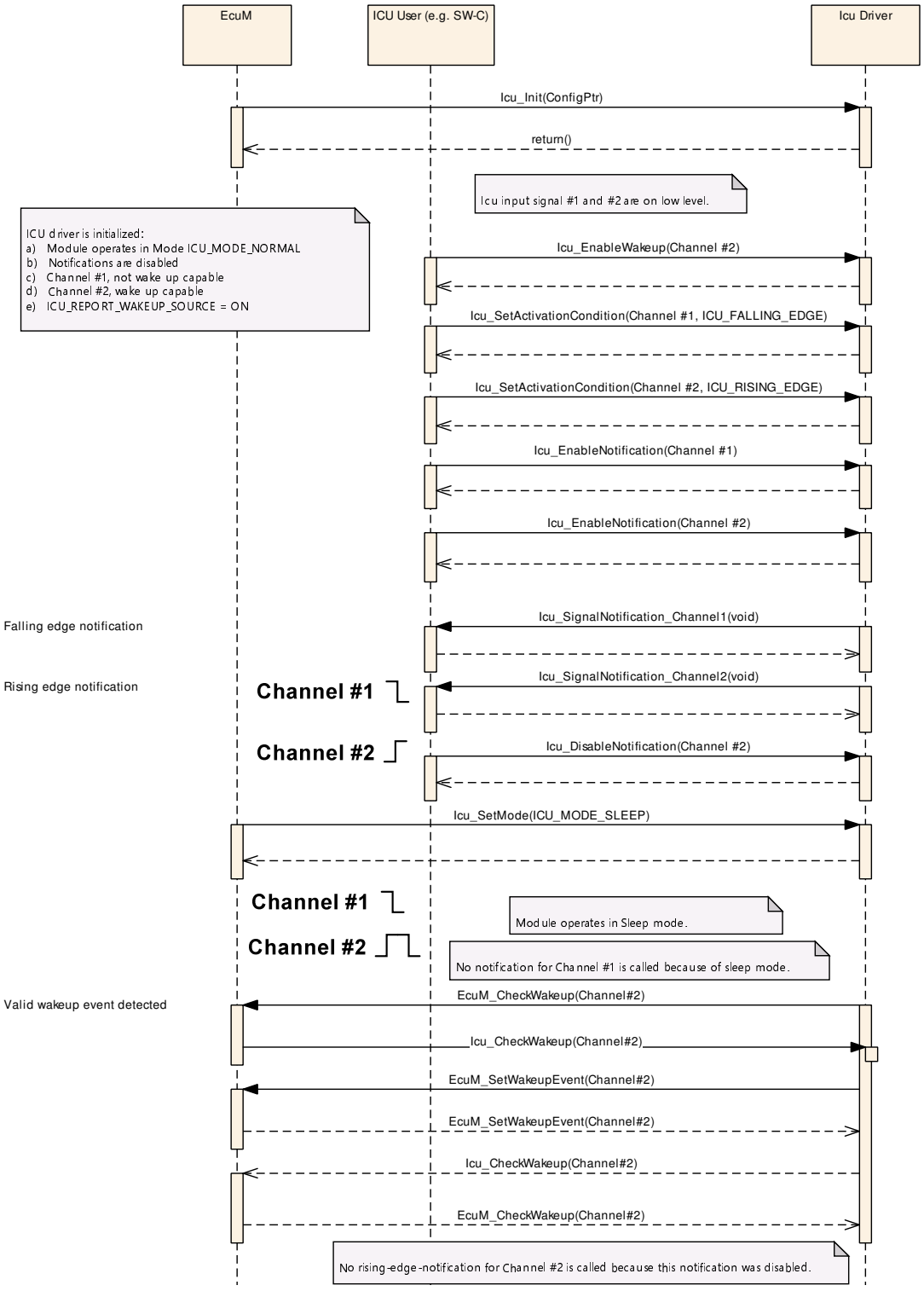


Figure 9.5: Disabled edge notification in SLEEP mode

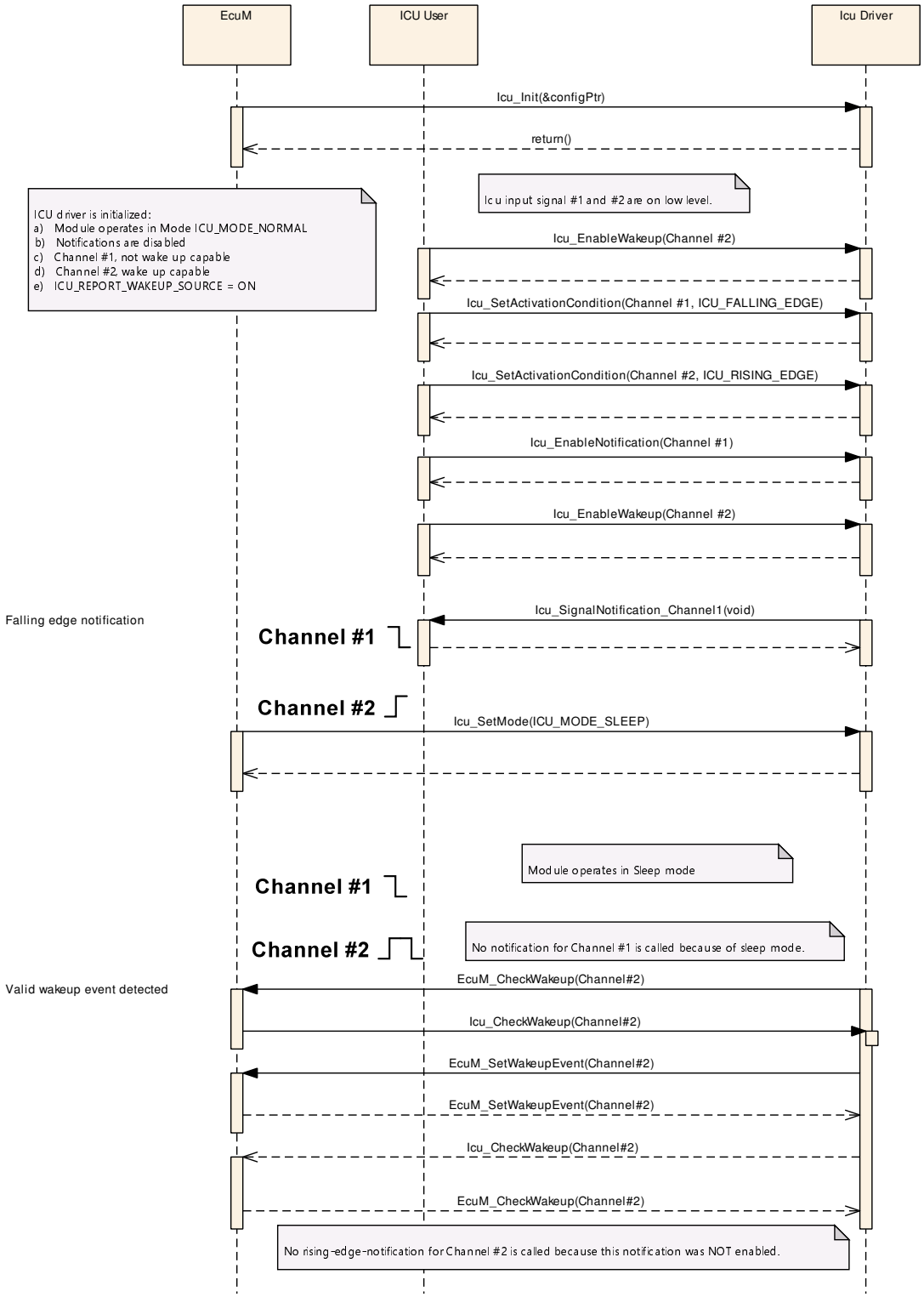


Figure 9.6: Un-Enabled reporting of notifications in SLEEP mode

9.5 Icu_DisableWakeup

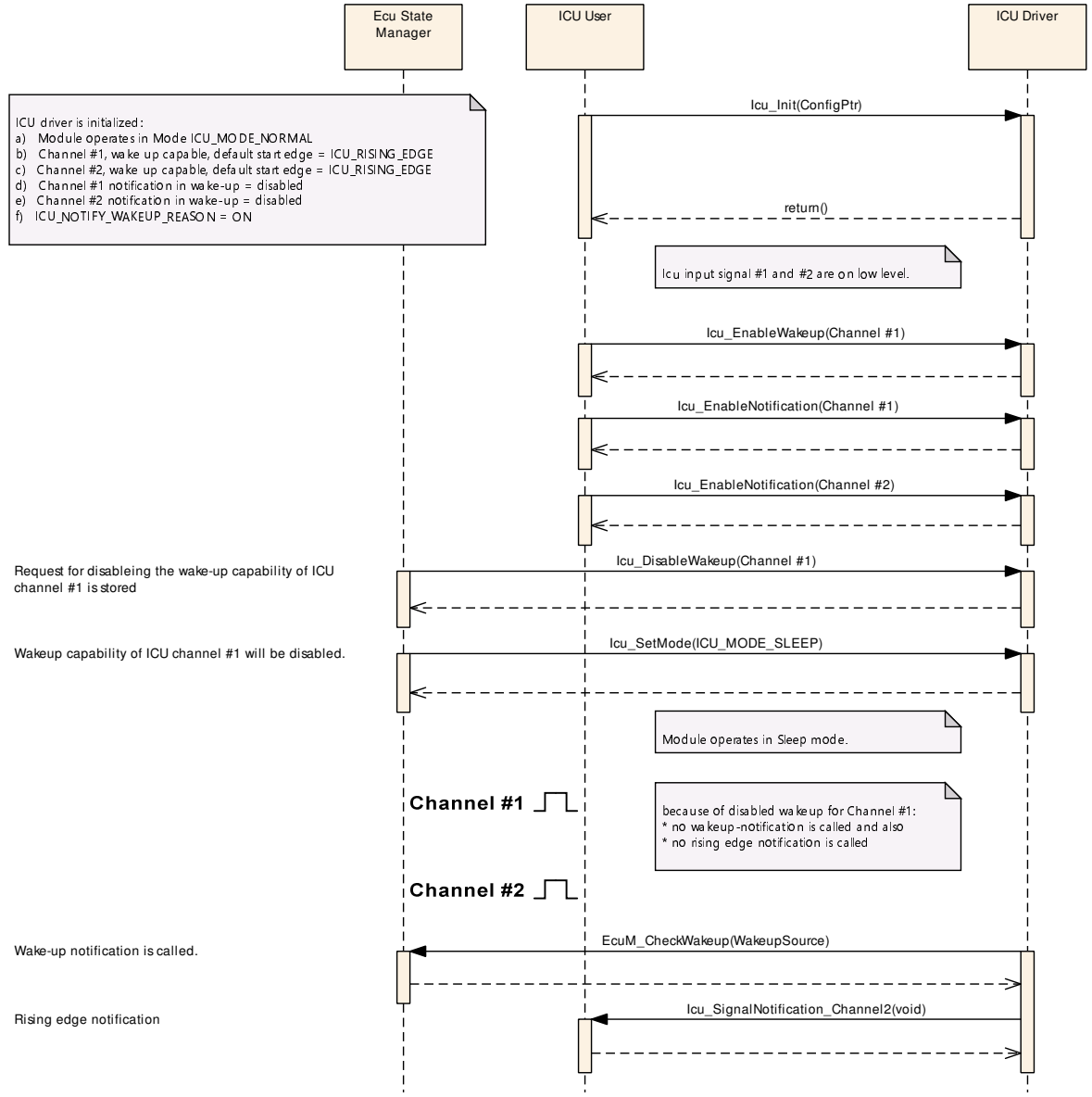


Figure 9.7: Disabling of wakeup-capabilities

9.6 Icu_EnableWakeup

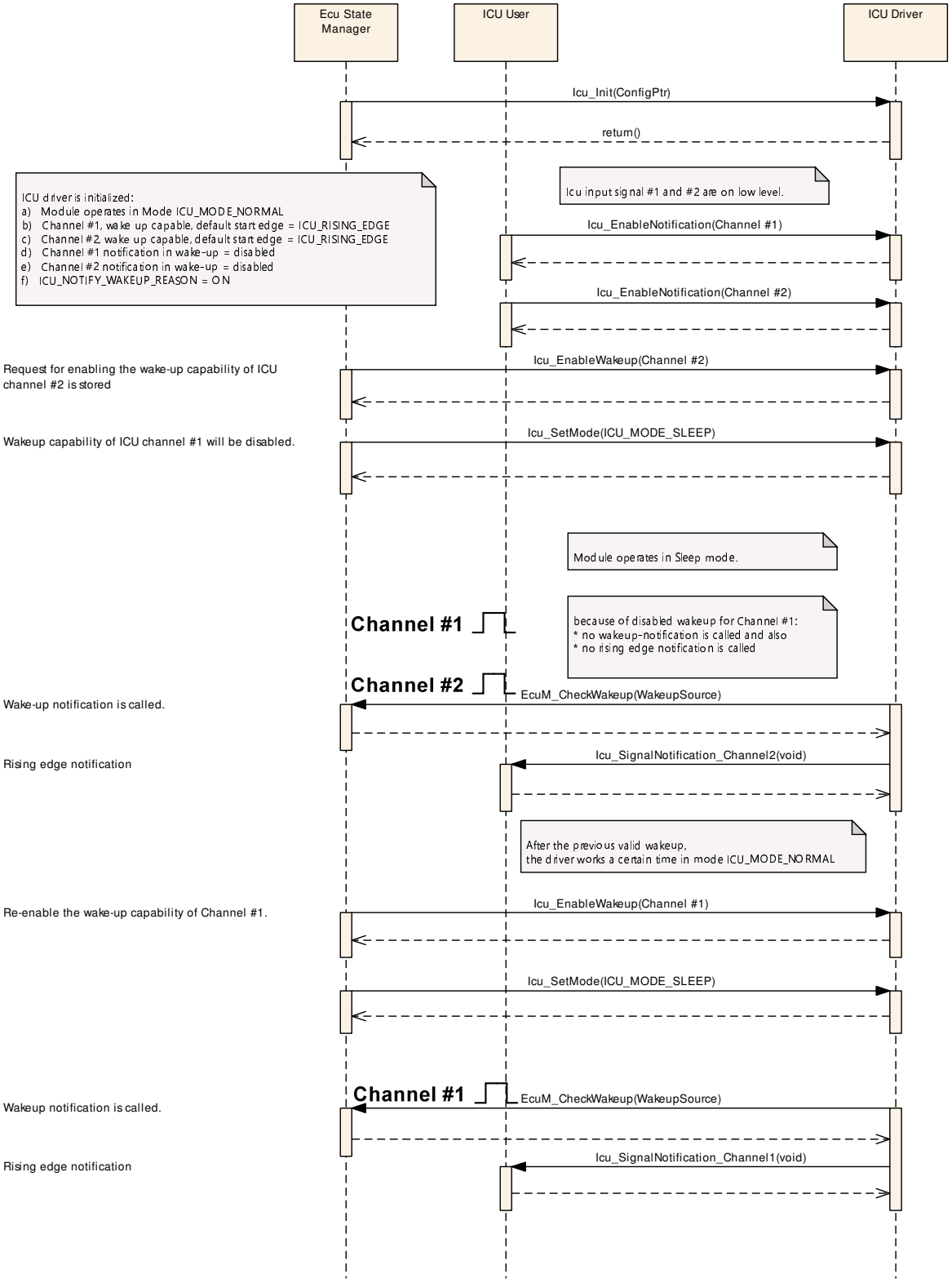


Figure 9.8: Enabling of wakeup-capabilities

9.7 Icu_SetActivationCondition

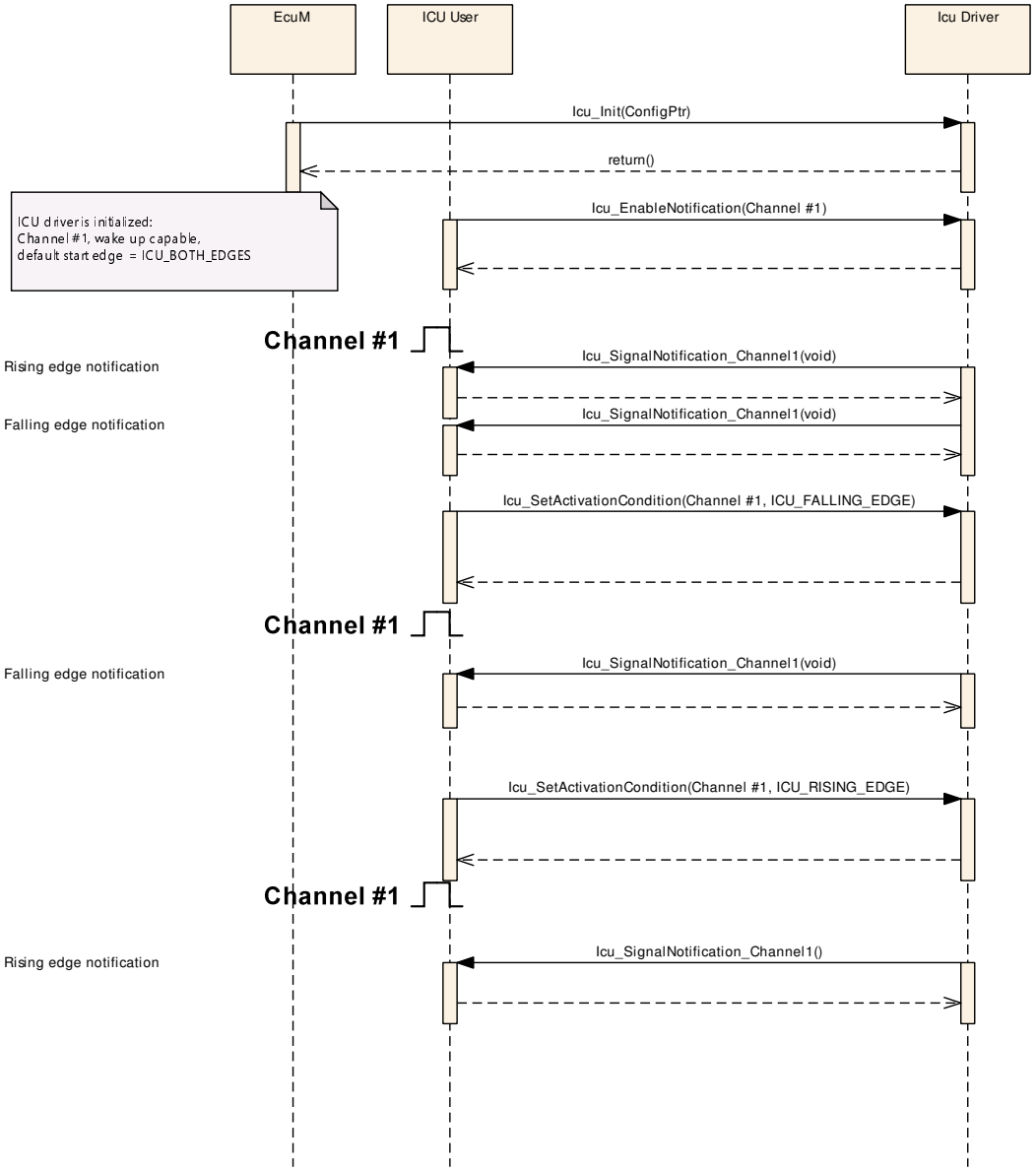


Figure 9.9: Setting up the activation condition for a channel

9.8 Icu_DisableNotification

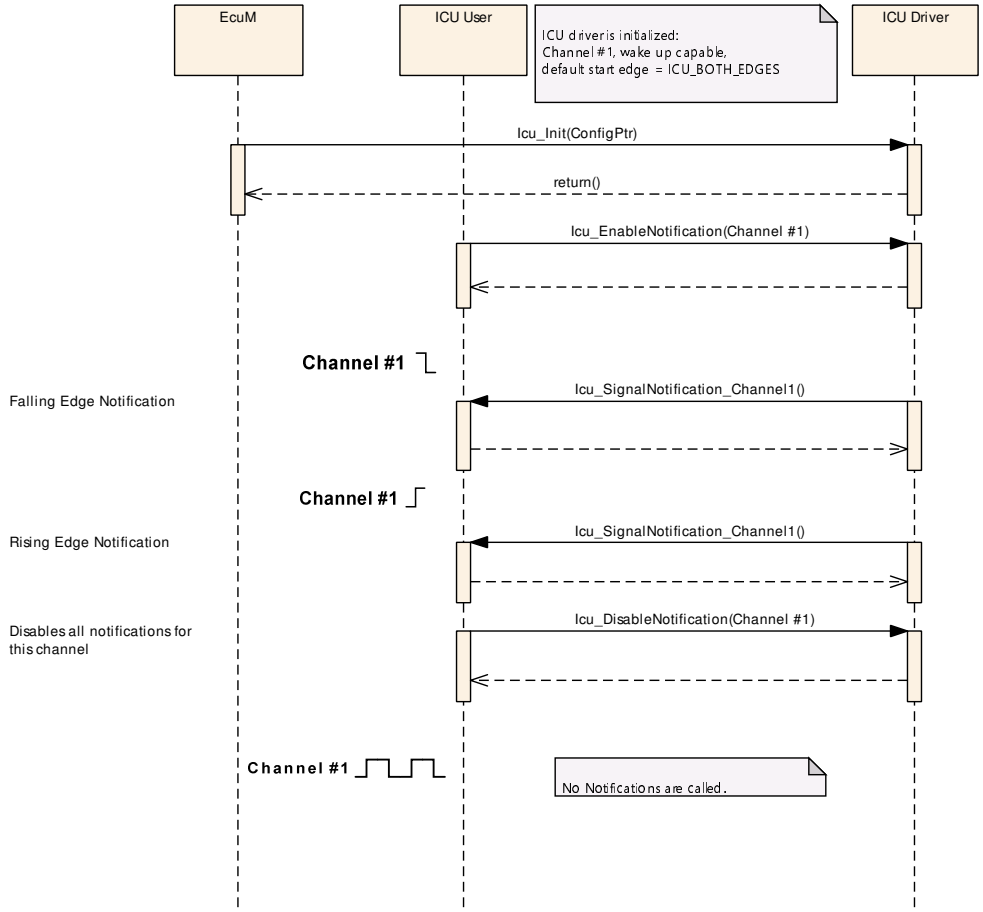


Figure 9.10: Disabling of the notification for a channel

9.9 Icu_EnableNotification

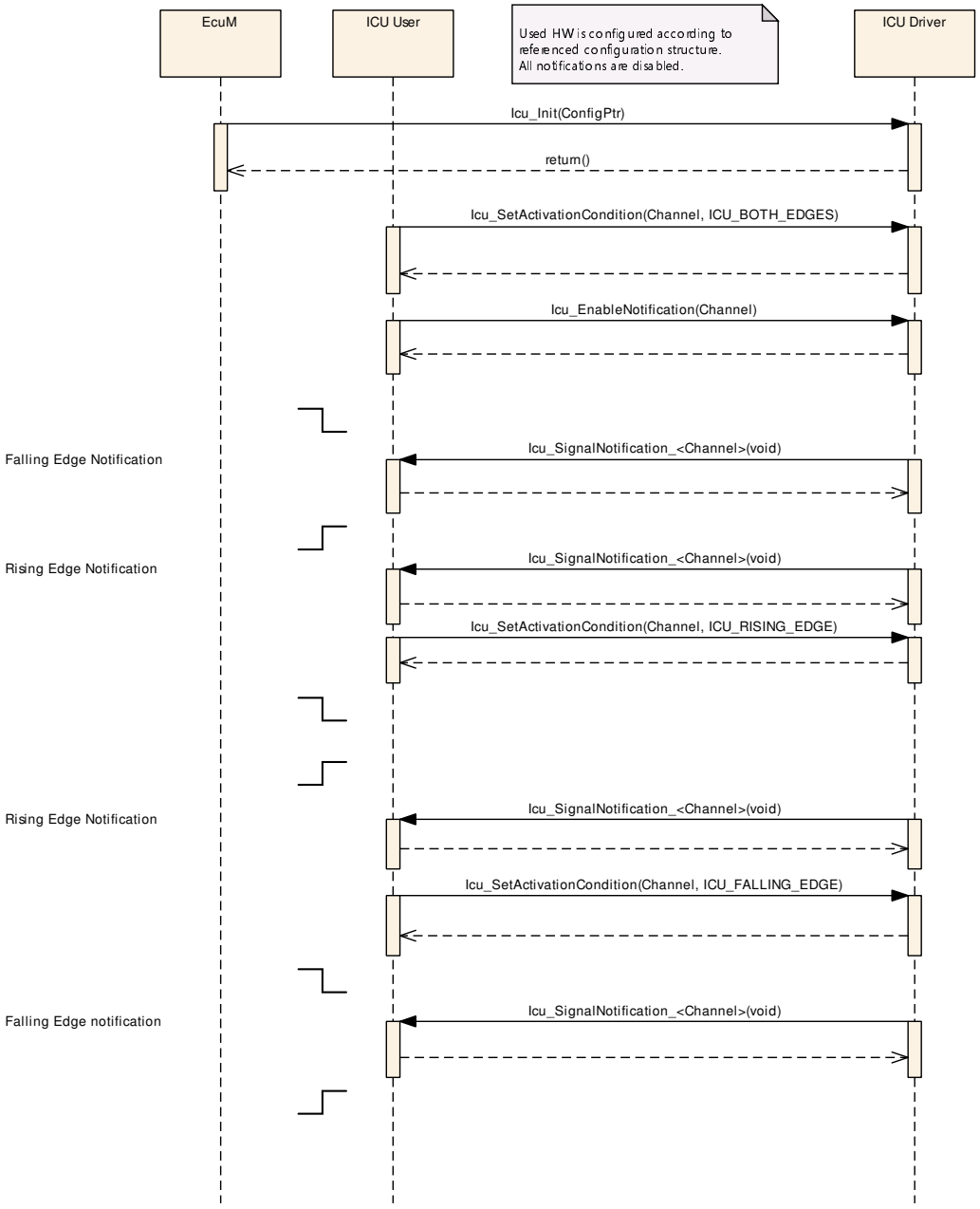


Figure 9.11: Enabling of the edge-notification for a channel

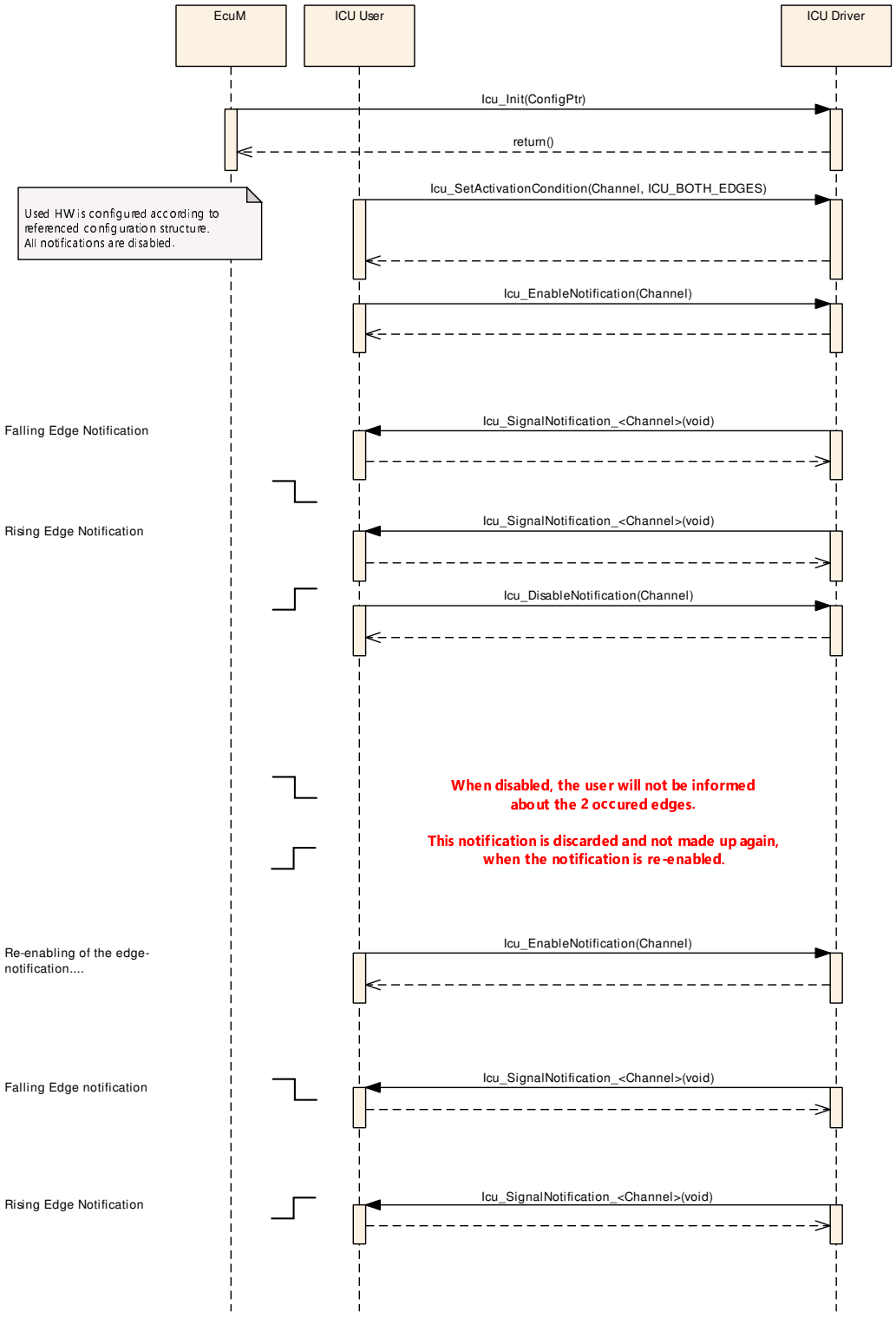


Figure 9.12: Re-enabling of the notification for a channel

9.10 Icu_GetInputState

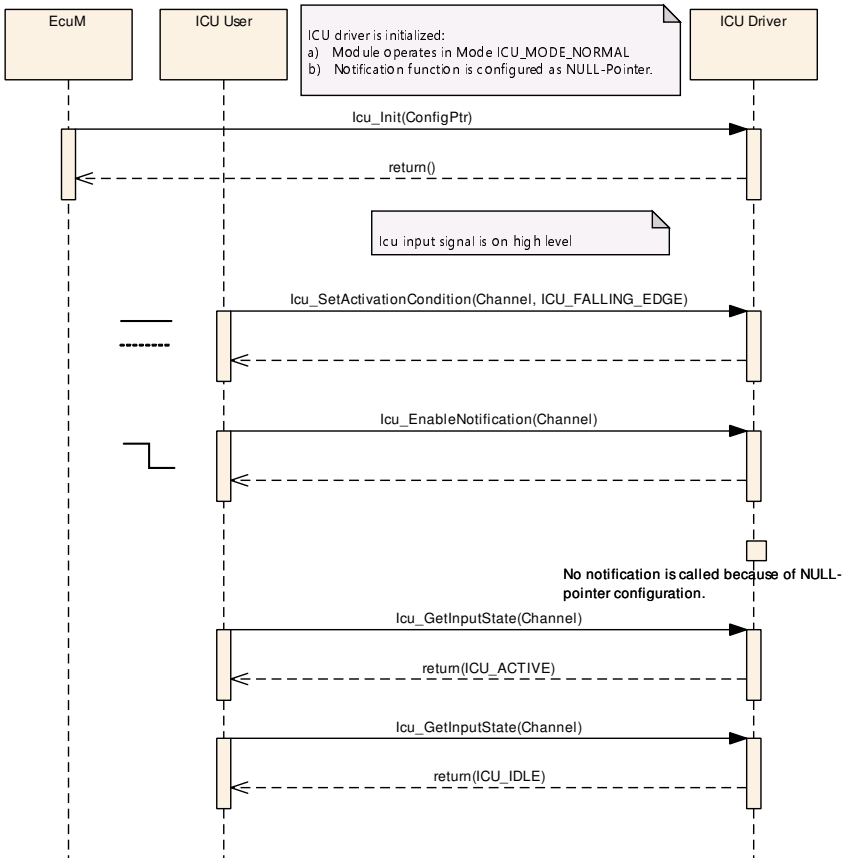


Figure 9.13: Polling of the channel status

9.11 Icu Timestamping

The following figure shall show the interactions between the different timestamp API-services.

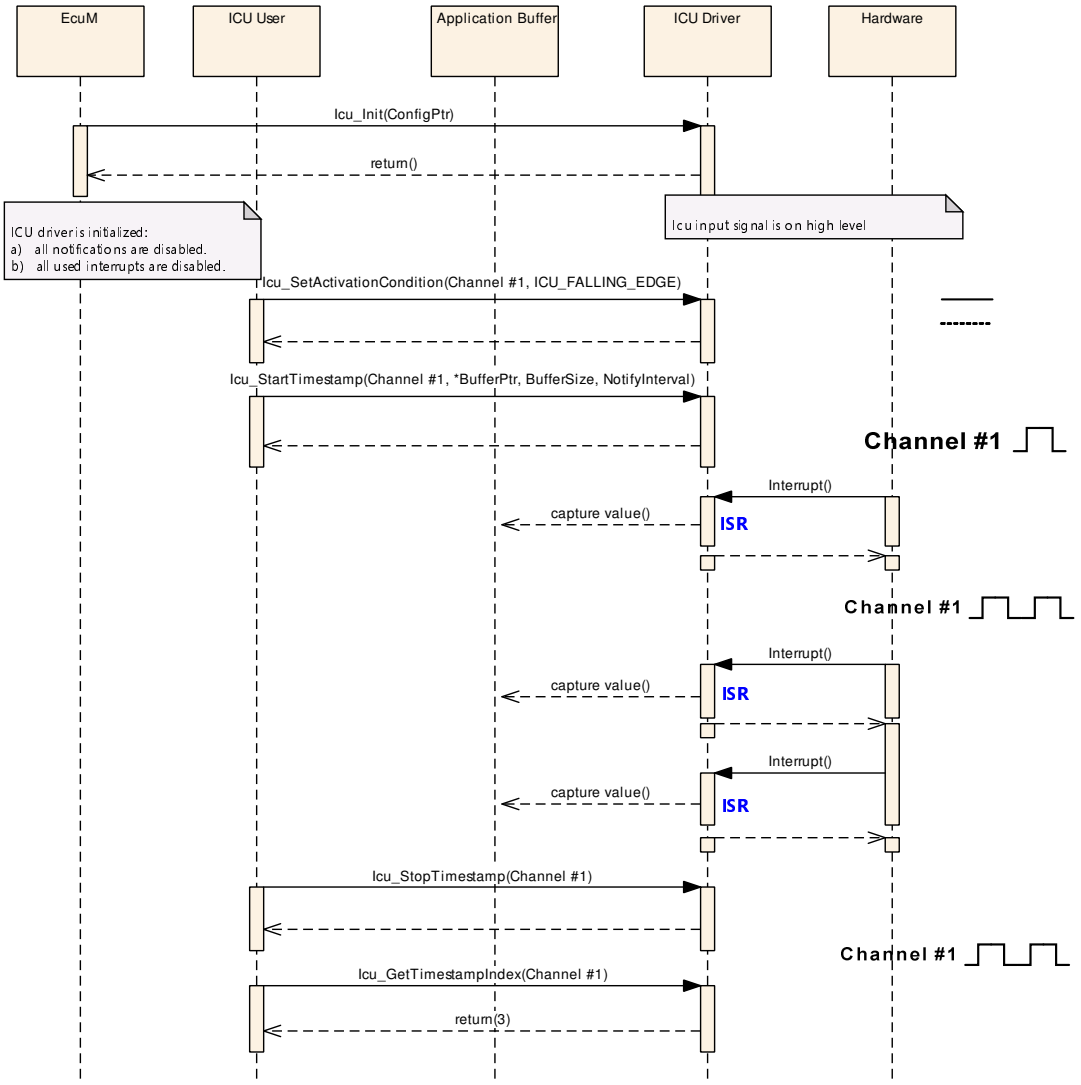
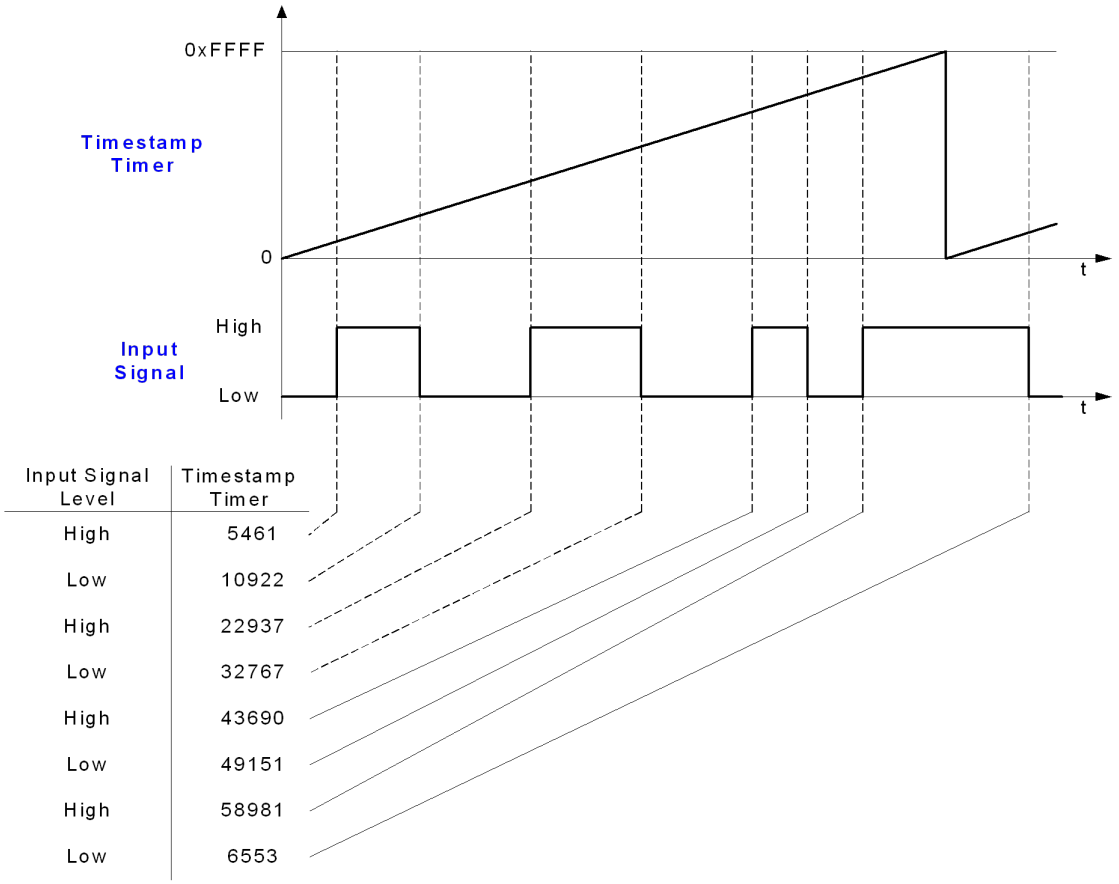


Figure 9.14: Overview of the timestamping functionality of the ICU driver

The Timestamping in general is shown in the following figure:



Input Signal Level	Timestamp Timer
High	5461
Low	10922
High	22937
Low	32767
High	43690
Low	49151
High	58981
Low	6553

Figure 9.15: Timestamping overview

9.12 Icu Edge Counting

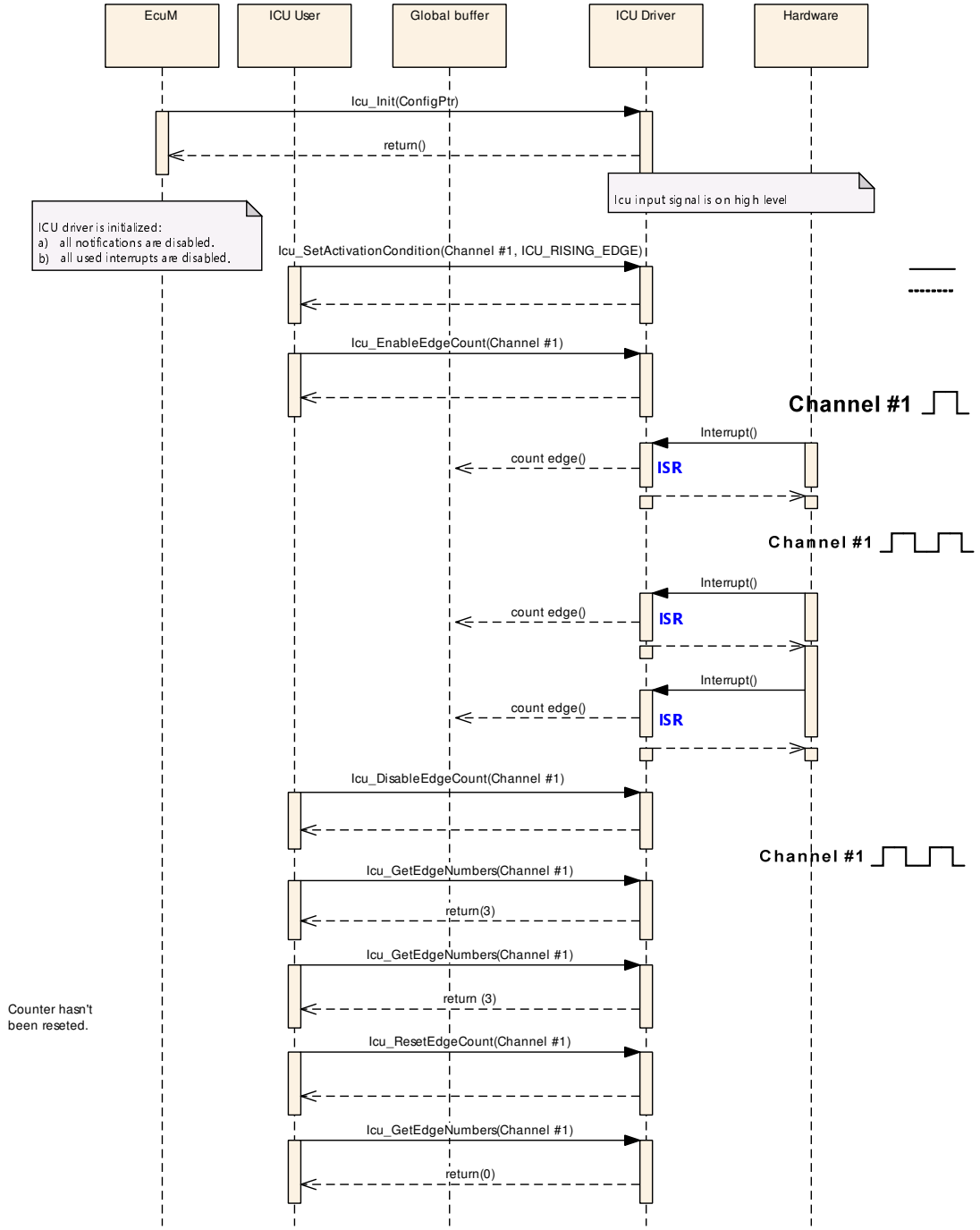


Figure 9.16: Inquire the number of counted edges

9.13 Icu_GetTimeElapsed

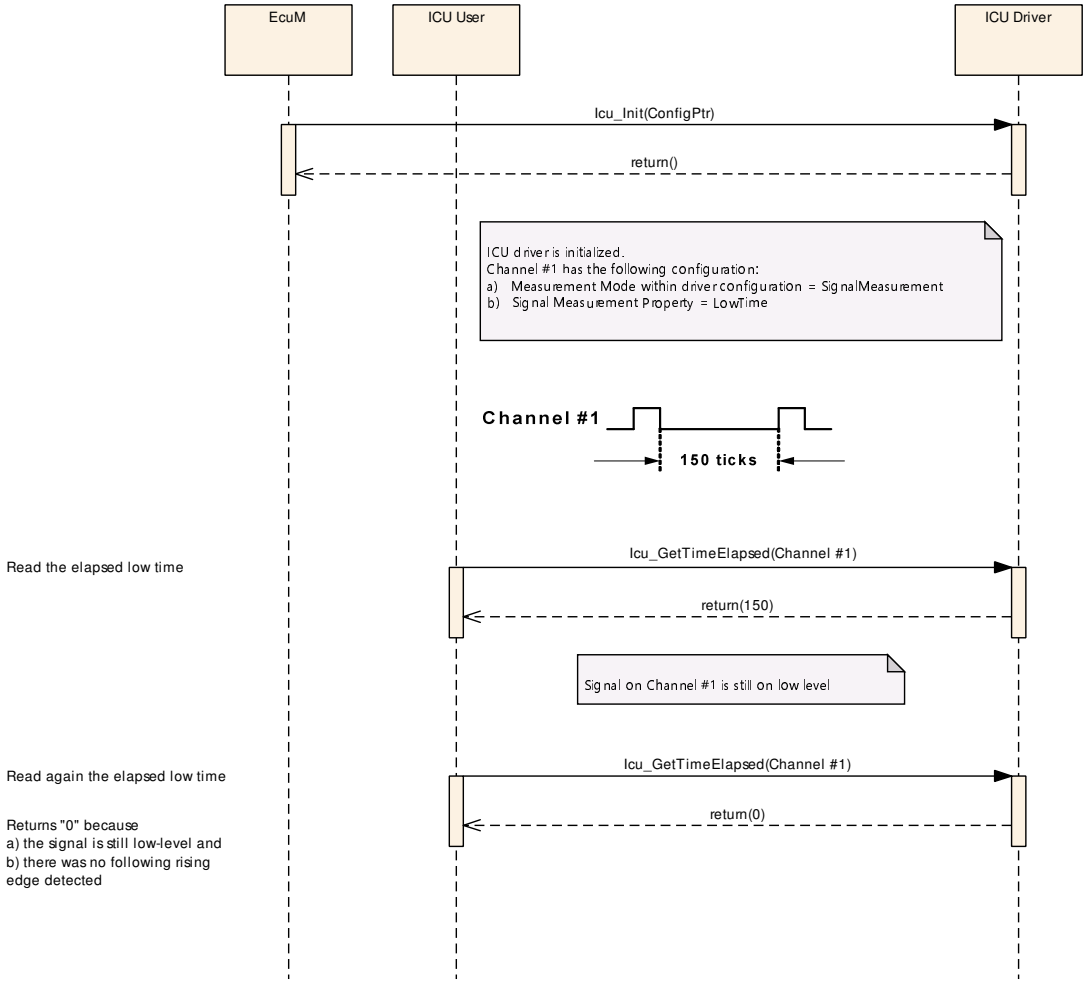


Figure 9.17: Inquire the elapsed level-time of a channel

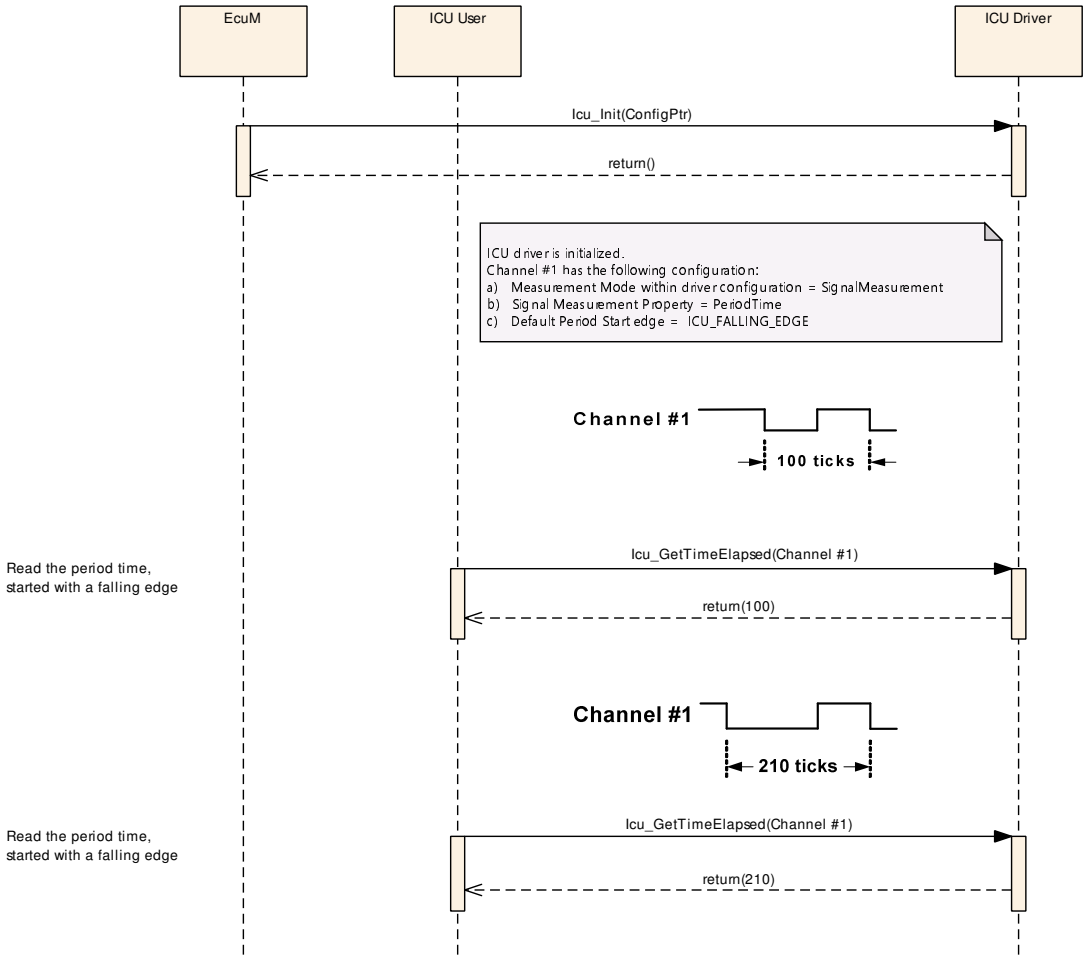


Figure 9.18: Inquire the elapsed period time of a channel

The following example shows the exemplary behaviour before, while and after capturing the "high" time of a signal.

The shown behaviour is also appropriate for the service `Icu_GetDutyCycleValues`.

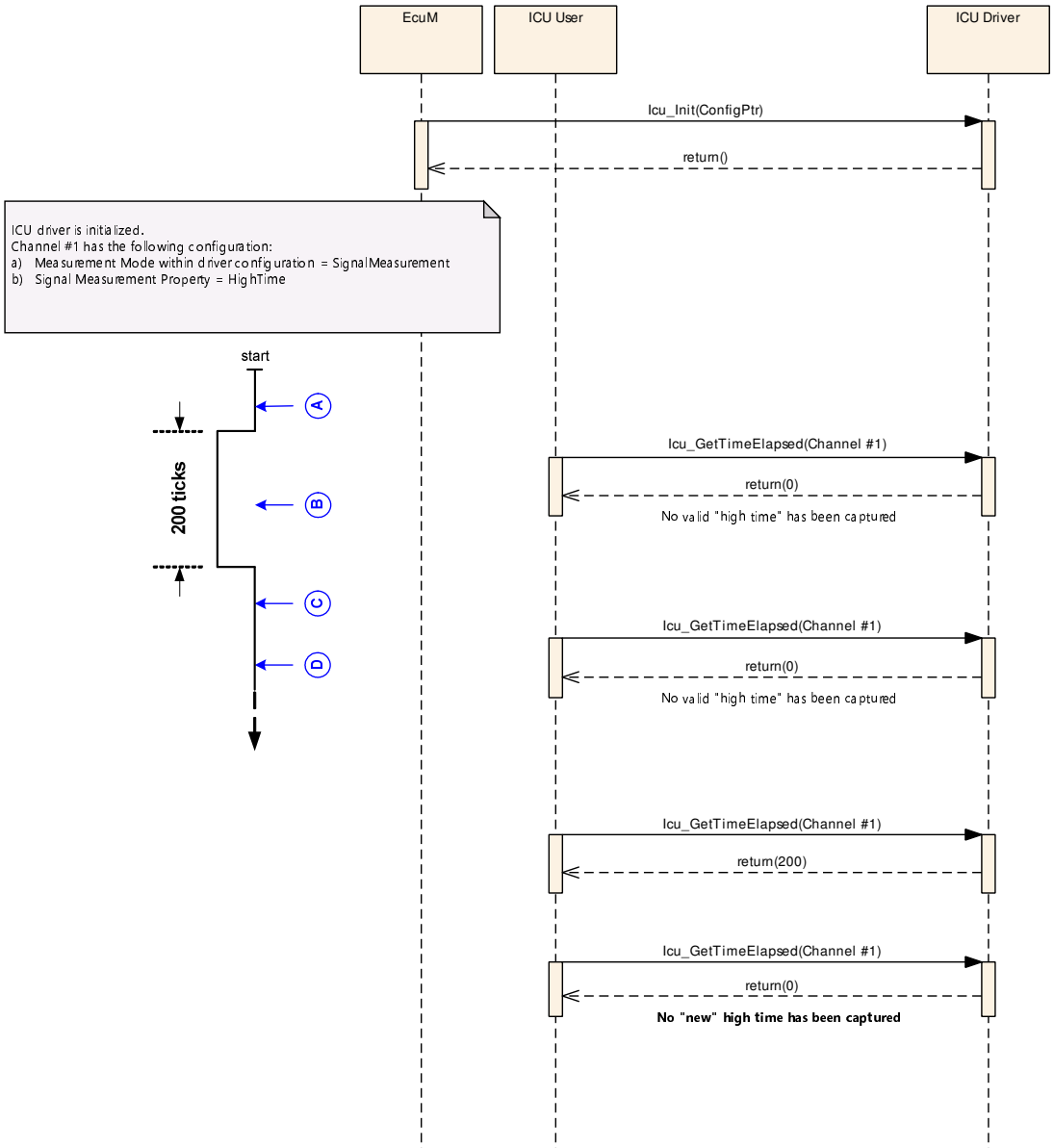


Figure 9.19: Inquire the elapsed high time of a channel

9.14 Icu_GetDutyCycleValues

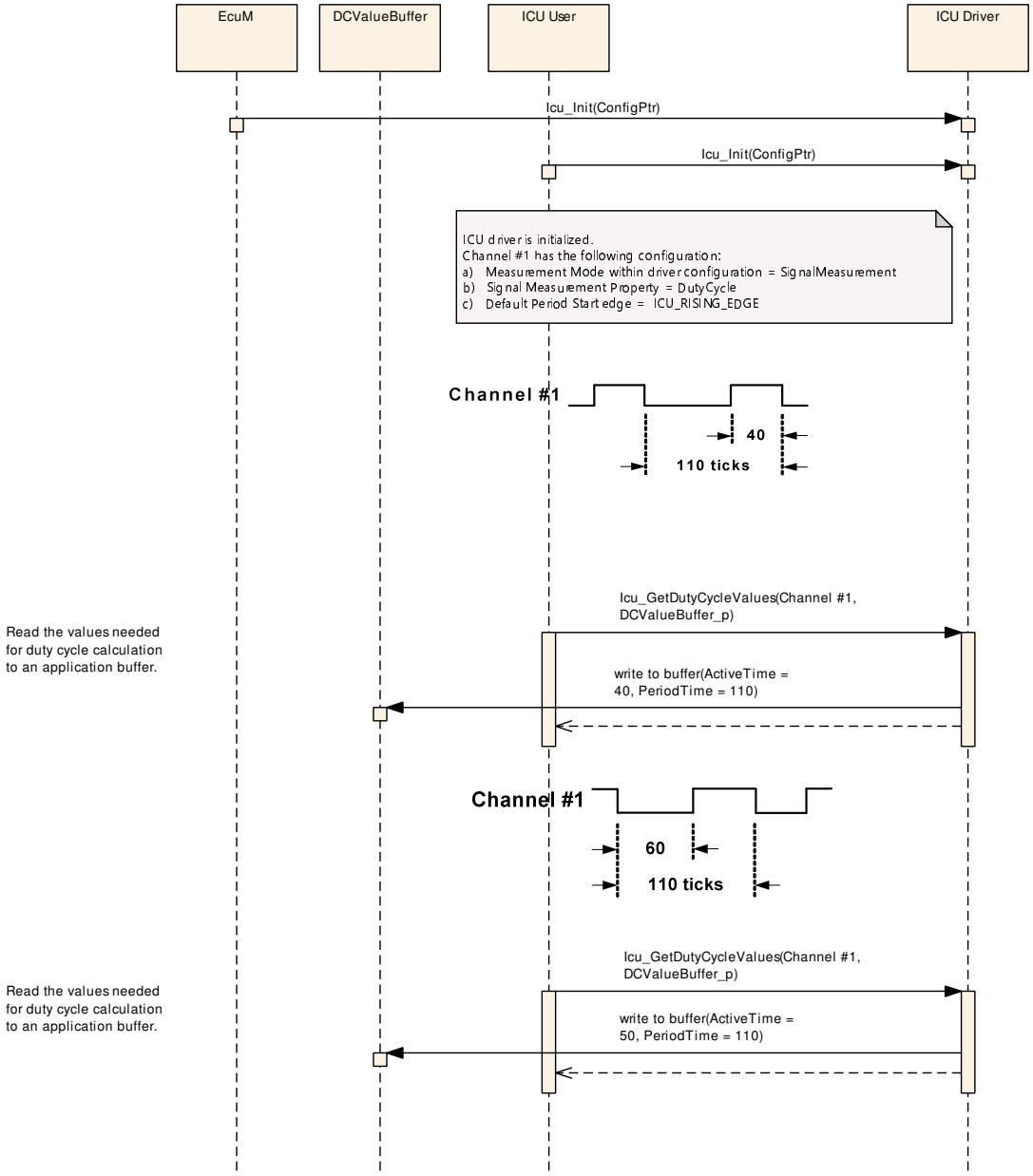


Figure 9.20: Measure the values needed for calculation of duty cycles

9.15 Icu_DisableNotificationAsync

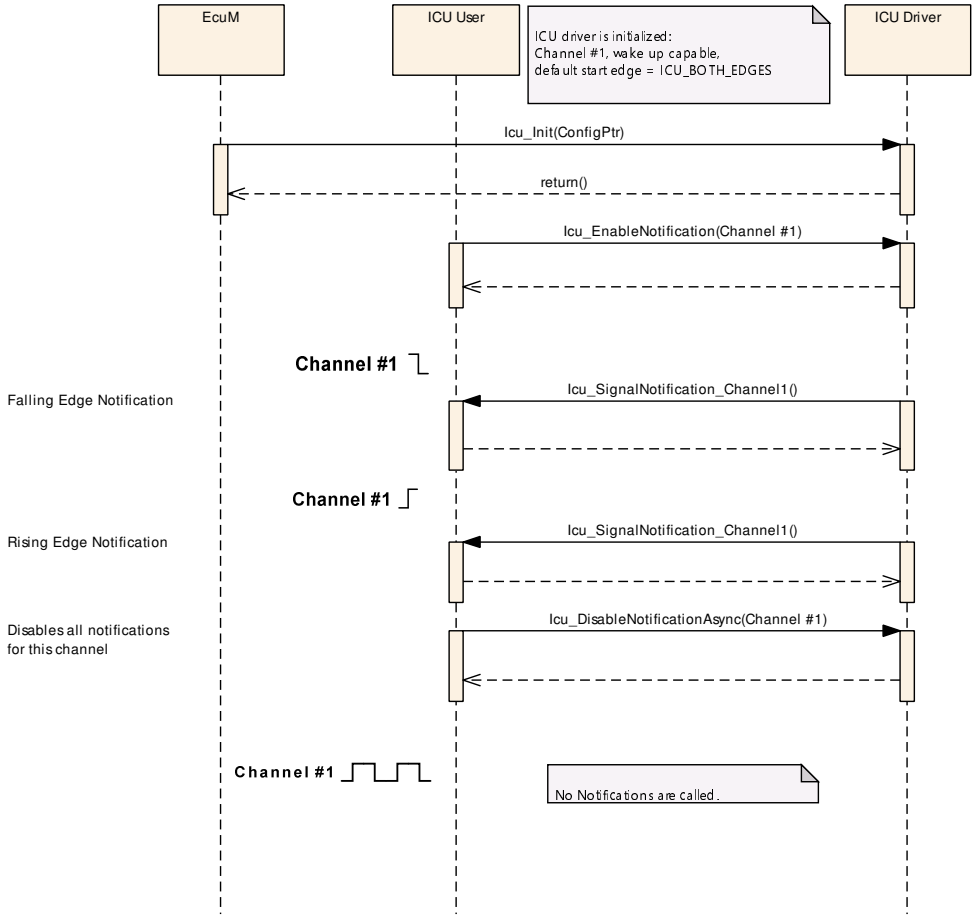


Figure 9.21: Async Disabling of the notification for a channel

9.16 Icu_SignalNotification and Icu_GetInputState

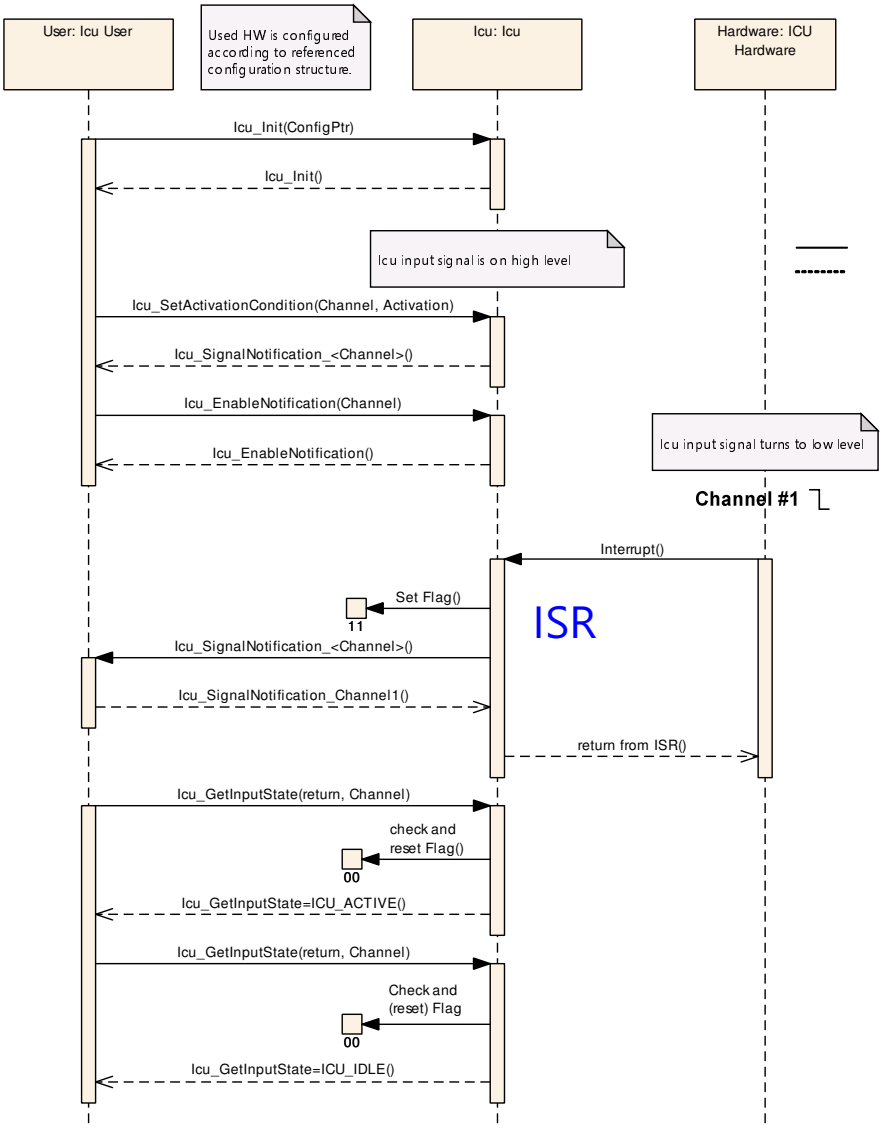


Figure 9.22: Cooperative usage of notification and polling mechanism

9.17 Icu_EnableNotificationAsync

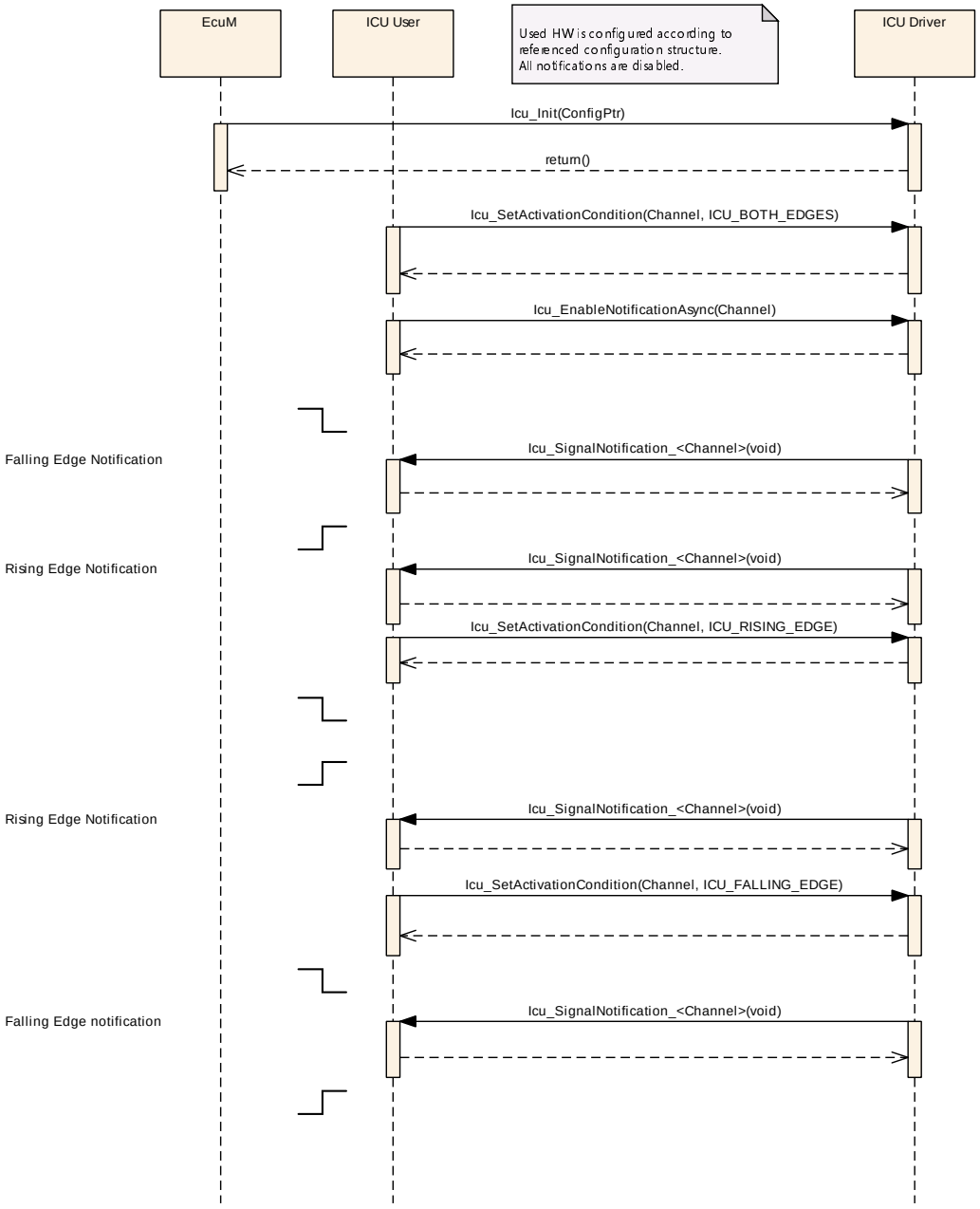


Figure 9.23: Enabling of the edge-notification for a channel via asynchronous API

10 Configuration specification

In general, this chapter defines configuration parameters and their clustering into containers. In order to support the specification Chapter 10.1 describes fundamentals. It also specifies a template (table) you shall use for the parameter specification. We intend to leave Chapter 10.1 in the specification to guarantee comprehension.

Chapter 10.2 specifies the structure (containers) and the parameters of the module ICU.

Chapter 10.3 specifies published information of the module ICU.

10.1 How to read this chapter

For details refer to [5] Chapter 10.1 *"Introduction to configuration specification"*.

10.2 Containers and configuration parameters

The following chapters summarize all configuration parameters. The detailed meanings of the parameters describe Chapter 7 and Chapter 8.

[SWS_Icu_00384] ⌈The Icu module shall reject configurations with partition mappings which are not supported by the implementation.⌋

10.2.1 Icu

[ECUC_Icu_00357] Definition of EcucModuleDef Icu ⌈

Module Name	Icu
Description	Configuration of the Icu (Input Capture Unit) module.
Post-Build Selectable Support	true
Post-Build Loadable Support	true

Included Containers		
Container Name	**Multiplicity**	**Description**
IcuConfigSet	1	This container contains the configuration parameters and sub containers of the AUTOSAR Icu module.
IcuGeneral	1	Configuration of general ICU parameters.
IcuOptionalApis	1	This container contains all configuration switches for configuring optional API services of the ICU driver.

⌋

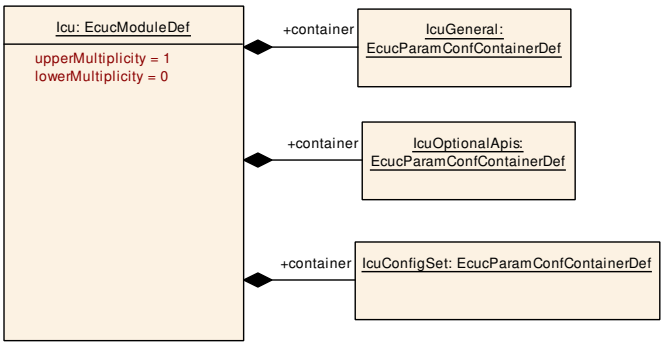


Figure 10.1: Icu

10.2.2 IcuGeneral

[ECUC_Icu_00026] Definition of EcucParamConfContainerDef IcuGeneral ⌈

Container Name	IcuGeneral
Parent Container	Icu
Upstream mapping defined	–
Description	Configuration of general ICU parameters.
Multiplicity	1
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
IcuDevErrorDetect	1	[ECUC_Icu_00232]
IcuReportWakeupSource	1	[ECUC_Icu_00233]
IcuEcucPartitionRef	0..*	[ECUC_Icu_00358]
IcuKernelEcucPartitionRef	0..1	[ECUC_Icu_00359]

No Included Containers

⌋

[ECUC_Icu_00232] Definition of EcucBooleanParamDef IcuDevErrorDetect ⌈

Parameter Name	IcuDevErrorDetect
Parent Container	IcuGeneral
Upstream mapping defined	–
Description	Switches the development error detection and notification on or off. • true: detection and notification is enabled. • false: detection and notification is disabled.
Multiplicity	1
Type	EcucBooleanParamDef
Default value	false

▽

△

Pre-Compile only	true

⌋

[ECUC_Icu_00233] Definition of EcucBooleanParamDef IcuReportWakeupSource ⌈

Parameter Name	IcuReportWakeupSource
Parent Container	IcuGeneral
Upstream mapping defined	–
Description	Switch for enabling Wakeup source reporting. true: Report Wakeup source. false: Do not report Wakeup source.
Multiplicity	1
Type	EcucBooleanParamDef
Default value	–
Pre-Compile only	true

⌋

[ECUC_Icu_00358] Definition of EcucReferenceDef IcuEcucPartitionRef ⌈

Parameter Name	IcuEcucPartitionRef	
Parent Container	IcuGeneral	
Upstream mapping defined	–	
Description	Maps the ICU driver to zero or multiple ECUC partitions to make the driver API available in the according partition.	
Multiplicity	0..*	
Type	Reference to EcucPartition	
Post-Build Multiplicity	-	Selectable
Post-Build Value	-	Selectable

⌋

[ECUC_Icu_00359] Definition of EcucReferenceDef IcuKernelEcucPartitionRef ⌈

Parameter Name	IcuKernelEcucPartitionRef	
Parent Container	IcuGeneral	
Upstream mapping defined	–	
Description	Maps the ICU kernel to zero or one ECUC partitions to assign the driver kernel to a certain core. The ECUC partition referenced is a subset of the ECUC partitions where the ICU driver is mapped to.	
Multiplicity	0..1	
Type	Reference to EcucPartition	
Post-Build Multiplicity	-	Selectable
Post-Build Value	-	Selectable

⌋

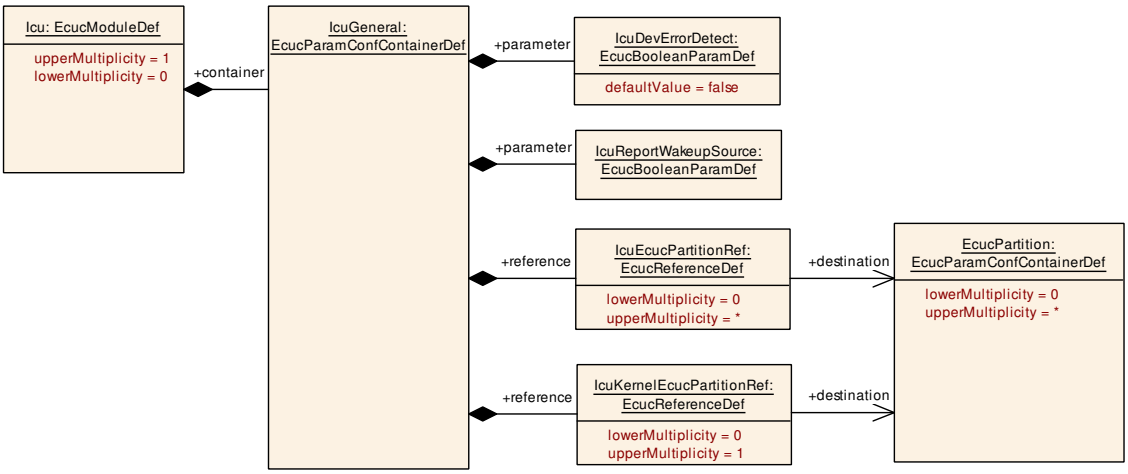


Figure 10.2: IcuGeneral

[SWS_Icu_CONSTR_00001] ⌈The ECUC partitions referenced by IcuKernelEcucPartitionRef shall be a subset of the ECUC partitions referenced by IcuEcucPartitionRef.⌋

[SWS_Icu_CONSTR_00003] ⌈If IcuEcucPartitionRef references one or more ECUC partitions, IcuKernelEcucPartitionRef shall have a multiplicity of one and reference one of these ECUC partitions as well.⌋

10.2.3 IcuOptionalApis

[ECUC_Icu_00114] Definition of EcucParamConfContainerDef IcuOptionalApis ⌈

Container Name	IcuOptionalApis
Parent Container	Icu
Upstream mapping defined	–
Description	This container contains all configuration switches for configuring optional API services of the ICU driver.
Multiplicity	1
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
IcuDeInitApi	1	[ECUC_Icu_00234]
IcuDisableWakeupApi	1	[ECUC_Icu_00235]
IcuEdgeCountApi	1	[ECUC_Icu_00124]
IcuEdgeDetectApi	1	[ECUC_Icu_00356]
IcuEnableWakeupApi	1	[ECUC_Icu_00236]
IcuGetDutyCycleValuesApi	1	[ECUC_Icu_00237]
IcuGetInputStateApi	1	[ECUC_Icu_00238]
IcuGetTimeElapsedApi	1	[ECUC_Icu_00239]

▽

△

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
IcuSetModeApi	1	[ECUC_Icu_00241]
IcuSignalMeasurementApi	1	[ECUC_Icu_00242]
IcuTimestampApi	1	[ECUC_Icu_00123]
IcuWakeupFunctionalityApi	1	[ECUC_Icu_00355]

No Included Containers

⌋

[ECUC_Icu_00234] Definition of EcucBooleanParamDef IcuDeInitApi ⌈

Parameter Name	IcuDeInitApi
Parent Container	IcuOptionalApis
Upstream mapping defined	–
Description	Adds / removes the service Icu_DeInit() from the code. true: Icu_DeInit() can be used. false: Icu_DeInit() can not be used.
Multiplicity	1
Type	EcucBooleanParamDef
Default value	–
Pre-Compile only	true

⌋

[ECUC_Icu_00235] Definition of EcucBooleanParamDef IcuDisableWakeupApi ⌈

Parameter Name	IcuDisableWakeupApi
Parent Container	IcuOptionalApis
Upstream mapping defined	–
Description	Adds / removes the service Icu_DisableWakeup() from the code. true: Icu_Disable Wakeup() can be used. false: Icu_DisableWakeup() can not be used.
Multiplicity	1
Type	EcucBooleanParamDef
Default value	–
Pre-Compile only	true

⌋

[ECUC_Icu_00124] Definition of EcucBooleanParamDef IcuEdgeCountApi ⌈

Parameter Name	IcuEdgeCountApi
Parent Container	IcuOptionalApis
Upstream mapping defined	–
Description	Adds / removes all services related to the edge counting functionality as listed below, from the code: Icu_ResetEdgeCount(), Icu_EnableEdgeCount(), Icu_DisableEdge Count(), Icu_GetEdgeNumbers(). true: The services listed above can be used. false: The services listed above can not be used.
Multiplicity	1

▽

△

Type	EcucBooleanParamDef
Default value	–
Pre-Compile only	true

⌋

[ECUC_Icu_00356] Definition of EcucBooleanParamDef IcuEdgeDetectApi ⌈

Parameter Name	IcuEdgeDetectApi
Parent Container	IcuOptionalApis
Upstream mapping defined	–
Description	Adds / removes the services related to the edge detection functionality, from the code: Icu_EnableEdgeDetection() and Icu_DisableEdgeDetection(). true: These services can be used. false: These services can not be used.
Multiplicity	1
Type	EcucBooleanParamDef
Default value	–
Pre-Compile only	true

⌋

[ECUC_Icu_00236] Definition of EcucBooleanParamDef IcuEnableWakeupApi ⌈

Parameter Name	IcuEnableWakeupApi
Parent Container	IcuOptionalApis
Upstream mapping defined	–
Description	Adds / removes the service Icu_EnableWakeup() from the code. true: Icu_Enable Wakeup() can be used. false: Icu_EnableWakeup() can not be used.
Multiplicity	1
Type	EcucBooleanParamDef
Default value	–
Pre-Compile only	true

⌋

[ECUC_Icu_00237] Definition of EcucBooleanParamDef IcuGetDutyCycleValues Api ⌈

Parameter Name	IcuGetDutyCycleValuesApi
Parent Container	IcuOptionalApis
Upstream mapping defined	–
Description	Adds / removes the service Icu_GetDutyCycleValues() from the code. true: Icu_Get DutyCycleValues() can be used. false: Icu_GetDutyCycleValues() can not be used. Dependency: If IcuSignalMeasurementApi==false this switch shall also be set to false.
Multiplicity	1
Type	EcucBooleanParamDef
Default value	–
Pre-Compile only	true

⌋

[ECUC_Icu_00238] Definition of EcucBooleanParamDef IcuGetInputStateApi ⌈

Parameter Name	IcuGetInputStateApi
Parent Container	IcuOptionalApis
Upstream mapping defined	–
Description	Adds / removes the service Icu_GetInputState() from the code. true: Icu_GetInput State() can be used. false: Icu_GetInputState() can not be used.
Multiplicity	1
Type	EcucBooleanParamDef
Default value	–
Pre-Compile only	true

⌋

[ECUC_Icu_00239] Definition of EcucBooleanParamDef IcuGetTimeElapsedApi ⌈

Parameter Name	IcuGetTimeElapsedApi
Parent Container	IcuOptionalApis
Upstream mapping defined	–
Description	Adds / removes the service Icu_GetTimeElapsed() from the code. true: Icu_GetTime Elapsed() can be used. false: Icu_GetTimeElapsed() can not be used. Dependency: If IcuSignalMeasurementApi==false this switch shall also be set to false.
Multiplicity	1
Type	EcucBooleanParamDef
Default value	–
Pre-Compile only	true

⌋

[ECUC_Icu_00241] Definition of EcucBooleanParamDef IcuSetModeApi ⌈

Parameter Name	IcuSetModeApi
Parent Container	IcuOptionalApis
Upstream mapping defined	–
Description	Adds / removes the service Icu_SetMode() from the code. true: Icu_SetMode() can be used. false: Icu_SetMode() can not be used.
Multiplicity	1
Type	EcucBooleanParamDef
Default value	–
Pre-Compile only	true

⌋

[ECUC_Icu_00242] Definition of EcucBooleanParamDef IcuSignalMeasurement Api ⌈

Parameter Name	IcuSignalMeasurementApi
Parent Container	IcuOptionalApis
Upstream mapping defined	–

▽

△

Description	Adds / removes the services Icu_StartSignalMeasurement() and Icu_StopSignal Measurement() from the code. true: Icu_StartSignalMeasurement() and Icu_Stop SignalMeasurement() can be used. false: Icu_StartSignalMeasurement() and Icu_Stop SignalMeasurement() can not be used.
Multiplicity	1
Type	EcucBooleanParamDef
Default value	–
Pre-Compile only	true

⌋

[ECUC_Icu_00123] Definition of EcucBooleanParamDef IcuTimestampApi ⌈

Parameter Name	IcuTimestampApi
Parent Container	IcuOptionalApis
Upstream mapping defined	–
Description	Adds / removes all services related to the timestamping functionality as listed below from the code: Icu_StartTimestamp(), Icu_StopTimestamp(), Icu_GetTimestamp Index(). true: The services listed above can be used. false: The services listed above can not be used.
Multiplicity	1
Type	EcucBooleanParamDef
Default value	–
Pre-Compile only	true

⌋

[ECUC_Icu_00355] Definition of EcucBooleanParamDef IcuWakeupFunctionality Api ⌈

Parameter Name	IcuWakeupFunctionalityApi
Parent Container	IcuOptionalApis
Upstream mapping defined	–
Description	Adds / removes the service Icu_CheckWakeup() from the code. true: Icu_Check Wakeup() can be used. false: Icu_CheckWakeup() cannot be used.
Multiplicity	1
Type	EcucBooleanParamDef
Default value	–
Pre-Compile only	true

⌋

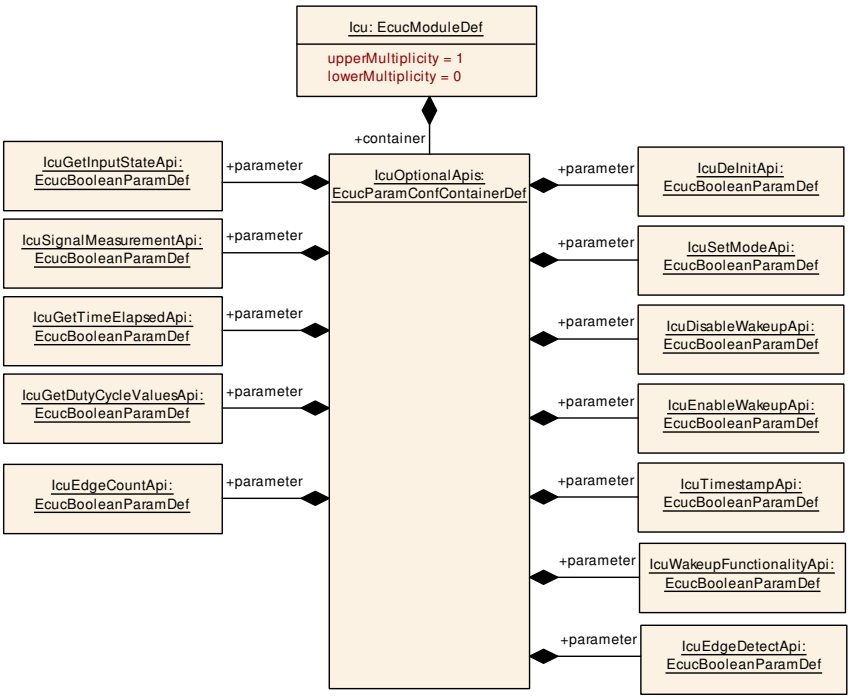


Figure 10.3: IcuOptionalApis

10.2.4 IcuChannel

[ECUC_Icu_00027] Definition of EcucParamConfContainerDef IcuChannel ⌈

Container Name	IcuChannel
Parent Container	IcuConfigSet
Upstream mapping defined	–
Description	Configuration of an individual ICU channel.
Multiplicity	1..*
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
IcuChannelId	1	[ECUC_Icu_00354]
IcuDefaultStartEdge	1	[ECUC_Icu_00222]
IcuMeasurementMode	1	[ECUC_Icu_00223]
IcuWakeupCapability	1	[ECUC_Icu_00224]
IcuChannelEcucPartitionRef	0..*	[ECUC_Icu_00360]

Included Containers		
Container Name	**Multiplicity**	**Description**
IcuSignalEdgeDetection	0..1	This container contains the configuration (parameters) in case the measurement mode is "IcuSignalEdgeDetection"
IcuSignalMeasurement	0..1	This container contains the configuration (parameters) in case the measurement mode is "IcuSignalMeasurement"
IcuTimestampMeasurement	0..1	This container contains the configuration (parameters) in case the measurement mode is "IcuTimestamp"
IcuWakeup	0..1	This container contains the configuration (parameters) needed to configure a wakeup capable channel

⌋

[ECUC_Icu_00354] Definition of EcucIntegerParamDef IcuChannelId ⌈

Parameter Name	IcuChannelId	
Parent Container	IcuChannel	
Upstream mapping defined	–	
Description	Channel Id of the ICU channel. This value will be assigned to the symbolic name derived of the IcuChannel container short name.	
Multiplicity	1	
Type	EcucIntegerParamDef (Symbolic Name generated for this parameter)	
Range	0 .. 65535	
Default value	–	
Pre-Compile only	true	

⌋

[ECUC_Icu_00222] Definition of EcucEnumerationParamDef IcuDefaultStartEdge ⌈

Parameter Name	IcuDefaultStartEdge	
Parent Container	IcuChannel	
Upstream mapping defined	–	
Description	Configures the default-activation-edge which shall be used for this channel if there was no activation-edge configured by the call of service Icu_SetActivationCondition(). In case the Measurement Mode is "IcuSignalMeasurement" and the properties "Duty Cycle" or "Period" are set, the edge configured here is used as Default Period Start Edge. Implementation Type: Icu_ActivationType	
Multiplicity	1	
Type	EcucEnumerationParamDef	
Range	ICU_BOTH_EDGES	As default, both edges are used.
	ICU_FALLING_EDGE	As default, falling edge is the used.
	ICU_RISING_EDGE	As default, rising edge is the used.
Post-Build Value	Loadable	Selectable

⌋

[ECUC_Icu_00223] Definition of EcucEnumerationParamDef IcuMeasurement Mode ⌈

Parameter Name	IcuMeasurementMode	
Parent Container	IcuChannel	
Upstream mapping defined	–	
Description	Configures the measurement mode of this channel. Implementation Type: Icu_MeasurementModeType Dependency: The possible measurement modes are depending on the pre-processor switches, which enable/disable optional API services.	
Multiplicity	1	
Type	EcucEnumerationParamDef	
Range	ICU_MODE_EDGE_COUNTER	The channnel is used to count the edges which are configured by the call of the service Icu_Set ActivationCondition(). The following API services support this mode: • Icu_EnableEdgeCount() • Icu_DisableEdgeCount() • Icu_GetEdgeNumbers() • Icu_ResetEdgeCount() This mode can only be configured if IcuEdge VountApi is switched on.
	ICU_MODE_SIGNAL_EDGE_ DETECT	The channel is used for detecting the edges which are configured by the call of the service Icu_SetActivationCondition(). The following API services support this mode: • Icu_EnableNotification() • Icu_DisableNotification() • Icu_GetInputState()
	ICU_MODE_SIGNAL_ MEASUREMENT	The channel is used to measure different times between various configurable edges. The configuration of the period-start edges are done by configuration and cannot be changed during runtime. The following API services support this mode: • Icu_GetTimeElapsed() • Icu_GetDutyCycleValues() • Icu_GetInputState() This mode can only be configured if at least one of the following switches are set to "true": • IcuGetDutyCycleValuesApi • IcuGetTimeElapsedApi
	ICU_MODE_TIMESTAMP	The channel is used to capture timer values on the edges which are configured by the call of the service Icu_SetActivationCondition(). The following API services support this mode: • Icu_StartTimestamp() • Icu_StopTimestamp() • Icu_GetTimestampIndex() This mode can only be configured if IcuTime StampApi is switched on.
Post-Build Value	Loadable	Selectable

⌋

[ECUC_Icu_00224] Definition of EcucBooleanParamDef IcuWakeupCapability ⌈

Parameter Name	IcuWakeupCapability	
Parent Container	IcuChannel	
Upstream mapping defined	–	
Description	Information about the wakeup-capability of this channel. true: Channel is wakeup capable. false: Channel is not wakeup capable.	
Multiplicity	1	
Type	EcucBooleanParamDef	
Default value	–	
Post-Build Value	Loadable	Selectable

⌋

[ECUC_Icu_00360] Definition of EcucReferenceDef IcuChannelEcucPartitionRef ⌈

Parameter Name	IcuChannelEcucPartitionRef	
Parent Container	IcuChannel	
Upstream mapping defined	–	
Description	Maps an ICU channel to zero or multiple ECUC partitions to limit the access to this channel. The ECUC partitions referenced are a subset of the ECUC partitions where the ICU driver is mapped to.	
Multiplicity	0..*	
Type	Reference to EcucPartition	
Post-Build Multiplicity	-	Selectable
Post-Build Value	-	Selectable

⌋

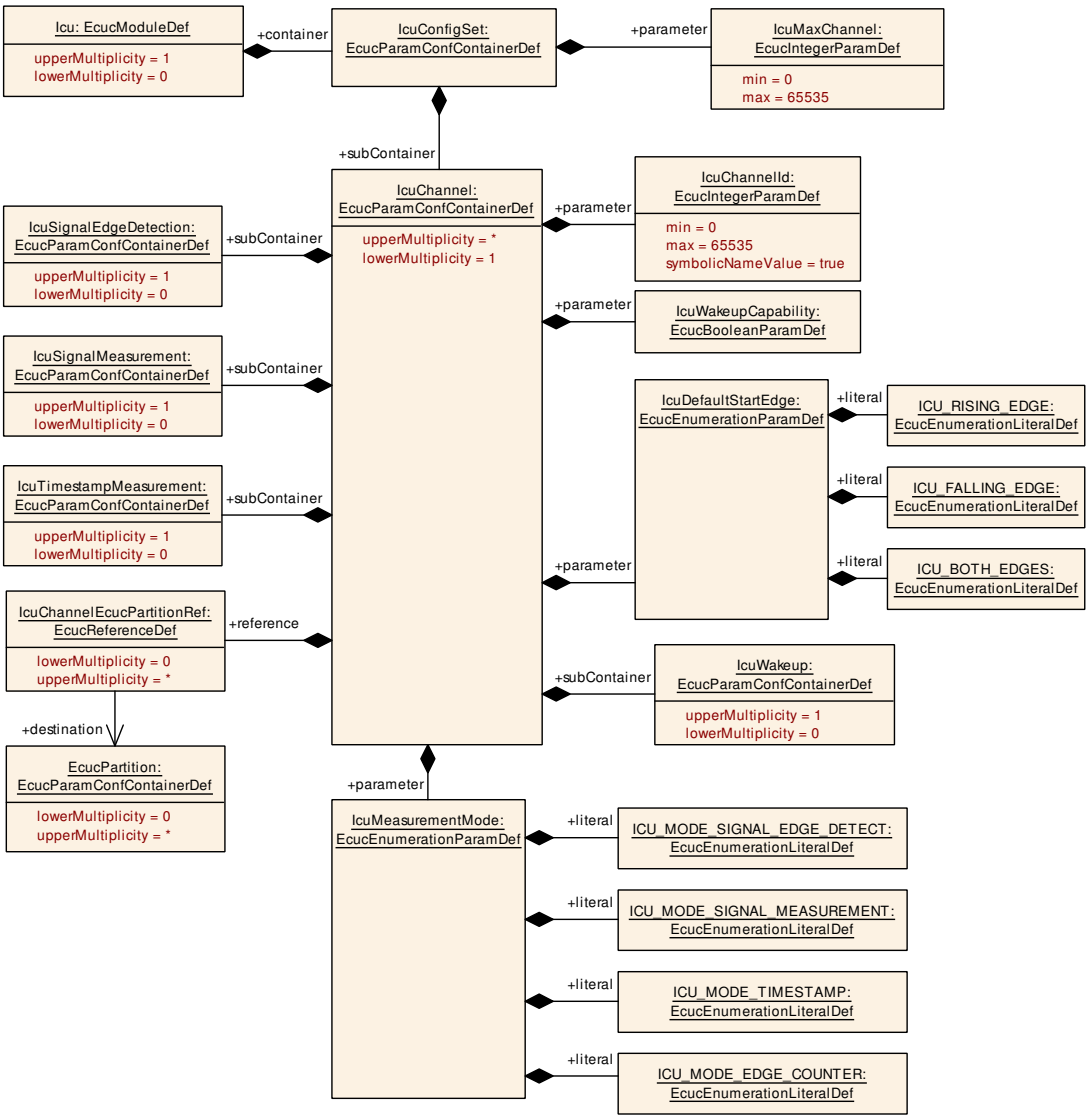


Figure 10.4: IcuChannel

[SWS_Icu_CONSTR_00002] ⌈The ECUC partitions referenced by IcuChannelEcucPartitionRef shall be a subset of the ECUC partitions referenced by IcuEcucPartitionRef.⌋

[SWS_Icu_CONSTR_00004] ⌈If IcuEcucPartitionRef references one or more ECUC partitions, IcuChannelEcucPartitionRef shall have a multiplicity of greater than one and reference one or several of these ECUC partitions as well.⌋

10.2.5 IcuSignalEdgeDetection

[ECUC_Icu_00219] Definition of EcucParamConfContainerDef IcuConfigSet ⌈

Container Name	IcuConfigSet
Parent Container	Icu
Upstream mapping defined	–
Description	This container contains the configuration parameters and sub containers of the AUTOSAR Icu module.
Multiplicity	1
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	Multiplicity	ECUC ID
IcuMaxChannel	1	[ECUC_Icu_00220]

Included Containers		
Container Name	Multiplicity	Description
IcuChannel	1..*	Configuration of an individual ICU channel.

⌋

[ECUC_Icu_00220] Definition of EcucIntegerParamDef IcuMaxChannel ⌈

Parameter Name	IcuMaxChannel	
Parent Container	IcuConfigSet	
Upstream mapping defined	–	
Description	This parameter contains the number of Channels configured. It will be gathered by tools during the configuration stage. calculationFormula = Number of configured Icu Channels Implementation Type: Icu_ChannelType	
Multiplicity	1	
Type	EcucIntegerParamDef	
Range	0 .. 65535	
Default value	–	
Post-Build Value	Loadable	Selectable

⌋

[ECUC_Icu_00126] Definition of EcucParamConfContainerDef IcuWakeup ⌈

Container Name	IcuWakeup
Parent Container	IcuChannel
Upstream mapping defined	–
Description	This container contains the configuration (parameters) needed to configure a wakeup capable channel
Multiplicity	0..1
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	Multiplicity	ECUC ID
IcuChannelWakeupInfo	0..1	[ECUC_Icu_00231]

No Included Containers

⌋

[ECUC_Icu_00231] Definition of EcucReferenceDef IcuChannelWakeupInfo ⌈

Parameter Name	IcuChannelWakeupInfo	
Parent Container	IcuWakeup	
Upstream mapping defined	–	
Description	If the wakeup-capability is true the wakeup source referenced is transmitted to the ECU State Manager (EcuM) . Implementation Type: reference to EcuM_WakeupSourceType Dependency: IcuWakeupCapability and IcuReportWakeupSource	
Multiplicity	0..1	
Type	Symbolic name reference to EcuMWakeupSource	
Post-Build Multiplicity	Loadable	Selectable
Post-Build Value	Loadable	Selectable

⌋

[ECUC_Icu_00228] Definition of EcucParamConfContainerDef IcuTimestampMeasurement ⌈

Container Name	IcuTimestampMeasurement
Parent Container	IcuChannel
Upstream mapping defined	–
Description	This container contains the configuration (parameters) in case the measurement mode is "IcuTimestamp"
Multiplicity	0..1
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
IcuTimestampMeasurementProperty	1	[ECUC_Icu_00229]
IcuTimestampNotification	0..1	[ECUC_Icu_00230]

No Included Containers

⌋

[ECUC_Icu_00229] Definition of EcucEnumerationParamDef IcuTimestampMeasurementProperty ⌈

Parameter Name	IcuTimestampMeasurementProperty
Parent Container	IcuTimestampMeasurement
Upstream mapping defined	–

▽

△

Description	Configures the handling of the buffer in case the mode is "Timestamp" Implementation Type: Icu_TimestampBufferType Dependency: IcuMeasurementMode	
Multiplicity	1	
Type	EcucEnumerationParamDef	
Range	ICU_CIRCULAR_BUFFER	After reaching the end of the buffer, the driver restarts at the beginning of the buffer
	ICU_LINEAR_BUFFER	The buffer will just be filled once
Post-Build Value	Loadable	Selectable

⌋

[ECUC_Icu_00230] Definition of EcucFunctionNameDef IcuTimestampNotification ⌈

Parameter Name	IcuTimestampNotification	
Parent Container	IcuTimestampMeasurement	
Upstream mapping defined	–	
Description	Notification function if the number of requested timestamps (Notification interval > 0) are acquired. Dependency: IcuTimestampApi	
Multiplicity	0..1	
Type	EcucFunctionNameDef	
Default value	–	
Regular Expression	–	
Post-Build Multiplicity	Loadable	Selectable
Post-Build Value	Loadable	Selectable

⌋

[ECUC_Icu_00226] Definition of EcucParamConfContainerDef IcuSignalMeasurement ⌈

Container Name	IcuSignalMeasurement
Parent Container	IcuChannel
Upstream mapping defined	–
Description	This container contains the configuration (parameters) in case the measurement mode is "IcuSignalMeasurement"
Multiplicity	0..1
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
IcuSignalMeasurementProperty	1	[ECUC_Icu_00227]

No Included Containers

⌋

[ECUC_Icu_00227] Definition of EcucEnumerationParamDef IcuSignalMeasurementProperty ⌈

Parameter Name	IcuSignalMeasurementProperty	
Parent Container	IcuSignalMeasurement	
Upstream mapping defined	–	
Description	Configures the property that could be measured in case the mode is "IcuSignal Measurement". This property can not be changed during runtime. Implementation Type: Icu_SignalMeasurementPropertyType Dependency: IcuMeasurementMode, IcuGetDutyCycleValuesApi, IcuGetTimeElapsed Api	
Multiplicity	1	
Type	EcucEnumerationParamDef	
Range	ICU_DUTY_CYCLE	The channel is configured to read values which are needed for calculating the duty cycle (coherent Active and Period Time).
	ICU_HIGH_TIME	The channel is configured for reading the elapsed Signal High Time
	ICU_LOW_TIME	The channel is configured for reading the elapsed Signal Low Time
	ICU_PERIOD_TIME	The channel is configured for reading the elapsed Signal Period Time
Post-Build Value	Loadable	Selectable

⌋

[ECUC_Icu_00021] Definition of EcucParamConfContainerDef IcuSignalEdgeDetection ⌈

Container Name	IcuSignalEdgeDetection
Parent Container	IcuChannel
Upstream mapping defined	–
Description	This container contains the configuration (parameters) in case the measurement mode is "IcuSignalEdgeDetection"
Multiplicity	0..1
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
IcuSignalNotification	0..1	[ECUC_Icu_00225]

No Included Containers

⌋

[ECUC_Icu_00225] Definition of EcucFunctionNameDef IcuSignalNotification ⌈

Parameter Name	IcuSignalNotification	
Parent Container	IcuSignalEdgeDetection	
Upstream mapping defined	–	
Description	Notification function for signal notification. Dependency: IcuMeasurementMode	
Multiplicity	0..1	
Type	EcucFunctionNameDef	
Default value	–	
Regular Expression	–	
Post-Build Multiplicity	Loadable	Selectable
Post-Build Value	Loadable	Selectable

⌋

10.3 Published Information

[SWS_Icu_00131]

Upstream requirements: SRS_BSW_00384

⌈The ICU driver shall describe which other modules (in which versions) are required. This description shall be done by the implementer.⌋

A Not applicable requirements

[SWS_Icu_NA_00999]

Upstream requirements: SRS_BSW_00383, SRS_BSW_00395, SRS_BSW_00397, SRS_BSW_00398, SRS_BSW_00399, SRS_BSW_00400, SRS_BSW_00409, SRS_BSW_00167, SRS_BSW_00168, SRS_BSW_00170, SRS_BSW_00416, SRS_BSW_00417, SRS_BSW_00422, SRS_BSW_00423, SRS_BSW_00424, SRS_BSW_00425, SRS_BSW_00426, SRS_BSW_00427, SRS_BSW_00428, SRS_BSW_00429, SRS_BSW_00432, SRS_BSW_00433, SRS_BSW_00437, SRS_SPAL_12068, SRS_SPAL_12077, SRS_SPAL_12092, SRS_SPAL_12265, SRS_SPAL_12463, SRS_BSW_00450, SRS_BSW_00469, SRS_BSW_00471, SRS_BSW_00472, SRS_BSW_00478

⌈These requirements are not applicable to this specification.⌋

B Change history of AUTOSAR traceable items

Please note that the lists in this chapter also include traceable items that have been removed from the specification in a later version. These items do not appear as hyperlinks in the document.

B.1 Traceable item history of this document according to AUTOSAR Release R26-11

B.1.1 Added Specification Items in R26-11

none

B.1.2 Changed Specification Items in R26-11

Number	Heading
[ECUC_Icu_00021]	Definition of EcucParamConfContainerDef IcuSignalEdgeDetection
[ECUC_Icu_00026]	Definition of EcucParamConfContainerDef IcuGeneral
[ECUC_Icu_00027]	Definition of EcucParamConfContainerDef IcuChannel
[ECUC_Icu_00114]	Definition of EcucParamConfContainerDef IcuOptionalApis
[ECUC_Icu_00126]	Definition of EcucParamConfContainerDef IcuWakeup
[ECUC_Icu_00219]	Definition of EcucParamConfContainerDef IcuConfigSet
[ECUC_Icu_00226]	Definition of EcucParamConfContainerDef IcuSignalMeasurement
[ECUC_Icu_00228]	Definition of EcucParamConfContainerDef IcuTimestampMeasurement
[SWS_Icu_00050]	
[SWS_Icu_00276]	Definition of imported datatypes of module Icu
[SWS_Icu_00382]	Definition of development errors in module Icu

Table B.1: Changed Specification Items in R26-11

B.1.3 Deleted Specification Items in R26-11

Number	Heading
[ECUC_Icu_00240]	Definition of EcucBooleanParamDef IcuGetVersionInfoApi
[SWS_Icu_00212]	Definition of API function Icu_GetVersionInfo
[SWS_Icu_00356]	

Table B.2: Deleted Specification Items in R26-11

B.1.4 Added Constraints in R26-11

none

B.1.5 Changed Constraints in R26-11

none

B.1.6 Deleted Constraints in R26-11

none

B.2 Traceable item history of this document according to AUTOSAR Release R25-11

B.2.1 Added Specification Items in R25-11

none

B.2.2 Changed Specification Items in R25-11

none

B.2.3 Deleted Specification Items in R25-11

none

B.2.4 Added Constraints in R25-11

none

B.2.5 Changed Constraints in R25-11

none

B.2.6 Deleted Constraints in R25-11

none