

<b>Document Title</b>	Specification of ICU Driver
<b>Document Owner</b>	AUTOSAR
<b>Document Responsibility</b>	AUTOSAR
<b>Document Identification No</b>	023
<b>Document Status</b>	Final
<b>Part of AUTOSAR Standard</b>	Classic Platform
<b>Part of Standard Release</b>	4.3.1

<b>Document Change History</b>			
<b>Date</b>	<b>Release</b>	<b>Changed by</b>	<b>Change Description</b>
2017-12-08	4.3.1	AUTOSAR Release Management	<ul style="list-style-type: none"> <li>Removed SWS_Icu_00116 and SWS_Icu_00190</li> <li>Added SRS_BSW_00450 to the list of non applicable requirements</li> <li>Renamed "default error" to "development error"</li> <li>SWS_Icu_00201: Icu_StartTimestamp: Parameter (IN): Icu_ValueType* BufferPtr changed to Parameters (out) type</li> <li>Changed ICU_E_NOT_STARTED from development error to runtime error</li> <li>Editorial Changes</li> </ul>
2016-11-30	4.3.0	AUTOSAR Release Management	<ul style="list-style-type: none"> <li>Removed chapter "10.2.1 Variants"</li> <li>Changed upper multiplicity of the ICU_EcuModuleDef to 1 in figure of section 10.2.2</li> <li>Removed config parameter IcuIndex(ECUC_Icu_00221) from IcuGeneral section 10.2.3 and in figure of section 10.2.3</li> <li>Requirement ID SWS_Icu_00383 given to additional test "EcuM_WakeupSourceType shall be imported from EcuM_Types.h"</li> <li>Removed requirement SWS_Icu_00346</li> <li>Editorial changes</li> </ul>

Document Change History			
Date	Release	Changed by	Change Description
2015-07-31	4.2.2	AUTOSAR Release Management	<ul style="list-style-type: none"> <li>Editorial changes</li> <li>DET renamed from "Development Error Tracer" to "Default Error Tracer".</li> <li>All references to obsolete SWS_Icu_00048 removed from the document</li> </ul>
2014-10-31	4.2.1	AUTOSAR Release Management	<ul style="list-style-type: none"> <li>IcuChannelId: postBuildVariantValue set to false</li> <li>SWS IDs with respect to NULL_PTR check for Icu_Init() removed</li> <li>ICU_E_PARAM_POINTER and ICU_E_INIT_FAILED added to Error classification</li> <li>ICU_E_PARAM_CONFIG and ICU_E_PARAM_BUFFER_PTR removed from Error classification</li> </ul>
2013-10-31	4.1.2	AUTOSAR Release Management	<ul style="list-style-type: none"> <li>ICU00354 - Check for a valid notification interval rephrased</li> <li>ICU078 - Removed the sentence "This is done by the hardware." from the note.</li> <li>ICU295 - Removed ICU_ACTIVE_TIME from the range of enumeration Icu_SignalMeasurementPropertyType</li> <li>Editorial changes</li> <li>Removed chapter(s) on change documentation</li> </ul>
2013-03-15	4.1.1	AUTOSAR Administration	<ul style="list-style-type: none"> <li>Modified the scope of the parameters from ECU/Module to local</li> <li>Reworked according to the new SWS_BSWGeneral</li> <li>Changed MemMap.h to Icu_MemMap.h</li> </ul>
2011-12-22	4.0.3	AUTOSAR Administration	<ul style="list-style-type: none"> <li>Corrected Type errors</li> <li>Updated description of Icu_IndexType</li> </ul>

<b>Document Change History</b>			
<b>Date</b>	<b>Release</b>	<b>Changed by</b>	<b>Change Description</b>
2010-09-30	3.1.5	AUTOSAR Administration	<ul style="list-style-type: none"> <li>• Services 'Icu_DisableEdgeDetection' and 'Icu_EnableEdgeDetection' were added.</li> <li>• Configuration parameters 'IcuEdgeDetectApi' and 'IcuWakeupFunctionalityApi' has been added.</li> <li>• Definition of 'duty cycle' has been corrected.</li> <li>• Corrected values of the parameter 'Icu_SignalMeasurementPropertyType'</li> </ul>
2008-08-13	3.1.1	AUTOSAR Administration	<ul style="list-style-type: none"> <li>• Legal disclaimer revised</li> </ul>
2008-02-01	3.0.2	AUTOSAR Administration	<ul style="list-style-type: none"> <li>• The code file structure of the module was completely reworked.</li> <li>• The following requirements were added: SWS_Icu_00088, SWS_Icu_00220, SWS_Icu_00221, SWS_Icu_00228 and SWS_Icu_00229.</li> <li>• The flow charts related to the ECU Wake-Up moved to the</li> <li>• SWS document of the ECU State Manager.</li> <li>• Document meta information extended</li> <li>• Small layout adaptations made</li> </ul>

<b>Document Change History</b>			
<b>Date</b>	<b>Release</b>	<b>Changed by</b>	<b>Change Description</b>
2007-12-21	3.0.1	AUTOSAR Administration	<ul style="list-style-type: none"> <li>• Default start edge is now used for edge configuration</li> <li>• Enable and Disable Notification can now be used for Timestamp functionality.</li> <li>• Edge detection functionality is now pre compile time configurable On/Off</li> <li>• Legal disclaimer revised</li> <li>• Release Notes added</li> <li>• “Advice for users” revised</li> <li>• “Revision Information” added</li> </ul>
2006-05-16	2.0	AUTOSAR Administration	<ul style="list-style-type: none"> <li>• Added the following services               <ul style="list-style-type: none"> <li>- Icu_SetActivationCondition</li> <li>- Icu_StartTimeStamp</li> <li>- Icu_StopTimeStamp</li> <li>- Icu_GetTimestampIndex</li> <li>- Icu_ResetEdgeCount</li> <li>- Icu_EnableEdgeCount</li> <li>- Icu_DisableEdgeCount</li> <li>- Icu_GetEdgeNumbers</li> <li>- Icu_GetTimeElapsed</li> <li>- Icu_GetDutyCycleValues</li> <li>- Icu_GetVersionInfo</li> </ul> </li> </ul>
2005-05-31	1.0	AUTOSAR Administration	<ul style="list-style-type: none"> <li>• Initial Release</li> </ul>

## 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 .....	9
2	Acronyms and abbreviations .....	10
3	Related documentation.....	11
3.1	Input documents.....	11
3.2	Related standards and norms .....	11
3.3	Related specification .....	11
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
5.6	File structure .....	15
5.6.1	Header file structure.....	15
6	Requirements traceability .....	18
7	Functional specification .....	28
7.1	General behavior.....	28
7.1.1	Background & Rationale .....	28
7.1.2	Requirements.....	28
7.1.3	Time Unit Ticks .....	29
7.1.3.1	Background & Rationale.....	29
7.1.3.2	Requirements .....	29
7.2	Error classification .....	29
7.2.1	Development Errors .....	29
7.2.2	Runtime Errors.....	30
7.2.3	Transient Faults .....	30
7.2.4	Production Errors .....	30
7.2.5	Extended Production Errors.....	30
7.3	Error detection.....	31
7.4	Debugging Concept.....	31
7.4.1	Background & Rationale .....	31
8	API specification.....	32
8.1	Imported types.....	32
8.2	Type definitions .....	32
8.2.1	Icu_ModeType .....	32
8.2.2	Icu_ChannelType.....	32
8.2.3	Icu_InputStateType.....	33
8.2.4	Icu_ConfigType.....	33

8.2.5	Icu_ActivationType.....	34
8.2.6	Icu_ValueType.....	34
8.2.7	Icu_DutyCycleType.....	35
8.2.8	Icu_IndexType.....	35
8.2.9	Icu_EdgeNumberType.....	35
8.2.10	Icu_MeasurementModeType.....	35
8.2.11	Icu_SignalMeasurementPropertyType.....	36
8.2.12	Icu_TimestampBufferType.....	36
8.3	Function definitions.....	36
8.3.1	Icu_Init.....	36
8.3.2	Icu_DeInit.....	38
8.3.3	Icu_SetMode.....	39
8.3.4	Icu_DisableWakeup.....	40
8.3.5	Icu_EnableWakeup.....	41
8.3.6	Icu_CheckWakeup.....	42
8.3.7	Icu_SetActivationCondition.....	43
8.3.8	Icu_DisableNotification.....	44
8.3.9	Icu_EnableNotification.....	45
8.3.10	Icu_GetInputState.....	46
8.3.11	Icu_StartTimestamp.....	47
8.3.12	Icu_StopTimestamp.....	49
8.3.13	Icu_GetTimestampIndex.....	50
8.3.14	Icu_ResetEdgeCount.....	51
8.3.15	Icu_EnableEdgeCount.....	52
8.3.16	Icu_EnableEdgeDetection.....	53
8.3.17	Icu_DisableEdgeDetection.....	54
8.3.18	Icu_DisableEdgeCount.....	55
8.3.19	Icu_GetEdgeNumbers.....	56
8.3.20	Icu_StartSignalMeasurement.....	57
8.3.21	Icu_StopSignalMeasurement.....	58
8.3.22	Icu_GetTimeElapsed.....	59
8.3.23	Icu_GetDutyCycleValues.....	64
8.3.24	Icu_GetVersionInfo.....	68
8.4	Callback notifications.....	68
8.5	Scheduled functions.....	68
8.6	Expected Interfaces.....	69
8.6.1	Mandatory Interfaces.....	69
8.6.2	Optional Interfaces.....	69
8.6.3	Configurable interfaces.....	70
9	Sequence diagrams.....	72
9.1	Icu_Init.....	72
9.2	Icu_DeInit.....	72
9.3	Check Wakeup Events.....	73
9.4	Icu_SetMode.....	74
9.5	Icu_DisableWakeup.....	78
9.6	Icu_EnableWakeup.....	79
9.7	Icu_SetActivationCondition.....	80
9.8	Icu_DisableNotification.....	81
9.9	Icu_EnableNotification.....	82

9.10	Icu_GetInputState .....	84
9.11	Icu Timestamping.....	85
9.12	Icu Edge Counting.....	87
9.13	Icu_GetTimeElapsed.....	88
9.14	Icu_GetDutyCycleValues .....	91
9.15	Icu_SignalNotification and Icu_GetInputState .....	92
10	Configuration specification.....	93
10.1	How to read this chapter .....	93
10.2	Containers and configuration parameters .....	94
10.2.1	Icu.....	94
10.2.2	IcuGeneral.....	94
10.2.3	IcuOptionalApis .....	95
10.2.4	IcuChannel .....	100
10.2.5	IcuSignalEdgeDetection .....	103
10.2.6	IcuSignalMeasurement.....	104
10.2.7	IcuTimestampMeasurement .....	104
10.2.8	IcuWakeup.....	105
10.2.9	IcuConfigSet.....	106
10.3	Published Information.....	107
11	Not applicable requirements .....	108



## 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

Abbreviation / Acronym:	Description:
Active Time	This depends on the starting edge of the signal to be captured. <ul style="list-style-type: none"> <li>▪ Start edge = falling edge =&gt; Active Time = Low Time</li> <li>▪ Start edge = rising edge =&gt; Active Time = High Time</li> <li>▪ Start edge = both edges =&gt; Active Time = High Time (if rising edge occurs initially)</li> <li>▪ Start edge = both edges =&gt; Active Time = Low Time (if falling edge occurs initially)</li> </ul>
DEM	Diagnostic Event Manager
DET	Default Error Tracer
EcuM	ECU State Manager
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 <u>not</u> isn't done by this driver. This shall be done by an upper layer.

## 3 Related documentation

### 3.1 Input documents

- [1] General Requirements on Basic Software Modules, AUTOSAR\_SRS\_BSWGeneral.pdf
- [2] General Requirements on SPAL, AUTOSAR\_SRS\_SPALGeneral.pdf
- [3] Specification of Standard Types, AUTOSAR\_SWS\_StandardTypes.pdf
- [4] List of Basic Software Modules, AUTOSAR\_TR\_BSWModuleList.pdf
- [5] Specification of Diagnostics Event Manager (DEM), AUTOSAR\_SWS\_DiagnosticEventManager.pdf
- [6] Specification of Default Error Tracer, AUTOSAR\_SWS\_DefaultErrorTracer.pdf
- [7] Requirements on ICU Driver, AUTOSAR\_SRS\_ICUDriver.pdf
- [8] Specification of ECU Configuration, AUTOSAR\_TPS\_ECUConfiguration.pdf
- [9] Layered Software Architecture, AUTOSAR\_EXP\_LayeredSoftwareArchitecture.pdf
- [10] Specification of ECU State Manager, AUTOSAR\_SWS\_ECUCStateManager.pdf
- [11] Basic Software Module Description Template, AUTOSAR\_TPS\_BSWModuleDescriptionTemplate.pdf
- [12] General Specification of Basic Software Modules AUTOSAR\_SWS\_BSWGeneral.pdf

### 3.2 Related standards and norms

- [13] IEC 7498-1 The Basic Model, IEC Norm, 1994

### 3.3 Related specification

AUTOSAR provides a General Specification on Basic Software modules [12] (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 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.

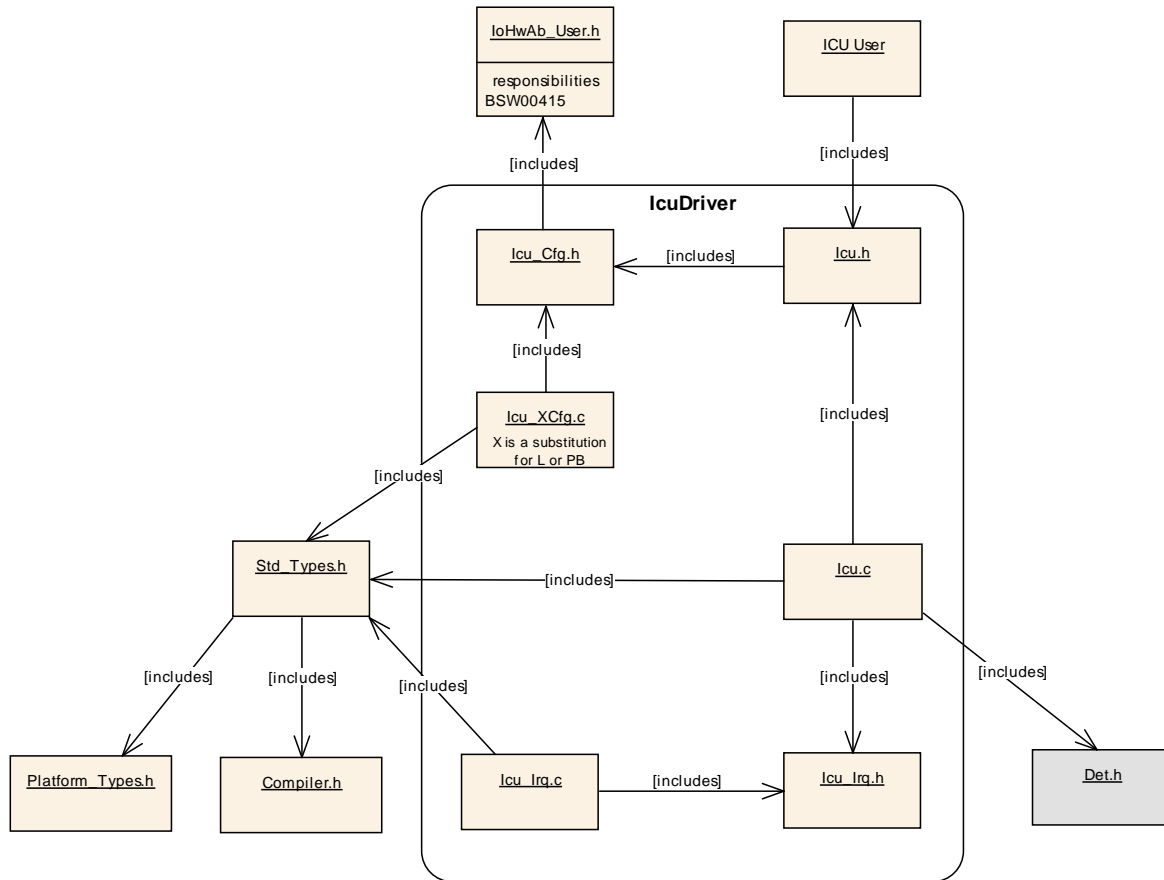
### 5.4 Module PORT

The configuration of port pins used for the ICU as inputs is done by the PORT driver. 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.] ()

## 5.6 File structure



### 5.6.1 Header file structure

The code file structure shall be as follows:

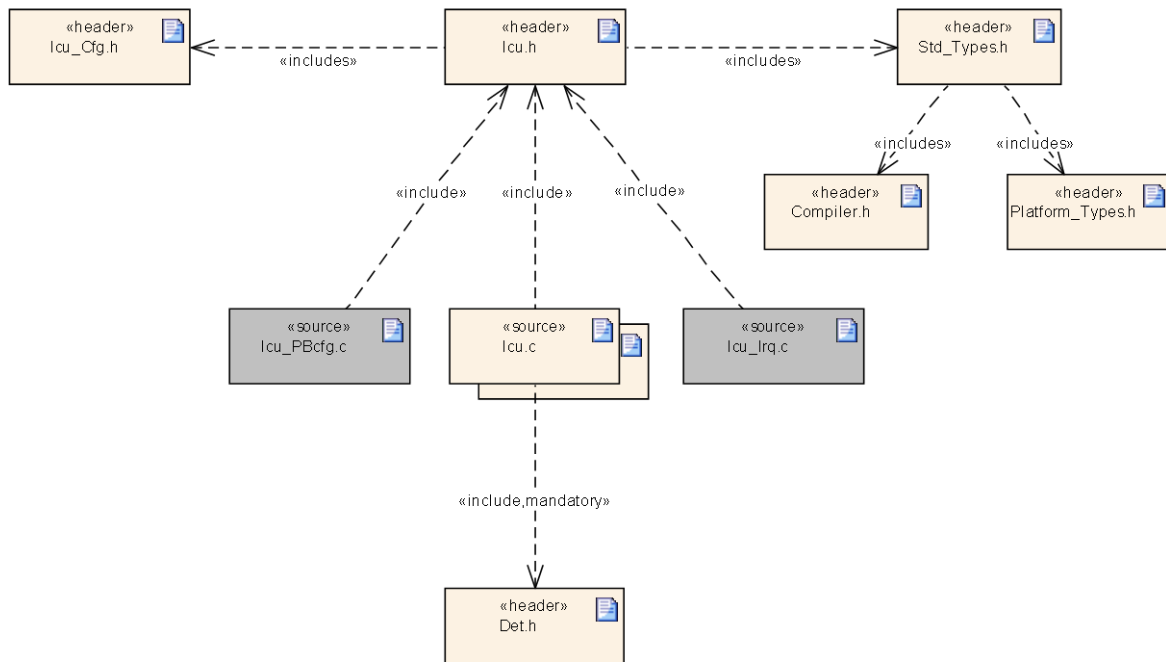


Figure 5.1: Header file structure

**[SWS\_Icu\_00245]** [Icu.h shall include Icu\_Cfg.h for the API pre-compiler switches.

Icu.c has access to the Icu\_Cfg.h via the implicitly included Icu.h file.] ()

**[SWS\_Icu\_00246]** [Icu\_Irq.c shall include Icu.h for the function which shall be called in the interrupt function.and Icu\_Irq.h for the declaration of interrupt functions.] ()

**[SWS\_Icu\_00247]** [The Type definitions for Icu\_Lcfg.c and Icu\_PBcfg.c are located in the file Icu\_Cfg.h or Icu.h.

The implicit include of Icu\_Cfg.h via Icu.h in the files Icu\_Lcfg.c and Icu\_PBcfg.c is necessary and can be solved like in the following construct:

Icu.h shall include EcuM\_Cbk.h, if wakeup functionality is configured.

```

Icu.h
-----
#if defined ICU_VERSION_INFO_API
Icu_GetVersionInfo(...)
#endif

Icu_Cfg.h
-----
#include "Icu.h"
    
```



```
#define ICU_VERSION_INFO_API ()
```

**[SWS\_Icu\_00248]** [Icu\_Lcfg.c shall include Icu\_Cbk.h for a link time configuration if the call back function is linked to the module via the ROM structure.] ()

**[SWS\_Icu\_00249]** [Icu\_PBcfg.c shall include Icu\_Cbk.h for post build time configuration if the call back function is linked to the module via the ROM structure.] ()

**[SWS\_Icu\_00250]** [Icu.c shall include Icu\_Cbk.h for pre-compile time configuration] ()

**[SWS\_Icu\_00251]** [Icu.c shall include Det.h, SchM\_Icu.h and Icu\_MemMap.h.] ()

**[SWS\_Icu\_00252]** [Icu\_Irq.c shall include Icu\_MemMap.h.] ()

**[SWS\_Icu\_00253]** [Icu\_Lcfg.c shall include [ICU.h and Icu\_MemMap.h.] ()

**[SWS\_Icu\_00254]** [Icu\_PBcfg.c shall include Icu\_MemMap.h and Icu.h.] ()

**[SWS\_Icu\_00256]** [Icu.h shall include EcuM\_Cbk.h.] ()

## 6 Requirements traceability

Requirement	Description	Satisfied by
SRS_BSW_00005	Modules of the $\mu$ C Abstraction Layer (MCAL) may not have hard coded horizontal interfaces	SWS_Icu_00380
SRS_BSW_00006	The source code of software modules above the $\mu$ C Abstraction Layer (MCAL) shall not be processor and compiler dependent.	SWS_Icu_00380
SRS_BSW_00007	All Basic SW Modules written in C language shall conform to the MISRA C 2012 Standard.	SWS_Icu_00380
SRS_BSW_00009	All Basic SW Modules shall be documented according to a common standard.	SWS_Icu_00380
SRS_BSW_00010	The memory consumption of all Basic SW Modules shall be documented for a defined configuration for all supported platforms.	SWS_Icu_00380
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_00160	Configuration files of AUTOSAR Basic SW module shall be readable for human beings	SWS_Icu_00380
SRS_BSW_00161	The AUTOSAR Basic Software shall provide a microcontroller abstraction layer which provides a standardized interface to higher software layers	SWS_Icu_00380
SRS_BSW_00162	The AUTOSAR Basic Software shall provide a hardware abstraction layer	SWS_Icu_00380
SRS_BSW_00164	The Implementation of interrupt service routines shall be done by the Operating System, complex drivers or modules	SWS_Icu_00380
SRS_BSW_00167	All AUTOSAR Basic Software Modules shall provide configuration rules and constraints to enable plausibility checks	SWS_Icu_00380
SRS_BSW_00168	SW components shall be tested by a function defined in a common API in the Basis-SW	SWS_Icu_00380
SRS_BSW_00170	The AUTOSAR SW Components shall provide	SWS_Icu_00380

	information about their dependency from faults, signal qualities, driver demands	
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_00172	The scheduling strategy that is built inside the Basic Software Modules shall be compatible with the strategy used in the system	SWS_Icu_00380
SRS_BSW_00300	All AUTOSAR Basic Software Modules shall be identified by an unambiguous name	SWS_Icu_00380
SRS_BSW_00301	All AUTOSAR Basic Software Modules shall only import the necessary information	SWS_Icu_00380
SRS_BSW_00302	All AUTOSAR Basic Software Modules shall only export information needed by other modules	SWS_Icu_00380
SRS_BSW_00304	All AUTOSAR Basic Software Modules shall use the following data types instead of native C data types	SWS_Icu_00380
SRS_BSW_00305	Data types naming convention	SWS_Icu_00380
SRS_BSW_00306	AUTOSAR Basic Software Modules shall be compiler and platform independent	SWS_Icu_00380
SRS_BSW_00307	Global variables naming convention	SWS_Icu_00380
SRS_BSW_00308	AUTOSAR Basic Software Modules shall not define global data in their header files, but in the C file	SWS_Icu_00380
SRS_BSW_00309	All AUTOSAR Basic Software Modules shall indicate all global data with read-only purposes by explicitly assigning the const keyword	SWS_Icu_00380
SRS_BSW_00310	API naming convention	SWS_Icu_00380
SRS_BSW_00312	Shared code shall be reentrant	SWS_Icu_00380
SRS_BSW_00314	All internal driver modules shall separate the interrupt frame definition from the service routine	SWS_Icu_00380
SRS_BSW_00318	Each AUTOSAR Basic Software Module file shall provide version	SWS_Icu_00380

	numbers in the header file	
SRS_BSW_00321	The version numbers of AUTOSAR Basic Software Modules shall be enumerated according specific rules	SWS_Icu_00380
SRS_BSW_00323	All AUTOSAR Basic Software Modules shall check passed API parameters for validity	SWS_Icu_00022, SWS_Icu_00024, SWS_Icu_00043, SWS_Icu_00125
SRS_BSW_00325	The runtime of interrupt service routines and functions that are running in interrupt context shall be kept short	SWS_Icu_00380
SRS_BSW_00327	Error values naming convention	SWS_Icu_00380
SRS_BSW_00328	All AUTOSAR Basic Software Modules shall avoid the duplication of code	SWS_Icu_00380
SRS_BSW_00330	It shall be allowed to use macros instead of functions where source code is used and runtime is critical	SWS_Icu_00380
SRS_BSW_00331	All Basic Software Modules shall strictly separate error and status information	SWS_Icu_00380
SRS_BSW_00333	For each callback function it shall be specified if it is called from interrupt context or not	SWS_Icu_00380
SRS_BSW_00334	All Basic Software Modules shall provide an XML file that contains the meta data	SWS_Icu_00380
SRS_BSW_00335	Status values naming convention	SWS_Icu_00380
SRS_BSW_00336	Basic SW module shall be able to shutdown	SWS_Icu_00037
SRS_BSW_00341	Module documentation shall contains all needed informations	SWS_Icu_00380
SRS_BSW_00342	It shall be possible to create an AUTOSAR ECU out of modules provided as source code and modules provided as object code, even mixed	SWS_Icu_00380
SRS_BSW_00344	BSW Modules shall support link-time configuration	SWS_Icu_00006
SRS_BSW_00347	A Naming separation of different instances of BSW drivers shall be in place	SWS_Icu_00380
SRS_BSW_00348	All AUTOSAR standard types and constants shall be placed and organized in a standard type header file	SWS_Icu_00380
SRS_BSW_00350	All AUTOSAR Basic Software Modules shall allow the	SWS_Icu_00380

	enabling/disabling of detection and reporting of development errors.	
SRS_BSW_00353	All integer type definitions of target and compiler specific scope shall be placed and organized in a single type header	SWS_Icu_00380
SRS_BSW_00357	For success/failure of an API call a standard return type shall be defined	SWS_Icu_00380
SRS_BSW_00358	The return type of init() functions implemented by AUTOSAR Basic Software Modules shall be void	SWS_Icu_00380
SRS_BSW_00359	All AUTOSAR Basic Software Modules callback functions shall avoid return types other than void if possible	SWS_Icu_00187
SRS_BSW_00360	AUTOSAR Basic Software Modules callback functions are allowed to have parameters	SWS_Icu_00380
SRS_BSW_00361	All mappings of not standardized keywords of compiler specific scope shall be placed and organized in a compiler specific type and keyword header	SWS_Icu_00380
SRS_BSW_00369	All AUTOSAR Basic Software Modules shall not return specific development error codes via the API	SWS_Icu_00049
SRS_BSW_00371	The passing of function pointers as API parameter is forbidden for all AUTOSAR Basic Software Modules	SWS_Icu_00380
SRS_BSW_00373	The main processing function of each AUTOSAR Basic Software Module shall be named according the defined convention	SWS_Icu_00380
SRS_BSW_00377	A Basic Software Module can return a module specific types	SWS_Icu_00380
SRS_BSW_00378	AUTOSAR shall provide a boolean type	SWS_Icu_00380
SRS_BSW_00379	All software modules shall provide a module identifier in the header file and in the module XML description file.	SWS_Icu_00380
SRS_BSW_00383	The Basic Software Module specifications shall specify which other configuration files from other modules they use at	SWS_Icu_00380

	least in the description	
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_00395	The Basic Software Module specifications shall list all configuration parameter dependencies	SWS_Icu_00380
SRS_BSW_00397	The configuration parameters in pre-compile time are fixed before compilation starts	SWS_Icu_00380
SRS_BSW_00398	The link-time configuration is achieved on object code basis in the stage after compiling and before linking	SWS_Icu_00380
SRS_BSW_00399	Parameter-sets shall be located in a separate segment and shall be loaded after the code	SWS_Icu_00380
SRS_BSW_00400	Parameter shall be selected from multiple sets of parameters after code has been loaded and started	SWS_Icu_00380
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
SRS_BSW_00406	A static status variable denoting if a BSW module is initialized shall be initialized with value 0 before any APIs of the BSW module is called	SWS_Icu_00022
SRS_BSW_00408	All AUTOSAR Basic Software Modules configuration parameters shall be named according to a specific naming rule	SWS_Icu_00380
SRS_BSW_00409	All production code error ID symbols are defined by the Dem module and shall be retrieved by the other BSW modules from Dem configuration	SWS_Icu_00380
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_BSW_00413	An index-based accessing of the instances of BSW modules shall be done	SWS_Icu_00380

SRS_BSW_00414	Init functions shall have a pointer to a configuration structure as single parameter	SWS_Icu_00380
SRS_BSW_00415	Interfaces which are provided exclusively for one module shall be separated into a dedicated header file	SWS_Icu_00380
SRS_BSW_00416	The sequence of modules to be initialized shall be configurable	SWS_Icu_00380
SRS_BSW_00417	Software which is not part of the SW-C shall report error events only after the DEM is fully operational.	SWS_Icu_00380
SRS_BSW_00422	Pre-de-bouncing of error status information is done within the DEM	SWS_Icu_00380
SRS_BSW_00423	BSW modules with AUTOSAR interfaces shall be describable with the means of the SW-C Template	SWS_Icu_00380
SRS_BSW_00424	BSW module main processing functions shall not be allowed to enter a wait state	SWS_Icu_00380
SRS_BSW_00425	The BSW module description template shall provide means to model the defined trigger conditions of schedulable objects	SWS_Icu_00380
SRS_BSW_00426	BSW Modules shall ensure data consistency of data which is shared between BSW modules	SWS_Icu_00380
SRS_BSW_00427	ISR functions shall be defined and documented in the BSW module description template	SWS_Icu_00380
SRS_BSW_00428	A BSW module shall state if its main processing function(s) has to be executed in a specific order or sequence	SWS_Icu_00380
SRS_BSW_00429	Access to OS is restricted	SWS_Icu_00380
SRS_BSW_00432	Modules should have separate main processing functions for read/receive and write/transmit data path	SWS_Icu_00380
SRS_BSW_00433	Main processing functions are only allowed to be called from task bodies provided by the BSW Scheduler	SWS_Icu_00380
SRS_BSW_00437	Memory mapping shall provide the possibility to define RAM segments which are not to be initialized during startup	SWS_Icu_00380
SRS_BSW_00439	Enable BSW modules to handle	SWS_Icu_00380

	interrupts	
SRS_BSW_00440	The callback function invocation by the BSW module shall follow the signature provided by RTE to invoke servers via Rte_Call API	SWS_Icu_00380
SRS_BSW_00441	Naming convention for type, macro and function	SWS_Icu_00380
SRS_BSW_00450	A Main function of a un-initialized module shall return immediately	SWS_Icu_00380
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_12368	The ICU driver shall support basic static configurations per channel	SWS_Icu_00039
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_00039, 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	SWS_Icu_00078



	available on an ICU channel	
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
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_12455	If circular buffer handling is configured, the driver shall restart at the beginning of the external buffer, when the end of the buffer is reached	SWS_Icu_00039
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_00039, SWS_Icu_00065
SRS_Icu_13100	Resetting 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_12068	The modules of the MCAL shall be initialized in a defined sequence	SWS_Icu_00380
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_12077	All drivers shall provide a non blocking implementation	SWS_Icu_00380
SRS_SPAL_12092	The driver's API shall be accessed by its handler or manager	SWS_Icu_00380
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
SRS_SPAL_12169	All driver modules that provide different operation modes shall provide a service for mode selection	SWS_Icu_00008
SRS_SPAL_12265	Configuration data shall be kept constant	SWS_Icu_00380
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
SRS_SPAL_12463	The register initialization settings shall be combined and forwarded	SWS_Icu_00380



## 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()
- Icu\_GetVersionInfo() ] ()

**[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

] ()

In ICU\_MODE\_NORMAL mode all notifications are available as

- **[SWS\_Icu\_00011]** [configured by service Icu\_SetActivationCondition() or IcuDefaultStartEdge.] (SRS\_SPAL\_12067)
- **[SWS\_Icu\_00259]** [selected by the Icu\_DisableNotification() and Icu\_EnableNotification() services before or after the call of Icu\_SetMode().] ()

In ICU\_MODE\_SLEEP mode

- **[SWS\_Icu\_00012]** [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()] (SRS\_SPAL\_12067)

- **[SWS\_Icu\_00260]** [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]** [The module `Icu` shall allow the configuration per channel of the definition on which edge the period starts.] (SRS\_Icu\_12425)

### 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

Section 7.x "Error Handling" of the document "General Specification of Basic Software Modules" 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

This chapter shall list all Development Errors that can be detected within this software module. For each error, a value shall be defined.

**[SWS\_Icu\_00382]** Development Error Types

<i>Type or error</i>	<i>Relevance</i>	<i>Related error code</i>	<i>Value [hex]</i>
API IS called with invalid pointer.]	Development	ICU_E_PARAM_POINTER	0x0A
API service used with an invalid channel identifier or channel was not configured for the functionality of the	Development	ICU_E_PARAM_CHANNEL	0x0B

<i>Type or error</i>	<i>Relevance</i>	<i>Related error code</i>	<i>Value [hex]</i>
calling API.] ()			
API service used with an invalid or not feasible activation.] ()	Development	ICU_E_PARAM_ACTIVATION	0x0C
Init function failed.] ()	Development	ICU_E_INIT_FAILED	0x0D
API service used with an invalid buffer size.] ()	Development	ICU_E_PARAM_BUFFER_SIZE	0x0E
API service Icu_SetMode used with an invalid mode.] ()	Development	ICU_E_PARAM_MODE	0x0F
API service used without module initialization.] ()	Development	ICU_E_UNINIT	0x14
API service Icu_SetMode is called while a running operation.] ()	Development	ICU_E_BUSY_OPERATION	0x16
API Icu_Init service is called and when the ICU driver and the Hardware are already initialized.] ()	Development	ICU_E_ALREADY_INITIALIZED	0x17
API Icu_StartTimeStamp is called and the parameter NotifyInterval is invalid (e.g."0", NotifyInterval < 1) ] ()	Development	ICU_E_PARAM_NOTIFY_INTERVAL	0x18
API Icu_GetVersionInfo is called and the parameter versioninfo is is invalid ( e.g. NULL ) ] ()	Development	ICU_E_PARAM_VINFO	0x19

## 7.2.2 Runtime Errors

<i>Type or error</i>	<i>Relevance</i>	<i>Related error code</i>	<i>Value [hex]</i>
API service Icu_StopTimeStamp called on a channel which was not started or already stopped] ()	Runtime	ICU_E_NOT_STARTED	0x15

## 7.2.3 Transient Faults

There are no transient faults.

## 7.2.4 Production Errors

There are no production errors.

## 7.2.5 Extended Production Errors

There are no extended production errors.

## 7.3 Error detection

**[SWS\_Icu\_00022]** [If development error detection for the Icu module is enabled: All Icu module functions, except for `Icu_Init` and `Icu_GetVersionInfo`, shall raise development error `ICU_E_UNINIT` when the function `Icu_Init` has not been called. ] (SRS\_BSW\_00323, SRS\_BSW\_00406)

## 7.4 Debugging Concept

### 7.4.1 Background & Rationale

The goal of the debugging module is to offer as much information as possible about the runtime behavior of the systems, 8  
it easier to spot the source of a problem when the integrated software does not behave as expected.

## 8 API specification

### 8.1 Imported types

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

[SWS\_Icu\_00383] [ EcuM\_WakeupSourceType shall be imported from EcuM\_Types.h.] ()

[SWS\_Icu\_00276] [

<i>Module</i>	<i>Imported Type</i>
EcuM	EcuM_WakeupSourceType
Std_Types	Std_ReturnType
	Std_VersionInfoType

] ()

### 8.2 Type definitions

#### 8.2.1 Icu\_ModeType

[SWS\_Icu\_00277] [

<b>Name:</b>	Icu_ModeType	
<b>Type:</b>	Enumeration	
<b>Range:</b>	ICU_MODE_NORMAL	Normal operation, all used interrupts are enabled according to the notification requests.
	ICU_MODE_SLEEP	Reduced power operation. In sleep mode only those notifications are available which are configured as wakeup capable.
<b>Description:</b>	Allow enabling / disabling of all interrupts which are not required for the ECU wakeup.	

] ()

#### 8.2.2 Icu\_ChannelType

[SWS\_Icu\_00278] [

<b>Name:</b>	Icu_ChannelType	
<b>Type:</b>	uint	
<b>Range:</b>	--	-- 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.
<b>Description:</b>	Numeric identifier of an ICU channel	

] ()



### 8.2.3 Icu\_InputStateType

[SWS\_Icu\_00279] [

<b>Name:</b>	Icu_InputStateType	
<b>Type:</b>	Enumeration	
<b>Range:</b>	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().
<b>Description:</b>	Input state of an ICU channel	

] ()

### 8.2.4 Icu\_ConfigType

[SWS\_Icu\_00280] [

<b>Name:</b>	Icu_ConfigType	
<b>Type:</b>	Structure	
<b>Range:</b>	--	Hardware and implementation dependent structure. The contents of the initialization data structure are microcontroller specific.
<b>Description:</b>	This type contains initialization data.	

] ()

[SWS\_Icu\_00281] [The Icu\_ConfigType shall contain:

Optional parameters

- MCU dependent properties for used HW units.
- Clock source with optional prescaler (if provided by HW).] ()

[SWS\_Icu\_00039] [The definition for each Channel within the Icu\_ConfigType shall contain:

Common parameters

- Default Start Edge
- Hardware Specific Settings per channel
- Measurement Mode
  - Signal Edge Detection / Notification
  - Signal Measurement
  - Timestamp
  - Edge Counter

Specific parameters

] (SRS\_Icu\_12368, SRS\_Icu\_12425, SRS\_Icu\_12455, SRS\_Icu\_12456)

[SWS\_Icu\_00283] [If the measurement mode for each Channel within the Icu\_ConfigType is configured as “signal edge detection” the notification function for signal notification shall be configurable.] ()

[SWS\_Icu\_00284] [If the measurement mode for each Channel within the Icu\_ConfigType is configured as “signal measurement”, the property that could be

measured shall be configurable. The values shall be as specified in

[SWS\\_Icu\\_00295.](#)] ()

**[SWS\_Icu\_00285]** [If the measurement mode for each Channel within the `Icu_ConfigType` is configured as “timestamp measurement”, buffer handling shall be configurable. The values shall be as specified in [SWS\\_Icu\\_00296.](#)] ()

**[SWS\_Icu\_00378]** [If the measurement mode for each Channel within the `Icu_ConfigType` is configured as “timestamp measurement”, the notification function for notifying the number of requested timestamps shall be configurable. ] ()

**[SWS\_Icu\_00286]** [If the measurement mode for each Channel within the `Icu_ConfigType` is configured as “edge counter”, the counting mode (activation edge) shall be configurable. The values shall be as specified in [SWS\\_Icu\\_00289.](#)] ()

**[SWS\_Icu\_00287]** [If in the definition for each Channel within the `Icu_ConfigType` the channel is configured as wakeup capable then the callout function for validation of wakeup reason shall be `EcuM_CheckWakeup.`] ()

**[SWS\_Icu\_00288]** [If, in the definition for each Channel within the `Icu_ConfigType`, the channel is configured as wakeup capable then the value transmitted to the `EcuM` shall be configurable.] ()

### 8.2.5 Icu\_ActivationType

**[SWS\_Icu\_00289]** [

<b>Name:</b>	Icu_ActivationType	
<b>Type:</b>	Enumeration	
<b>Range:</b>	ICU_RISING_EDGE	-- An appropriate action shall be executed when a rising edge occurs on the ICU input signal.
	ICU_FALLING_EDGE	-- An appropriate action shall be executed when a falling edge occurs on the ICU input signal.
	ICU_BOTH_EDGES	-- An appropriate action shall be executed when either a rising or falling edge occur on the ICU input signal.
<b>Description:</b>	Definition of the type of activation of an ICU channel.	

] ()

### 8.2.6 Icu\_ValueType

**[SWS\_Icu\_00290]** [

<b>Name:</b>	Icu_ValueType	
<b>Type:</b>	uint	
<b>Range:</b>	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.
<b>Description:</b>	Width of the buffer for timestamp ticks and measured elapsed timeticks.	

] ()

### 8.2.7 Icu\_DutyCycleType

[SWS\_Icu\_00291] [

<b>Name:</b>	Icu_DutyCycleType		
<b>Type:</b>	Structure		
<b>Element:</b>	Icu_ValueType	ActiveTime	This shall be the coherent active-time measured on a channel
	Icu_ValueType	PeriodTime	This shall be the coherent period-time measured on a channel
<b>Description:</b>	Type which shall contain the values, needed for calculating duty cycles.		

] ()

### 8.2.8 Icu\_IndexType

[SWS\_Icu\_00292] [

<b>Name:</b>	Icu_IndexType		
<b>Type:</b>	uint		
<b>Range:</b>	--	--	Implementation specific. This type shall be chosen in order to have the most efficient implementation on a specific microcontroller platform.
<b>Description:</b>	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.		

] ()

### 8.2.9 Icu\_EdgeNumberType

[SWS\_Icu\_00293] [

<b>Name:</b>	Icu_EdgeNumberType		
<b>Type:</b>	uint		
<b>Range:</b>	--	--	Implementation specific. This type shall be chosen in order to have the most efficient implementation on a specific microcontroller platform.
<b>Description:</b>	Type, to abstract the return value of the service Icu_GetEdgeNumbers().		

] ()

### 8.2.10 Icu\_MeasurementModeType

[SWS\_Icu\_00294] [

<b>Name:</b>	Icu_MeasurementModeType		
<b>Type:</b>	Enumeration		
<b>Range:</b>	ICU_MODE_SIGNAL_EDGE_DETECT	--	Mode for detecting edges
	ICU_MODE_SIGNAL_MEASUREMENT	--	Mode for measuring different times between various configurable edges
	ICU_MODE_TIMESTAMP	--	Mode for capturing timer values on configurable edges
	ICU_MODE_EDGE_COUNTER	--	Mode for counting edges on configurable

		edges
<b>Description:</b>	Definition of the measurement mode type	

] ()

### 8.2.11 Icu\_SignalMeasurementPropertyType

[SWS\_Icu\_00295] [

<b>Name:</b>	Icu_SignalMeasurementPropertyType	
<b>Type:</b>	Enumeration	
<b>Range:</b>	ICU_LOW_TIME	-- The channel is configured for reading the elapsed Signal Low Time
	ICU_HIGH_TIME	-- The channel is configured for reading the elapsed Signal High Time
	ICU_PERIOD_TIME	-- The channel is configured for reading the elapsed Signal Period Time
	ICU_DUTY_CYCLE	-- The channel is configured to read values which are needed for calculating the duty cycle (coherent Active and Period Time).
<b>Description:</b>	Definition of the measurement property type	

] ()

### 8.2.12 Icu\_TimestampBufferType

[SWS\_Icu\_00296] [

<b>Name:</b>	Icu_TimestampBufferType	
<b>Type:</b>	Enumeration	
<b>Range:</b>	ICU_LINEAR_BUFFER	-- The buffer will just be filled once
	ICU_CIRCULAR_BUFFER	-- After reaching the end of the buffer, the driver restarts at the beginning of the buffer
<b>Description:</b>	Definition of the timestamp measurement property type	

] ()

## 8.3 Function definitions

This is a list of functions provided for upper layer modules.

### 8.3.1 Icu\_Init

[SWS\_Icu\_00191] [

<b>Service name:</b>	Icu_Init	
<b>Syntax:</b>	void Icu_Init( const Icu_ConfigType* ConfigPtr )	
<b>Service ID[hex]:</b>	0x00	
<b>Sync/Async:</b>	Synchronous	
<b>Reentrancy:</b>	Non Reentrant	
<b>Parameters (in):</b>	ConfigPtr	Pointer to a selected configuration structure
<b>Parameters (inout):</b>	None	
<b>Parameters (out):</b>	None	

<b>Return value:</b>	None
<b>Description:</b>	This function initializes the driver.

] ()

**[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]** [The function `Icu_Init` shall initialize all relevant registers of the configured hardware with the values of the structure referenced by the parameter `ConfigPtr`.] (SRS\_BSW\_00344, SRS\_BSW\_00404, SRS\_BSW\_00405, SRS\_BSW\_00101, SRS\_SPAL\_12057, SRS\_SPAL\_12461)

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

- **[SWS\_Icu\_00051]** [If the hardware allows for only one usage of the register, the driver module implementing that functionality is responsible for initializing the register.] (SRS\_SPAL\_12461)
- **[SWS\_Icu\_00052]** [If the register can affect several hardware modules and if it is an I/O register it shall be initialized by the PORT driver.] (SRS\_SPAL\_12461)
- **[SWS\_Icu\_00053]** [If the register can affect several hardware modules and if it is not an I/O register it shall be initialized by the MCU driver.] (SRS\_SPAL\_12461)
- **[SWS\_Icu\_00128]** [One-time writable registers that require initialization directly after reset shall be initialized by the start-up code.] (SRS\_SPAL\_12461)
- **[SWS\_Icu\_00129]** [All other registers shall be initialized by the startup code.] (SRS\_SPAL\_12461)

**[SWS\_Icu\_00061]** [The function `Icu_Init` shall disable all notifications.] (SRS\_SPAL\_12057, SRS\_Icu\_12407)

**[SWS\_Icu\_00121]** [The function `Icu_Init` shall disable the wakeup-capability of all channels.] ()

**[SWS\_Icu\_00040]** [The function `Icu_Init` shall set all used ICU channels to status `ICU_IDLE`.] (SRS\_SPAL\_12057, SRS\_Icu\_12407)

**[SWS\_Icu\_00060]** [The function `Icu_Init` shall set the module mode to `ICU_MODE_NORMAL`.] (SRS\_SPAL\_12057)

**[SWS\_Icu\_00054]** [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).] (SRS\_SPAL\_12125)

**[SWS\_Icu\_00220]** [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.

### 8.3.2 Icu\_DeInit

**[SWS\_Icu\_00193]** [

<b>Service name:</b>	<code>Icu_DeInit</code>
<b>Syntax:</b>	<code>void Icu_DeInit(     void )</code>
<b>Service ID[hex]:</b>	0x01
<b>Sync/Async:</b>	Synchronous
<b>Reentrancy:</b>	Non Reentrant
<b>Parameters (in):</b>	None
<b>Parameters (inout):</b>	None
<b>Parameters (out):</b>	None
<b>Return value:</b>	None
<b>Description:</b>	This function de-initializes the ICU module.

] ()

**[SWS\_Icu\_00036]** [The function `Icu_DeInit` shall set the state of the peripherals used by configuration as the same after power on reset.] (SRS\_SPAL\_12163, SRS\_Icu\_12429)

**[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]** [The function `Icu_DeInit` shall disable all used interrupts and notifications.] (SRS\_BSW\_00336, SRS\_SPAL\_12163)

**[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]** [The function `Icu_DeInit` shall be pre compile time configurable by configuration parameter `IcuDeInitApi`.] (SRS\_BSW\_00410, SRS\_BSW\_00171)

**[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\\_00022](#) apply to the function `Icu_DeInit`.] ()

### 8.3.3 Icu\_SetMode

**[SWS\_Icu\_00194]** [

<b>Service name:</b>	<code>Icu_SetMode</code>
<b>Syntax:</b>	<code>void Icu_SetMode(     Icu_ModeType Mode )</code>
<b>Service ID[hex]:</b>	0x02
<b>Sync/Async:</b>	Synchronous
<b>Reentrancy:</b>	Non Reentrant
<b>Parameters (in):</b>	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.
<b>Parameters (inout):</b>	None
<b>Parameters (out):</b>	None
<b>Return value:</b>	None
<b>Description:</b>	This function sets the ICU mode.

] ()

**[SWS\_Icu\_00008]** [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.] (SRS\_SPAL\_12067, SRS\_SPAL\_12169, SRS\_Icu\_12370)

**[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]** [The function `Icu_SetMode` shall be pre-compile time configurable by the configuration parameter `IcuSetModeApi`.] (SRS\_BSW\_00410, SRS\_BSW\_00171)

**[SWS\_Icu\_00303]** [The function `Icu_SetMode` shall be configurable ON/OFF by the configuration parameter `IcuSetModeApi`.] ()

**[SWS\_Icu\_00125]** [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.] (SRS\_BSW\_00323)

**[SWS\_Icu\_00133]** [This service can be called during running operations. If so, an ongoing operation that generates interrupts on a wakeup capable channel like e.g. time stamping or edge counting might lead to the ICU module not being able to properly enter sleep mode. This is then a system or ECU configuration issue not a problem of this specification.

[SWS\\_Icu\\_00022](#) apply to the function `Icu_SetMode`.] (SRS\_SPAL\_12064)

### 8.3.4 Icu\_DisableWakeup

**[SWS\_Icu\_00195]** [

<b>Service name:</b>	Icu_DisableWakeup	
<b>Syntax:</b>	<pre>void Icu_DisableWakeup(     Icu_ChannelType Channel )</pre>	
<b>Service ID[hex]:</b>	0x03	
<b>Sync/Async:</b>	Synchronous	
<b>Reentrancy:</b>	Reentrant (limited according to ICU050)	
<b>Parameters (in):</b>	Channel	Numeric identifier of the ICU channel
<b>Parameters (inout):</b>	None	



<b>Parameters (out):</b>	None
<b>Return value:</b>	None
<b>Description:</b>	This function disables the wakeup capability of a single ICU channel.

] ()

**[SWS\_Icu\_00013]** [The function `Icu_DisableWakeup` shall disable the wakeup capability of a single ICU channel.] (SRS\_Icu\_12408)

**[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]** [The function `Icu_DisableWakeup` shall be pre compile time configurable by the configuration parameter `IcuDisableWakeupApi`.] (SRS\_BSW\_00410, SRS\_BSW\_00171)

**[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]** [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.] (SRS\_BSW\_00323)

**[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\\_00022](#) apply to the function `Icu_DisableWakeup`.] ()

### 8.3.5 Icu\_EnableWakeup

**[SWS\_Icu\_00196]** [

<b>Service name:</b>	<code>Icu_EnableWakeup</code>
<b>Syntax:</b>	<code>void Icu_EnableWakeup(     Icu_ChannelType Channel )</code>
<b>Service ID[hex]:</b>	0x04
<b>Sync/Async:</b>	Synchronous
<b>Reentrancy:</b>	Reentrant (limited according to ICU050)

<b>Parameters (in):</b>	Channel	Numeric identifier of the ICU channel
<b>Parameters (inout):</b>	None	
<b>Parameters (out):</b>	None	
<b>Return value:</b>	None	
<b>Description:</b>	This function (re-)enables the wakeup capability of the given ICU channel.	

] ()

**[SWS\_Icu\_00307]** [The function `Icu_EnableWakeup` shall be re-entrant.] ()

**[SWS\_Icu\_00014]** [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.] (SRS\_Icu\_12408)

**[SWS\_Icu\_00097]** [The function `Icu_EnableWakeup` shall be pre compile time configurable by configuration parameter `IcuEnableWakeupApi`.] (SRS\_BSW\_00410, SRS\_BSW\_00171)

**[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\\_00022](#) apply to the function `Icu_EnableWakeup`.] ()

### 8.3.6 Icu\_CheckWakeup

**[SWS\_Icu\_00358]** [

<b>Service name:</b>	<code>Icu_CheckWakeup</code>	
<b>Syntax:</b>	<pre>void Icu_CheckWakeup(     EcuM_WakeupSourceType WakeupSource )</pre>	
<b>Service ID[hex]:</b>	0x15	
<b>Sync/Async:</b>	Synchronous	
<b>Reentrancy:</b>	Reentrant (limited according to ICU050)	
<b>Parameters (in):</b>	WakeupSource	Informatin on wakeup source to be checked. The associated ICU channel can be determined from

	configuration data.
<b>Parameters (inout):</b>	None
<b>Parameters (out):</b>	None
<b>Return value:</b>	None
<b>Description:</b>	Checks if a wakeup capable ICU channel is the source for a wakeup event and calls the ECU state manager service <code>EcuM_SetWakeupEvent</code> in case of a valid ICU channel wakeup event.

] ()

**[SWS\_Icu\_00359]** [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]** [

<b>Service name:</b>	<code>Icu_SetActivationCondition</code>	
<b>Syntax:</b>	<pre>void Icu_SetActivationCondition(     Icu_ChannelType Channel,     Icu_ActivationType Activation )</pre>	
<b>Service ID[hex]:</b>	0x05	
<b>Sync/Async:</b>	Synchronous	
<b>Reentrancy:</b>	Reentrant (limited according to ICU050)	
<b>Parameters (in):</b>	Channel	Numeric identifier of the ICU channel
	Activation	Type of activation (if supported by hardware) - <code>ICU_RISING_EDGE</code> - <code>ICU_FALLING_EDGE</code> - <code>ICU_BOTH_EDGES</code>
<b>Parameters</b>	None	

<b>(inout):</b>	
<b>Parameters (out):</b>	None
<b>Return value:</b>	None
<b>Description:</b>	This function sets the activation-edge for the given channel.

] ()

**[SWS\_Icu\_00090]** [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

`IcuMeasurementMode` (for details refer to 8.2.10)

- `ICU_MODE_SIGNAL_EDGE_DETECT`
- `ICU_MODE_TIMESTAMP`
- `ICU_MODE_EDGE_COUNTER`] (SRS\_BSW\_00410)

**[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 re-entrant.] ()

**[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]** [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.

[SWS\\_Icu\\_00022](#) apply to the function `Icu_SetActivationCondition`.] (SRS\_BSW\_00323)

### 8.3.8 Icu\_DisableNotification

**[SWS\_Icu\_00198]** [

<b>Service name:</b>	<code>Icu_DisableNotification</code>
<b>Syntax:</b>	<code>void Icu_DisableNotification(     Icu_ChannelType Channel )</code>
<b>Service ID[hex]:</b>	<code>0x06</code>
<b>Sync/Async:</b>	Synchronous
<b>Reentrancy:</b>	Reentrant (limited according to ICU050)

<b>Parameters (in):</b>	Channel	Numeric identifier of the ICU channel
<b>Parameters (inout):</b>	None	
<b>Parameters (out):</b>	None	
<b>Return value:</b>	None	
<b>Description:</b>	This function disables the notification of a channel.	

] ()

**[SWS\_Icu\_00009]** [The function `Icu_DisableNotification` shall disable the notification on the given channel.] (SRS\_Icu\_12305)

**[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\\_00022](#) apply to the function `Icu_DisableNotification`.] ()

### 8.3.9 Icu\_EnableNotification

**[SWS\_Icu\_00199]** [

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

] ()

**[SWS\_Icu\_00010]** [The function `Icu_EnableNotification` shall enable the notification on the given channel.] (SRS\_Icu\_12305)

**[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\\_00022](#) apply to the function `Icu_EnableNotification. ] ()`

### 8.3.10 Icu\_GetInputState

**[SWS\_Icu\_00200]** [

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

] ()

**[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]** [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`] (`SRS_SPAL_00157`, `SRS_Icu_12371`)

**[SWS\_Icu\_00312]** [The function `Icu_GetInputState` shall be re-entrant.] ()

**[SWS\_Icu\_00031]** [If an activation edge has been detected the function `Icu_GetInputState` shall return `ICU_ACTIVE` for Edge Detection channels.] (`SRS_Icu_12371`)

**[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]** [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.] (SRS\_Icu\_12371)

**[SWS\_Icu\_00122]** [The function `Icu_GetInputState` shall be pre compile time configurable by the configuration parameter `IcuGetInputStateApi`.] (SRS\_BSW\_00410, SRS\_BSW\_00171)

**[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]** [If development error detection is enabled the function `Icu_GetInputState` shall return `ICU_IDLE` if an error is detected.

[SWS\\_Icu\\_00022](#) apply to the function `Icu_GetInputState`.] (SRS\_SPAL\_12448, SRS\_BSW\_00369)

### 8.3.11 Icu\_StartTimestamp

**[SWS\_Icu\_00201]** [

<b>Service name:</b>	Icu_StartTimestamp	
<b>Syntax:</b>	<pre>void Icu_StartTimestamp(     Icu_ChannelType Channel,     Icu_ValueType* BufferPtr,     uint16 BufferSize,     uint16 NotifyInterval )</pre>	
<b>Service ID[hex]:</b>	0x09	
<b>Sync/Async:</b>	Asynchronous	
<b>Reentrancy:</b>	Reentrant (limited according to ICU050)	
<b>Parameters (in):</b>	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.
<b>Parameters (inout):</b>	None	
<b>Parameters (out):</b>	BufferPtr	Pointer to the buffer-array where the timestamp values shall be placed.
<b>Return value:</b>	None	
<b>Description:</b>	This function starts the capturing of timer values on the edges.	

] ()

**[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]** [The function `Icu_StartTimestamp` shall start the capturing of timer values on the edges

- activated by the service `Icu_SetActivationCondition()` (rising / falling / both edges)] (SRS\_BSW\_00410, SRS\_SPAL\_12063, SRS\_SPAL\_12075, SRS\_Icu\_12430, SRS\_Icu\_12438)

**[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]** [If linear buffer handling is configured, when the capture functionality reaches the end of the buffer, the Icu module shall stop capturing timer values.] (SRS\_Icu\_12456)

**[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]** [The function `Icu_StartTimeStamP` shall only be available in Measurement Mode "ICU\_MODE\_TIMESTAMP".] (SRS\_Icu\_12430)

**[SWS\_Icu\_00098]** [The function `Icu_StartTimestamp` shall be pre-compile time configurable by the configuration parameter: `ICU_TIMESTAMP_API`.] (SRS\_BSW\_00171)

**[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]** [ 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\\_00022](#) apply to the function `Icu_StartTimestamp`.] (SRS\_SPAL\_12448)

### 8.3.12 Icu\_StopTimestamp

**[SWS\_Icu\_00202]** [

<b>Service name:</b>	<code>Icu_StopTimestamp</code>
<b>Syntax:</b>	<code>void Icu_StopTimestamp(     Icu_ChannelType Channel )</code>
<b>Service ID[hex]:</b>	0x0a
<b>Sync/Async:</b>	Synchronous
<b>Reentrancy:</b>	Reentrant (limited according to ICU050)
<b>Parameters (in):</b>	<code>Channel</code> Numeric identifier of the ICU channel
<b>Parameters (inout):</b>	None
<b>Parameters (out):</b>	None
<b>Return value:</b>	None
<b>Description:</b>	This function stops the timestamp measurement of the given channel.

] ()

**[SWS\_Icu\_00067]** [The function `Icu_StopTimestamp` shall stop the timestamp measurement of the given channel.] (SRS\_Icu\_12431)

**[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]** [The function `Icu_StopTimestamp` shall be pre-compile time configurable by the configuration parameter: `IcuTimestampApi` (see also chapter 10.2.4. [Configuration of optional API services](#))] (SRS\_BSW\_00410, SRS\_BSW\_00171)

**[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\\_00022](#) apply to the function `Icu_StopTimestamp.`] ()

### 8.3.13 Icu\_GetTimestampIndex

**[SWS\_Icu\_00203]** [

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

] ()

**[SWS\_Icu\_00071]** [The function `Icu_GetTimestampIndex` shall read the timestamp index of the given channel, which is the next to be written.] (SRS\_Icu\_12453)

**[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]** [The function `Icu_GetTimestampIndex` shall be pre compile time configurable by the configuration parameter: `IcuTimestampApi`] (SRS\_BSW\_00410, SRS\_BSW\_00171)

**[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]** [If development error detection is enabled the function `Icu_GetTimestampIndex` shall return “0” if an error is detected.

[SWS\\_Icu\\_00022](#) apply to the function `Icu_GetTimestampIndex`. ] (SRS\_SPAL\_12448)

### 8.3.14 Icu\_ResetEdgeCount

**[SWS\_Icu\_00204]** [

<b>Service name:</b>	<code>Icu_ResetEdgeCount</code>	
<b>Syntax:</b>	<pre>void Icu_ResetEdgeCount(     Icu_ChannelType Channel )</pre>	
<b>Service ID[hex]:</b>	0x0c	
<b>Sync/Async:</b>	Synchronous	
<b>Reentrancy:</b>	Reentrant (limited according to ICU050)	
<b>Parameters (in):</b>	<code>Channel</code>	Numeric identifier of the ICU channel
<b>Parameters (inout):</b>	None	
<b>Parameters (out):</b>	None	
<b>Return value:</b>	None	
<b>Description:</b>	This function resets the value of the counted edges to zero.	

] ()

**[SWS\_Icu\_00072]** [The function `Icu_ResetEdgeCount` shall reset the value of the counted edges to zero.] (SRS\_Icu\_12439, SRS\_Icu\_13100)

**[SWS\_Icu\_00326]** [The function `Icu_ResetEdgeCount` shall be re-entrant.] ()

**[SWS\_Icu\_00101]** [The function `Icu_ResetEdgeCount` shall be pre-compile time configurable by the configuration parameter `ICU_EDGE_COUNT_API`.] (SRS\_BSW\_00410, SRS\_BSW\_00171)

**[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\\_00022](#) apply to the function `Icu_ResetEdgeCount`.] ()

### 8.3.15 Icu\_EnableEdgeCount

**[SWS\_Icu\_00205]** [

<b>Service name:</b>	<code>Icu_EnableEdgeCount</code>
<b>Syntax:</b>	<code>void Icu_EnableEdgeCount(     Icu_ChannelType Channel )</code>
<b>Service ID[hex]:</b>	0x0d
<b>Sync/Async:</b>	Synchronous
<b>Reentrancy:</b>	Reentrant (limited according to ICU050)
<b>Parameters (in):</b>	Channel      Numeric identifier of the ICU channel
<b>Parameters (inout):</b>	None
<b>Parameters (out):</b>	None
<b>Return value:</b>	None
<b>Description:</b>	This function enables the counting of edges of the given channel.

] ()

**[SWS\_Icu\_00078]** [The function `Icu_EnableEdgeCount` shall enable the counting of edges of the given channel.] (SRS\_Icu\_12432)

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

**[SWS\_Icu\_00073]** [The function `Icu_EnableEdgeCount` shall only count the configured<sup>1</sup> edges (rising edge / falling edge / both edges).] (SRS\_Icu\_12439)

**[SWS\_Icu\_00074]** [The function `Icu_EnableEdgeCount` shall be available for each ICU channel in Measurement Mode “Edge Counter”.] (SRS\_Icu\_12439)

**[SWS\_Icu\_00328]** [The function `Icu_EnableEdgeCount` shall be re-entrant.] ()

**[SWS\_Icu\_00102]** [The function `Icu_EnableEdgeCount` shall be pre-compile time configurable by the configuration parameter `ICU_EDGE_COUNT_API`] (SRS\_BSW\_00410, SRS\_BSW\_00171)

**[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\\_00022](#) apply to the function `Icu_EnableEdgeCount`.] ()

### 8.3.16 Icu\_EnableEdgeDetection

**[SWS\_Icu\_00364]** [

<b>Service name:</b>	<code>Icu_EnableEdgeDetection</code>
<b>Syntax:</b>	<code>void Icu_EnableEdgeDetection(     Icu_ChannelType Channel )</code>
<b>Service ID[hex]:</b>	0x16
<b>Sync/Async:</b>	Synchronous
<b>Reentrancy:</b>	Reentrant (limited according to ICU050)
<b>Parameters (in):</b>	Channel      Numeric identifier of the ICU channel
<b>Parameters (inout):</b>	None
<b>Parameters (out):</b>	None
<b>Return value:</b>	None
<b>Description:</b>	This function enables / re-enables the detection of edges of the given channel.

] ()

**[SWS\_Icu\_00365]** [The function `Icu_EnableEdgeDetection` shall enable the detection of edges for the given channel.] ()

<sup>1</sup> Configured edge after the call of `Icu_Init()` (default-edge) or `Icu_SetActivationCondition()`.

**[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\\_00022](#) apply to the function `Icu_EnableEdgeDetection`] ()

### 8.3.17 Icu\_DisableEdgeDetection

**[SWS\_Icu\_00377]** [

<b>Service name:</b>	<code>Icu_DisableEdgeDetection</code>
<b>Syntax:</b>	<code>void Icu_DisableEdgeDetection(     Icu_ChannelType Channel )</code>
<b>Service ID[hex]:</b>	0x17
<b>Sync/Async:</b>	Synchronous
<b>Reentrancy:</b>	Reentrant (limited according to ICU050)
<b>Parameters (in):</b>	Channel      Numeric identifier of the ICU channel
<b>Parameters (inout):</b>	None
<b>Parameters (out):</b>	None
<b>Return value:</b>	None
<b>Description:</b>	This function disables the detection of edges of the given channel.

] ()

**[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\\_00022](#) apply to the function `Icu_DisableEdgeDetection`.] ()

### 8.3.18 Icu\_DisableEdgeCount

**[SWS\_Icu\_00206]** [

<b>Service name:</b>	<code>Icu_DisableEdgeCount</code>
<b>Syntax:</b>	<code>void Icu_DisableEdgeCount(     Icu_ChannelType Channel )</code>
<b>Service ID[hex]:</b>	0x0e
<b>Sync/Async:</b>	Synchronous
<b>Reentrancy:</b>	Reentrant (limited according to ICU050)
<b>Parameters (in):</b>	<code>Channel</code>   Numeric identifier of the ICU channel
<b>Parameters (inout):</b>	None
<b>Parameters (out):</b>	None
<b>Return value:</b>	None
<b>Description:</b>	This function disables the counting of edges of the given channel.

] ()

**[SWS\_Icu\_00079]** [The function `Icu_DisableEdgeCount` shall disable the counting of edges of the given channel.] (SRS\_Icu\_12433)

**[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]** [The function `Icu_DisableEdgeCount` shall be pre-compile time configurable by the configuration parameter `IcuEdgeCountApi`.] (SRS\_BSW\_00410, SRS\_BSW\_00171)

[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\\_00022](#) apply to the function `Icu_DisableEdgeCount`.] ()

### 8.3.19 Icu\_GetEdgeNumbers

[SWS\_Icu\_00207] [

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

] ()

[SWS\_Icu\_00080] [The function `Icu_GetEdgeNumbers` shall read the number of counted edges after the last call of `Icu_ResetEdgeCount` (.] (SRS\_Icu\_12434)

[SWS\_Icu\_00332] [The function `Icu_GetEdgeNumbers` shall be re-entrant.] ()

[SWS\_Icu\_00104] [The function `Icu_GetEdgeNumbers` shall be pre compile time configurable by the configuration parameter: `ICU_EDGE_COUNT_API`] (SRS\_BSW\_00410, SRS\_BSW\_00171)

[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\\_00022](#) apply to the function `Icu_GetEdgeNumbers.] ()`

### 8.3.20 Icu\_StartSignalMeasurement

**[SWS\_Icu\_00208]** [

<b>Service name:</b>	<code>Icu_StartSignalMeasurement</code>
<b>Syntax:</b>	<code>void Icu_StartSignalMeasurement(     Icu_ChannelType Channel )</code>
<b>Service ID[hex]:</b>	0x13
<b>Sync/Async:</b>	Asynchronous
<b>Reentrancy:</b>	Reentrant (limited according to ICU050)
<b>Parameters (in):</b>	Channel      Numeric identifier of the ICU channel
<b>Parameters (inout):</b>	None
<b>Parameters (out):</b>	None
<b>Return value:</b>	None
<b>Description:</b>	This function starts the measurement of signals.

`] ()`

**[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\\_00022](#) apply to the function `Icu_StartSignalMeasurement`.] ()

### 8.3.21 Icu\_StopSignalMeasurement

**[SWS\_Icu\_00209]** [

<b>Service name:</b>	<code>Icu_StopSignalMeasurement</code>
<b>Syntax:</b>	<code>void Icu_StopSignalMeasurement(     Icu_ChannelType Channel )</code>
<b>Service ID[hex]:</b>	0x14
<b>Sync/Async:</b>	Synchronous
<b>Reentrancy:</b>	Reentrant (limited according to ICU050)
<b>Parameters (in):</b>	<code>Channel</code> Numeric identifier of the ICU channel
<b>Parameters (inout):</b>	None
<b>Parameters (out):</b>	None
<b>Return value:</b>	None
<b>Description:</b>	This function stops the measurement of signals of the given channel.

] ()

**[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\\_00022](#) apply to the function `Icu_StopSignalMeasurement.()`

### 8.3.22 Icu\_GetTimeElapsed

**[SWS\_Icu\_00210]** [

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

] ()

**[SWS\_Icu\_00338]** [The function `Icu_GetTimeElapsed` shall be re-entrant.] ()

**[SWS\_Icu\_00081]** [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.] (SRS\_SPAL\_12063, SRS\_Icu\_12442)

**[SWS\_Icu\_00082]** [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.] (SRS\_SPAL\_12063, SRS\_Icu\_12435)

**[SWS\_Icu\_00083]** [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.] (SRS\_SPAL\_12063, SRS\_Icu\_12443)

**[SWS\_Icu\_00136]** [The function Icu\_GetTimeElapsed shall return “0” in case no requested time has been captured (see

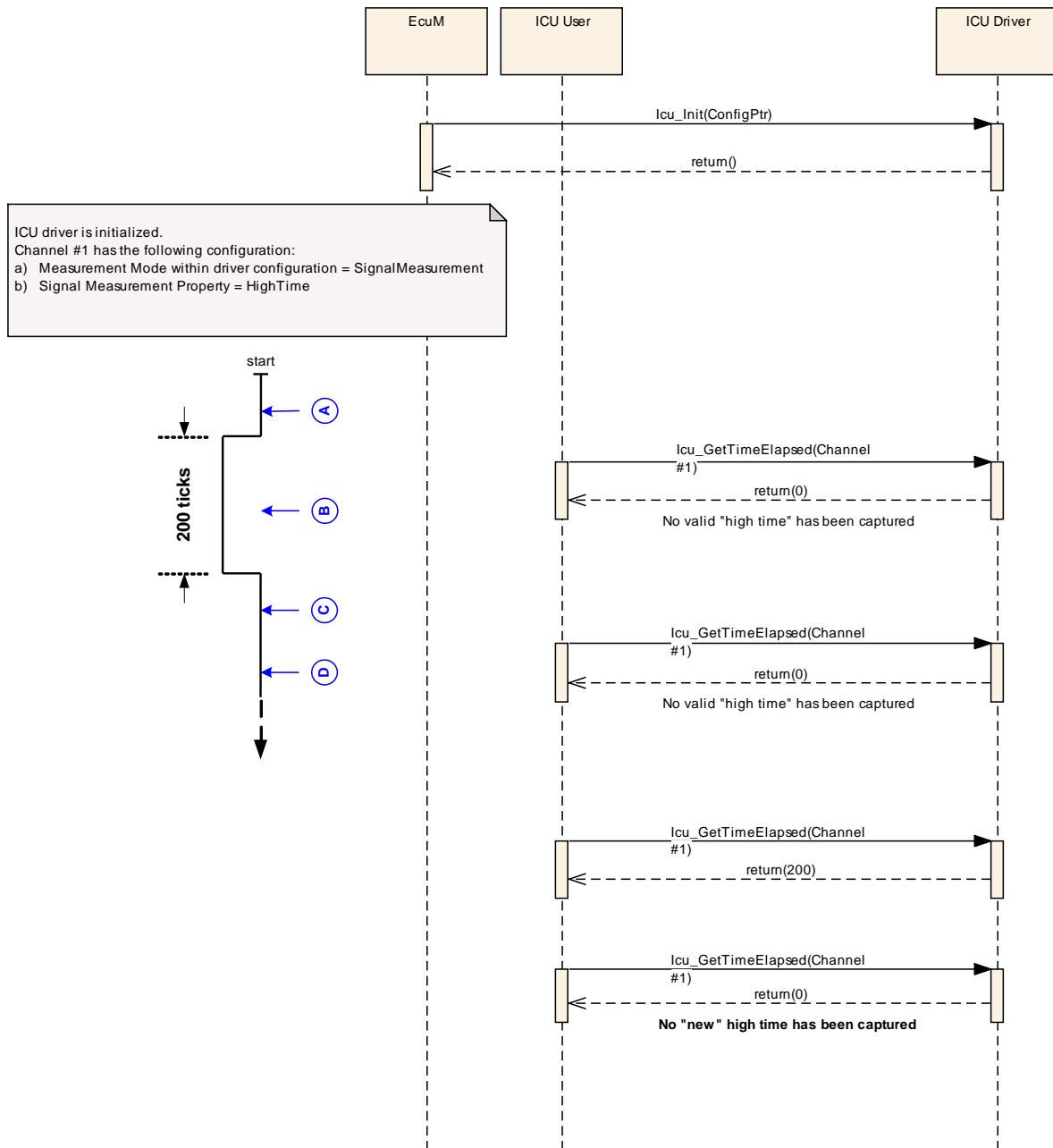


Figure 9.19, letter “A”).] ()

**[SWS\_Icu\_00339]** [The function `Icu_GetTimeElapsed` shall return “0” in case the capturing of a requested time is ongoing and not finished (see

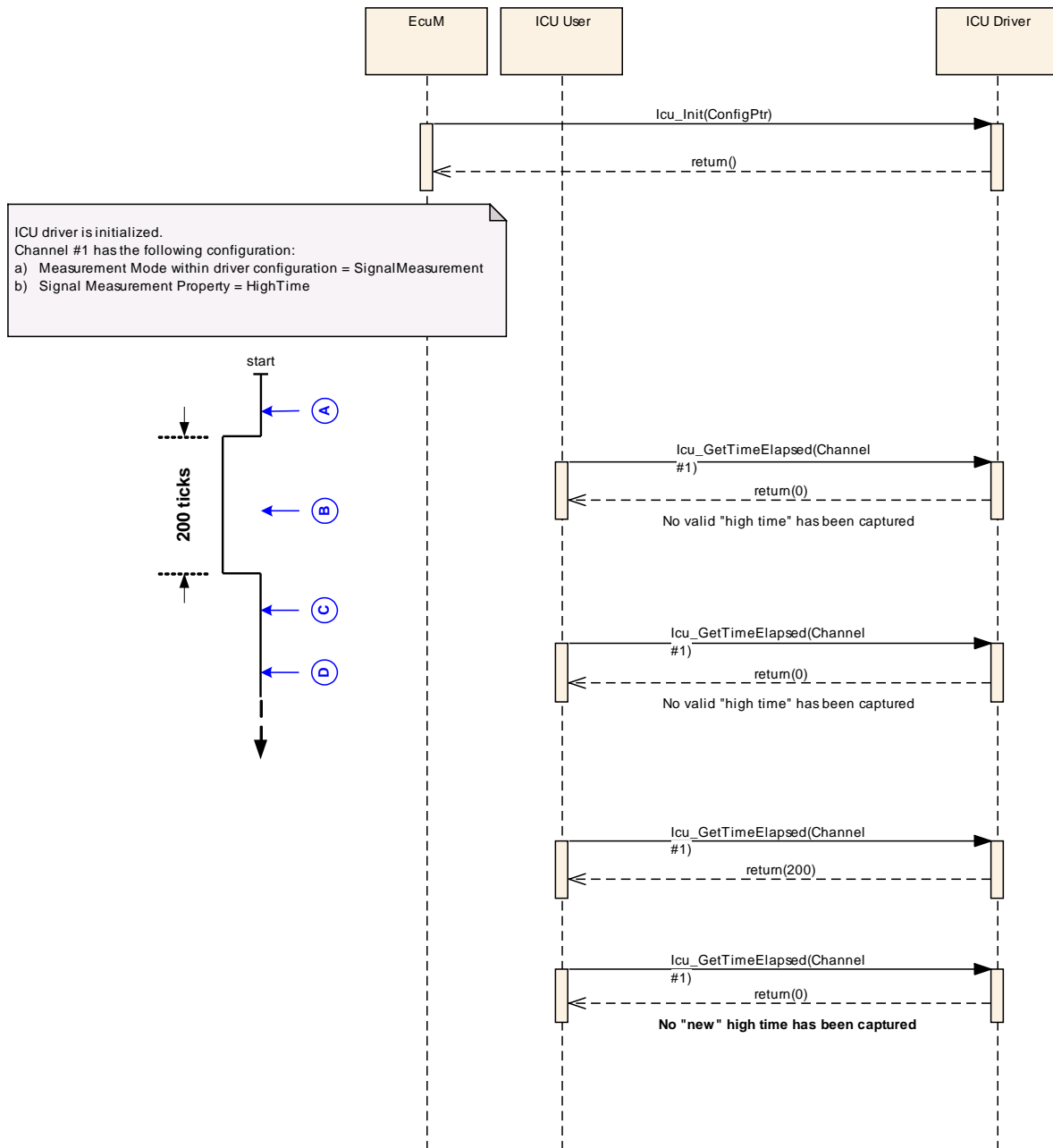


Figure 9.19, letter “B”) ] ()

**[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

(see

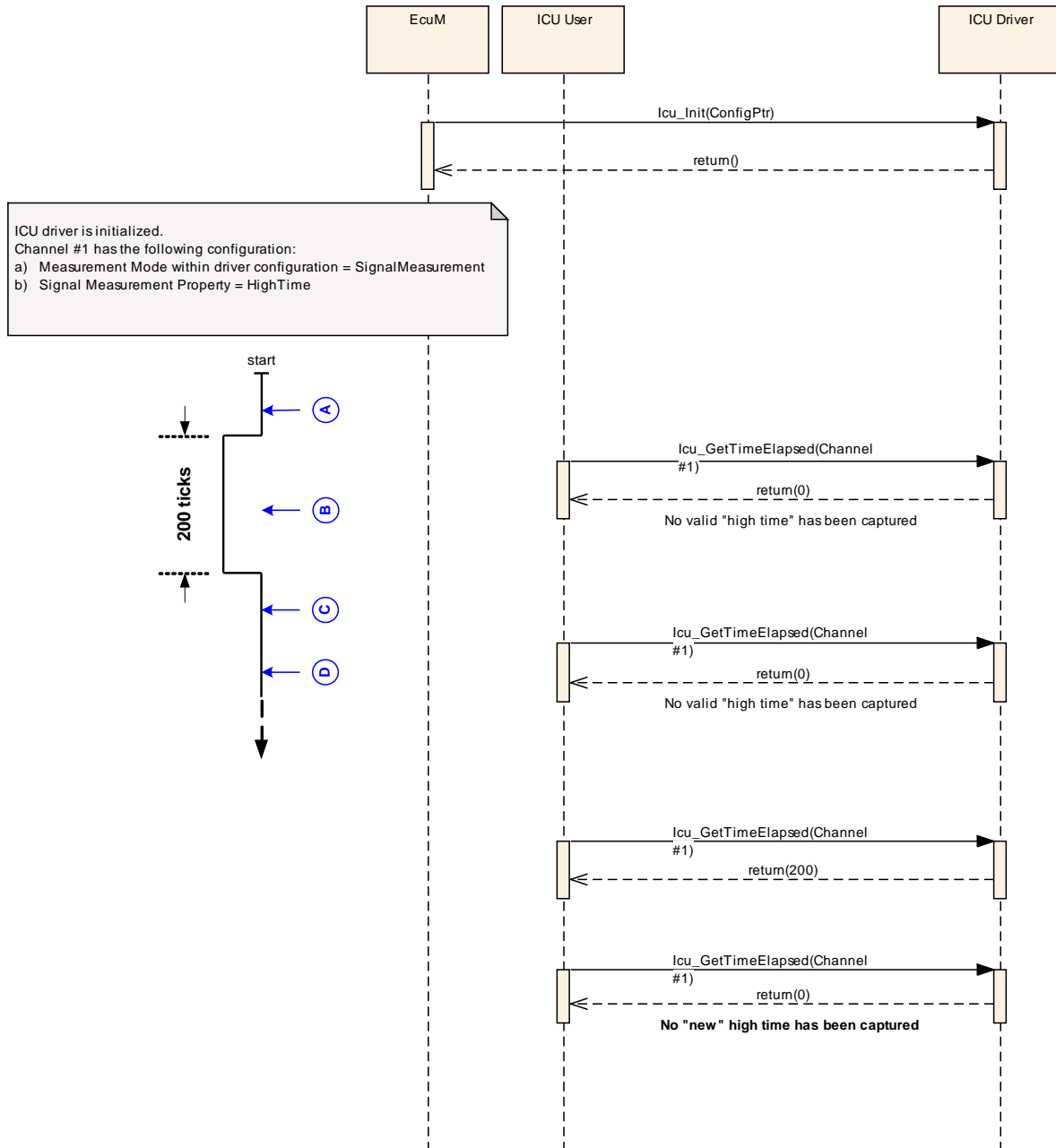


Figure 9.19, letter "D") ] ( )

**[SWS\_Icu\_00105]** [The function `Icu_GetTimeElapsed` shall be pre compile time configurable by the configuration parameter `IcuGetTimeElapsedApi.`] (SRS\_BSW\_00410, SRS\_BSW\_00171)

**[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\\_00022](#) apply to the function `Icu_GetTimeElapsed`.] ()

### 8.3.23 Icu\_GetDutyCycleValues

**[SWS\_Icu\_00211]** [

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

] ()

**[SWS\_Icu\_00342]** [The function `Icu_GetDutyCycleValues` shall be re-entrant. ]  
()

**[SWS\_Icu\_00084]** [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".] (SRS\_Icu\_12436)



**[SWS\_Icu\_00137]** [The function `Icu_GetDutyCycleValues` shall return “0” in case no coherent active- and period time has been captured (similar to

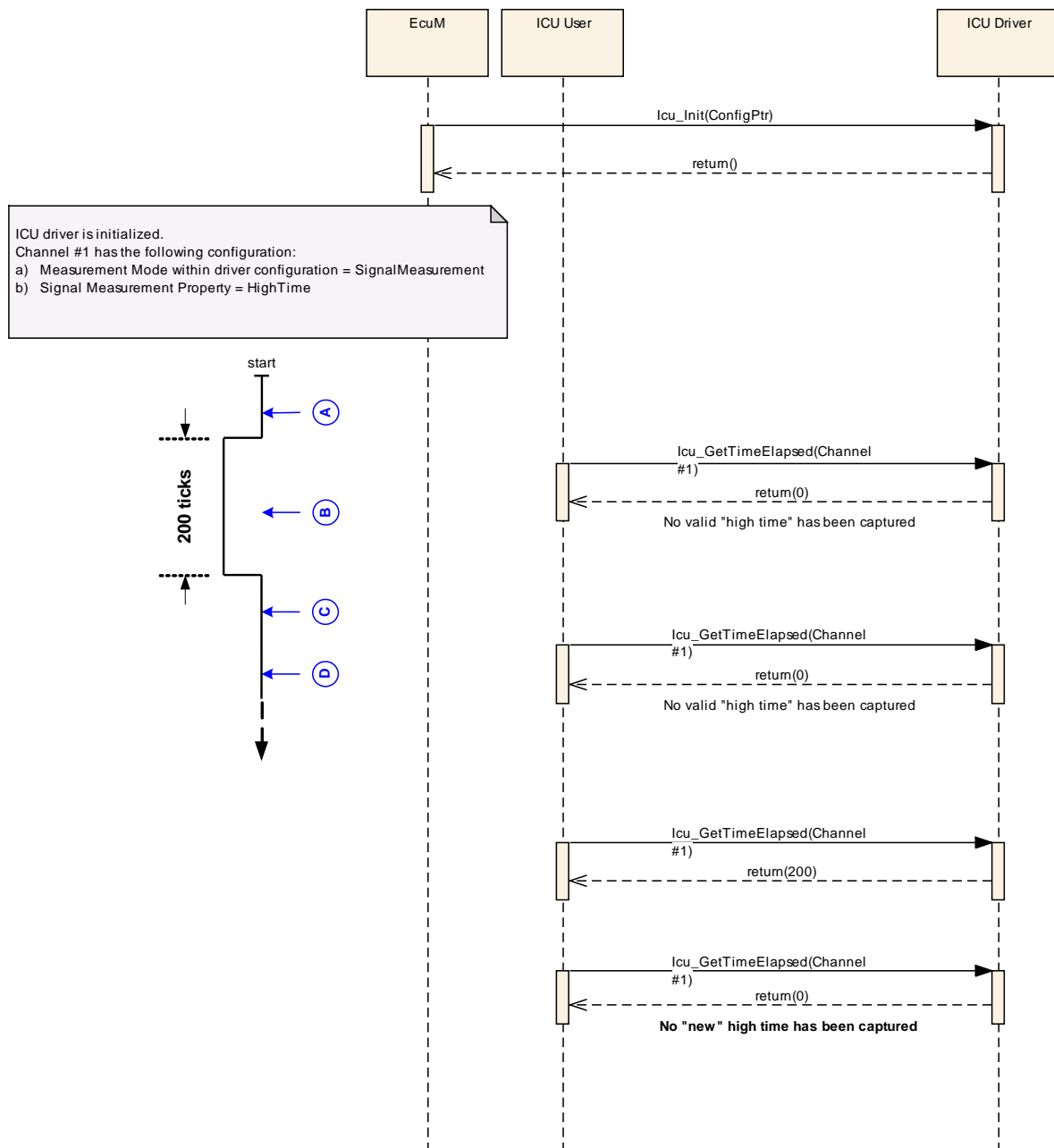


Figure 9.19, letter “A”).] ( )

**[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) (similar to

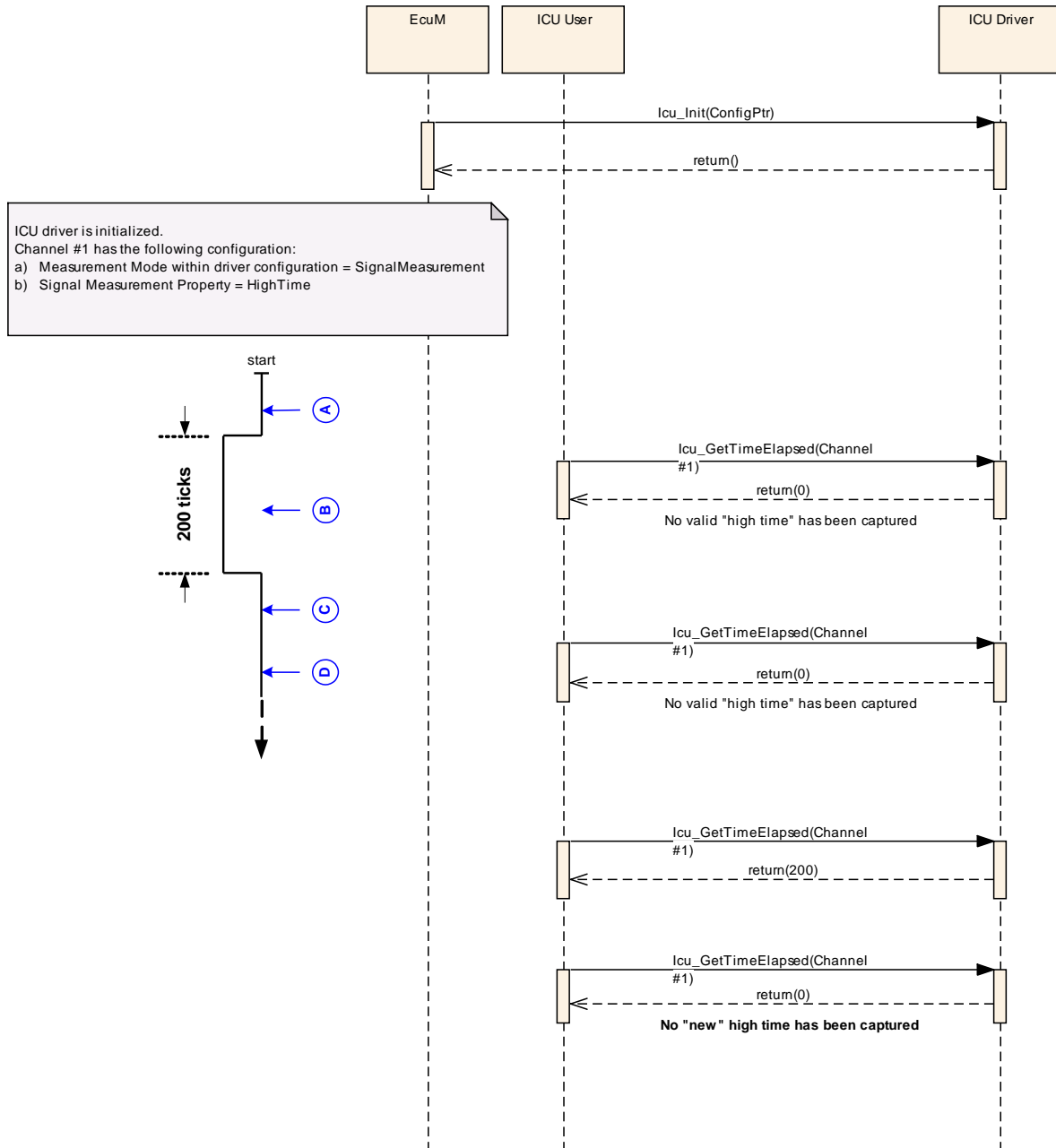


Figure 9.19, letter "B").] ()

**[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

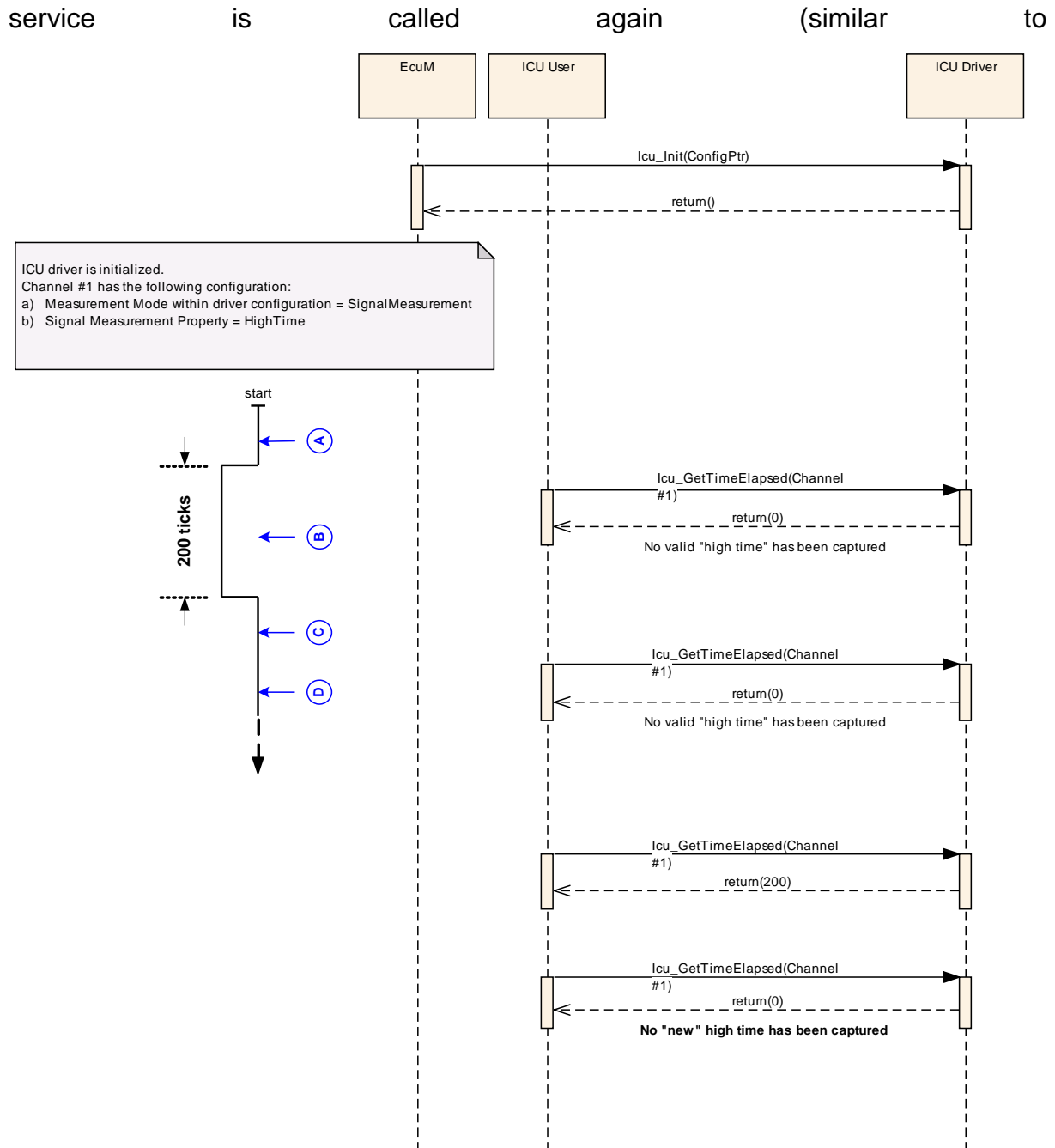


Figure 9.19, letter "D") ] ( )

**[SWS\_Icu\_00106]** [The function `Icu_GetDutyCycleValues` shall be pre compile time configurable by the configuration parameter `IcuGetDutyCycleValuesApi.`] (SRS\_BSW\_00410, SRS\_BSW\_00171)

**[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\\_00022](#) apply to the function `Icu_GetDutyCycleValues`.] ()

### 8.3.24 Icu\_GetVersionInfo

**[SWS\_Icu\_00212]** [

<b>Service name:</b>	<code>Icu_GetVersionInfo</code>
<b>Syntax:</b>	<code>void Icu_GetVersionInfo(     Std_VersionInfoType* versioninfo )</code>
<b>Service ID[hex]:</b>	0x12
<b>Sync/Async:</b>	Synchronous
<b>Reentrancy:</b>	Reentrant
<b>Parameters (in):</b>	None
<b>Parameters (inout):</b>	None
<b>Parameters (out):</b>	<code>versioninfo</code>   Pointer to where to store the version information of this module.
<b>Return value:</b>	None
<b>Description:</b>	This function returns the version information of this module.

] ()

**[SWS\_Icu\_00356]** [If development error detection for the Icu module is enabled: The function `Icu_GetVersionInfo` shall check the parameter `versioninfo` for not being `NULL` and shall raise the development error code `ICU_E_PARAM_VINFO` if the check fails.] ()

## 8.4 Callback notifications

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

## 8.5 Scheduled functions

None.

## 8.6 Expected Interfaces

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

### 8.6.1 Mandatory Interfaces

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

#### [SWS\_Icu\_91001] [

API function	Description
Det_ReportRuntimeError	Service to report runtime errors. If a callout has been configured then this callout shall be called.

] ()

### 8.6.2 Optional Interfaces

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

#### [SWS\_Icu\_00213] [

API function	Description
Det_ReportError	Service to report development errors.
EcuM_CheckWakeup	This callout is called by the EcuM to poll a wakeup source. It shall 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	Sets the wakeup event.

] ()

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

- **[SWS\_Icu\_00055]** [The static configuration parameter `IcuReportWakeupSource` is set to “ON” ] (SRS\_SPAL\_12069, SRS\_BSW\_00410)
- **[SWS\_Icu\_00056]** [The module is in mode `ICU_MODE_SLEEP`] (SRS\_SPAL\_12069)
- **[SWS\_Icu\_00057]** [A wakeup event occurs on a wakeup capable ICU channel.] (SRS\_SPAL\_12069)

**[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.] ()

### 8.6.3 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]** [The ISRs shall reset the interrupt flags (if needed by hardware) and call the corresponding notification functions.] (SRS\_SPAL\_12129)

**[SWS\_Icu\_00018]** [The Icu notification functions shall be configurable as function pointers within the initialization data structure (Icu\_ConfigType).] (SRS\_SPAL\_12056)

**[SWS\_Icu\_00187]** [The Icu module's notification functions shall have no parameters and no return value.] (SRS\_BSW\_00359)

**[SWS\_Icu\_00214]** [

<b>Service name:</b>	Icu_SignalNotification_<Channel>
<b>Syntax:</b>	void Icu_SignalNotification_<Channel>(void)
<b>Sync/Async:</b>	Synchronous
<b>Reentrancy:</b>	Reentrancy of interface not relevant for this module. (in general it is in this case not reentrant).
<b>Parameters (in):</b>	None
<b>Parameters (inout):</b>	None
<b>Parameters (out):</b>	None
<b>Return value:</b>	None
<b>Description:</b>	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).

] ()

**[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]** [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).] (SRS\_SPAL\_00157, SRS\_Icu\_12369)

**[SWS\_Icu\_00044]** [Only those edge notifications shall be provided, which are supported by hardware.] (SRS\_Icu\_12305)

**[SWS\_Icu\_00042]** [After a call of `Icu_DisableNotification`, the Icu module shall not call the notification function `Icu_SignalNotification_<Channel>`.] (SRS\_Icu\_12305)

**[SWS\_Icu\_00215]** [

<b>Service name:</b>	<code>Icu_TimestampNotification_&lt;Channel&gt;</code>
<b>Syntax:</b>	<code>void Icu_TimestampNotification_&lt;Channel&gt;(void)</code>
<b>Sync/Async:</b>	Synchronous
<b>Reentrancy:</b>	Reentrancy of interface not relevant for this module. (in general it is in this case not reentrant).
<b>Parameters (in):</b>	None
<b>Parameters (inout):</b>	None
<b>Parameters (out):</b>	None
<b>Return value:</b>	None
<b>Description:</b>	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 <code>Icu_EnableNotification()</code> .

] (SRS\_Icu\_12444)

**[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

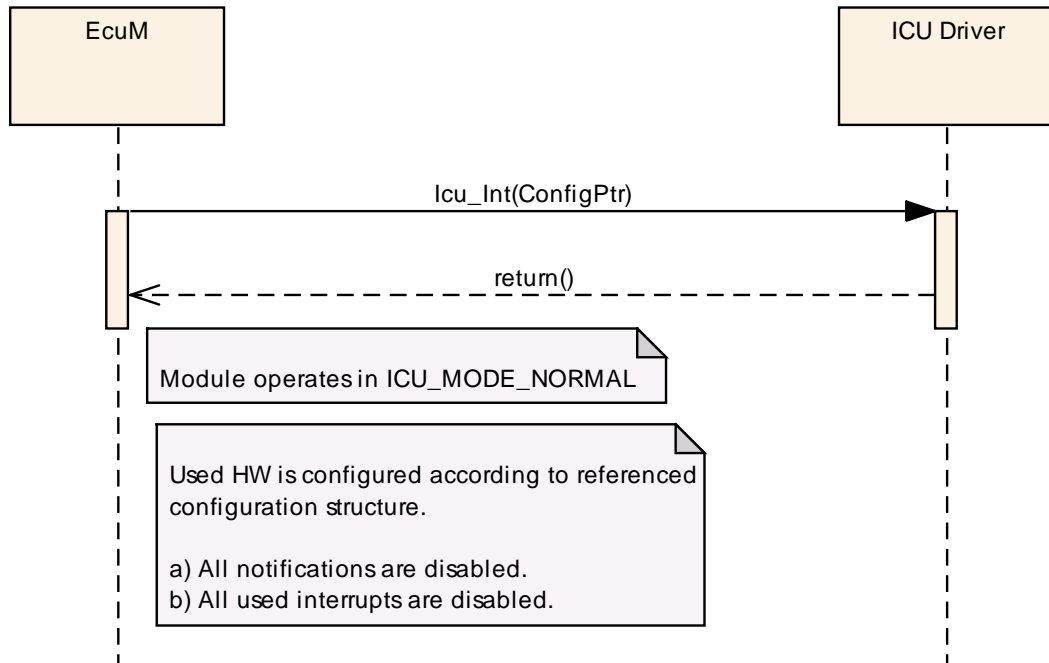


Figure 9.1: Initialization of the ICU driver

### 9.2 Icu\_DeInit

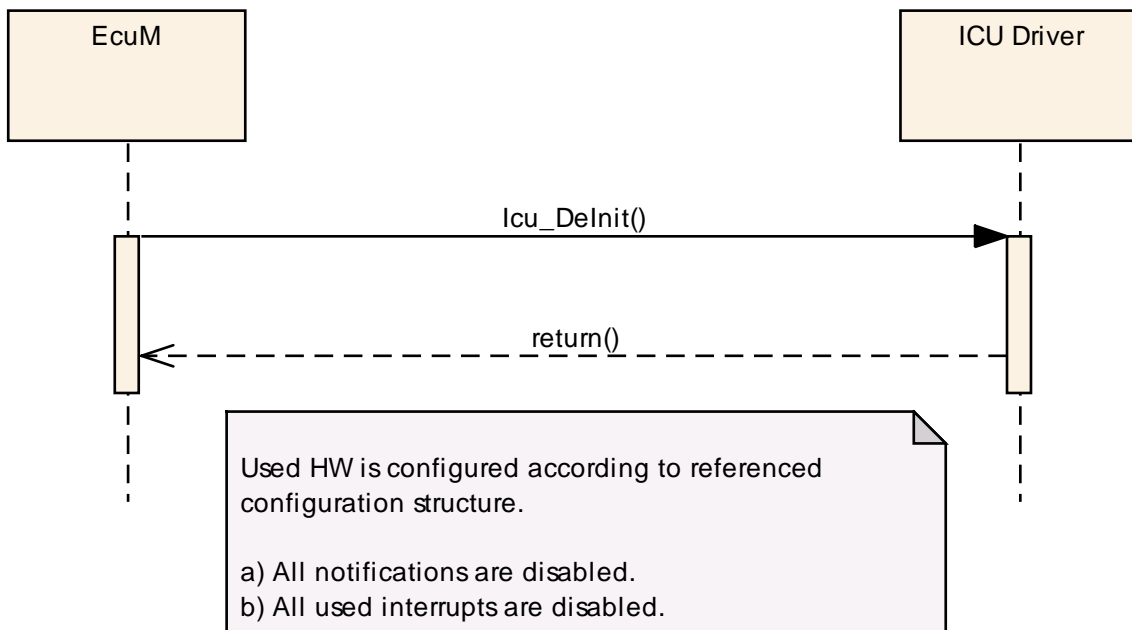


Figure 9.2: De-Initialization of the ICU driver



### 9.3 Check Wakeup Events

Note: The Sequence charts for the ICU can be found in the ECU State Manager specification [10]

**9.4 Icu\_SetMode**

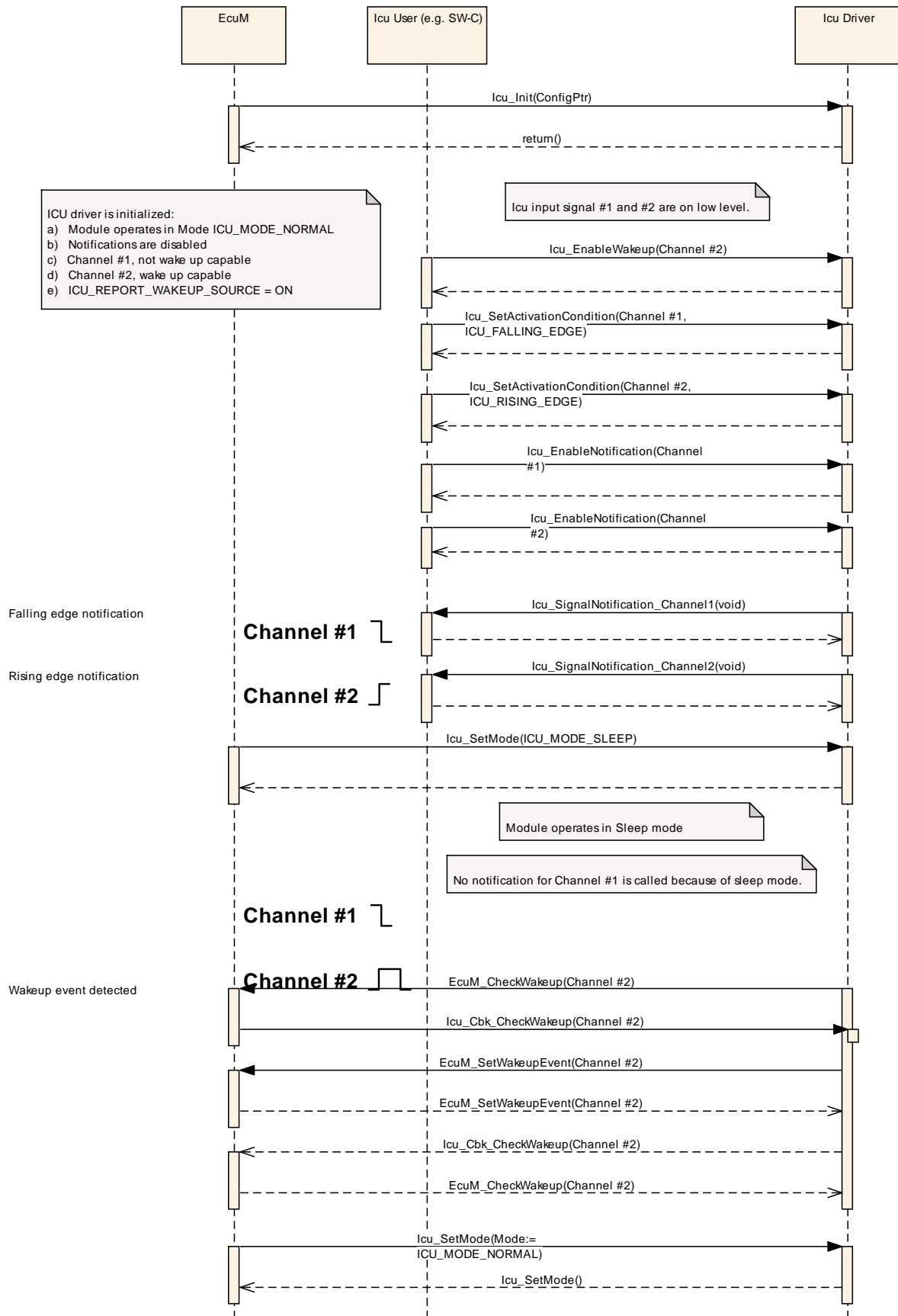


Figure 9.3: Enabled notifications in SLEEP mode

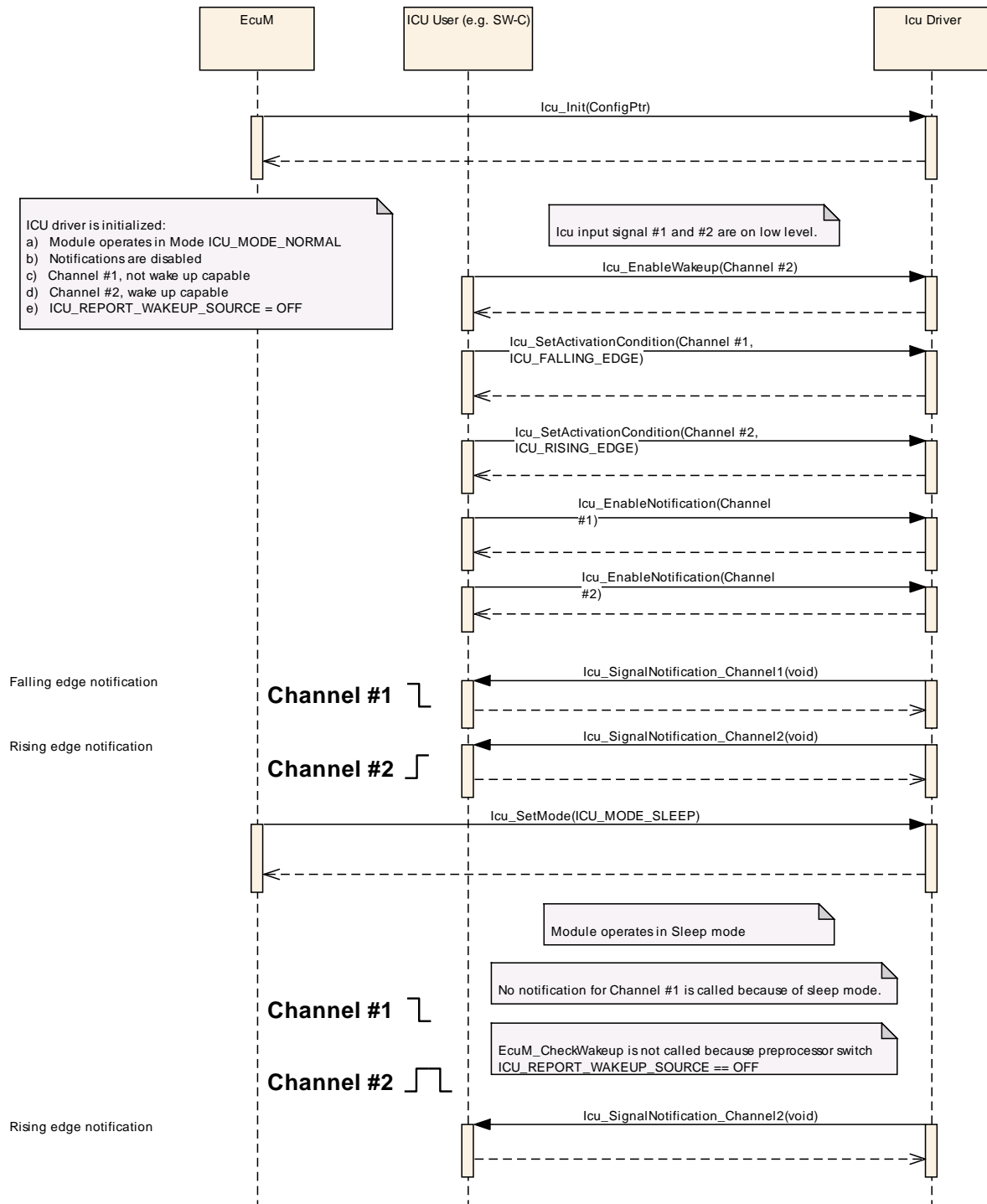


Figure 9.4: Disabled reporting of wakeup sources in SLEEP mode

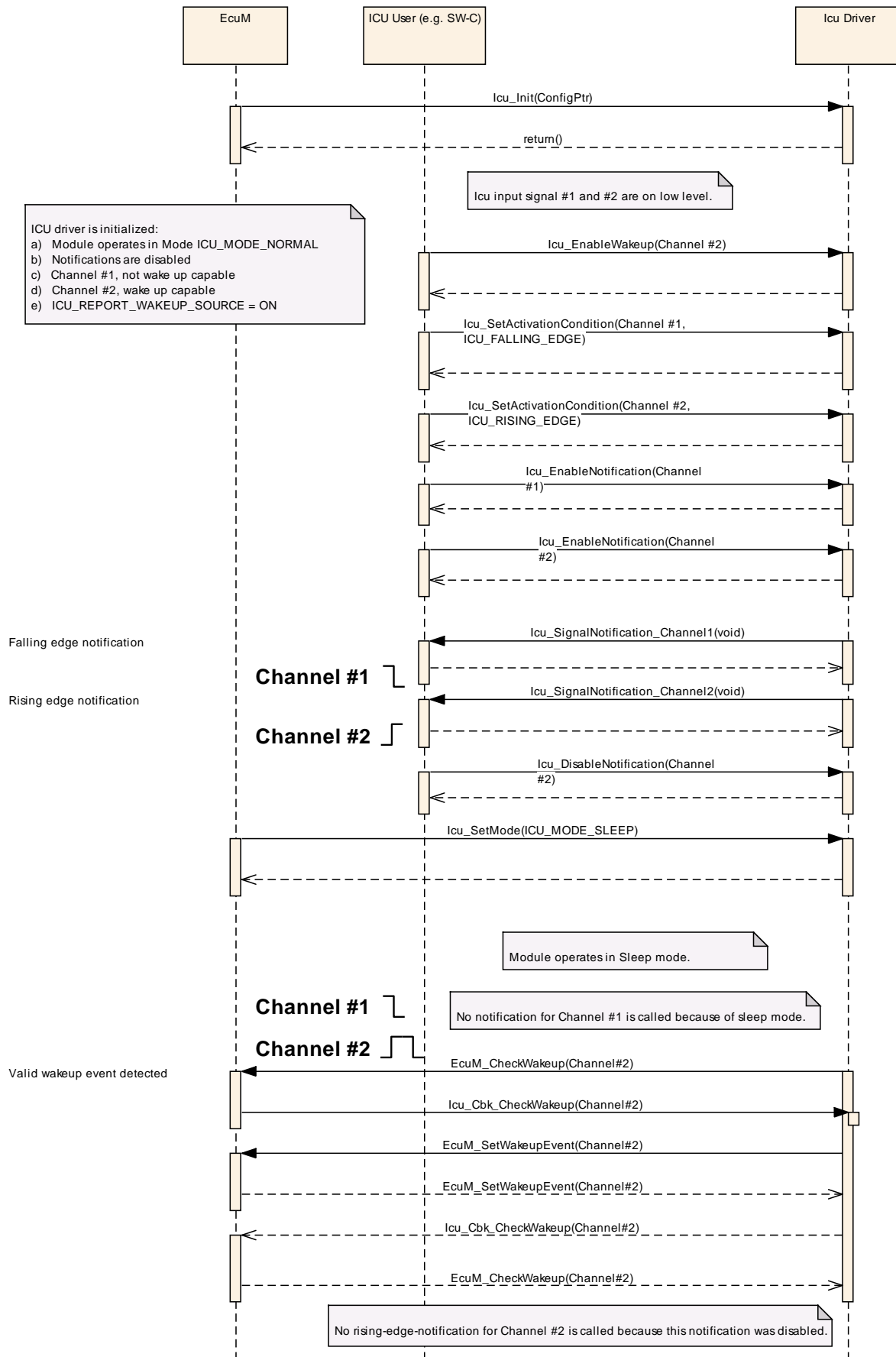


Figure 9.5: Disabled edge notification in SLEEP mode

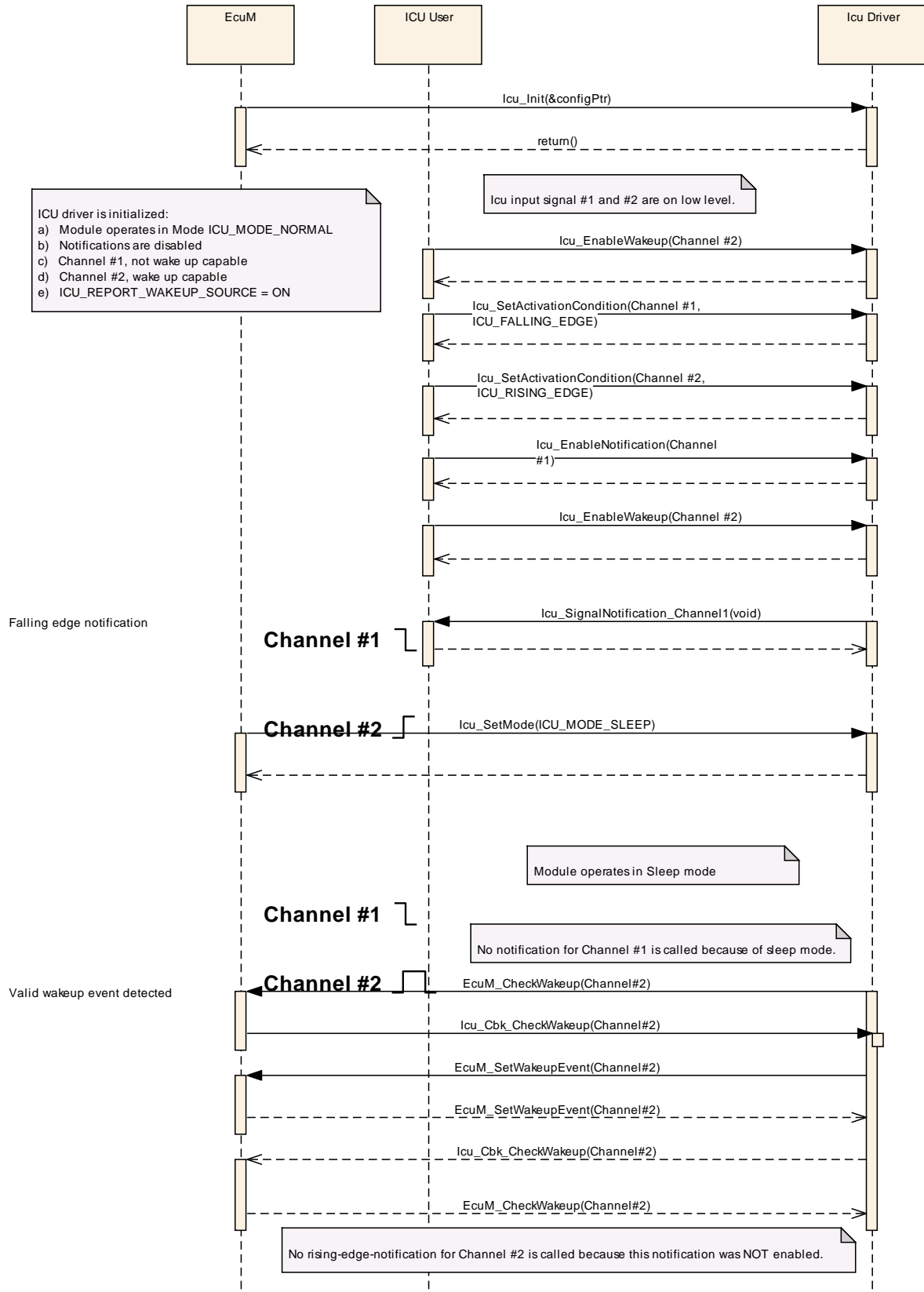


Figure 9.6: Un-Enabled reporting of notifications in SLEEP mode

### 9.5 Icu\_DisableWakeup

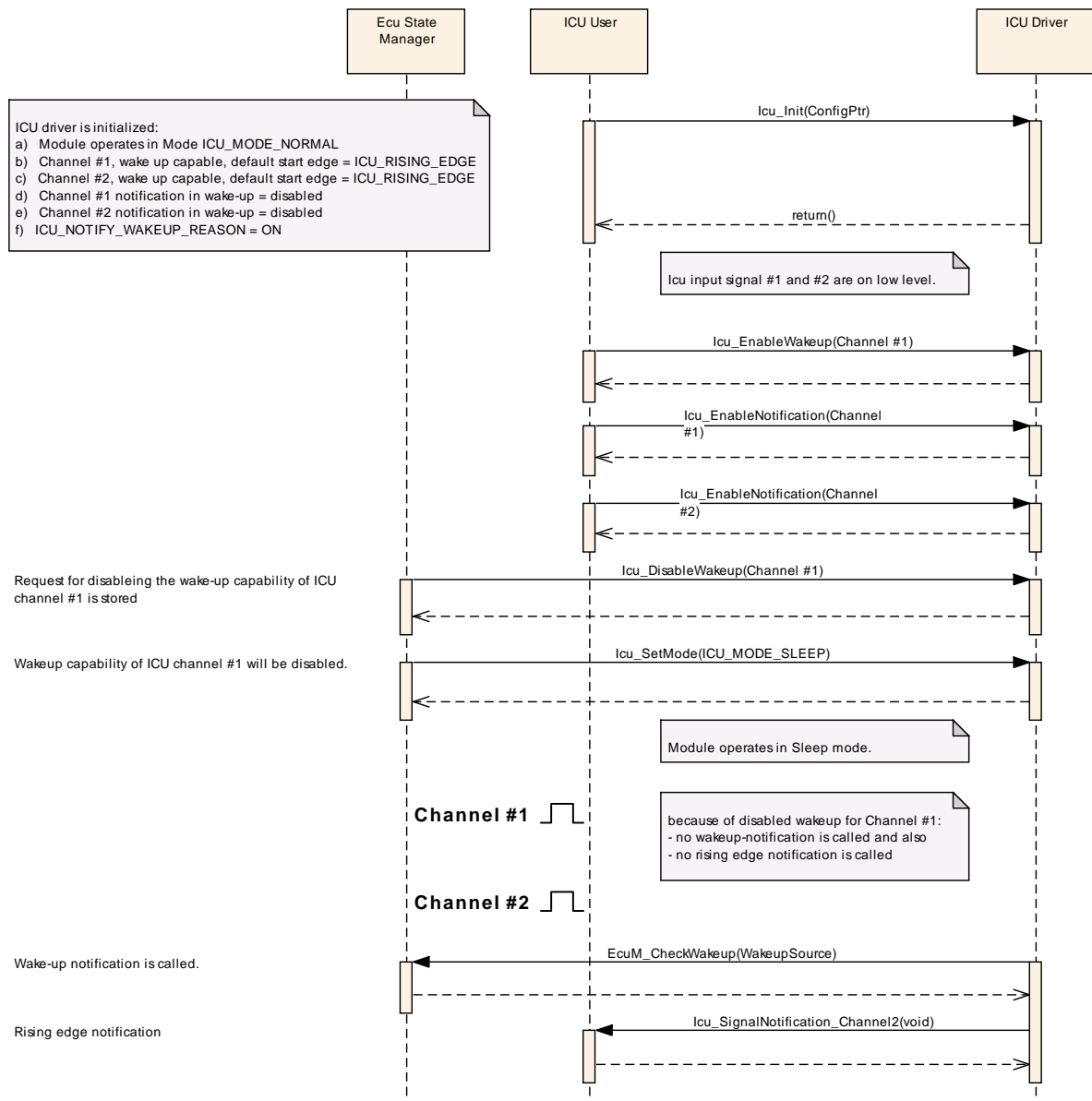


Figure 9.7: Disabling of wakeup-capabilities

### 9.6 Icu\_EnableWakeup

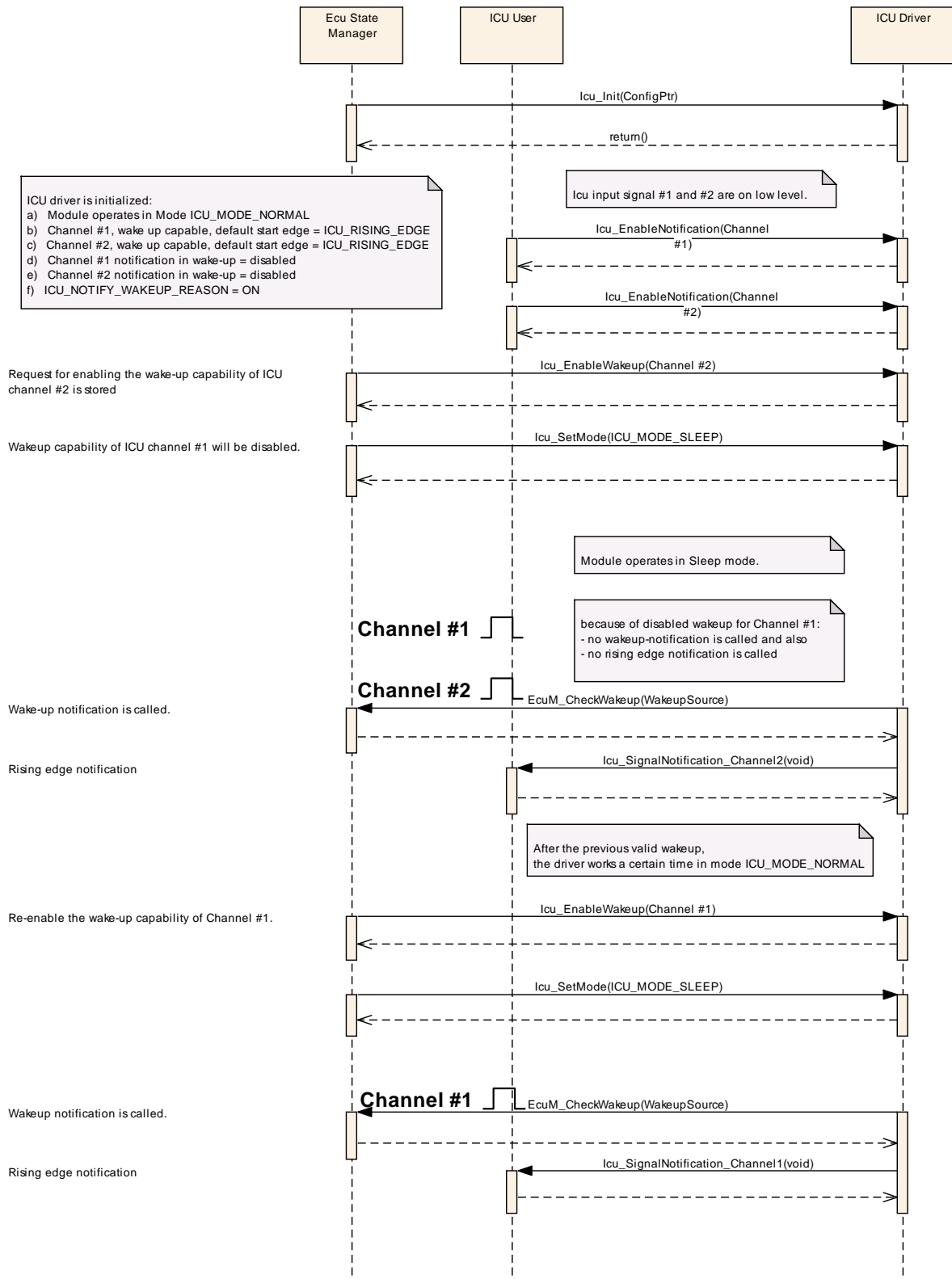


Figure 9.8: Enabling of wakeup-capabilities

### 9.7 Icu\_SetActivationCondition

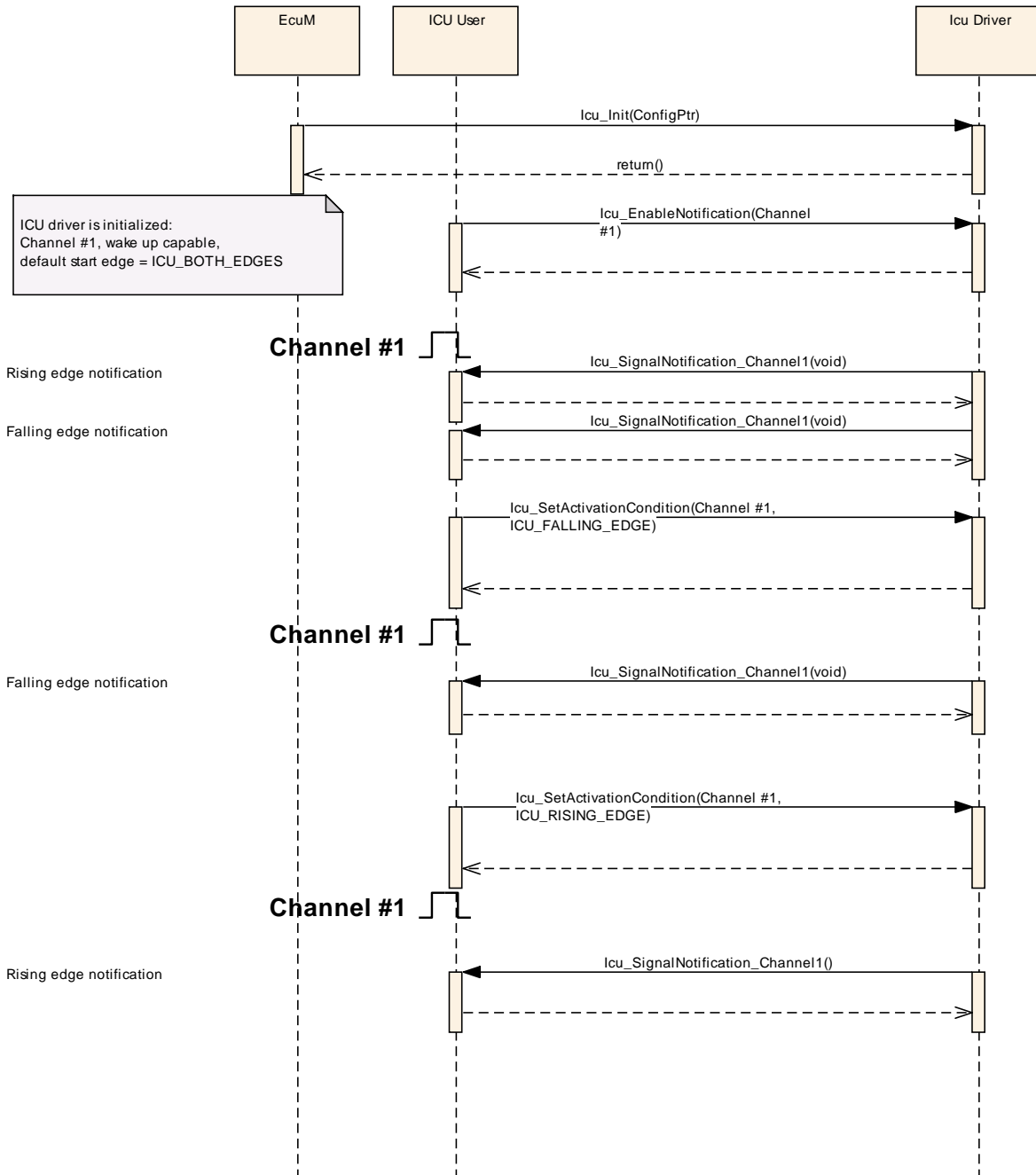


Figure 9.9: Setting up the activation condition for a channel



### 9.8 Icu\_DisableNotification

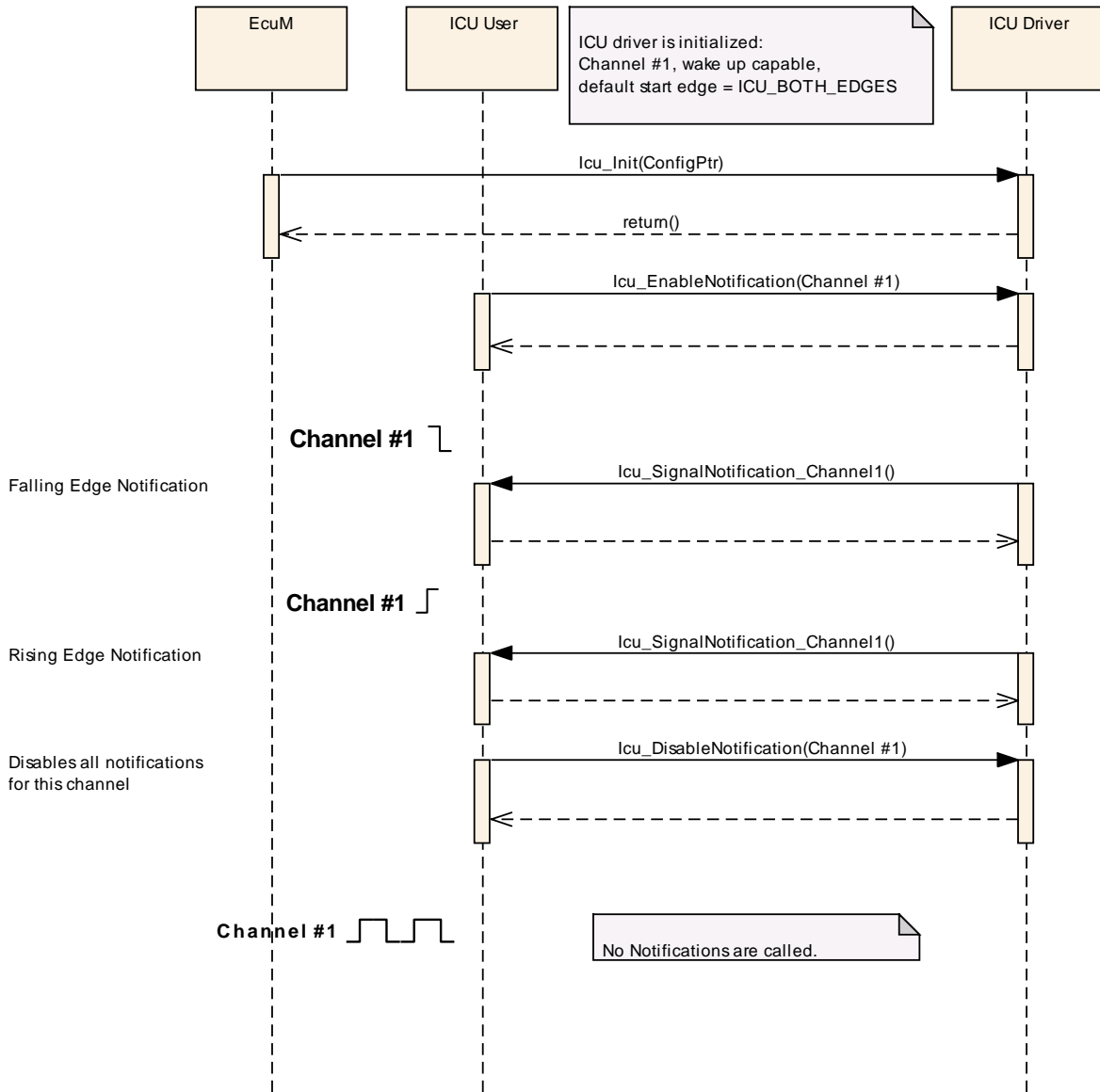


Figure 9.10: Disabling of the notification for a channel

### 9.9 Icu\_EnableNotification

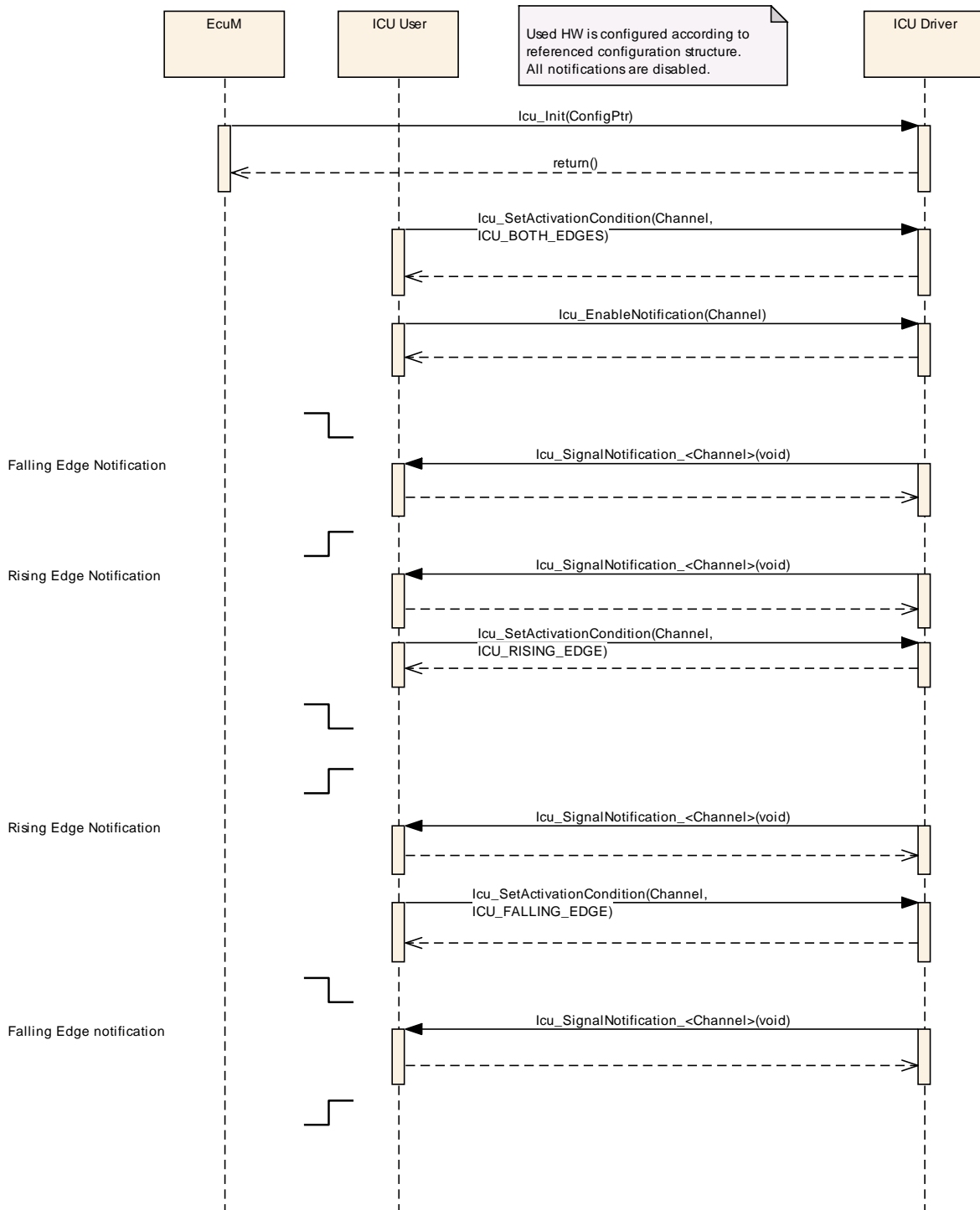


Figure 9.11: Enabling of the edge-notification for a channel

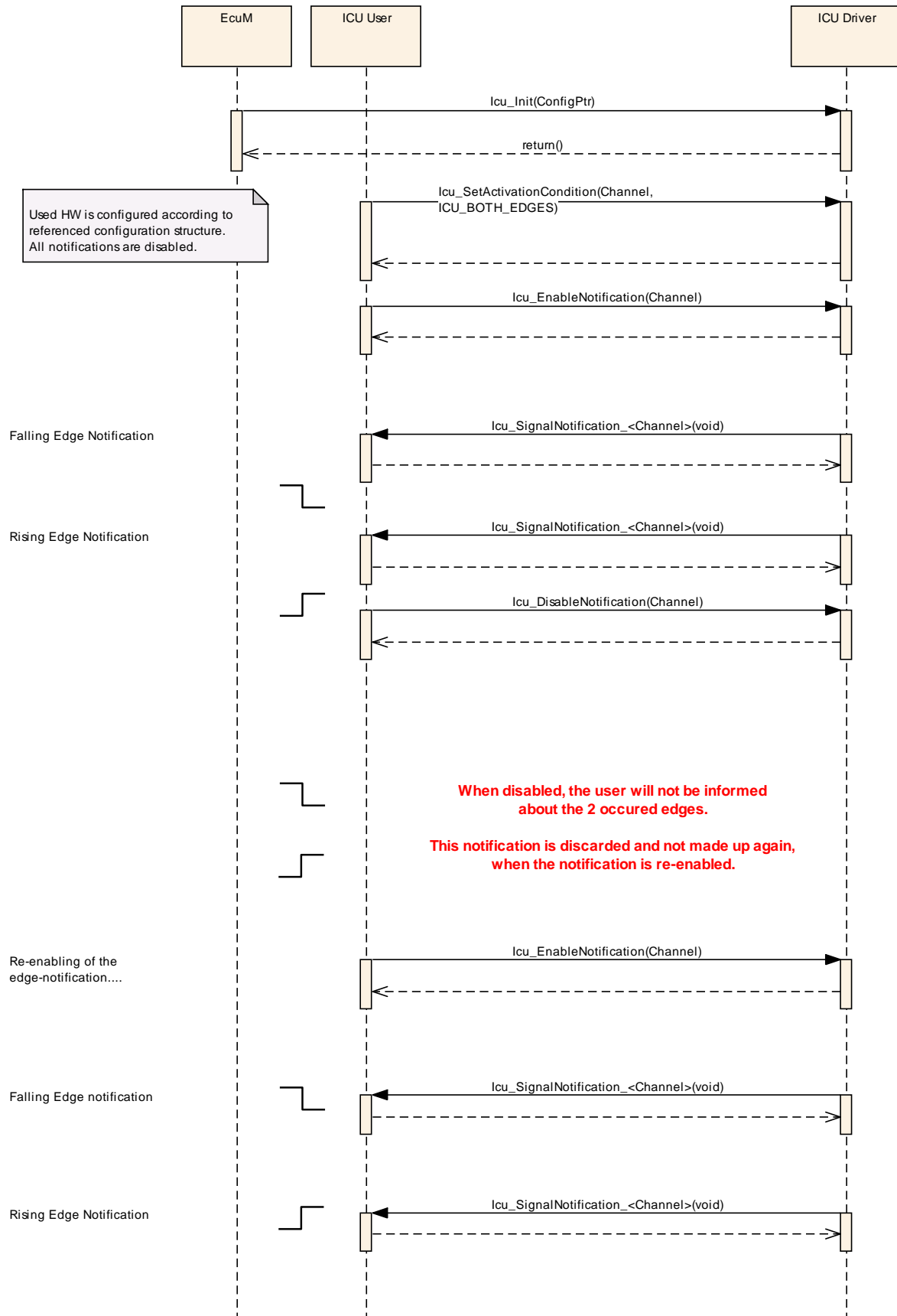


Figure 9.12: Re-enabling of the notification for a channel

### 9.10 Icu\_GetInputState

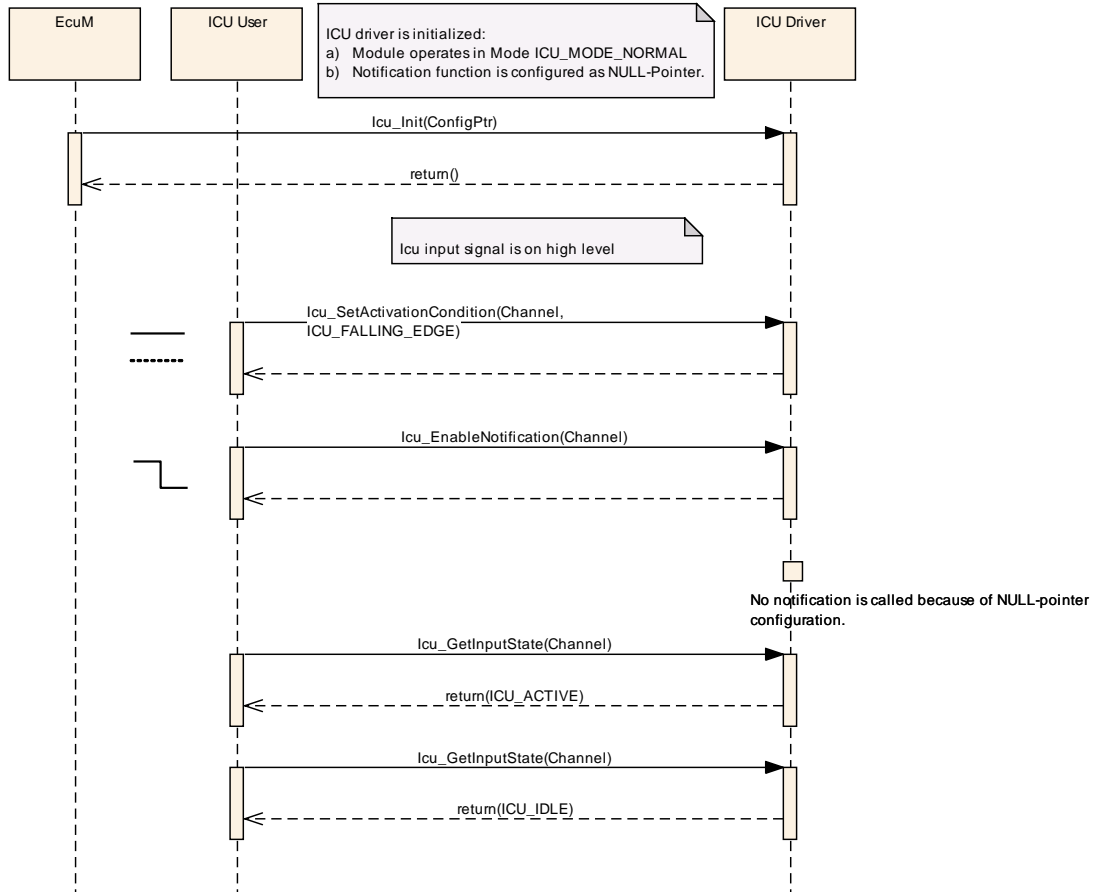


Figure 9.13: Polling of the channel status

### 9.11 Icu Timestamping

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

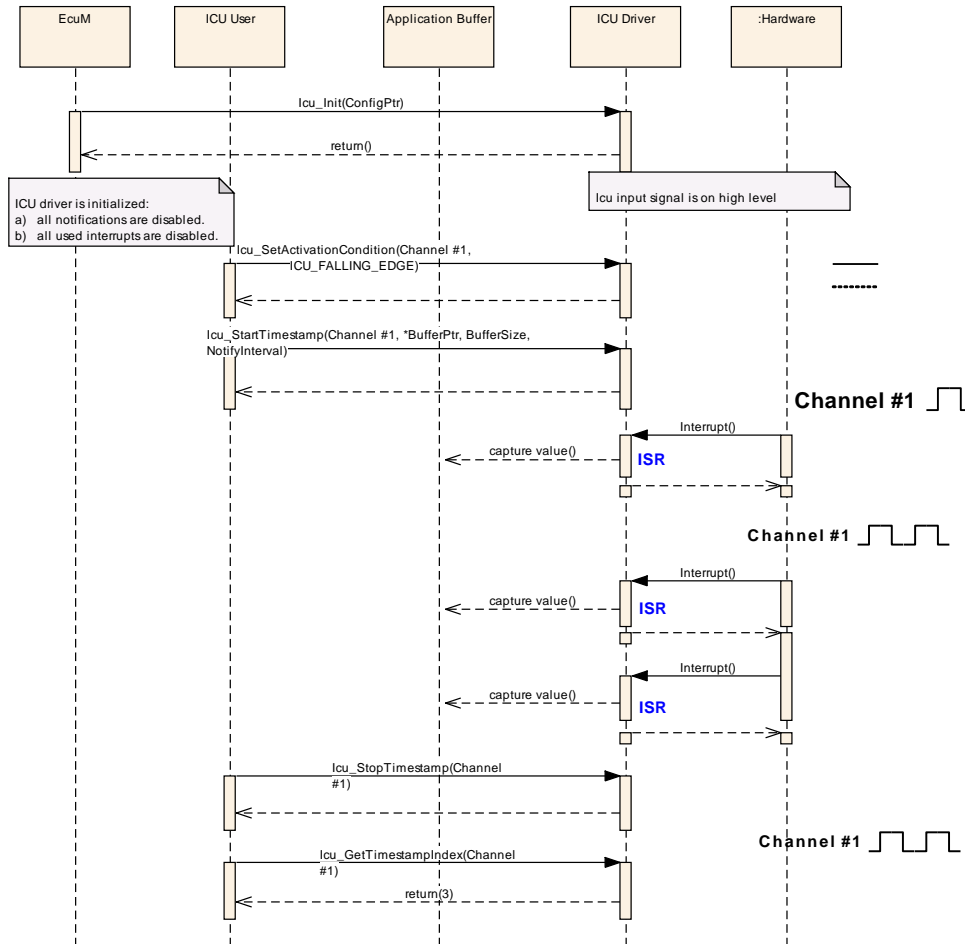


Figure 9.14: Overview of the timestamping functionality of the ICU driver

The Timestamping in general is shown in the following figure:

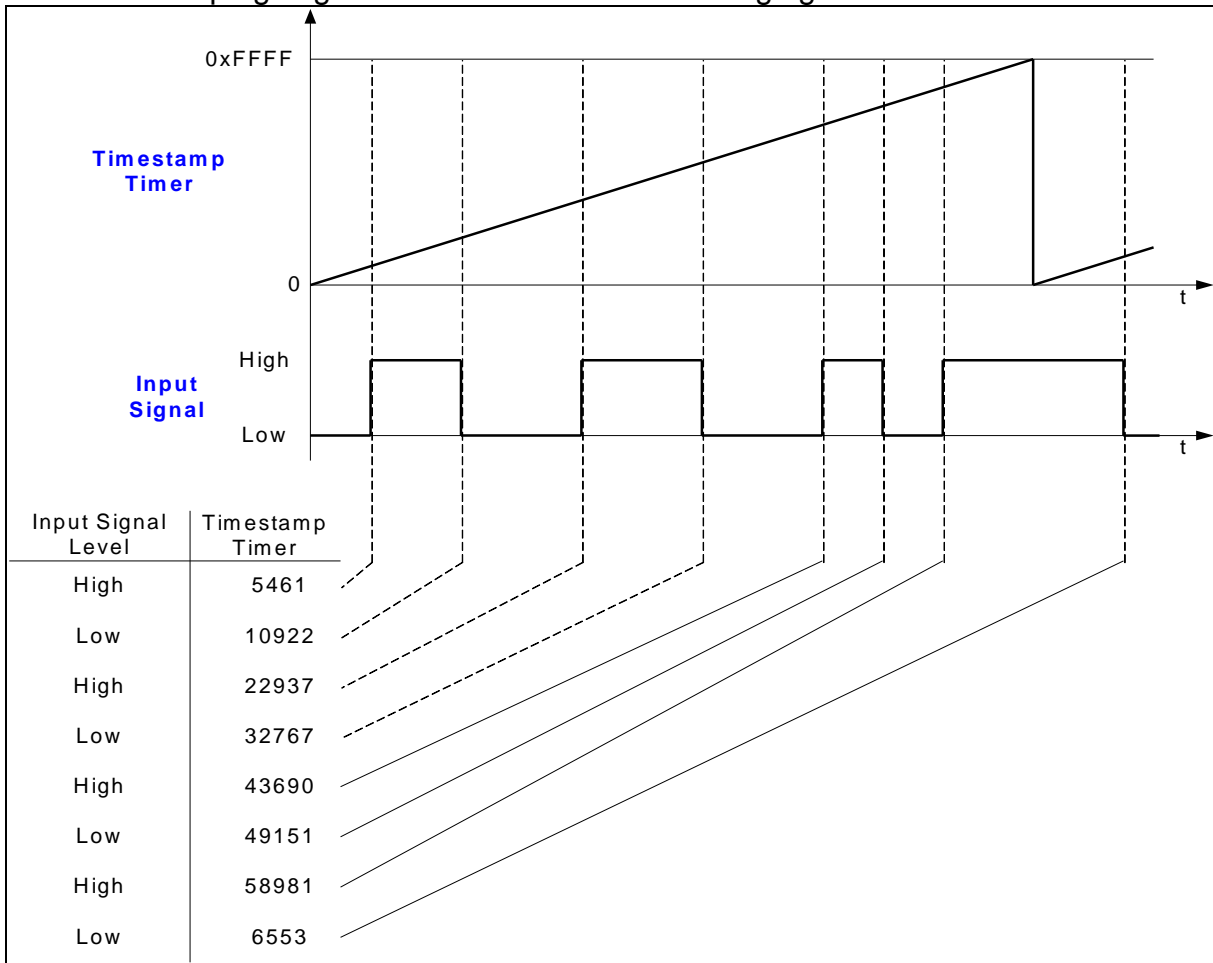


Figure 9.15: Timestamping overview

### 9.12 Icu Edge Counting

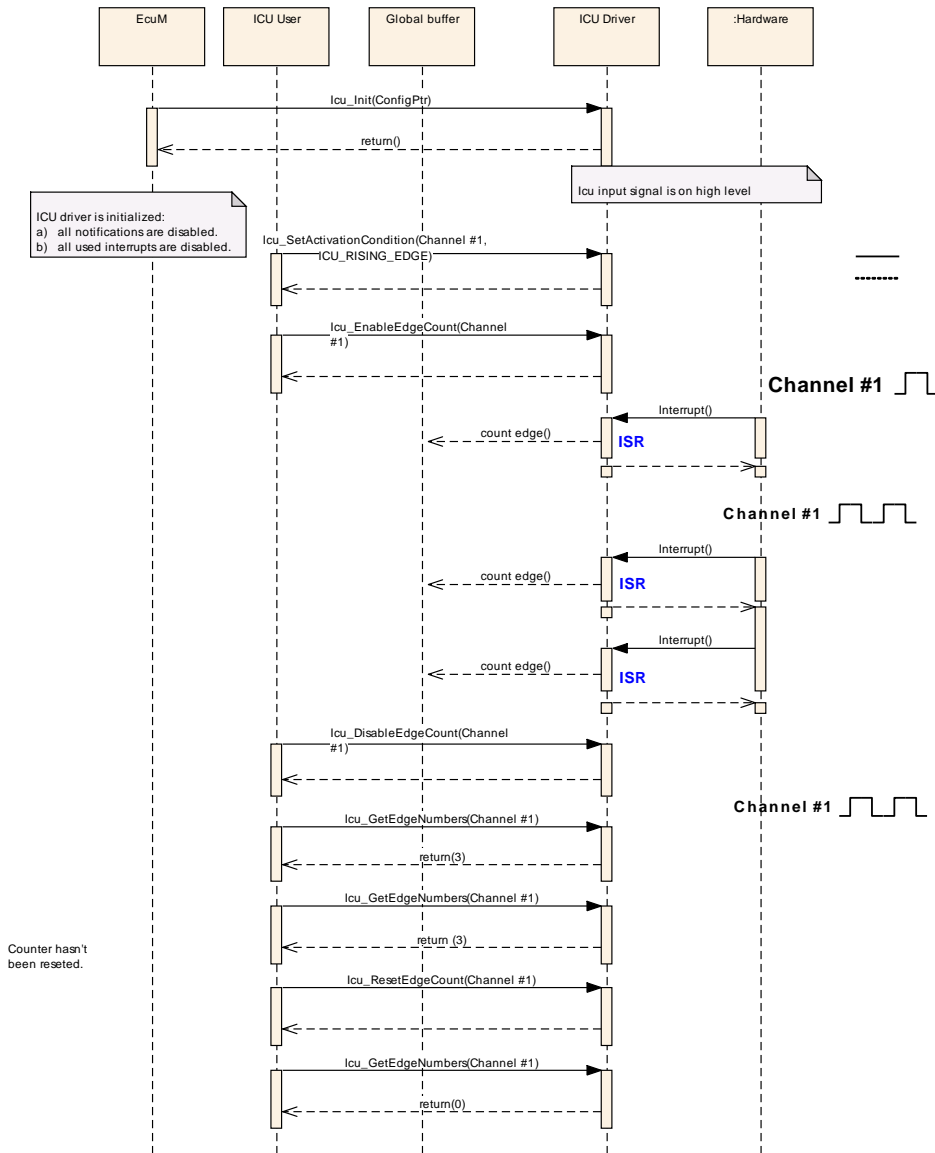


Figure 9.16: Inquire the number of counted edges

### 9.13 Icu\_GetTimeElapsed

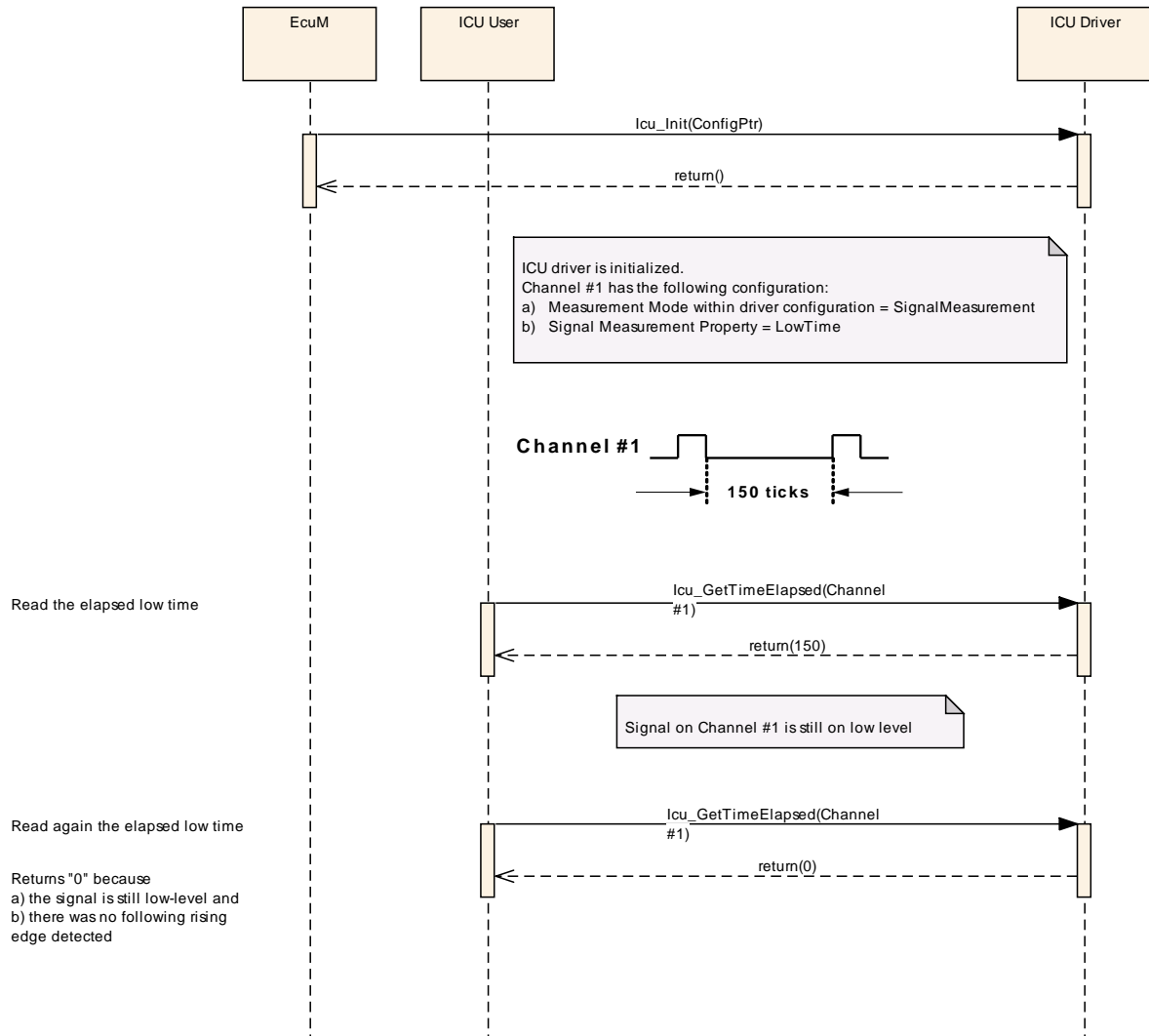


Figure 9.17: Inquire the elapsed level-time of a channel



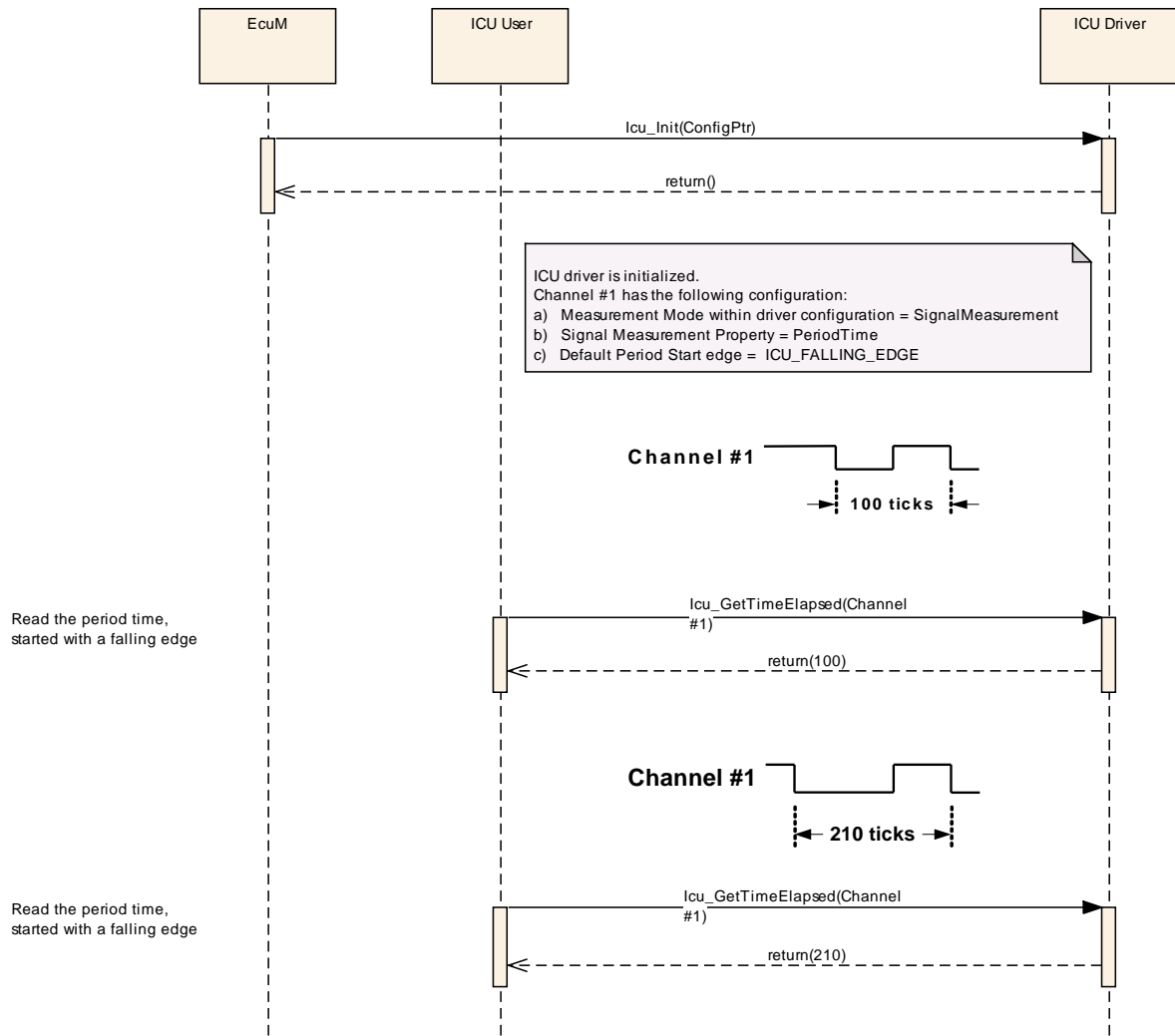


Figure 9.18: Inquire the elapsed period time of a channel

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()`.**

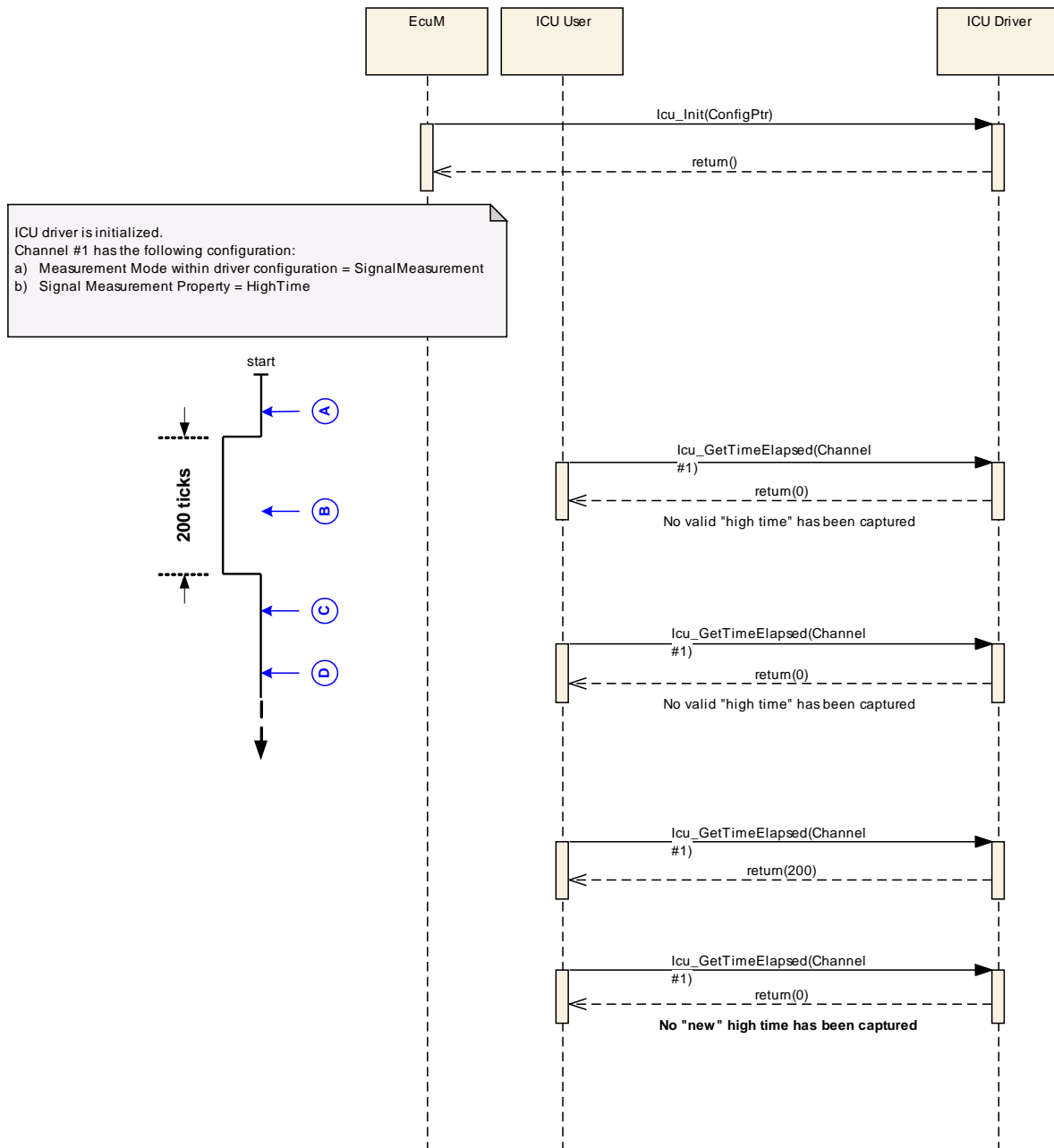


Figure 9.19: Inquire the elapsed high time of a channel

### 9.14 Icu\_GetDutyCycleValues

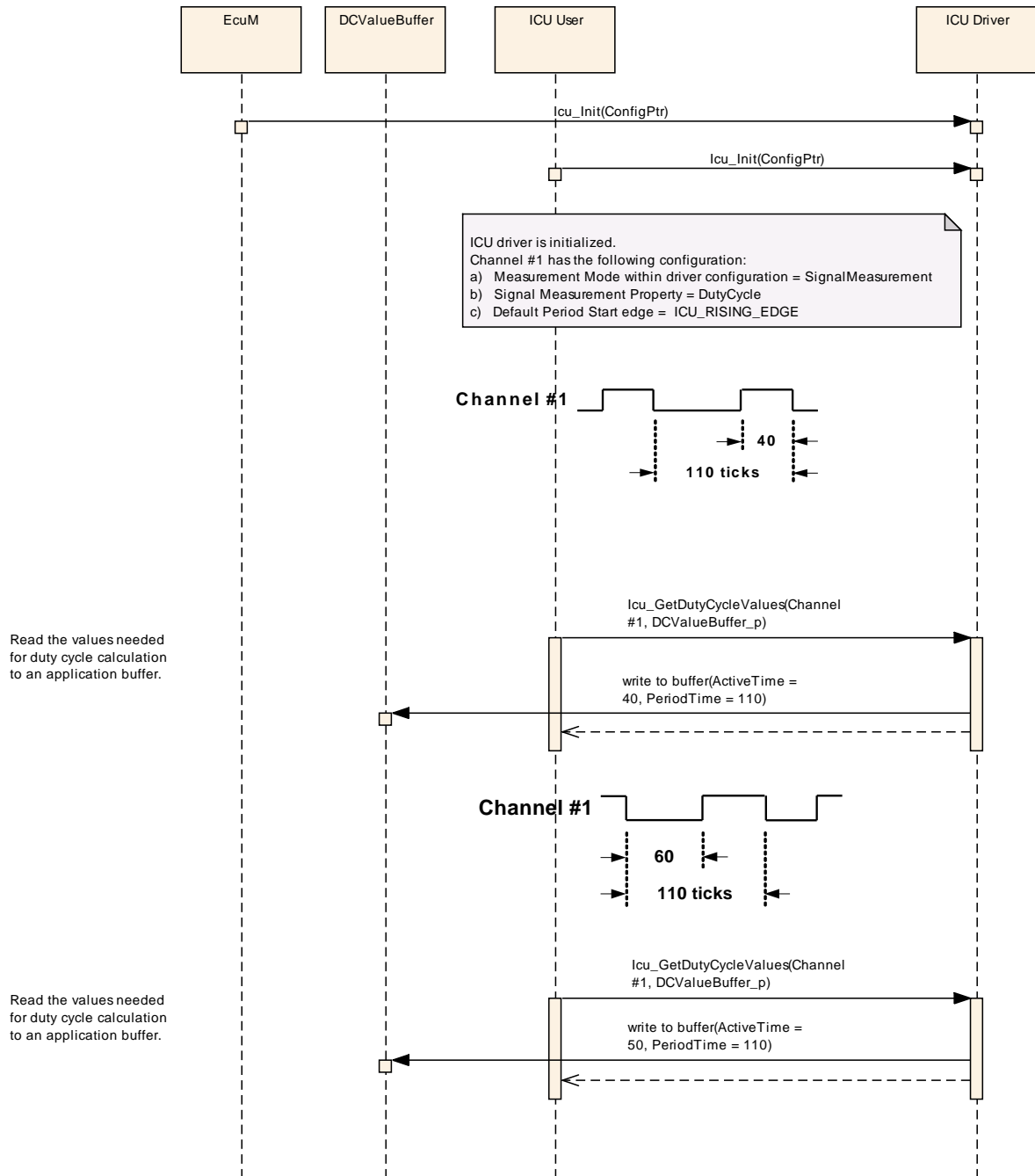


Figure 9.20: Measure the values needed for calculation of duty cycles

### 9.15 Icu\_SignalNotification and Icu\_GetInputState

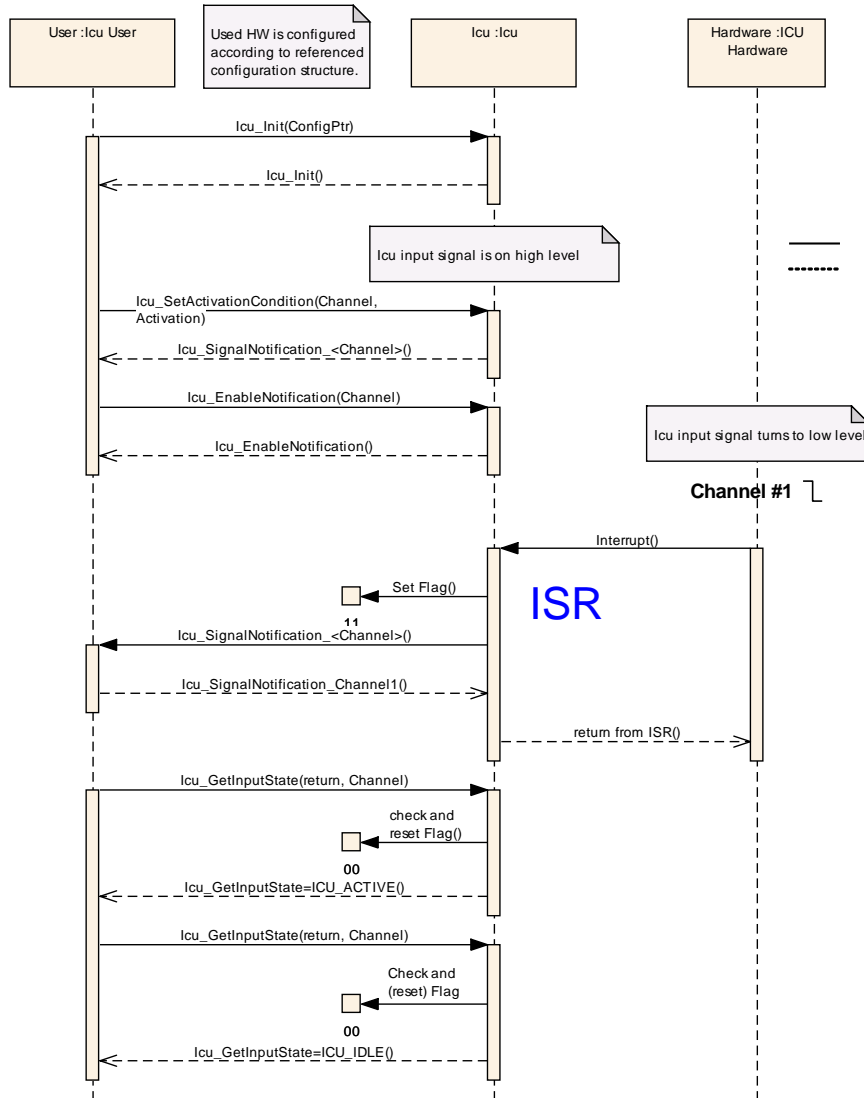


Figure 9.21: Cooperative usage of notification and polling mechanism

## 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 the chapter 10.1 “Introduction to configuration specification” in *SWS\_BSWGeneral*.

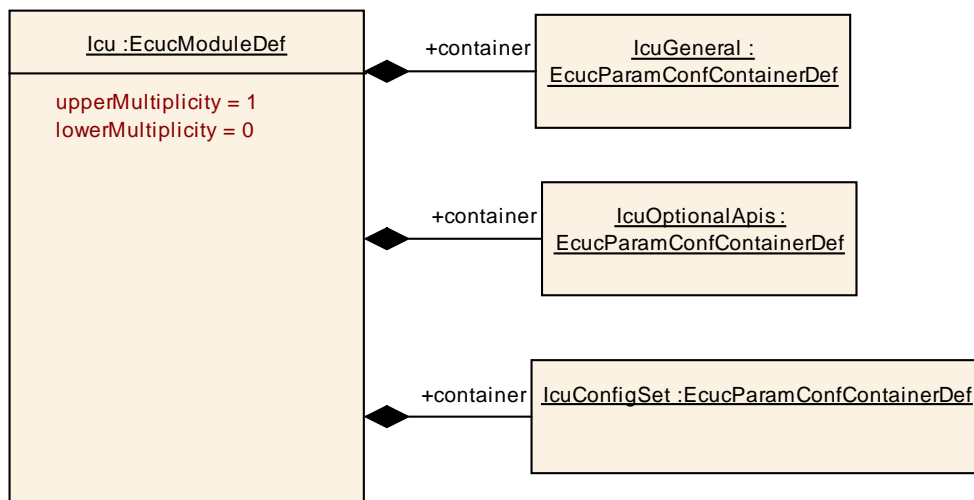
## 10.2 Containers and configuration parameters

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

### 10.2.1 Icu

<b>SWS Item</b>	<b>ECUC_Icu_00357 :</b>
<b>Module Name</b>	Icu
<b>Module Description</b>	Configuration of the Icu (Input Capture Unit) module.
<b>Post-Build Variant Support</b>	true
<b>Supported Config Variants</b>	VARIANT-POST-BUILD, VARIANT-PRE-COMPILE

Included Containers		
Container Name	Multiplicity	Scope / Dependency
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.



### 10.2.2 IcuGeneral

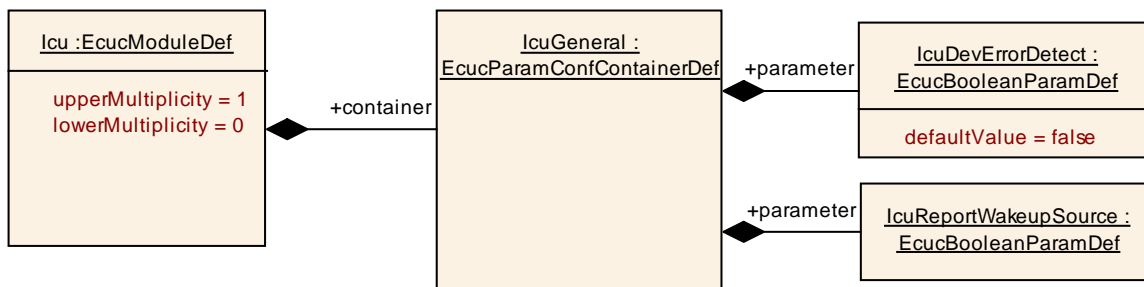
<b>SWS Item</b>	<b>ECUC_Icu_00026 :</b>
<b>Container Name</b>	IcuGeneral
<b>Description</b>	Configuration of general ICU parameters.
<b>Configuration Parameters</b>	

<b>SWS Item</b>	<b>ECUC_Icu_00232 :</b>
<b>Name</b>	IcuDevErrorDetect
<b>Parent Container</b>	IcuGeneral
<b>Description</b>	Switches the development error detection and notification on or off. <ul style="list-style-type: none"> <li>true: detection and notification is enabled.</li> </ul>

	<ul style="list-style-type: none"> <li>false: detection and notification is disabled.</li> </ul>		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucBooleanParamDef		
<b>Default value</b>	false		
<b>Post-Build Variant Value</b>	false		
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	All Variants
	<b>Link time</b>	--	
	<b>Post-build time</b>	--	
<b>Scope / Dependency</b>	scope: local		

<b>SWS Item</b>	<b>ECUC_Icu_00233 :</b>		
<b>Name</b>	IcuReportWakeupSource		
<b>Parent Container</b>	IcuGeneral		
<b>Description</b>	Switch for enabling Wakeup source reporting. true: Report Wakeup source. false: Do not report Wakeup source.		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucBooleanParamDef		
<b>Default value</b>	--		
<b>Post-Build Variant Value</b>	false		
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	All Variants
	<b>Link time</b>	--	
	<b>Post-build time</b>	--	
<b>Scope / Dependency</b>	scope: local		

**No Included Containers**



### 10.2.3 IcuOptionalApis

<b>SWS Item</b>	<b>ECUC_Icu_00114 :</b>
<b>Container Name</b>	IcuOptionalApis
<b>Description</b>	This container contains all configuration switches for configuring optional API services of the ICU driver.
<b>Configuration Parameters</b>	

<b>SWS Item</b>	<b>ECUC_Icu_00234 :</b>
<b>Name</b>	IcuDelnitApi
<b>Parent Container</b>	IcuOptionalApis
<b>Description</b>	Adds / removes the service Icu_Delnit() from the code. true: Icu_Delnit() can be used. false: Icu_Delnit() can not be used.
<b>Multiplicity</b>	1

<b>Type</b>	EcucBooleanParamDef		
<b>Default value</b>	--		
<b>Post-Build Variant Value</b>	false		
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	All Variants
	<b>Link time</b>	--	
	<b>Post-build time</b>	--	
<b>Scope / Dependency</b>	scope: local		

<b>SWS Item</b>	<b>ECUC_Icu_00235 :</b>		
<b>Name</b>	IcuDisableWakeupApi		
<b>Parent Container</b>	IcuOptionalApis		
<b>Description</b>	Adds / removes the service Icu_DisableWakeup() from the code. true: Icu_DisableWakeup() can be used. false: Icu_DisableWakeup() can not be used.		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucBooleanParamDef		
<b>Default value</b>	--		
<b>Post-Build Variant Value</b>	false		
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	All Variants
	<b>Link time</b>	--	
	<b>Post-build time</b>	--	
<b>Scope / Dependency</b>	scope: local		

<b>SWS Item</b>	<b>ECUC_Icu_00124 :</b>		
<b>Name</b>	IcuEdgeCountApi		
<b>Parent Container</b>	IcuOptionalApis		
<b>Description</b>	Adds / removes all services related to the edge counting functionality as listed below, from the code: Icu_ResetEdgeCount(), Icu_EnableEdgeCount(), Icu_DisableEdgeCount(), Icu_GetEdgeNumbers(). true: The services listed above can be used. false: The services listed above can not be used.		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucBooleanParamDef		
<b>Default value</b>	--		
<b>Post-Build Variant Value</b>	false		
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	All Variants
	<b>Link time</b>	--	
	<b>Post-build time</b>	--	
<b>Scope / Dependency</b>	scope: local		

<b>SWS Item</b>	<b>ECUC_Icu_00356 :</b>		
<b>Name</b>	IcuEdgeDetectApi		
<b>Parent Container</b>	IcuOptionalApis		
<b>Description</b>	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.		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucBooleanParamDef		
<b>Default value</b>	--		
<b>Post-Build Variant Value</b>	false		
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	All Variants
	<b>Link time</b>	--	



	<b>Post-build time</b>	--	
<b>Scope / Dependency</b>	scope: local		

<b>SWS Item</b>	<b>ECUC_Icu_00236 :</b>		
<b>Name</b>	IcuEnableWakeupApi		
<b>Parent Container</b>	IcuOptionalApis		
<b>Description</b>	Adds / removes the service Icu_EnableWakeup() from the code. true: Icu_EnableWakeup() can be used. false: Icu_EnableWakeup() can not be used.		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucBooleanParamDef		
<b>Default value</b>	--		
<b>Post-Build Variant Value</b>	false		
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	All Variants
	<b>Link time</b>	--	
	<b>Post-build time</b>	--	
<b>Scope / Dependency</b>	scope: local		

<b>SWS Item</b>	<b>ECUC_Icu_00237 :</b>		
<b>Name</b>	IcuGetDutyCycleValuesApi		
<b>Parent Container</b>	IcuOptionalApis		
<b>Description</b>	Adds / removes the service Icu_GetDutyCycleValues() from the code. true: Icu_GetDutyCycleValues() can be used. false: Icu_GetDutyCycleValues() can not be used.		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucBooleanParamDef		
<b>Default value</b>	--		
<b>Post-Build Variant Value</b>	false		
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	All Variants
	<b>Link time</b>	--	
	<b>Post-build time</b>	--	
<b>Scope / Dependency</b>	scope: local dependency: If IcuSignalMeasurementApi==false this switch shall also be set to false.		

<b>SWS Item</b>	<b>ECUC_Icu_00238 :</b>		
<b>Name</b>	IcuGetInputStateApi		
<b>Parent Container</b>	IcuOptionalApis		
<b>Description</b>	Adds / removes the service Icu_GetInputState() from the code. true: Icu_GetInputState() can be used. false: Icu_GetInputState() can not be used.		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucBooleanParamDef		
<b>Default value</b>	--		
<b>Post-Build Variant Value</b>	false		
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	All Variants
	<b>Link time</b>	--	
	<b>Post-build time</b>	--	
<b>Scope / Dependency</b>	scope: local		

<b>SWS Item</b>	<b>ECUC_Icu_00239 :</b>		
<b>Name</b>	IcuGetTimeElapsedApi		
<b>Parent Container</b>	IcuOptionalApis		
<b>Description</b>	Adds / removes the service Icu_GetTimeElapsed() from the code. true: Icu_GetTimeElapsed() can be used. false: Icu_GetTimeElapsed() can not be used.		

<b>Multiplicity</b>	1		
<b>Type</b>	EcucBooleanParamDef		
<b>Default value</b>	--		
<b>Post-Build Variant Value</b>	false		
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	All Variants
	<b>Link time</b>	--	
	<b>Post-build time</b>	--	
<b>Scope / Dependency</b>	scope: local dependency: If IcuSignalMeasurementApi==false this switch shall also be set to false.		

<b>SWS Item</b>	<b>ECUC_Icu_00240 :</b>		
<b>Name</b>	IcuGetVersionInfoApi		
<b>Parent Container</b>	IcuOptionalApis		
<b>Description</b>	Adds / removes the service Icu_GetVersionInfo() from the code. true: Icu_GetVersionInfo() can be used. false: Icu_GetVersionInfo() can not be used.		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucBooleanParamDef		
<b>Default value</b>	false		
<b>Post-Build Variant Value</b>	false		
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	All Variants
	<b>Link time</b>	--	
	<b>Post-build time</b>	--	
<b>Scope / Dependency</b>	scope: local		

<b>SWS Item</b>	<b>ECUC_Icu_00241 :</b>		
<b>Name</b>	IcuSetModeApi		
<b>Parent Container</b>	IcuOptionalApis		
<b>Description</b>	Adds / removes the service Icu_SetMode() from the code. true: Icu_SetMode() can be used. false: Icu_SetMode() can not be used.		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucBooleanParamDef		
<b>Default value</b>	--		
<b>Post-Build Variant Value</b>	false		
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	All Variants
	<b>Link time</b>	--	
	<b>Post-build time</b>	--	
<b>Scope / Dependency</b>	scope: local		

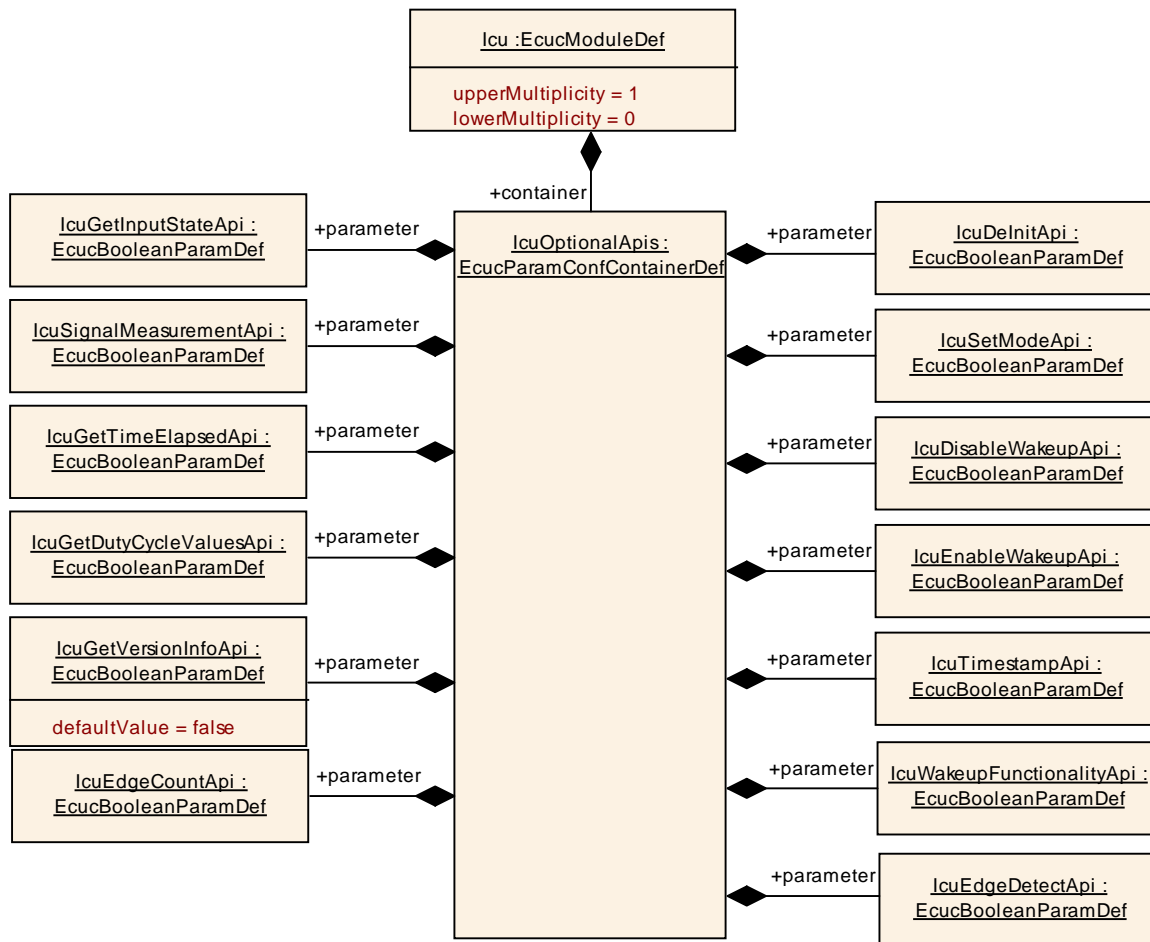
<b>SWS Item</b>	<b>ECUC_Icu_00242 :</b>		
<b>Name</b>	IcuSignalMeasurementApi		
<b>Parent Container</b>	IcuOptionalApis		
<b>Description</b>	Adds / removes the services Icu_StartSignalMeasurement() and Icu_StopSignalMeasurement() from the code. true: Icu_StartSignalMeasurement() and Icu_StopSignalMeasurement() can be used. false: Icu_StartSignalMeasurement() and Icu_StopSignalMeasurement() can not be used.		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucBooleanParamDef		
<b>Default value</b>	--		
<b>Post-Build Variant Value</b>	false		
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	All Variants
	<b>Link time</b>	--	

	<b>Post-build time</b>	--	
<b>Scope / Dependency</b>	scope: local		

<b>SWS Item</b>	<b>ECUC_Icu_00123 :</b>		
<b>Name</b>	IcuTimestampApi		
<b>Parent Container</b>	IcuOptionalApis		
<b>Description</b>	Adds / removes all services related to the timestamping functionality as listed below from the code: Icu_StartTimestamp(), Icu_StopTimestamp(), Icu_GetTimestampIndex(). true: The services listed above can be used. false: The services listed above can not be used.		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucBooleanParamDef		
<b>Default value</b>	--		
<b>Post-Build Variant Value</b>	false		
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	All Variants
	<b>Link time</b>	--	
	<b>Post-build time</b>	--	
<b>Scope / Dependency</b>	scope: local		

<b>SWS Item</b>	<b>ECUC_Icu_00355 :</b>		
<b>Name</b>	IcuWakeupFunctionalityApi		
<b>Parent Container</b>	IcuOptionalApis		
<b>Description</b>	Adds / removes the service Icu_CheckWakeup() from the code. true: Icu_CheckWakeup() can be used. false: Icu_CheckWakeup() cannot be used.		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucBooleanParamDef		
<b>Default value</b>	--		
<b>Post-Build Variant Value</b>	false		
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	All Variants
	<b>Link time</b>	--	
	<b>Post-build time</b>	--	
<b>Scope / Dependency</b>	scope: local		

**No Included Containers**



### 10.2.4 IcuChannel

<b>SWS Item</b>	<b>ECUC_Icu_00027 :</b>
<b>Container Name</b>	IcuChannel
<b>Description</b>	Configuration of an individual ICU channel.
<b>Configuration Parameters</b>	

<b>SWS Item</b>	<b>ECUC_Icu_00354 :</b>		
<b>Name</b>	IcuChannelId		
<b>Parent Container</b>	IcuChannel		
<b>Description</b>	Channel Id of the ICU channel. This value will be assigned to the symbolic name derived of the IcuChannel container short name.		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucIntegerParamDef (Symbolic Name generated for this parameter)		
<b>Range</b>	0 .. 65535		
<b>Default value</b>	--		
<b>Post-Build Variant Value</b>	false		
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	All Variants
	<b>Link time</b>	--	
	<b>Post-build time</b>	--	
<b>Scope / Dependency</b>	scope: ECU		

<b>SWS Item</b>	<b>ECUC_Icu_00222 :</b>
<b>Name</b>	IcuDefaultStartEdge

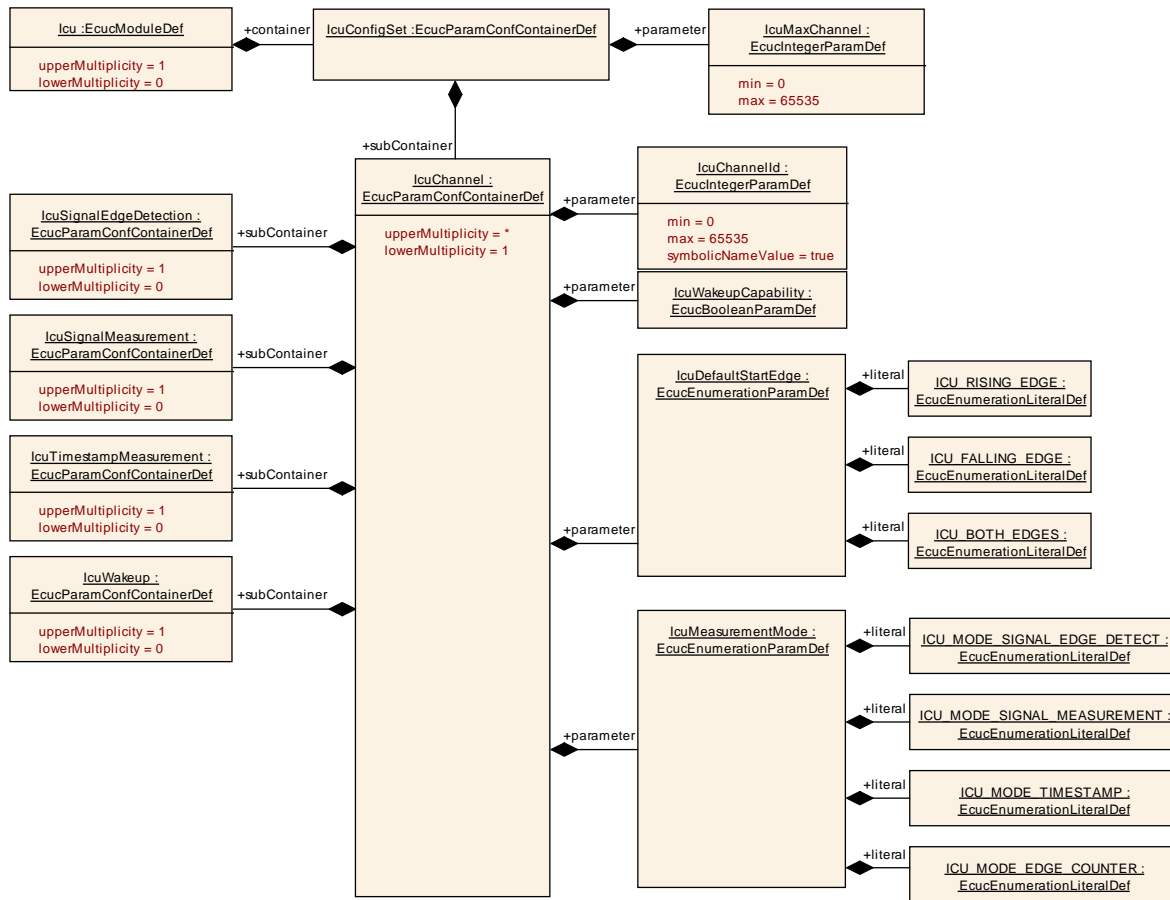
<b>Parent Container</b>	IcuChannel		
<b>Description</b>	<p>Configures the default-activation-edge which shall be used for this channel if there was no activation-edge configured by the call of service Icu_SetActivationCondition().</p> <p>In case the Measurement Mode is "IcuSignalMeasurement" and the properties "DutyCycle" or "Period" are set, the edge configured here is used as Default Period Start Edge.</p> <p>Implementation Type: Icu_ActivationType</p>		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucEnumerationParamDef		
<b>Range</b>	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.	
<b>Post-Build Variant Value</b>	true		
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	--	
	<b>Post-build time</b>	X	VARIANT-POST-BUILD
<b>Scope / Dependency</b>	scope: local		

<b>SWS Item</b>	ECUC_Icu_00223 :		
<b>Name</b>	IcuMeasurementMode		
<b>Parent Container</b>	IcuChannel		
<b>Description</b>	<p>Configures the measurement mode of this channel.</p> <p>Implementation Type: Icu_MeasurementModeType</p>		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucEnumerationParamDef		
<b>Range</b>	ICU_MODE_EDGE_COUNTER	<p>The channel is used to count the edges which are configured by the call of the service Icu_SetActivationCondition().</p> <p>The following API services support this mode:</p> <ul style="list-style-type: none"> <li>- Icu_EnableEdgeCount()</li> <li>- Icu_DisableEdgeCount()</li> <li>- Icu_GetEdgeNumbers()</li> <li>- Icu_ResetEdgeCount()</li> </ul> <p>This mode can only be configured if IcuEdgeVountApi is switched on.</p>	
	ICU_MODE_SIGNAL_EDGE_DETECT	<p>The channel is used for detecting the edges which are configured by the call of the service Icu_SetActivationCondition().</p> <p>The following API services support this mode:</p> <ul style="list-style-type: none"> <li>- Icu_EnableNotification()</li> <li>- Icu_DisableNotification()</li> <li>- Icu_GetInputState()</li> </ul>	
	ICU_MODE_SIGNAL_MEASUREMENT	<p>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.</p> <p>The following API services support this</p>	

		<p>mode:</p> <ul style="list-style-type: none"> <li>- Icu_GetTimeElapsed()</li> <li>- Icu_GetDutyCycleValues()</li> <li>- Icu_GetInputState()</li> </ul> <p>This mode can only be configured if at least one of the following switches are set to "true":</p> <ul style="list-style-type: none"> <li>- IcuGetDutyCycleValuesApi</li> <li>- IcuGetTimeElapsedApi</li> </ul>	
	ICU_MODE_TIMESTAMP	<p>The channel is used to capture timer values on the edges which are configured by the call of the service Icu_SetActivationCondition().</p> <p>The following API services support this mode:</p> <ul style="list-style-type: none"> <li>- Icu_StartTimestamp()</li> <li>- Icu_StopTimestamp()</li> <li>- Icu_GetTimestampIndex()</li> </ul> <p>This mode can only be configured if IcuTimeStampApi is switched on.</p>	
<b>Post-Build Variant Value</b>	true		
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	--	
	<b>Post-build time</b>	X	VARIANT-POST-BUILD
<b>Scope / Dependency</b>	<p>scope: local</p> <p>dependency: The possible measurement modes are depending on the pre-processor switches, which enable/disable optional API services.</p>		

<b>SWS Item</b>	<b>ECUC_Icu_00224 :</b>		
<b>Name</b>	IcuWakeupCapability		
<b>Parent Container</b>	IcuChannel		
<b>Description</b>	<p>Information about the wakeup-capability of this channel.</p> <p>true: Channel is wakeup capable.</p> <p>false: Channel is not wakeup capable.</p>		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucBooleanParamDef		
<b>Default value</b>	--		
<b>Post-Build Variant Value</b>	true		
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	--	
	<b>Post-build time</b>	X	VARIANT-POST-BUILD
<b>Scope / Dependency</b>	scope: local		

<b>Included Containers</b>		
<b>Container Name</b>	<b>Multiplicity</b>	<b>Scope / Dependency</b>
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



### 10.2.5 IcuSignalEdgeDetection

<b>SWS Item</b>	<b>ECUC_Icu_00021 :</b>
<b>Container Name</b>	IcuSignalEdgeDetection
<b>Description</b>	This container contains the configuration (parameters) in case the measurement mode is "IcuSignalEdgeDetection"
<b>Configuration Parameters</b>	

<b>SWS Item</b>	<b>ECUC_Icu_00225 :</b>		
<b>Name</b>	IcuSignalNotification		
<b>Parent Container</b>	IcuSignalEdgeDetection		
<b>Description</b>	Notification function for signal notification.		
<b>Multiplicity</b>	0..1		
<b>Type</b>	EcucFunctionNameDef		
<b>Default value</b>	--		
<b>maxLength</b>	--		
<b>minLength</b>	--		
<b>regularExpression</b>	--		
<b>Post-Build Variant Multiplicity</b>	true		
<b>Post-Build Variant Value</b>	true		
<b>Multiplicity Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	--	
	<b>Post-build time</b>	X	VARIANT-POST-BUILD
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE

	<b>Link time</b>	--	
	<b>Post-build time</b>	X	VARIANT-POST-BUILD
<b>Scope / Dependency</b>	scope: local dependency: IcuMeasurementMode		

**No Included Containers**

### 10.2.6 IcuSignalMeasurement

<b>SWS Item</b>	<b>ECUC_Icu_00226 :</b>
<b>Container Name</b>	IcuSignalMeasurement
<b>Description</b>	This container contains the configuration (parameters) in case the measurement mode is "IcuSignalMeasurement"
<b>Configuration Parameters</b>	

<b>SWS Item</b>	<b>ECUC_Icu_00227 :</b>		
<b>Name</b>	IcuSignalMeasurementProperty		
<b>Parent Container</b>	IcuSignalMeasurement		
<b>Description</b>	Configures the property that could be measured in case the mode is "IcuSignalMeasurement". This property can not be changed during runtime.  Implementation Type: Icu_SignalMeasurementPropertyType		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucEnumerationParamDef		
<b>Range</b>	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
<b>Post-Build Variant Value</b>	true		
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	--	
	<b>Post-build time</b>	X	VARIANT-POST-BUILD
<b>Scope / Dependency</b>	scope: local dependency: IcuMeasurementMode, IcuGetDutyCycleValuesApi, IcuGetTimeElapsedApi		

**No Included Containers**

### 10.2.7 IcuTimestampMeasurement

<b>SWS Item</b>	<b>ECUC_Icu_00228 :</b>
<b>Container Name</b>	IcuTimestampMeasurement
<b>Description</b>	This container contains the configuration (parameters) in case the measurement mode is "IcuTimestamp"
<b>Configuration Parameters</b>	



<b>SWS Item</b>	<b>ECUC_Icu_00229 :</b>		
<b>Name</b>	IcuTimestampMeasurementProperty		
<b>Parent Container</b>	IcuTimestampMeasurement		
<b>Description</b>	Configures the handling of the buffer in case the mode is "Timestamp" Implementation Type: Icu_TimestampBufferType		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucEnumerationParamDef		
<b>Range</b>	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	
<b>Post-Build Variant Value</b>	true		
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	--	
	<b>Post-build time</b>	X	VARIANT-POST-BUILD
<b>Scope / Dependency</b>	scope: local dependency: IcuMeasurementMode		

<b>SWS Item</b>	<b>ECUC_Icu_00230 :</b>		
<b>Name</b>	IcuTimestampNotification		
<b>Parent Container</b>	IcuTimestampMeasurement		
<b>Description</b>	Notification function if the number of requested timestamps (Notification interval > 0) are acquired.		
<b>Multiplicity</b>	0..1		
<b>Type</b>	EcucFunctionNameDef		
<b>Default value</b>	--		
<b>maxLength</b>	--		
<b>minLength</b>	--		
<b>regularExpression</b>	--		
<b>Post-Build Variant Multiplicity</b>	true		
<b>Post-Build Variant Value</b>	true		
<b>Multiplicity Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	--	
	<b>Post-build time</b>	X	VARIANT-POST-BUILD
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	--	
	<b>Post-build time</b>	X	VARIANT-POST-BUILD
<b>Scope / Dependency</b>	scope: local dependency: IcuTimestampApi		

**No Included Containers**

## 10.2.8 IcuWakeup

<b>SWS Item</b>	<b>ECUC_Icu_00126 :</b>
<b>Container Name</b>	IcuWakeup
<b>Description</b>	This container contains the configuration (parameters) needed to configure a wakeup capable channel
<b>Configuration Parameters</b>	

<b>SWS Item</b>	<b>ECUC_Icu_00231 :</b>
-----------------	-------------------------

<b>Name</b>	IcuChannelWakeupInfo		
<b>Parent Container</b>	IcuWakeup		
<b>Description</b>	If the wakeup-capability is true the wakeup source referenced is transmitted to the ECU State Manager (EcuM) . Implementation Type: reference to EcuM_WakeupSourceType		
<b>Multiplicity</b>	0..1		
<b>Type</b>	Symbolic name reference to [ EcuMWakeupSource ]		
<b>Post-Build Variant Multiplicity</b>	true		
<b>Post-Build Variant Value</b>	true		
<b>Multiplicity Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	--	
	<b>Post-build time</b>	X	VARIANT-POST-BUILD
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	--	
	<b>Post-build time</b>	X	VARIANT-POST-BUILD
<b>Scope / Dependency</b>	scope: local dependency: IcuWakeupCapability and IcuReportWakeupSource		

**No Included Containers**

### 10.2.9 IcuConfigSet

<b>SWS Item</b>	<b>ECUC_Icu_00219 :</b>		
<b>Container Name</b>	IcuConfigSet		
<b>Description</b>	This container contains the configuration parameters and sub containers of the AUTOSAR Icu module.		
<b>Configuration Parameters</b>			

<b>SWS Item</b>	<b>ECUC_Icu_00220 :</b>		
<b>Name</b>	IcuMaxChannel		
<b>Parent Container</b>	IcuConfigSet		
<b>Description</b>	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		
<b>Multiplicity</b>	1		
<b>Type</b>	EcuIntegerParamDef		
<b>Range</b>	0 .. 65535		
<b>Default value</b>	--		
<b>Post-Build Variant Value</b>	true		
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	--	
	<b>Post-build time</b>	X	VARIANT-POST-BUILD
<b>Scope / Dependency</b>	scope: local		

<b>Included Containers</b>		
<b>Container Name</b>	<b>Multiplicity</b>	<b>Scope / Dependency</b>
IcuChannel	1..*	Configuration of an individual ICU channel.

### 10.3 Published Information

**[SWS\_Icu\_00131]** [The ICU driver shall describe which other modules (in which versions) are required. This description shall be done by the implementer.] (SRS\_BSW\_00384)

## 11 Not applicable requirements

**[SWS\_Icu\_00380]** [ These requirements are not applicable to this specification.]  
(SRS\_BSW\_00300, SRS\_BSW\_00301, SRS\_BSW\_00302, SRS\_BSW\_00304,  
SRS\_BSW\_00305, SRS\_BSW\_00306, SRS\_BSW\_00307, SRS\_BSW\_00308,  
SRS\_BSW\_00309, SRS\_BSW\_00310, SRS\_BSW\_00312, SRS\_BSW\_00314,  
SRS\_BSW\_00318, SRS\_BSW\_00321, SRS\_BSW\_00325, SRS\_BSW\_00327,  
SRS\_BSW\_00328, SRS\_BSW\_00330, SRS\_BSW\_00331, SRS\_BSW\_00333,  
SRS\_BSW\_00334, SRS\_BSW\_00335, SRS\_BSW\_00341, SRS\_BSW\_00342,  
SRS\_BSW\_00347, SRS\_BSW\_00348, SRS\_BSW\_00350, SRS\_BSW\_00353,  
SRS\_BSW\_00357, SRS\_BSW\_00358, SRS\_BSW\_00360, SRS\_BSW\_00361,  
SRS\_BSW\_00371, SRS\_BSW\_00373, SRS\_BSW\_00377, SRS\_BSW\_00378,  
SRS\_BSW\_00379, SRS\_BSW\_00383, SRS\_BSW\_00395, SRS\_BSW\_00397,  
SRS\_BSW\_00398, SRS\_BSW\_00399, SRS\_BSW\_00400, SRS\_BSW\_00408,  
SRS\_BSW\_00409, SRS\_BSW\_00413, SRS\_BSW\_00414, SRS\_BSW\_00005,  
SRS\_BSW\_00006, SRS\_BSW\_00007, SRS\_BSW\_00009, SRS\_BSW\_00010,  
SRS\_BSW\_00160, SRS\_BSW\_00161, SRS\_BSW\_00162, SRS\_BSW\_00164,  
SRS\_BSW\_00167, SRS\_BSW\_00168, SRS\_BSW\_00170, SRS\_BSW\_00172,  
SRS\_BSW\_00415, 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\_BSW\_00439, SRS\_BSW\_00440,  
SRS\_BSW\_00441, SRS\_SPAL\_12068, SRS\_SPAL\_12077, SRS\_SPAL\_12092,  
SRS\_SPAL\_12265, SRS\_SPAL\_12463, SRS\_BSW\_00450)