Document Title	Specification of ICU Driver
Document Owner	AUTOSAR GbR
Document Responsibility	AUTOSAR GbR
Document Version	2.1.0
Document Status	Draft
Part of Release	2.1
Revision	0014

Document Change History

Date	Version	Changed by	Change Description
31.01.2007	2.1.0	AUTOSAR Administration	• Default start edge is now used for edge configuration • Enable and Disable Notification can now be used for Timestamp functionality. • Edge detection functionality is now pre compile time configurable On/Off • Legal disclaimer revised • Release Notes added • “Advice for users” revised • “Revision Information” added
30.01.2006	2.0.0	AUTOSAR Administration	Added the following services • Icu_SetActivationCondition • Icu_StartTimestamp • Icu_StopTimestamp • Icu_GetTimestampIndex • Icu_ResetEdgeCount • Icu_EnableEdgeCount • Icu_DisableEdgeCount • Icu_GetEdgeNumbers • Icu_GetTimeElapsed • Icu_GetDutyCycleValues • Icu_GetVersionInfo
30.06.2005	1.0.0	AUTOSAR Administration	Initial Release

Release Notes

Errata and known deficiencies

The wakeup concept is currently neither harmonized nor consistent throughout all wakeup related specification documents and therefore subject to change.

Known and potential problems resulting from known deficiencies

Due to the fact that the wakeup concept is not harmonized, inconsistent assumptions may lead to

- the duplication of functionalities across multiple modules
- proprietary implementation extensions
- difficulties during integration

Changes planned for next release

The harmonized wakeup concept throughout all wakeup related documents will result in

- adapted specification texts in Chapter 7 of the specification documents
- adapted APIs in Chapter 8 of the specification documents
- adapted wakeup sequences in Chapter 9 of the specification documents

Disclaimer

Advice to users of AUTOSAR Specification Documents:

AUTOSAR Specification Documents may contain exemplary items (exemplary reference models, "use cases", and/or references to exemplary technical solutions, devices, processes or software).
Any such exemplary items are contained in the Specification Documents for illustration purposes only, and they themselves are not part of the AUTOSAR Standard. Neither their presence in such Specification Documents, nor any later documentation of AUTOSAR conformance of products actually implementing such exemplary items, imply that intellectual property rights covering such exemplary items are licensed under the same rules as applicable to the AUTOSAR Standard.

Table of Contents

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 timestamping, usable for the acquisition of non-periodic signals
- Edge counting

2 Acronyms and abbreviations

Abbreviation / Acronym:	Description:
Active Time	This depends on the start edge a signal shall be captured. ▪ Start edge = falling edge => Active Time = Low Time ▪ Start edge = rising edge => Active Time = High Time
DEM	Diagnostic Event Manager
DET	Development 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 an handle with a name. With this handle each channel and its related properties can be found within the configuration structure. In “C” programming language this can be realized e.g. by #defines and enums.
Wakeup event	A wakeup event is understood as a pattern of edges, which will lead to the wake up of this driver. Nevertheless the decision whether a pattern is valid or not isn’t done by this driver. This shall be done by an upper layer.

3 Related documentation

3.1 Input documents

[1] General Requirements on Basic Software Modules,
https://svn.autosar.org/repos/10Releases/
AUTOSAR_SRS_General.pdf

[2] General Requirements on SPAL,
https://svn.autosar.org/repos/10Releases/
AUTOSAR_SRS_SPAL_General.pdf

[3] Specification of Standard Types,
https://svn.autosar.org/repos/10Releases/
AUTOSAR_SWS_StandardTypes.pdf

[4] List of Basic Software Modules,
https://svn.autosar.org/repos/10Releases/
AUTOSAR_BasicSoftwareModules.pdf

[5] Specification of Diagnostics Event Manager (DEM),
https://svn.autosar.org/repos/10Releases/
AUTOSAR_SWS_DEM.pdf

[6] Specification of Development Error Tracer,
https://svn.autosar.org/repos/10Releases/
AUTOSAR_SWS_Development_Error_Tracer.pdf

[7] Requirements on ICU Driver,
https://svn.autosar.org/repos/10Releases/
AUTOSAR_SRS_ICU_Driver.pdf

[8] Specification of ECU Configuration,
https://svn.autosar.org/repos/10Releases/
AUTOSAR_ECU_Configuration.pdf

[9] Layered Software Architecture,
https://svn.autosar.org/repos/10Releases/
AUTOSAR_LayeredSoftwareArchitecture.pdf

[10] Specification of ECU State Manager,
https://svn.autosar.org/repos/10Releases/
AUTOSAR_SWS_EcuStateManager.pdf

3.2 Related standards and norms

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

4 Constraints and assumptions

4.1 Limitations

No limitations.

4.2 Applicability to car domains

No restrictions.

5 Dependencies to other modules

Module DET (Development Error Tracer)
In development mode the DET will be called.

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.

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.

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 behavior.

Module EcuM
The ICU driver will do the reporting of wakeup interrupts to the EcuM.

5.1 File structure

5.1.1 Code file structure

ICU115: The code file structure shall not be defined within this specification.

5.1.2 Header file structure

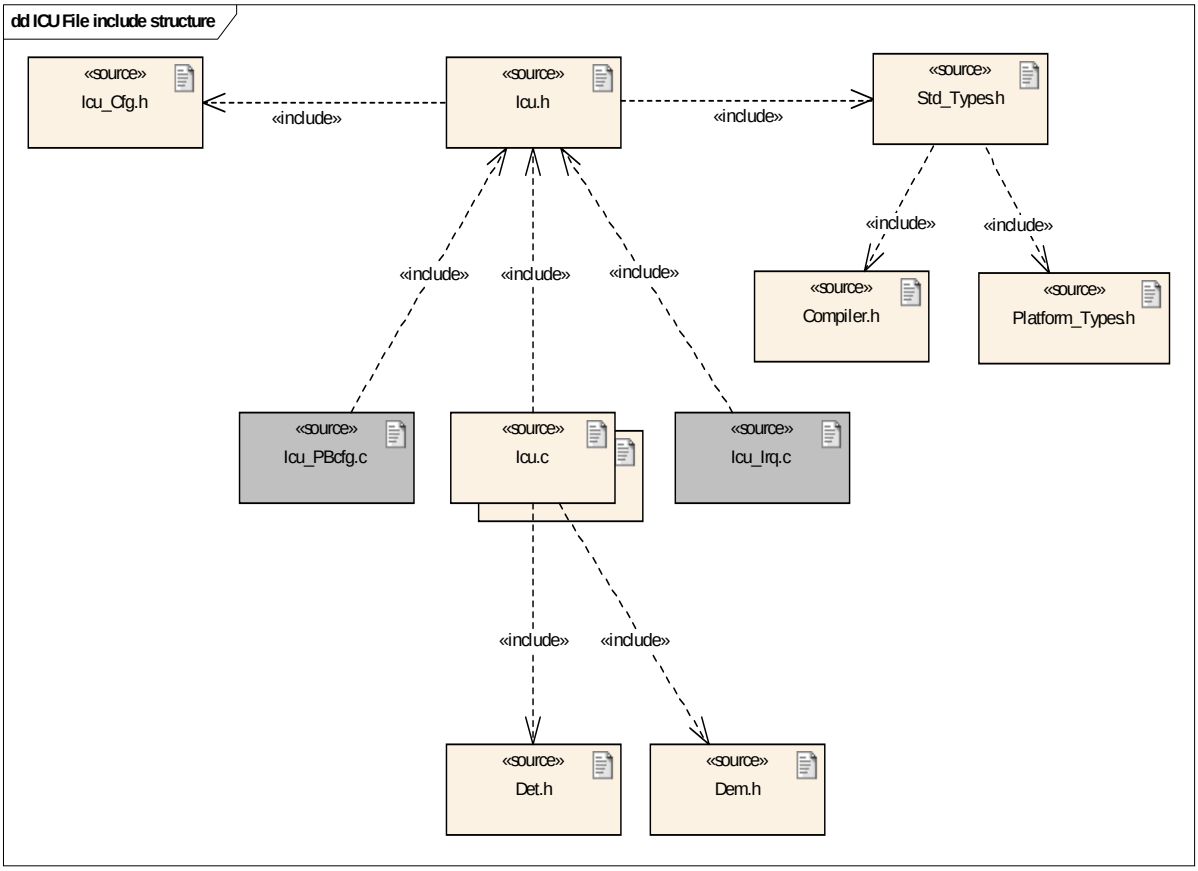


Figure 5.1: Header file structure

ICU116: The module shall include the Dem.h file. By this inclusion the API's to report errors as well as the required Event Id symbols are included.
This specification defines the name of the Event Id symbols which are provided by XML to the DEM configuration tool. The DEM configuration tool assigns ECU dependent values to the Event Id symbols and publishes the symbols in Dem_IntErrId.h.

6 Requirements traceability

Document: General Requirements on Basic Software Modules (see Literature [1])

Requirement	***Satisfied by***
[**BSW003**] Version identification	ICU028
[**BSW00300**] Module naming convention	Not applicable (non-functional requirement)
[**BSW00301**] Limit imported information	Not applicable (non-functional requirement)
[**BSW00302**] Limit exported information	Not applicable (non-functional requirement)
[**BSW00304**] AUTOSAR integer data types	Not applicable (non-functional requirement)
[**BSW00305**] Self-defined data types naming convention	Not applicable (non-functional requirement)
[**BSW00306**] Avoid direct use of compiler and platform specific keywords	Not applicable (non-functional requirement)
[**BSW00307**] Global variables naming convention	Not applicable (non-functional requirement)
[**BSW00308**] Definition of global data	Not applicable (non-functional requirement)
[**BSW00309**] Global data with read-only constraint	Not applicable (non-functional requirement)
[**BSW00310**] API naming convention	Not applicable (non-functional requirement)
[**BSW00312**] Shared code shall be reentrant	Not applicable (non-functional requirement)
[**BSW00314**] Separation of interrupt frames and service routines	Not applicable (non-functional requirement)
[**BSW00318**] Format of module version numbers	Not applicable (non-functional requirement)
[**BSW00321**] Enumeration of module version numbers	Not applicable (non-functional requirement)
[**BSW00323**] API parameter checking	ICU022, ICU023, ICU024, ICU043, ICU048, ICU120, ICU125
[**BSW00324**] Do not use HIS I/O Library	Not applicable (non-functional requirement)
[**BSW00325**] Runtime of interrupt service routines	Not applicable (implementation design requirement)
[**BSW00326**] Transition from ISRs to OS tasks	Not applicable (non-functional requirement)
[**BSW00327**] Error values naming convention	Not applicable (non-functional requirement)
[**BSW00328**] Avoid duplication of code	Not applicable (non-functional requirement)
[**BSW00329**] Avoidance of generic interfaces	Not applicable (non-functional requirement)
[**BSW00330**] Usage of macros / inline functions instead of functions	Not applicable (non-functional requirement)
[**BSW00331**] Separation of error and status values	Not applicable (non-functional requirement)
[**BSW00333**] Documentation of callback function context	Not applicable (non-functional requirement)
[**BSW00334**] Provision of XML file	Not applicable (non-functional requirement)
[**BSW00335**] Status values naming convention	Not applicable (non-functional requirement)

Requirement	*Satisfied by*
[**BSW00336**] Shutdown interface	ICU035, ICU037
[**BSW00337**] Classification of errors	ICU001, ICU004
[**BSW00338**] Detection and Reporting of development errors	ICU002, ICU111
[**BSW00339**] Reporting of production relevant errors and exceptions	ICU003
[**BSW00341**] Microcontroller compatibility documentation	Not applicable (non-functional requirement)
[**BSW00342**] Usage of source code and object code	Not applicable (non-functional requirement)
[**BSW00343**] Specification and configuration of time	ICU085
[**BSW00344**] Reference to link-time configuration	ICU027, ICU006
[**BSW00345**] Pre-compile-time configuration	ICU026, Figure 5.1: Header file structure
[**BSW00346**] Basic set of module files	 Figure 5.1: Header file structure
[**BSW00347**] Naming separation of different instances of BSW drivers	Not applicable (non-functional requirement)
[**BSW00348**] Standard type header	Not applicable (non-functional requirement)
[**BSW00350**] Development error detection keyword	Not applicable (non-functional requirement)
[**BSW00353**] Platform specific type header	Not applicable (non-functional requirement)
[**BSW00355**] Do not redefine AUTOSAR integer data types	Not applicable (non-functional requirement)
[**BSW00357**] Standard API return type	Not applicable (non-functional requirement)
[**BSW00358**] Return type of init() functions	Not applicable (non-functional requirement)
[**BSW00359**] Return type of callback functions	ICU019, ICU069
[**BSW00360**] Parameters of callback functions	Not applicable (non-functional requirement)
[**BSW00361**] Compiler specific language extension header	Not applicable (non-functional requirement)
[**BSW00369**] Do not return development error codes via API	ICU002, ICU049
[**BSW00370**] Separation of callback interface from API	Not applicable (non-functional requirement)
[**BSW00371**] Do not pass function pointers via API	Not applicable (non-functional requirement)
[**BSW00373**] Main processing function naming convention.	Not applicable (this module does not provide a schedulable main function)
[**BSW00374**] Module vendor identification	ICU028
[**BSW00376**] Return type and parameters of main processing functions	Not applicable (non-functional requirement)
[**BSW00377**] Module specific API return types	Not applicable (non-functional requirement)
[**BSW00378**] AUTOSAR boolean type	Not applicable (non-functional requirement)
[**BSW00379**] Module identification	Not applicable (non-functional requirement)
[**BSW00380**] Separate C-File for configuration parameters	Figure 5.1: Header file structure
[**BSW00381**] Separate H-File for configuration	

Requirement	*Satisfied by*
parameters	Figure 5.1: Header file structure
[**BSW00383**] List dependent Files	Not applicable (this module does not use configuration files from other modules)
[**BSW00384**] List dependencies to other modules	ICU131
[**BSW00385**] List possible error notifications	ICU001
[**BSW00386**] Configuration for detecting an error	See chapter 7.2.1
[**BSW00387**] Specify the configuration class of callback function	Not applicable (this module does not provide any callback routines)
[**BSW00388**] Introduce containers	See chapter 10.2
[**BSW00389**] Containers shall have names	See chapter 10.2
[**BSW00390**] Parameter content shall be unique within the module	See chapter 8.3
[**BSW00391**] Parameter shall have unique names	See chapter 8.3See chapter 8.3
[**BSW00392**] Parameters shall have a type	See chapter 8.3
[**BSW00393**] Parameters shall have a range	See chapter 8.3
[**BSW00394**] Specify the scope of the parameters	See chapter 8.3
[**BSW00395**] List the required parameters (per parameter)	Not applicable (none of the parameters of this module are dependent on other parameters)
[**BSW00396**] Configuration classes	See chapter 10.2
[**BSW00397**] Pre-compile-time parameters	Not applicable (only #define's as pre-compile time parameters)
[**BSW00398**] Link-time parameters	Not applicable (this is just a definement)
[**BSW00399**] Loadable Post-build time parameters	Not applicable (this is just a definement)
[**BSW004**] Version check	ICU005
[**BSW00400**] Selectable Post-build time parameters	Not applicable (this is just a definement)
[**BSW00401**] Documentation of multiple instances of configuration parameters	See chapter 10.2
[**BSW00402**] Published information	See chapter 10.3
[**BSW00404**] Reference to post build time configuration	ICU006
[**BSW00405**] Reference to multiple configuration sets	ICU006
[**BSW00406**] Check module initialization	ICU022
[**BSW00407**] Function to read out published parameters	ICU093
[**BSW00408**] Configuration parameter naming convention	Not applicable (non-functional requirement)
[**BSW00409**] Header files for production code error ID's	Not applicable (no production relevant error status, only error events)
[**BSW00410**] Compiler switches shall have specified values.	ICU055, ICU090, ICU092, ICU094, ICU095, ICU096, ICU097, ICU122, ICU063, ICU099, ICU100, ICU101, ICU102, ICU103, ICU104, ICU105, ICU106, ICU111,
[**BSW00411**] Get version info keyword	ICU094
[**BSW00412**] Separate H-File for configuration parameters	Figure 5.1: Header file structure
[**BSW00413**] Accessing instances of BSW modules	Not applicable (all configuration parameters are single instance only)
[**BSW00414**] Parameter of init function	Not applicable (Due to a SPAL Team decision, the parameter will be kept in any variant)

Requirement	*Satisfied by*
[**BSW005]** No hard coded horizontal interfaces within MCAL	Not applicable (non-functional requirement)
[**BSW006]** Platform independency	Not applicable (non-functional requirement)
[**BSW007**] HIS MISRA C	Not applicable (non-functional requirement)
[**BSW009**] Module User Documentation	Not applicable (non-functional requirement)
[**BSW010**] Memory resource documentation	Not applicable (non-functional requirement)
[**BSW101**] Initialization interface	ICU006
[**BSW158**] Separation of configuration from implementation.	Figure 5.1: Header file structure
[**BSW159**] Tool-based configuration	Both, static and runtime configuration parameters are located outside the source code of the module. This is the prerequisite for automatic configuration. See Figure 5.1: Header file structure
[**BSW160**] Human-readable configuration data	Not applicable (non-functional requirement)
[**BSW161**] Microcontroller abstraction	Not applicable (non-functional requirement)
[**BSW162**] ECU layout abstraction	Not applicable (non-functional requirement)
[**BSW164**] Implementation of interrupt service routines	Not applicable (non-functional requirement)
[**BSW167**] Static configuration checking	Not applicable (requirement for a configuration tool)
[**BSW168**] Diagnostic interface	Not applicable (this module does not support a special diagnostic interface)
[**BSW170**] Data for reconfiguration of SW-components	Not applicable (this driver has no interdependencies on other drivers)
[**BSW171**] Configurability of optional functionality	ICU092, ICU094, ICU095, ICU096, ICU097, ICU098, ICU099, ICU100, ICU101, ICU102, ICU103, ICU104, ICU105, ICU106, ICU026, ICU114, ICU122, ICU123, ICU124
[**BSW172**] Compatibility and documentation of scheduling strategy	Not applicable (non-functional requirement)
[**BSW00375**] Notification of wakeup reason	See Figure 9.3
[**BSW00415**] User dependent include files	Not applicable (this is a basic software module)
[**BSW00416**] Sequence of Initialization	Not applicable (requirement on system design, not on a single module)
[**BSW00417**] Reporting of Error Events by Non-Basic Software	Not applicable (this is a basic software module)
[**BSW00419**] Separate C-Files for pre-compile time configuration parameters	See Figure 9.3
[**BSW00420**] Production relevant error event rate detection	Not applicable (no production relevant error status, only error events)
[**BSW00421**] Reporting of production relevant error events	Not applicable (no production relevant error status, only error events)
[**BSW00422**] Debouncing of production relevant error	Not applicable

Requirement	***Satisfied by***
status	(no production relevant error status, only error events)
[**BSW00423**] Usage of SW-C template to describe BSW modules with AUTOSAR Interfaces.	Not applicable (non-functional requirement)
[**BSW00424**] BSW main processing function task allocation	Not applicable (this module does not provide a schedulable main function)
[**BSW00425**] Trigger conditions for schedulable objects	Not applicable (no internal scheduling policy)
[**BSW00426**] Exclusive areas in BSW modules	Not applicable (no exclusive areas specified for this module)
[**BSW00427**] ISR description for BSW modules	Not applicable (requirement on implementation, not on specification)
[**BSW00428**] Execution order dependencies of main processing functions	Not applicable (this module does not provide a schedulable main function)
[**BSW00429**] Restricted BSW OS functionality access	Not applicable (this module doesn't require OS objects/services)
[**BSW00431**] The BSW Scheduler module implements task bodies	Not applicable (requirement on system design, not on a single module)
[**BSW00432**] Modules should have separate main processing functions for read/receive and write/transmit data path	Not applicable (this module does not provide a schedulable main function)
[**BSW00433**] Calling of main processing functions	Not applicable (this module does not provide a schedulable main function)
[**BSW00434**] The Schedule Module shall provide an API for exclusive areas	Not applicable (no internal scheduling policy)

Document: General Requirements on SPAL (see Literature [2])

Requirement	***Satisfied by***
[**BSW12056**] Configuration of notification mechanisms	ICU018, ICU020, ICU027
[**BSW12057**] Driver module initialization	ICU006, ICU040, ICU041, ICU060, ICU061
[**BSW12063**] Raw value mode	ICU063, ICU081, ICU082, ICU083
[**BSW12064**] Change of operation mode during running operation	ICU133
[**BSW12067**] Setting of wakeup conditions	ICU008, ICU011, ICU012
[**BSW12068**] MCAL initialization sequence	Not applicable (requirement on system design, not on a single module)
[**BSW12069**] Wakeup notification of ECU State Manager	See Figure 9.3, ICU055, ICU056, ICU057
[**BSW12075**] Use of application buffers	ICU063, ICU109, ICU110
[**BSW12077**] Non-blocking implementation	Not applicable (requirement on implementation, not on specification)
[**BSW12078**] Runtime and memory efficiency	ICU114
[**BSW12092**] Access to drivers	Not applicable (requirement on system design, not on a single module)
[**BSW12125**] Initialization of hardware resources	ICU054
[**BSW12129**] Resetting of interrupt flags	ICU119
[**BSW12163**] Driver module deinitialization	ICU035, ICU036, ICU037
[**BSW12169**] Control of operation mode	ICU008
[**BSW12263**] Object code compatible configuration concept	ICU027
[**BSW12264**] Specification of configuration items	See chapter 10 “Configuration specification”
[**BSW12265**] Configuration data shall be kept constant	Not applicable (requirement on implementation, not on specification)
[**BSW12267**] Configuration of wakeup sources	ICU126
[**BSW12448**] Behavior after development error detection	ICU048, ICU049, ICU107, ICU108
[**BSW157**] Notification mechanisms of drivers and handlers	ICU021, ICU030, ICU002, ICU003
[**BSW12461**] Responsibility for register initialization	ICU006, ICU051, ICU052, ICU053, ICU128, ICU129
[**BSW12462**] Provide settings for register initialization	Chapter 10.3 “Published Information”
[**BSW12463**] Combine and forward settings for register initialization	Not applicable (requirement for a configuration tool)

Document: Requirements on ICU Driver (see Literature [7])

Requirement	***Satisfied by***
[**BSW12305**] Enable/Disable notification during runtime	ICU009, ICU010, ICU042, ICU044
[**BSW12327**] ICU global configuration	ICU038
[**BSW12368**] ICU channel/group configuration	ICU039
[**BSW12369**] Notification on signal edge	ICU021, ICU045
[**BSW12370**] Sleep mode selection service	ICU008
[**BSW12371**] ICU Channel status function	ICU030, ICU031, ICU032, ICU033
[**BSW12407**] Initialization of ICU	ICU040, ICU041, ICU061
[**BSW12408**] Wakeup enable / disable service	ICU013, ICU014
[**BSW12425**] Measured property of ICU Channel	ICU039, ICU088
[**BSW12429**] ICU Deinitialization	ICU036
[**BSW12430**] ICU start timestamp service	ICU063, ICU066
[**BSW12431**] ICU cancel timestamp service	ICU067
[**BSW12432**] Enable ICU edge counting service	ICU078
[**BSW12433**] Disable ICU edge counting service	ICU079
[**BSW12434**] ICU edge counting read service	ICU080
[**BSW12435**] Get elapsed Signal High Time for an ICU Channel	ICU082
[**BSW12436**] Get Duty Cycle input values for an ICU Channel	ICU084
[**BSW12437**] ICU driver time unit	ICU085
[**BSW12438**] Timestamps of elapsed time	ICU063
[**BSW12439**] Edge counting	ICU072, ICU073, ICU074
[**BSW12442**] Get elapsed Signal Low Time for an ICU Channel	ICU081
[**BSW12443**] Get elapsed Period Time for an ICU Channel	ICU083
[**BSW12444**] ICU timestamp notification	ICU068
[**BSW12453**] ICU get timestamp index service	ICU071
[**BSW12455**] External circular buffer handling	ICU064, ICU039
[**BSW12456**] External linear buffer handling	ICU065, ICU039
[**BSW13100**] Reset the value of counted edges	ICU072

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

ICU050: Most services are re-entrant for different ICU channel numbers. The exceptions are the following services:

- `Icu_Init()`
- `Icu_DeInit()`
- `Icu_SetMode()`
- `Icu_GetVersionInfo()`

For a single ICU channel number, the users are responsible to check the integrity if several calls for the same ICU channel are used during runtime in different tasks or ISR's. In order to keep a simple driver implementation, no check will be performed by the driver.

7.1.3 Version check

7.1.3.1 Background & Rationale

The integration of incompatible files shall be avoided. Minimum implementation is the version check of the header file inside the C file (version numbers of C and H file shall be identical)

7.1.3.2 Requirements

ICU005: The integration of incompatible files shall be avoided. Minimum implementation is the version check of the header file.
For included header files:

- ICU_AR_MAJOR_VERSION
- ICU_AR_MINOR_VERSION

shall be identical.

For the module internal c and h files:

- ICU_SW_MAJOR_VERSION
- ICU_SW_MINOR_VERSION
- ICU_AR_MAJOR_VERSION
- ICU_AR_MINOR_VERSION
- ICU_AR_PATCH_VERSION

shall be identical.

7.1.4 Time Unit Ticks

7.1.4.1 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.

7.1.4.2 Requirements

ICU085: All time units used within the API services of the ICU driver shall be unit ticks.

7.2 Error classification

7.2.1 Background & Rationale

The error classification depends on the time of error occurrence according to product life cycle:

- Development Errors
  Development errors shall be detected and fixed during the development phase. The detection of errors that shall only occur during development can be switched off for production code (by static configuration namely pre-processor switches).

- Production / series
  Those errors are hardware errors and software exceptions that cannot be avoided and are also expected to occur in production code.

7.2.2 Requirements

ICU117: Values for production code event ID's are assigned externally by the configuration of the DEM. They are published in the file Dem_IntErrId.h and included via Dem.h.

ICU118: Development error values are of type uint8.

ICU001: The following errors and exceptions shall be detectable by the ICU driver depending on its build version (development/production mode):

Type or error	*Relevance*	*Related error code*	*Value [hex]*
API Icu_Init service called with wrong parameter	Development	ICU_E_PARAM_CONFIG	0x0A
API service used with an invalid channel identifier or channel was not configured for the functionality of the calling API	Development	ICU_E_PARAM_CHANNEL	0x0B
API service used with an invalid or not feasible activation	Development	ICU_E_PARAM_ACTIVATION	0x0C
API service used with an invalid application-buffer pointer	Development	ICU_E_PARAM_BUFFER_PTR	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_StopTimestamp called on a channel which was not started or already stopped	Development	ICU_E_NOT_STARTED	0x15
API service Icu_SetMode is called while a running operation	Development	ICU_E_BUSY_OPERATION	0x16
None	Production	None	Assigned by DEM

7.3 Error detection

ICU111: The detection of development errors is configurable (*ON* / *OFF*) at pre-compile time. The switch ICU_DEV_ERROR_DETECT (see chapter 10) shall activate or deactivate the detection of all development errors.

ICU112: If the switch ICU_DEV_ERROR_DETECT is enabled, API parameter checking is enabled. The detailed description of the detected errors can be found in chapter 7.2 and chapter 8.

ICU113: The detection of production code errors cannot be switched off.

7.4 Error notification

ICU002: Detected development errors shall be reported to the Det_ReportError service of the Development Error Tracer (DET) if the pre-processor switch ICU_DEV_ERROR_DETECT is set (see ICU026, chapter 10.2.2).

ICU003: Production errors shall be reported to the Diagnostic Event Manager (DEM).

ICU004: Additional errors that are detected because of specific implementation and/or specific hardware properties shall be added in the ICU device specific implementation specification. The classification and enumeration shall be compatible with the errors listed above.

8 API specification

8.1 Imported types

8.1.1 Standard types

In this chapter all types included from the Std_Types.h file are listed:
- Std_VersionInfoType

8.1.2 EcuM types

In this chapter all types included from module EcuM are listed:
- EcuM_WakeupSourceType

8.2 Type definitions

8.2.1 Icu_ModeType

Type:	enumeration	
Range:	`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.
Description:	Allow enabling / disabling of all interrupts which are not required for the ECU wakeup.	

8.2.2 Icu_ChannelType

Type:	uint8 … uint32
Range:	This is implementation specific but not all values may be valid within the type. This type shall be chosen in order to have the most efficient implementation on a specific microcontroller platform.
Description:	Numeric identifier of an ICU channel

8.2.3 Icu_InputStateType

Type:	enumeration	
Range:	`ICU_ACTIVE`	An activation edge has been detected
	`ICU_IDLE`	No activation edge has been detected since the last call of Icu_GetInputState() or Icu_Init().
Description:	Input state of an ICU channel	

8.2.4 Icu_ConfigType

Type:	structure
Range:	Hardware and **implementation** dependent structure. The contents of the initialization data structure are microcontroller specific.
Description:	**ICU038**: This type of the external data structure shall contain the initialization data for the ICU driver. It shall contain: ▪ Wakeup Module Info (in case the wakeup-capability is true) ▪ MCU dependent properties for used HW units ▪ Clock source with optional prescaler (if provided by HW) **ICU039**: The definition for each Channel shall contain: <u>Common parameters</u> ▪ Wakeup capability (true / false) ▪ Default Start Edge ▪ Hardware Specific Settings per channel ▪ Measurement Mode - Signal Edge Detection / Notification - Signal Measurement - Timestamp - Edge Counter <u>Specific parameters</u> ▪ If measurement mode is “signal edge detection” the notification function for signal notification shall be configurable ▪ If measurement mode is “signal measurement” the property that could be measured shall be configurable. The values shall be: - High Time - Low Time - Period Time - Duty Cycle Values (High Time and Period Time) ▪ If measurement mode is “timestamp measurement”, buffer handling shall be configurable. The values shall be: - Circular buffer handling - Linear buffer handling Also the notification function for notifying the number of requested timestamps shall be configurable ▪ If measurement mode is “edge counter”, the counting mode (activation edge) shall be configurable. The values shall be: - Rising Edge - Falling Edge - Both edges ▪ If the channel is configured as wakeup capable, - the callout function for validation of wakeup reason shall be configurable - The value transmitted to the EcuM shall be configurable ▪ Assigned capture register(s) (can also be none for channels which provide only edge detection like an external interrupt) ▪ Assigned capture timer (can also be none for channels which provide only edge detection like an external interrupt)

8.2.5 Icu_ActivationType

Type:	enumeration	
Range:	`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.
Description:	Definition of the type of activation of an ICU channel.	

8.2.6 Icu_ValueType

Type:	uint8 … uint32
Range:	0 ... width of the timer register Implementation specific. This type shall be chosen in order to have the most efficient implementation on a specific microcontroller platform.
Description:	Width of the buffer for timestamp ticks and measured elapsed timeticks.

8.2.7 Icu_DutyCycleType

Type:	structure	
Range:	`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
Description:	Type which shall contain the values, needed for calculating duty cycles.	

8.2.8 Icu_IndexType

Type:	uint8 … uint32
Range:	Implementation specific. This type shall be chosen in order to have the most efficient implementation on a specific microcontroller platform.
Description:	Type, to abstract the return value of the service Icu_GetTimestampIndex().

8.2.9 Icu_EdgeNumberType

Type:	uint8 … uint32
Range:	Implementation specific. This type shall be chosen in order to have the most efficient implementation on a specific microcontroller platform.
Description:	Type, to abstract the return value of the service Icu_GetEdgeNumbers().

8.2.10 Icu_MeasurementModeType

Type:	enumeration	
Range:	`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
Description:	Definition of the measurement mode type	

8.2.11 Icu_SignalMeasurementPropertyType

Type:	enumeration	
Range:	`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).
Description:	Definition of the measurement property type	

8.2.12 Icu_TimestampBufferType

Type:	enumeration	
Range:	`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
Description:	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

Service name:	Icu_Init
Syntax:	`void Icu_Init ( const Icu_ConfigType *ConfigPtr )`
Service ID [hex]:	0x00
Sync/Async:	Synchronous
Reentrancy:	Non reentrant

Parameters (in):	`ConfigPtr`	Pointer to a selected configuration structure **ICU023**: If development error detection is enabled, the parameter `ConfigPtr` shall be checked for being within the allowed range. If this check fails the development error code `ICU_E_PARAM_CONFIG` shall be reported to the Development Error Tracer. **ICU048**: If development error detection is enabled and an error is detected the desired functionality shall be skipped and the requested service is left without any action.
Parameters (out):	None	
Return value:	None	
Description:	**ICU006**: Service for driver initialization. The Initialization function shall initialize all relevant registers of the configured hardware with the values of the structure referenced by the parameter `ConfigPtr.` The following rules regarding initialization of controller registers shall apply to this driver implementation: ▪ **ICU051**: If the hardware allows for only one usage of the register, the driver module implementing that functionality is responsible for initializing the register. ▪ **ICU052**: If the register can affect several hardware modules and if it is an I/O register it shall be initialized by the PORT driver. ▪ **ICU053**: If the register can affect several hardware modules and if it is not an I/O register it shall be initialized by the MCU driver. ▪ **ICU128**: One-time writable registers that require initialization directly after reset shall be initialized by the startup code. ▪ **ICU129**: All other registers shall be initialized by the startup code. **ICU061**: This service shall disable all notifications. **ICU121**: The wakeup-capability of a channel shall be disabled after the call of this service. **ICU040**: All appropriate ICU channel status shall be set to ICU_IDLE. **ICU060**: After initialization the module mode shall be ICU_MODE_NORMAL. **ICU054**: Only the specified resources that are configured in the configuration file shall be set (including clearing of pending interrupt flags).	
Caveats:	This service shall not be called during a running operation.	
Configuration:	See 8.2.4 Icu_ConfigType and 10.2.2	

8.3.2 Icu_DeInit

Service name:	Icu_DeInit

Syntax:	`void Icu_DeInit`
	`(`
	`  void`

	)
Service ID [hex]:	0x01
Sync/Async:	Synchronous
Reentrancy:	Non re-entrant
Parameters (in):	None
Parameters (out):	None
Return value:	None
Description:	**ICU035**: Service for ICU De-Initialization **ICU036**: After the call of this service, the state of the peripherals used by configuration shall be the same as after power on reset. Values of registers which are not writeable are excluded. **ICU091**: The service influences only the peripherals which are allocated by static configuration and/or the runtime configuration set passed by the previous call of Icu_Init(). **ICU037**: This service shall disable all used interrupts and notifications. **ICU022**:The service `Icu_Init()` shall be called first before calling any other ICU services. If not respected, the error code `ICU_E_UNINIT` will be reported to the Development Error Tracer (if development error detection is enabled).
Caveats:	This service shall not be called during a running operation.
Configuration:	**ICU092**: This service shall be pre compile time configurable `On/Off` by the configuration parameter: `ICU_DE_INIT_API.` See also chapter 10.2.8, “Configuration of optional API services”

8.3.3 Icu_SetMode

Service name:	Icu_SetMode

Syntax:	`void Icu_SetMode`
	`(`
	`  Icu_ModeType Mode`
	`)`

Service ID [hex]:	0x02
Sync/Async:	Synchronous
Reentrancy:	Non re-entrant
Parameters (in):	`Mode` `ICU_MODE_NORMAL`: Normal operation, all used interrupts are enabled according to the notification requests. `ICU_MODE_SLEEP`: Reduced power mode. In sleep mode only those notifications are available which are configured as wakeup capable. **ICU125**: If development error detection is enabled, the parameter `Mode` shall be checked by this service. If `Mode` is invalid, then the error `ICU_E_PARAM_MODE` shall be reported to the Development Error Tracer. ICU048 If development error detection is enabled and an error is detected the desired functionality shall be skipped and the requested service is left without any action.
Parameters (out):	None

Return value:	None
Description:	**ICU008**: Service for ICU mode selection. This service shall set the operation mode to the given mode parameter. **ICU011**: In ICU_MODE_NORMAL mode all notifications are available as ▪ configured by service Icu_SetActivationCondition() or Icu_DefaultStartEdge ▪ selected by the Icu_DisableNotification() and Icu_EnableNotification() services before or after the call of Icu_SetMode(). **ICU012**: In ICU_MODE_SLEEP mode ▪ only those wakeup events are available which are configured as wakeup capable, enabled via Icu_EnableWakeup() after Icu_Init() and which are not disabled via service Icu_DisableWakeup(). ▪ 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. 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 **ICU133**: This service shall not be called during running operations (e. g. timestamp measurement or edge counting). If so, the running operation shall be maintained. If development error detection is enabled the error code ICU_E_BUSY_OPERATION will be reported to the Development Error Tracer. ICU022 The service Icu_Init() shall be called first before calling any other ICU services. If not respected, the error code ICU_E_UNINIT will be reported to the Development Error Tracer (if development error detection is enabled).
Caveats:	This service influences the functionality of the ICU channels. Therefore the mode switching of the module shall be compatible to the overall state of the ECU.
Configuration:	**ICU095**: This service shall be pre compile time configurable On/Off by the configuration parameter: ICU_SET_MODE_API. See also chapter 10.2.8, “Configuration of optional API services”.

8.3.4 Icu_DisableWakeup

Service name:	Icu_DisableWakeup
Syntax:	void Icu_DisableWakeup (Icu_ChannelType Channel)
Service ID [hex]:	0x03
Sync/Async:	Synchronous
Reentrancy:	re-entrant (limited, ICU050)

Parameters (in):	`Channel`	Numeric identifier of the ICU channel **ICU024**: If development error detection is enabled, the parameter `Channel` shall be checked by this service. If `Channel` is invalid, then the error `ICU_E_PARAM_CHANNEL` shall be reported to the Development Error Tracer. **ICU059**: If development error detection is enabled, the parameter `Channel` shall be checked by this service. If `Channel` is indexing an ICU channel statically not configured as wakeup capable, then the error `ICU_E_PARAM_CHANNEL` shall be reported to the Development Error Tracer. ICU048 If development error detection is enabled and an error is detected the desired functionality shall be skipped and the requested service is left without any action.
Parameters (out):	None	
Return value:	None	
Description:	**ICU013**: This service shall disable the wakeup capability of a single ICU channel. This service is only feasible for ICU channels configured statically as wakeup capable true. ICU022 The service `Icu_Init()` shall be called first before calling any other ICU services. If not respected, the error code `ICU_E_UNINIT` will be reported to the Development Error Tracer (if development error detection is enabled).	
Caveats:	The settings are only relevant in ICU_MODE_SLEEP.	
Configuration:	**ICU096**: This service shall be pre compile time configurable `On/Off` by the configuration parameter: `ICU_DISABLE_WAKEUP_API.` See also chapter 10.2.8, “Configuration of optional API services”.	

8.3.5 Icu_EnableWakeup

Service name:	Icu_EnableWakeup

Syntax:	`void Icu_EnableWakeup`
	`(`
	`  Icu_ChannelType Channel`
	`)`

Service ID [hex]:	0x04
Sync/Async:	Synchronous
Reentrancy:	re-entrant (limited, ICU050)

Parameters (in):	`Channel`	Numeric identifier of the ICU channel ICU024 If development error detection is enabled, the parameter `Channel` shall be checked by this service. If `Channel` is invalid, then the error `ICU_E_PARAM_CHANNEL` shall be reported to the Development Error Tracer. ICU059 If development error detection is enabled, the parameter `Channel` shall be checked by this service. If `Channel` is indexing an ICU channel statically not configured as wakeup capable, then the error `ICU_E_PARAM_CHANNEL` shall be reported to the Development Error Tracer. ICU048 If development error detection is enabled and an error is detected the desired functionality shall be skipped and the requested service is left without any action.
Parameters (out):	None	
Return value:	None	
Description:	**ICU014**: This service 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. ICU022 The service `Icu_Init()` shall be called first before calling any other ICU services. If not respected, the error code `ICU_E_UNINIT` will be reported to the Development Error Tracer (if development error detection is enabled).	
Caveats:	To make the selection effective a call of the function Icu_SetMode, requesting the mode ICU_MODE_SLEEP is required.	
Configuration:	**ICU097**: This service shall be pre compile time configurable `On/Off` by the configuration parameter: `ICU_ENABLE_WAKEUP_API.` See also chapter 10.2.8, “Configuration of optional API services”.	

8.3.6 Icu_SetActivationCondition

Service name:	Icu_SetActivationCondition	
Syntax:	`void Icu_SetActivationCondition ( Icu_ChannelType Channel, Icu_ActivationType Activation )`	
Service ID [hex]:	0x05	
Sync/Async:	Synchronous	
Reentrancy:	re-entrant (limited, ICU050)	
Parameters (in):	`Channel`	Numeric identifier of the ICU channel ICU024 If development error detection is enabled, the parameter `Channel` shall be checked by this service. If `Channel` is invalid, then the error `ICU_E_PARAM_CHANNEL` shall be reported to the Development Error Tracer.

	`Activation`	Type of activation (if supported by hardware) ▪ ICU_RISING_EDGE ▪ ICU_FALLING_EDGE ▪ ICU_BOTH_EDGES **ICU043**: If development error detection is enabled, the parameter Activation shall be checked by this service. If Activation is invalid or not feasible for the requested ICU channel, then the error `ICU_E_PARAM_ACTIVATION` shall be reported to the Development Error Tracer.
Parameters (out):	None	
Return value:	None	
Description:	**ICU090**: This service shall set the activation-edge according to Activation parameter for the given channel. This service shall support channels which are configured for the following Icu_MeasurementMode: ▪ `ICU_MODE_SIGNAL_EDGE_DETECT` ▪ `ICU_MODE_TIMESTAMP` ▪ `ICU_MODE_EDGE_COUNTER` **ICU139**: This service shall reset the state for the given channel to ICU_IDLE. ICU022 The service `Icu_Init()` shall be called first before calling any other ICU services. If not respected, the error code `ICU_E_UNINIT` will be reported to the Development Error Tracer (if development error detection is enabled). ICU048 If development error detection is enabled and an error is detected the desired functionality shall be skipped and the requested service is left without any action.	
Caveats:	None	
Configuration:	None	

8.3.7 Icu_DisableNotification

Service name:	Icu_DisableNotification

Syntax:	`void Icu_DisableNotification`
	`(`
	`  Icu_ChannelType Channel`
	`)`

Service ID [hex]:	0x06
Sync/Async:	Synchronous
Reentrancy:	re-entrant (limited, ICU050)

Parameters (in):	`Channel`	Numeric identifier of the ICU channel ICU024 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_EDGE_DETECT` or `ICU_MODE_TIMESTAMP`), then the error `ICU_E_PARAM_CHANNEL` shall be reported to the Development Error Tracer. ICU048 If development error detection is enabled and an error is detected the desired functionality shall be skipped and the requested service is left without any action.
Parameters (out):	None	
Return value:	None	
Description:	**ICU009**: This service shall disable the notification of this channel in case of ▪ Signal edge detection ▪ Time stamping ICU022 The service `Icu_Init()` shall be called first before calling any other ICU services. If not respected, the error code `ICU_E_UNINIT` will be reported to the Development Error Tracer (if development error detection is enabled).	
Caveats:	None	
Configuration:	None	

8.3.8 Icu_EnableNotification

Service name:	Icu_EnableNotification	
Syntax:	`void Icu_EnableNotification ( Icu_ChannelType Channel )`	
Service ID [hex]:	0x07	
Sync/Async:	Synchronous	
Reentrancy:	re-entrant (limited, ICU050)	
Parameters (in):	`Channel`	Numeric identifier of the ICU channel ICU024 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_EDGE_DETECT` or `ICU_MODE_TIMESTAMP`), then the error `ICU_E_PARAM_CHANNEL` shall be reported to the Development Error Tracer. ICU048 If development error detection is enabled and an error is detected the desired functionality shall be skipped and the requested service is left without any action.
Parameters (out):	None	
Return value:	None	
Description:	**ICU010**: This service shall enable the notification on the given channel.	

	ICU127: This service shall cancel pending interrupts in case the channel is configured for edge detection functionality.. ICU022 The service `Icu_Init()` shall be called first before calling any other ICU services. If not respected, the error code `ICU_E_UNINIT` will be reported to the Development Error Tracer (if development error detection is enabled).
Caveats:	None
Configuration:	None

8.3.9 Icu_GetInputState

Service name:	Icu_GetInputState	
Syntax:	`Icu_InputStateType Icu_GetInputState ( Icu_ChannelType Channel )`	
Service ID [hex]:	0x08	
Sync/Async:	Synchronous	
Reentrancy:	re-entrant (limited, ICU050)	
Parameters (in):	`Channel`	Numeric identifier of the ICU channel ICU024 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 modes `ICU_MODE_SIGNAL_EDGE_DETECT` or `ICU_MODE_SIGNAL_MEASUREMENT`, then the error `ICU_E_PARAM_CHANNEL` shall be reported to the Development Error Tracer. ICU048 If development error detection is enabled and an error is detected the desired functionality shall be skipped and the requested service is left without any action.
Parameters (out):	None	
Return value:	ICU_ACTIVE	An activation edge has been detected
	ICU_IDLE	No activation edge has been detected since the last call of `Icu_GetInputState()` or `Icu_Init()`.
Description:	**ICU030**: This service shall return the status of the ICU input. Only channels which are configured for the following Icu_MeasurementMode shall be supported: ▪ `ICU_MODE_SIGNAL_EDGE_DETECT` ▪ `ICU_MODE_SIGNAL_MEASUREMENT` **ICU031**: If an activation edge has been detected this service will return ICU_ACTIVE. **ICU032**: Once the service has returned the status ICU_ACTIVE the stored status will be set to ICU_IDLE until the next edge is detected. **ICU033**: If no activation edge has been detected this service will return ICU_IDLE. **ICU049**: If development error detection is enabled and an error is detected the service returns ICU_IDLE.	

	ICU022 The service `Icu_Init()` shall be called first before calling any other ICU services. If not respected, the error code `ICU_E_UNINIT` will be reported to the Development Error Tracer (if development error detection is enabled).
Caveats:	None
Configuration:	**ICU122**: This service shall be pre compile time configurable `On/Off` by the configuration parameter: `ICU_GET_INPUT_STATE_API`. See also chapter 10.2.8, “Configuration of optional API services”.

8.3.10 Icu_StartTimestamp

Service name:	Icu_StartTimestamp	
Syntax:	`void Icu_StartTimestamp ( Icu_ChannelType Channel, Icu_ValueType *BufferPtr, uint16 BufferSize, uint16 NotifyInterval )`	
Service ID [hex]:	0x09	
Sync/Async:	Asynchronous	
Reentrancy:	re-entrant (limited, ICU050)	
Parameters (in):	`Channel`	Numeric identifier of the ICU channel ICU024 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_TIMESTAMP), then the error `ICU_E_PARAM_CHANNEL` shall be reported to the Development Error Tracer.
	`BufferPtr`	**ICU109**: Pointer to the buffer-array where the timestamp values shall be placed. **ICU120**: If development error detection is enabled, the parameter `BufferPtr` shall be checked by this service. If `BufferPtr` is invalid (e.g. “0”), then the error `ICU_E_PARAM_BUFFER_PTR` shall be reported to the Development Error Tracer.
	`BufferSize`	Size of the external buffer (number of entries) **ICU108**: If development error detection is enabled, the parameter `BufferSize` shall be checked by this service. If `BufferSize` is invalid (e.g. “0”), then the error `ICU_E_PARAM_BUFFER_SIZE` shall be reported to the Development Error Tracer.
	`NotifyInterval`	Notification interval[1] (number of events). This parameter can not be checked in a reasonable way.
Parameters (out):	None	

[1] Interval = number of timestamp-events on which a notification will be called

Return value:	None
Description:	**ICU063**: This service starts the capturing of timer values on the edges ▪ activated by the service `Icu_SetActivationCondition()` (rising / falling / both edges) ▪ to an external buffer ▪ at the beginning of the buffer **ICU064**: If circular buffer handling is configured (for the given channel), when the capture functionality reaches the end of the buffer, the driver restarts at the beginning of the buffer. **ICU134**: A notification function shall only be called if the following facts are given: ▪ A notification function is configured (see Icu_TimestampNotification_<Channel>, chapter 8.6.3) ▪ The notification has been enabled by the call of `Icu_EnableNotification()` ▪ NotifyInterval > 0 ▪ The number of events specified by NotifyInterval has been captured **ICU065**: If linear buffer handling is configured, when the capture functionality reaches the end of the buffer, the driver stops capturing timer values. **ICU066**: This service shall only be available in Measurement Mode "ICU_MODE_TIMESTAMP". ICU022 The service `Icu_Init()` shall be called first before calling any other ICU services. If not respected, the error code `ICU_E_UNINIT` will be reported to the Development Error Tracer (if development error detection is enabled). ICU048 If development error detection is enabled and an error is detected the desired functionality shall be skipped and the requested service is left without any action.
Caveats:	None
Configuration:	**ICU098**: This service shall be pre compile time configurable `On/Off` by the configuration parameter: `ICU_TIMESTAMP_API`. See also chapter 10.2.8, "Configuration of optional API services".

8.3.11 Icu_StopTimestamp

Service name:	Icu_StopTimestamp

Syntax:	`void Icu_StopTimestamp`
	`(`
	`  Icu_ChannelType Channel`
	`)`

Service ID [hex]:	0x0A
Sync/Async:	Synchronous
Reentrancy:	re-entrant (limited, ICU050)

Parameters (in):	`Channel`	Numeric identifier of the ICU channel ICU024 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_TIMESTAMP), then the error `ICU_E_PARAM_CHANNEL` shall be reported to the Development Error Tracer. ICU048 If development error detection is enabled and an error is detected the desired functionality shall be skipped and the requested service is left without any action.
Parameters (out):	None	
Return value:	None	
Description:	**ICU067**: This service stops the timestamp measurement of the given channel. **ICU132**: Stopping the timestamp measurement of a given channel, which is not active (has not started or has already stopped) will not return an error in production mode. If development error detection is enabled the error code `ICU_E_NOT_STARTED` will be reported to the Development Error Tracer. ICU022 The service `Icu_Init()` shall be called first before calling any other ICU services. If not respected, the error code `ICU_E_UNINIT` will be reported to the Development Error Tracer (if development error detection is enabled). ICU066 This service shall only be available in Measurement Mode "ICU_MODE_TIMESTAMP".	
Caveats:	None	
Configuration:	**ICU099**: This service shall be pre compile time configurable `On/Off` by the configuration parameter: `ICU_TIMESTAMP_API`. See also chapter 10.2.8, "Configuration of optional API services".	

8.3.12 Icu_GetTimestampIndex

Service name:	Icu_GetTimestampIndex
Syntax:	`Icu_IndexType Icu_GetTimestampIndex ( Icu_ChannelType Channel )`
Service ID [hex]:	0xB
Sync/Async:	Synchronous
Reentrancy:	re-entrant (limited, ICU050)

Parameters (in):	`Channel`	Numeric identifier of the ICU channel ICU024 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_TIMESTAMP), then the error `ICU_E_PARAM_CHANNEL` shall be reported to the Development Error Tracer. ICU048 If development error detection is enabled and an error is detected the desired functionality shall be skipped and the requested service is left without any action. **ICU107**: If development error detection is enabled and an error is detected this service shall return `"0"`.
Parameters (out):	`None`	
Return value:	`Icu_IndexType`	Abstract return type to cover different microcontrollers. This type is described in chapter 8.2.8 on page 25.
Description:	**ICU071**: This service reads the timestamp index of the given channel, which is the next to be written. **ICU135**: The service shall return "0" in case the service is called before Icu_StartTimestamp (no buffer is defined in this case). ICU022 The service `Icu_Init()` shall be called first before calling any other ICU services. If not respected, the error code `ICU_E_UNINIT` will be reported to the Development Error Tracer (if development error detection is enabled). ICU066 This service shall only be available in Measurement Mode "ICU_MODE_TIMESTAMP".	
Caveats:	None	
Configuration:	**ICU100**: This service shall be pre compile time configurable `On/Off` by the configuration parameter: `ICU_TIMESTAMP_API`. See also chapter 10.2.8, "Configuration of optional API services".	

8.3.13 Icu_ResetEdgeCount

Service name:	Icu_ResetEdgeCount
Syntax:	`void Icu_ResetEdgeCount ( Icu_ChannelType Channel )`
Service ID [hex]:	0xC
Sync/Async:	Synchronous
Reentrancy:	re-entrant (limited, ICU050)

Parameters (in):	`Channel`	Numeric identifier of the ICU channel ICU024 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_EDGE_COUNTER`), then the error `ICU_E_PARAM_CHANNEL` shall be reported to the Development Error Tracer. ICU048 If development error detection is enabled and an error is detected the desired functionality shall be skipped and the requested service is left without any action.
Parameters (out):	None	
Return value:	None	
Description:	**ICU072**: The value of the counted edges shall be reset to zero with the call of this service. ICU022 The service `Icu_Init()` shall be called first before calling any other ICU services. If not respected, the error code `ICU_E_UNINIT` will be reported to the Development Error Tracer (if development error detection is enabled).	
Caveats:	None	
Configuration:	**ICU101**: This service shall be pre compile time configurable `On/Off` by the configuration parameter: `ICU_EDGE_COUNT_API`. See also chapter 10.2.8, “Configuration of optional API services”.	

8.3.14 Icu_EnableEdgeCount

Service name:	Icu_EnableEdgeCount	
Syntax:	`void Icu_EnableEdgeCount ( Icu_ChannelType Channel )`	
Service ID [hex]:	0xD	
Sync/Async:	Synchronous	
Reentrancy:	re-entrant (limited, ICU050)	
Parameters (in):	`Channel`	Numeric identifier of the ICU channel ICU024 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_EDGE_COUNTER`), then the error `ICU_E_PARAM_CHANNEL` shall be reported to the Development Error Tracer. ICU048 If development error detection is enabled and an error is detected the desired functionality shall be skipped and the requested service is left without any action.
Parameters (out):	None	
Return value:	None	
Description:	**ICU078**: This service shall enable the counting of edges of the given channel.	

	Note: This service doesn't do the real counting itself. This is done by the hardware. **ICU073**: Only the configured[2] edges shall be counted (rising edge / falling edge / both edges). **ICU074**: This service shall be available for each ICU channel in Measurement Mode "Edge Counter". ICU022 The service `Icu_Init()` shall be called first before calling any other ICU services. If not respected, the error code `ICU_E_UNINIT` will be reported to the Development Error Tracer (if development error detection is enabled).
Caveats:	None
Configuration:	**ICU102**: This service shall be pre compile time configurable `On/Off` by the configuration parameter: `ICU_EDGE_COUNT_API`. See also chapter 10.2.8, "Configuration of optional API services".

8.3.15 Icu_DisableEdgeCount

Service name:	Icu_DisableEdgeCount
Syntax:	`void Icu_DisableEdgeCount ( Icu_ChannelType Channel )`
Service ID [hex]:	0x0E
Sync/Async:	Synchronous
Reentrancy:	re-entrant (limited, ICU050)
Parameters (in):	`Channel` Numeric identifier of the ICU channel ICU024 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_EDGE_COUNTER`), then the error `ICU_E_PARAM_CHANNEL` shall be reported to the Development Error Tracer. ICU048 If development error detection is enabled and an error is detected the desired functionality shall be skipped and the requested service is left without any action.
Parameters (out):	None
Return value:	None
Description:	**ICU079**: This service shall disable the counting of edges of the given channel. To reset the edge counter, the service `Icu_ResetEdgeCount()` is available. ICU022 The service `Icu_Init()` shall be called first before calling any other ICU services. If not respected, the error code `ICU_E_UNINIT` will be reported to the Development Error Tracer (if development error detection is enabled).
Caveats:	None
Configuration:	**ICU103**: This service shall be pre compile time configurable `On/Off` by the

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

	configuration parameter: ICU_EDGE_COUNT_API. See also chapter 10.2.8, “Configuration of optional API services”.

8.3.16 Icu_GetEdgeNumbers

Service name:	Icu_GetEdgeNumbers
Syntax:	`Icu_EdgeNumberType Icu_GetEdgeNumbers ( Icu_ChannelType Channel )`
Service ID [hex]:	0x0F
Sync/Async:	Synchronous
Reentrancy:	re-entrant (limited, ICU050)
Parameters (in):	`Channel` Numeric identifier of the ICU channel ICU024 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_EDGE_COUNTER`), then the error `ICU_E_PARAM_CHANNEL` shall be reported to the Development Error Tracer. ICU048 If development error detection is enabled and an error is detected the desired functionality shall be skipped and the requested service is left without any action. ICU107 If development error detection is enabled and an error is detected this service shall returns `“0”`.
Parameters (out):	None
Return value:	`Icu_EdgeNumberType` Abstract return type to cover different microcontrollers. This type is described in chapter 8.2.9 on page 25.
Description:	**ICU080**: This service shall read the number of counted edges after the last call of `Icu_ResetEdgeCount()`. ICU022 The service `Icu_Init()` shall be called first before calling any other ICU services. If not respected, the error code `ICU_E_UNINIT` will be reported to the Development Error Tracer (if development error detection is enabled).
Caveats:	None
Configuration:	**ICU104**: This service shall be pre compile time configurable `On/Off` by the configuration parameter: `ICU_EDGE_COUNT_API`. See also chapter 10.2.8, “Configuration of optional API services”.

8.3.17 Icu_StartSignalMeasurement

Service name:	Icu_StartSignalMeasurement
Syntax:	`void Icu_StartSignalMeasurement ( Icu_ChannelType Channel`

	)
Service ID [hex]:	0x13
Sync/Async:	Asynchronous
Reentrancy:	re-entrant (limited, ICU050)
Parameters (in):	`Channel` Numeric identifier of the ICU channel ICU024 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 Development Error Tracer.
Parameters (out):	None
Return value:	None
Description:	**ICU140**: This service starts the measurement of signals beginning with the configured default start edge which occurs first after the call of this service. **ICU141**: This service shall only be available in Measurement Mode "ICU_MODE_SIGNAL_MEASUREMENT". **ICU146**: This service shall reset the state for the given channel to ICU_IDLE. ICU022 The service `Icu_Init()` shall be called first before calling any other ICU services. If not respected, the error code `ICU_E_UNINIT` will be reported to the Development Error Tracer (if development error detection is enabled). ICU048 If development error detection is enabled and an error is detected the desired functionality shall be skipped and the requested service is left without any action.
Caveats:	None
Configuration:	**ICU142**: This service shall be pre compile time configurable `On/Off` by the configuration parameter `ICU_SIGNAL_MEASUREMENT_API`. See also chapter 10.2.8, "Configuration of optional API services".

8.3.18 Icu_StopSignalMeasurement

Service name:	Icu_StopSignalMeasurement
Syntax:	`void Icu_StopSignalMeasurement ( Icu_ChannelType Channel, )`
Service ID [hex]:	0x14
Sync/Async:	Synchronous
Reentrancy:	re-entrant (limited, ICU050)

Parameters (in):	`Channel`	Numeric identifier of the ICU channel ICU024 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 Development Error Tracer.
Parameters (out):	None	
Return value:	None	
Description:	**ICU143**: This service stops the measurement of signals of the given channel. **ICU144**: This service shall only be available in Measurement Mode "ICU_MODE_SIGNAL_MEASUREMENT". ICU022 The service `Icu_Init()` shall be called first before calling any other ICU services. If not respected, the error code `ICU_E_UNINIT` will be reported to the Development Error Tracer (if development error detection is enabled). ICU048 If development error detection is enabled and an error is detected the desired functionality shall be skipped and the requested service is left without any action.	
Caveats:	None	
Configuration:	**ICU145**: This service shall be pre compile time configurable `On/Off` by the configuration parameter `ICU_SIGNAL_MEASUREMENT_API`. See also chapter 10.2.8, "Configuration of optional API services".	

8.3.19 Icu_GetTimeElapsed

Service name:	Icu_GetTimeElapsed

Syntax:	`Icu_ValueType Icu_GetTimeElapsed`
	`(`
	`  Icu_ChannelType Channel`
	`)`

Service ID [hex]:	0x10
Sync/Async:	Synchronous
Reentrancy:	re-entrant (limited, ICU050)

Parameters (in):	`Channel`	Numeric identifier of the ICU channel ICU024 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 Development Error Tracer. ICU048 If development error detection is enabled and an error is detected the desired functionality shall be skipped and the requested service is left without any action. ICU107 If development error detection is enabled and an error is detected this service shall returns `"0"`.
Parameters (out):	`None`	
Return value:	`Icu_ValueType`	see Description
Description:	**ICU081**: This service reads 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. **ICU082**: This service reads 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. **ICU083**: This service reads 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 (see 10.2.3). **ICU136**: This service shall return "0" in case ▪ no requested time has been captured (see Figure 9.21, letter "A") ▪ the capturing of a requested time is ongoing and not finished (see Figure 9.21, letter "B") ▪ a captured time was already returned once by this service and this service is called again (see Figure 9.21, letter "D") ICU022 The service `Icu_Init()` shall be called first before calling any other ICU services. If not respected, the error code `ICU_E_UNINIT` will be reported to the Development Error Tracer (if development error detection is enabled).	
Caveats:	None	
Configuration:	**ICU105**: This service shall be pre compile time configurable `On/Off` by the configuration parameter: `ICU_GET_TIME_ELAPSED_API`. See also chapter 10.2.8, "Configuration of optional API services".	

8.3.20 Icu_GetDutyCycleValues

Service name:	Icu_GetDutyCycleValues
Syntax:	`void Icu_GetDutyCycleValues`

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

Service ID [hex]:	0x11
Sync/Async:	Synchronous
Reentrancy:	re-entrant (limited, ICU050)
Parameters (in):	`Channel` Numeric identifier of the ICU channel ICU024 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`, Duty Cycle Values), then the error `ICU_E_PARAM_CHANNEL` shall be reported to the Development Error Tracer.
Parameters (out):	`DutyCycleValues` **ICU110**: Pointer to a buffer where the results (high time and period time) shall be placed. ICU120 If development error detection is enabled, the parameter `DutyCycleValues` shall be checked by this service. If `DutyCycleValues` is invalid, then the error `ICU_E_PARAM_BUFFER_PTR` shall be reported to the Development Error Tracer.
Return value:	None
Description:	**ICU084**: This service reads the coherent active time and period time for the given ICU Channel, if it is configured in Measurement Mode “Signal Measurement, Duty Cycle Values”. **ICU088**: The definition on which edge the period starts shall be configurable per channel. **ICU137**: This service shall return “0” in case ▪ no coherent active- and period time has been captured (similar to Figure 9.21, letter “A”) ▪ the capturing of a requested high- and period time is ongoing and not finished (similar to Figure 9.21, letter “B”) ▪ captured duty cycle values were already returned once by this service and this service is called again (similar to Figure 9.21, letter “D”) ICU022 The service `Icu_Init()` shall be called first before calling any other ICU services. If not respected, the error code `ICU_E_UNINIT` will be reported to the Development Error Tracer (if development error detection is enabled). ICU048 If development error detection is enabled and an error is detected the desired functionality shall be skipped and the requested service is left without any action.
Caveats:	None
Configuration:	**ICU106**: This service shall be pre compile time configurable `On/Off` by the configuration parameter: `ICU_GET_DUTY_CYCLE_VALUES_API`. See also chapter 10.2.8, “Configuration of optional API services”.

8.3.21 Icu_GetVersionInfo

Service name:	Icu_GetVersionInfo
Syntax:	`void Icu_GetVersionInfo ( Std_VersionInfoType *VersionInfo )`
Service ID [hex]:	0x12
Sync/Async:	Synchronous
Reentrancy:	non reentrant
Parameters (in):	`none`
Parameters (out):	`versioninfo` Pointer to where to store the version information of this module.
Return value:	none
Description:	**ICU093**: This service returns the version information of this module. It shall be possible to call this function at any time. The version information includes: - Module Id (See Literature [4]) - Vendor Id - Vendor specific version numbers. Hint: If source code for caller and callee of this function is available this function should be realized as a macro. The macro should be defined in the modules header file.
Caveats:	None
Configuration:	**ICU094**: This service shall be pre compile time configurable `On/Off` by the configuration parameter: `ICU_GET_VERSION_INFO_API`. See also chapter 10.2.8, “Configuration of optional API services”.

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

None

8.6.2 Optional Interfaces

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

API function	Det_ReportError
Module	DET
Description	Development error notification
Configuration parameter (description see chapter 10)	ICU_DEV_ERROR_DETECT

API function	EcuM_ValidateWakeupEvent
Module	EcuM
Description	This service will be called if all of the following are true: ▪ **ICU055**: the static configuration parameter ICU_REPORT_WAKEUP_SOURCE is set to “ON” ▪ **ICU056**: the module is in mode ICU_MODE_SLEEP ▪ **ICU057**: a wakeup event occurs on a wakeup capable ICU channel. This behavior is shown in Figure 9.3 and Figure 9.4 (page 50).
Configuration parameter (description see chapter 10)	ICU_REPORT_WAKEUP_SOURCE

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 kind of interfaces are not fixed because they are configurable.

ICU119: The ISR´s shall be responsible for resetting the interrupt flags (if needed by hardware) and calling the corresponding notification functions.

ICU018: The notification functions shall be configurable as function pointers within the initialization data structure (`Icu_ConfigType`).

ICU020: If a notification function is configured as a null pointer, no call shall be executed.

Name:	Icu_SignalNotification_<Channel>
Syntax:	`void NotificationName ( void )`
Reentrancy:	Reentrancy of interface not relevant for this module (in general it is in this case not reentrant).
Parameters (in):	None
Parameters (out):	None
Return value:	None

Description:	**ICU021**: According to the last call of `Icu_EnableNotification`, this notification function shall be called if the requested signal edge (rising / falling / both edges) occurs (once per edge). **ICU019**: The notification functions shall have no parameters and no return value. **ICU044**: Only those edge notifications shall be provided, which are supported by hardware. **ICU042**: After a call of `Icu_DisableNotification` this function must not be called.
Caveats:	None
Configuration:	None

Name:	Icu_TimestampNotification_<Channel>

Syntax:	`void TimestampNotificationName`
	`(`
	`  void`
	`)`

Reentrancy:	Reentrancy of interface not relevant for this module (in general it is in this case not reentrant).
Parameters (in):	None
Parameters (out):	None
Return value:	None
Description:	**ICU068**: This notification shall be called if the number of requested timestamps (Notification interval > 0) are acquired and if the notification has been enabled by the call of `Icu_EnableNotification()`.After a call of `Icu_DisableNotification()` this function must not be called. **ICU069**: The notification functions shall have no parameters and no return value.
Caveats:	None
Configuration:	This notification depends on pre-processor switch `ICU_TIMESTAMP_API`. See also chapter 10.2.8, “Configuration of optional API services”.

Name:	Icu_TimestampNotification_<Channel>

Syntax:	`void TimestampNotificationName`
	`(`
	`  void`
	`)`

Reentrancy:	Reentrancy of interface not relevant for this module (in general it is in this case not reentrant).
Parameters (in):	None
Parameters (out):	None
Return value:	None
Description:	**ICU068**: This notification shall be called if the number of requested timestamps (Notification interval > 0) are acquired. **ICU069**: The notification functions shall have no parameters and no return value.
Caveats:	None
Configuration:	This notification depends on pre-processor switch `ICU_TIMESTAMP_API`. See also chapter 10.2.8, “Configuration of optional API services”.

9 Sequence diagrams

9.1 Icu_Init

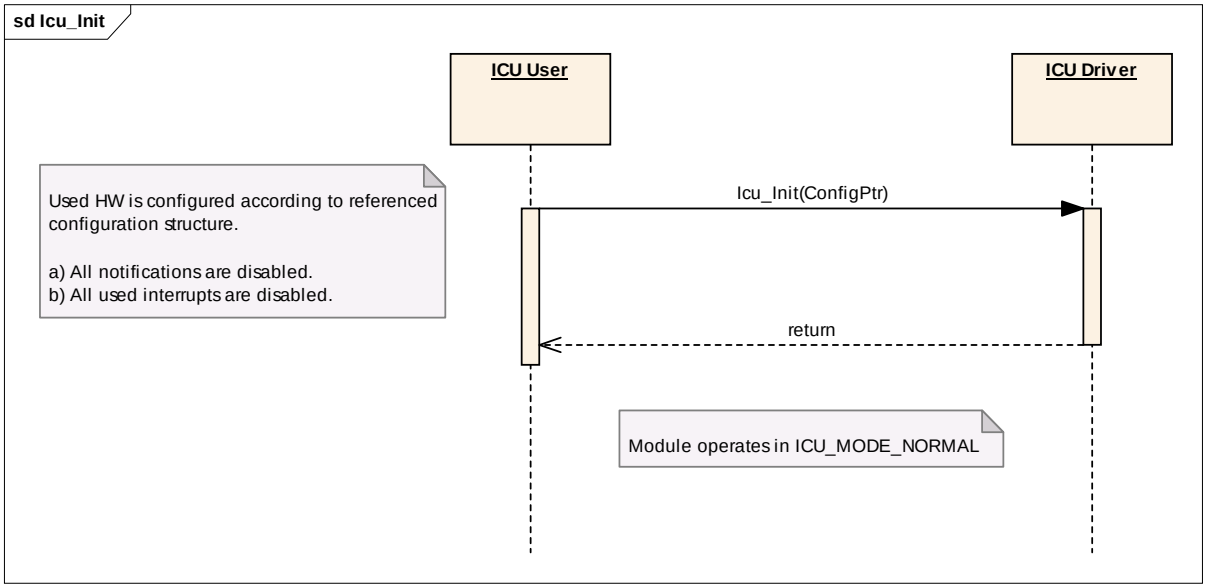


Figure 9.1: Initialization of the ICU driver

9.2 Icu_DeInit

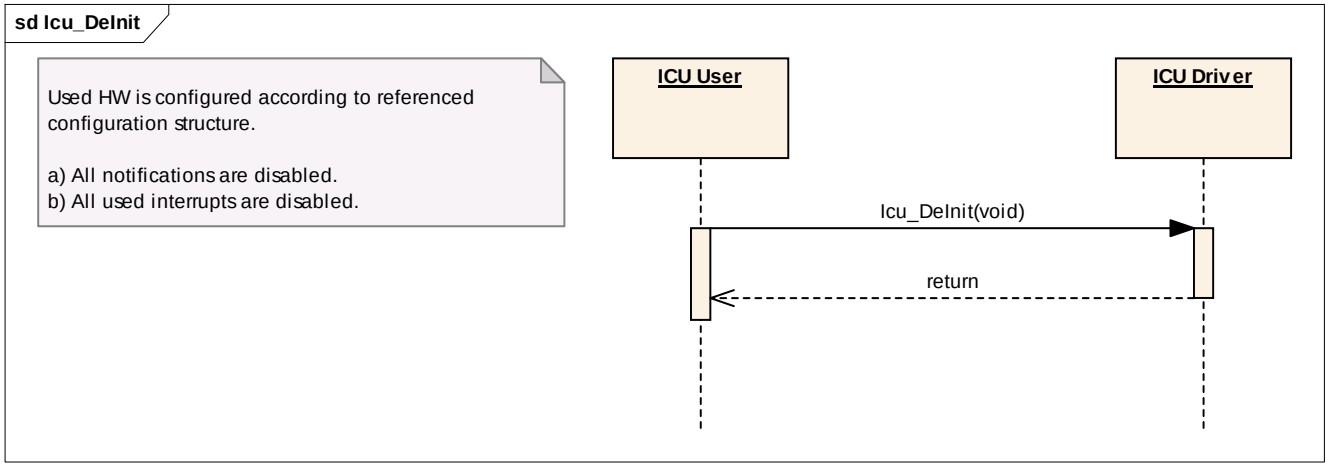


Figure 9.2: De-Initialization of the ICU driver

9.3 Validate Wakeup Events

Precondition of the following 2 figures are ICU_REPORT_WAKEUP_SOURCE = ON.
The following 2 figures shall show the different wakeup interrupt handling within different microcontrollers.

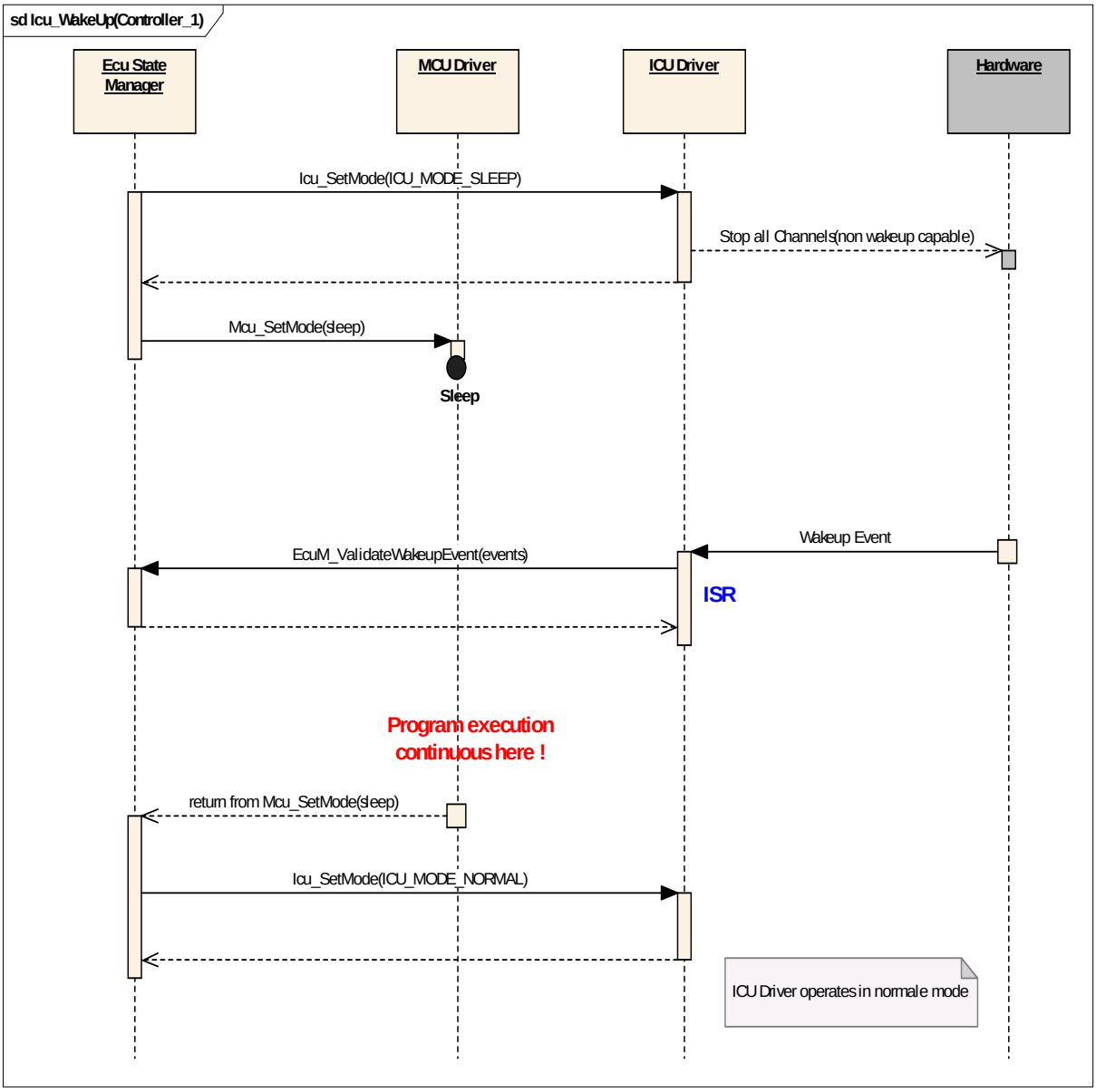


Figure 9.3: Validation of wakeup events 1

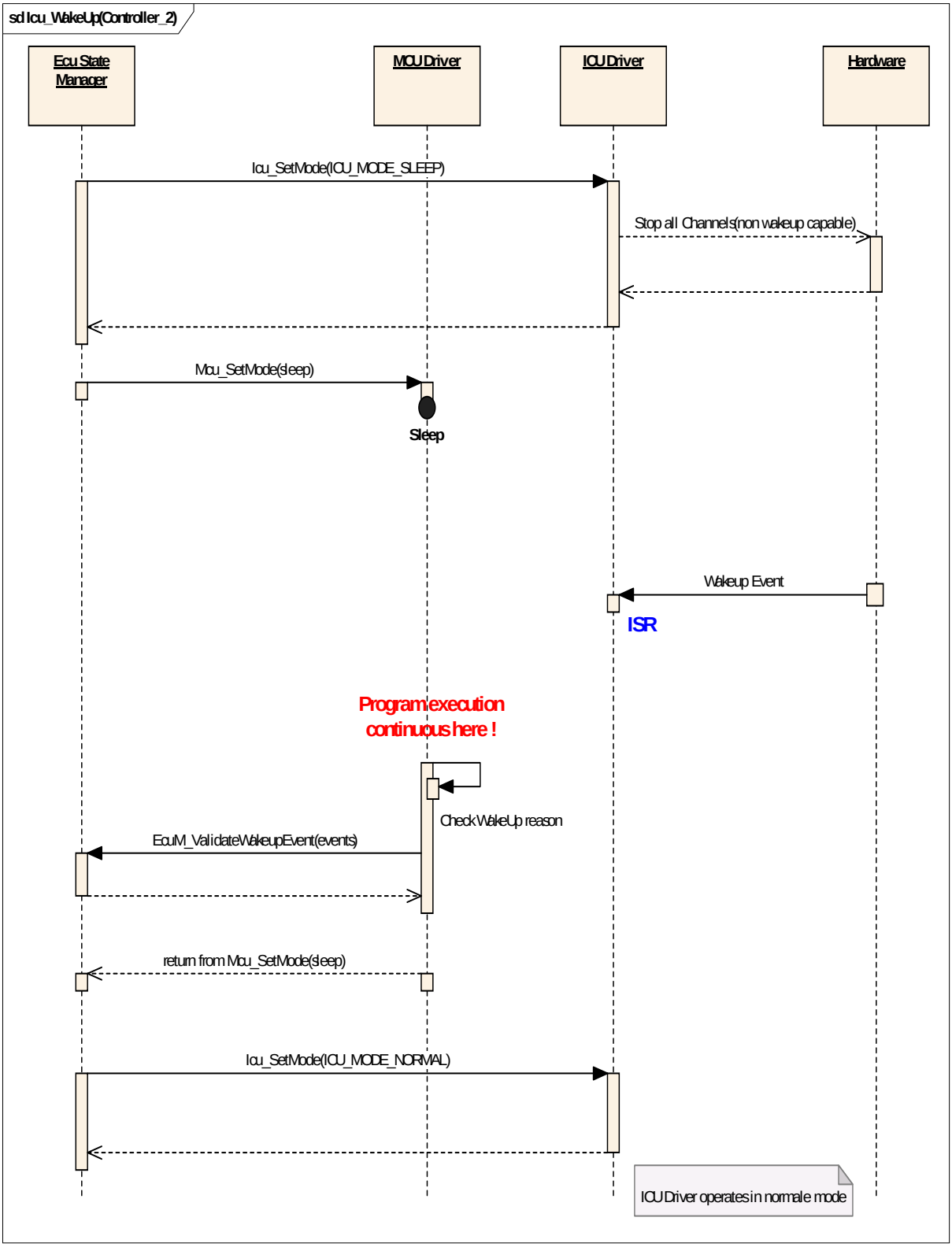


Figure 9.4: Validation of wakeup events 2

9.4 Icu_SetMode

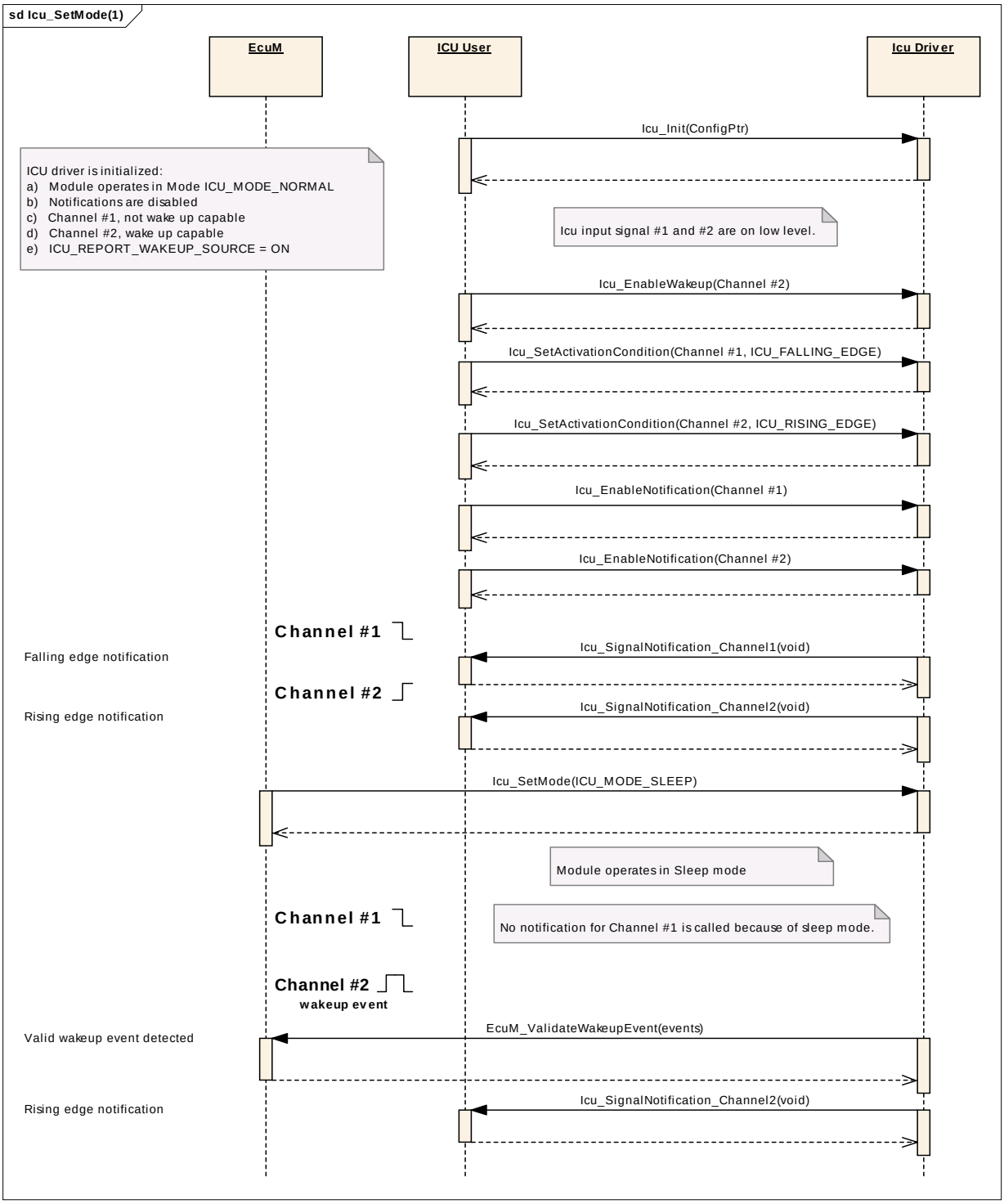


Figure 9.5: Enabled notifications in SLEEP mode

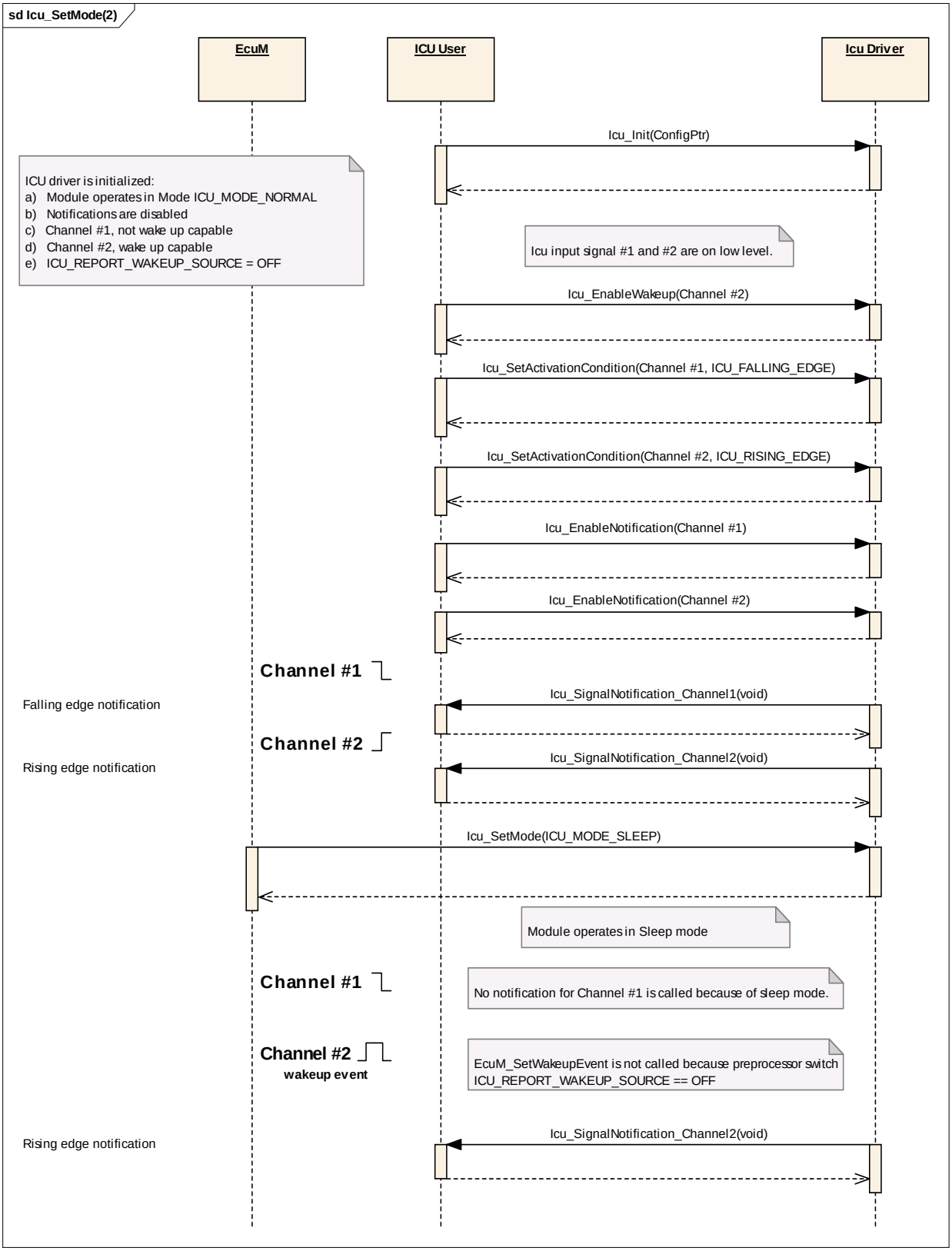


Figure 9.6: Disabled reporting of wakeup sources in SLEEP mode

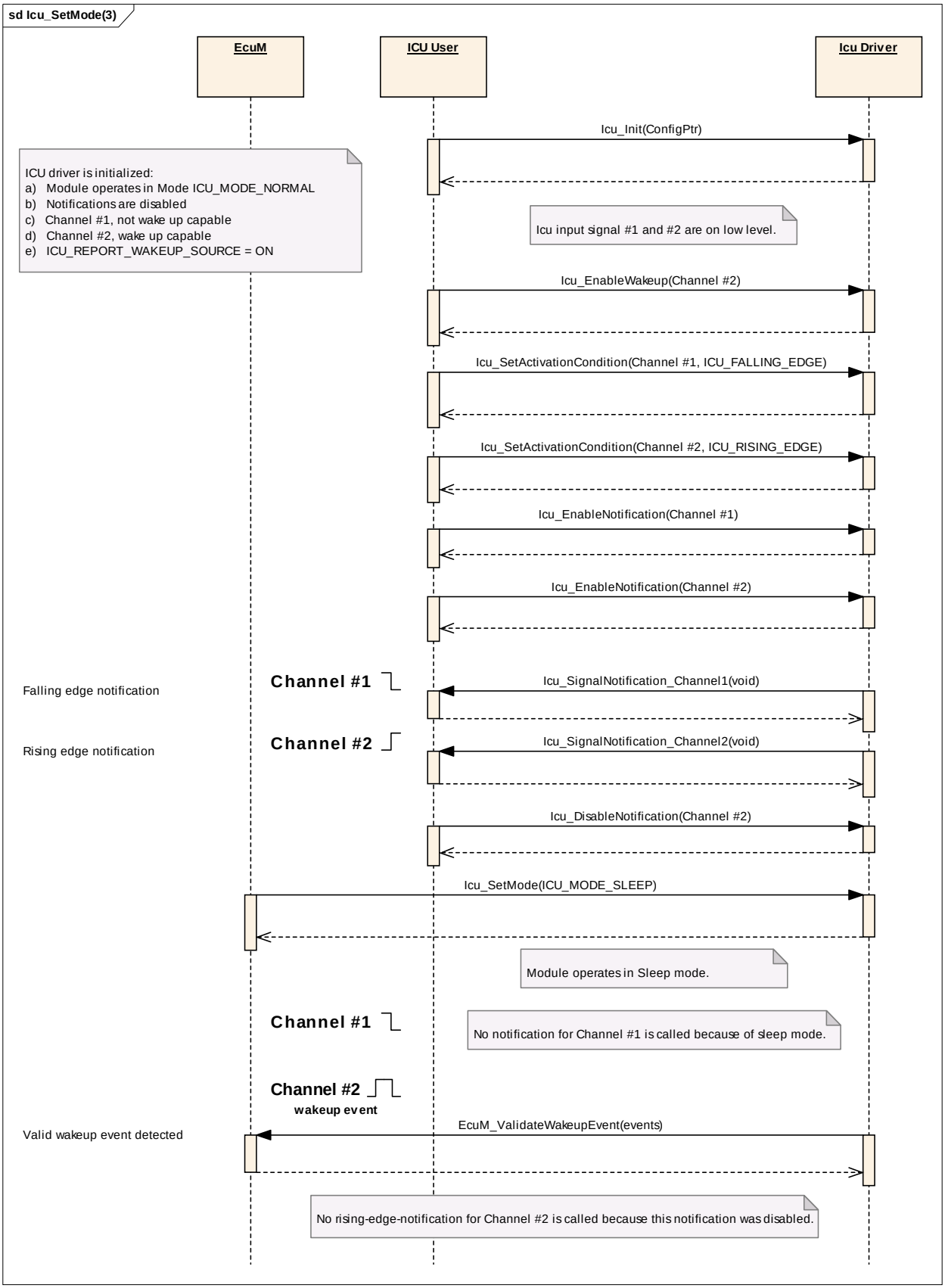


Figure 9.7: Disabled edge notification in SLEEP mode

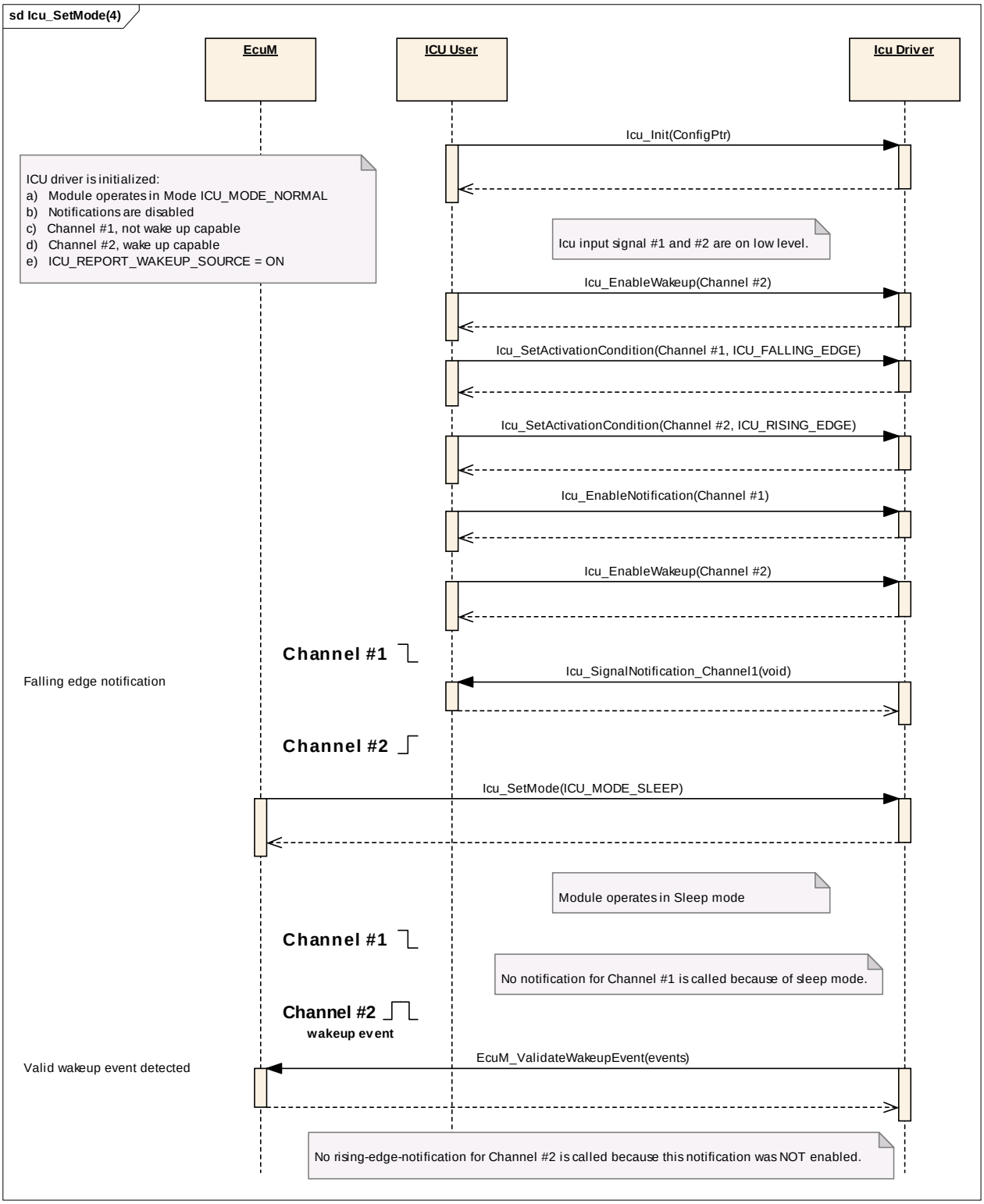


Figure 9.8: Un-Enabled reporting of notifications in SLEEP mode

9.5 Icu_DisableWakeup

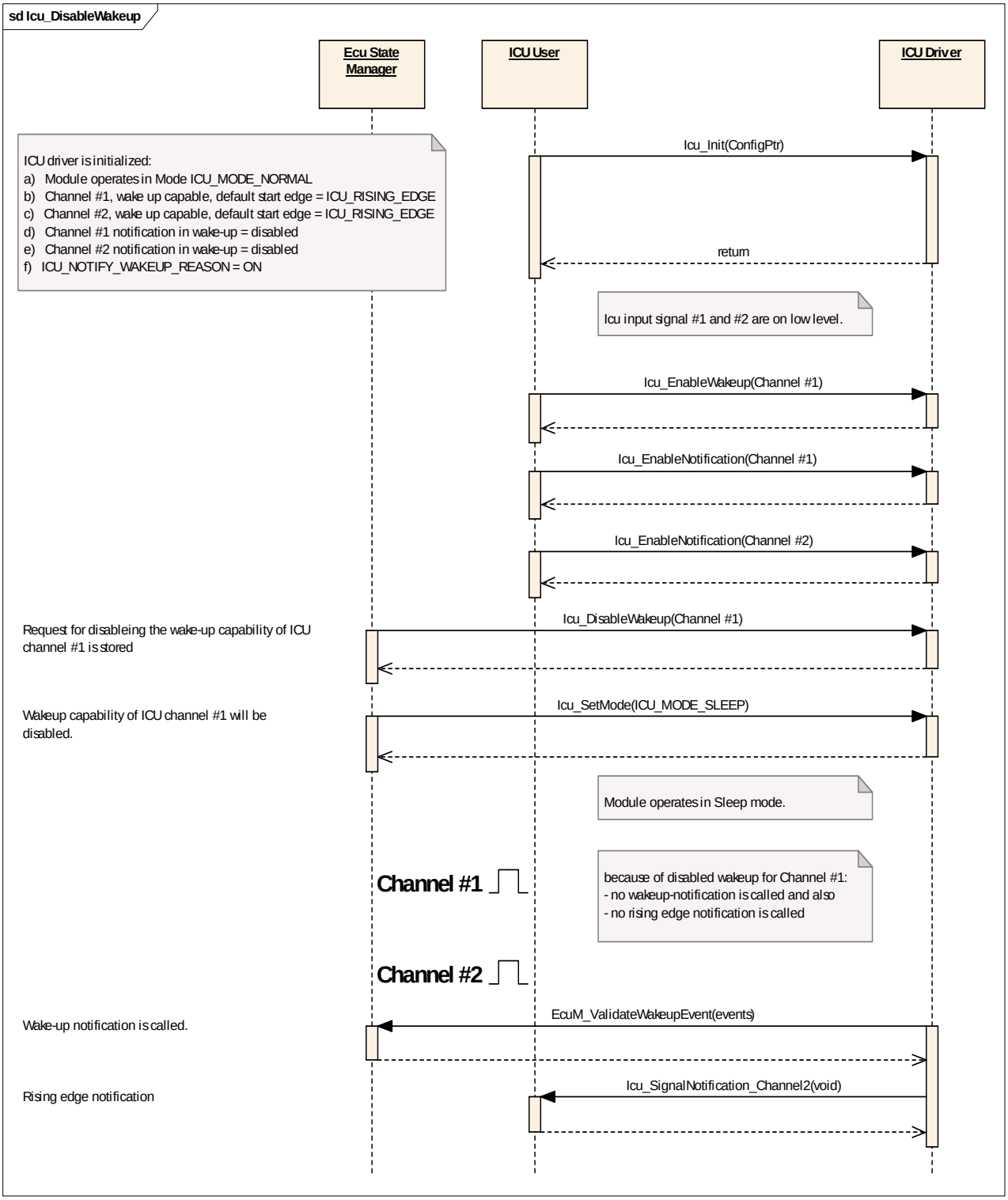


Figure 9.9: Disabling of wakeup-capabilities

9.6 Icu_EnableWakeup

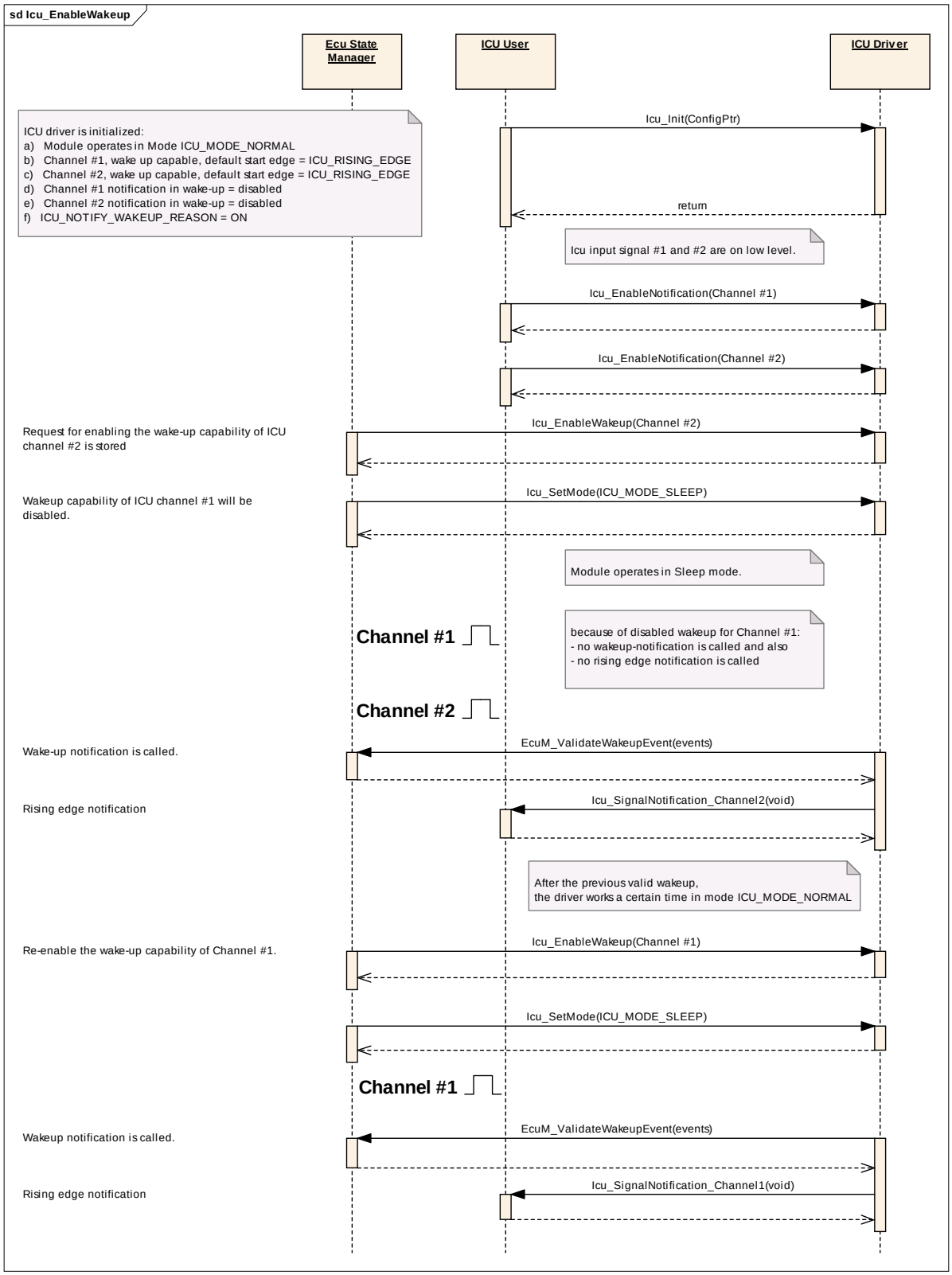


Figure 9.10: Enabling of wakeup-capabilities

9.7 Icu_SetActivationCondition

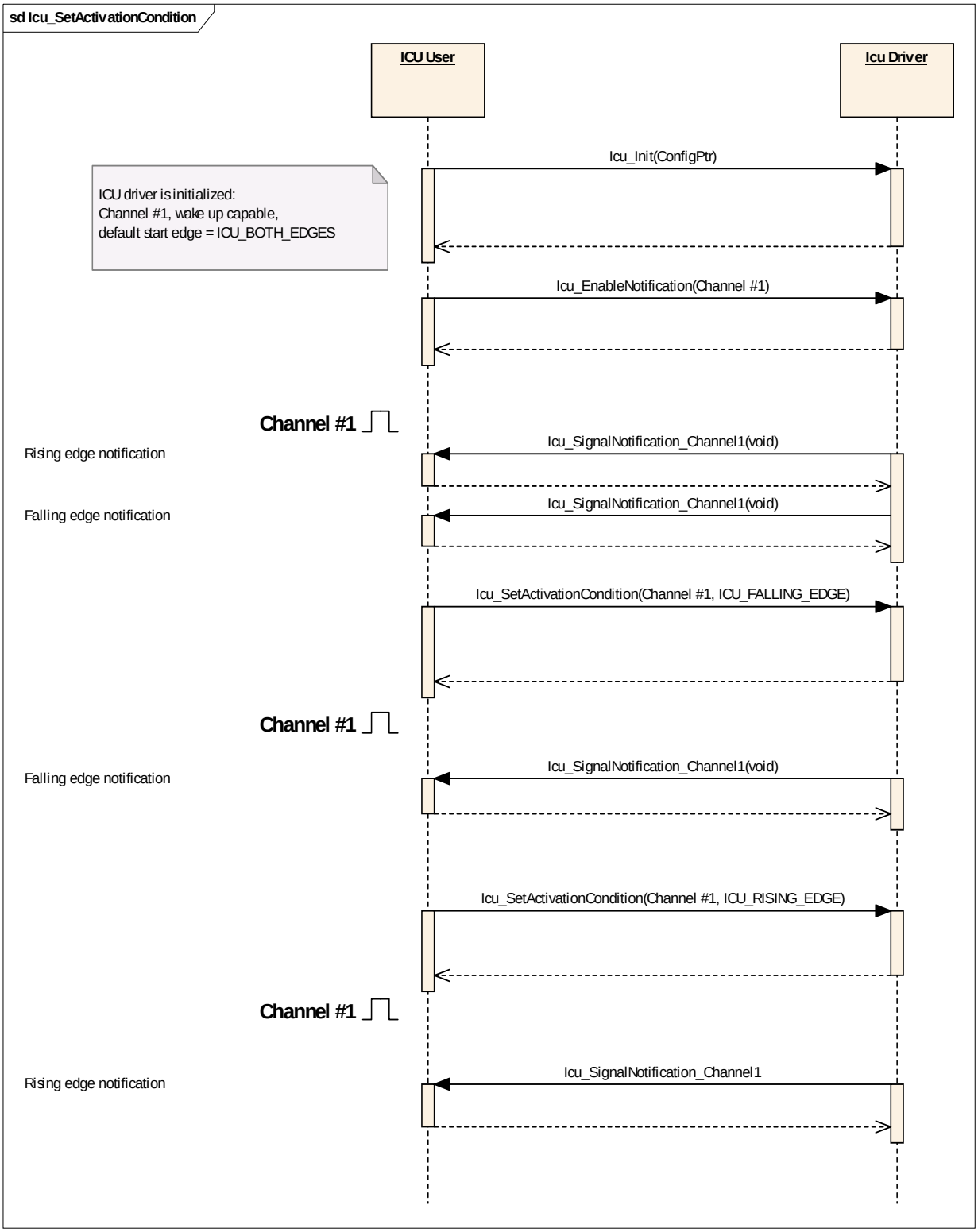


Figure 9.11: Setting up the activation condition for a channel

9.8 Icu_DisableNotification

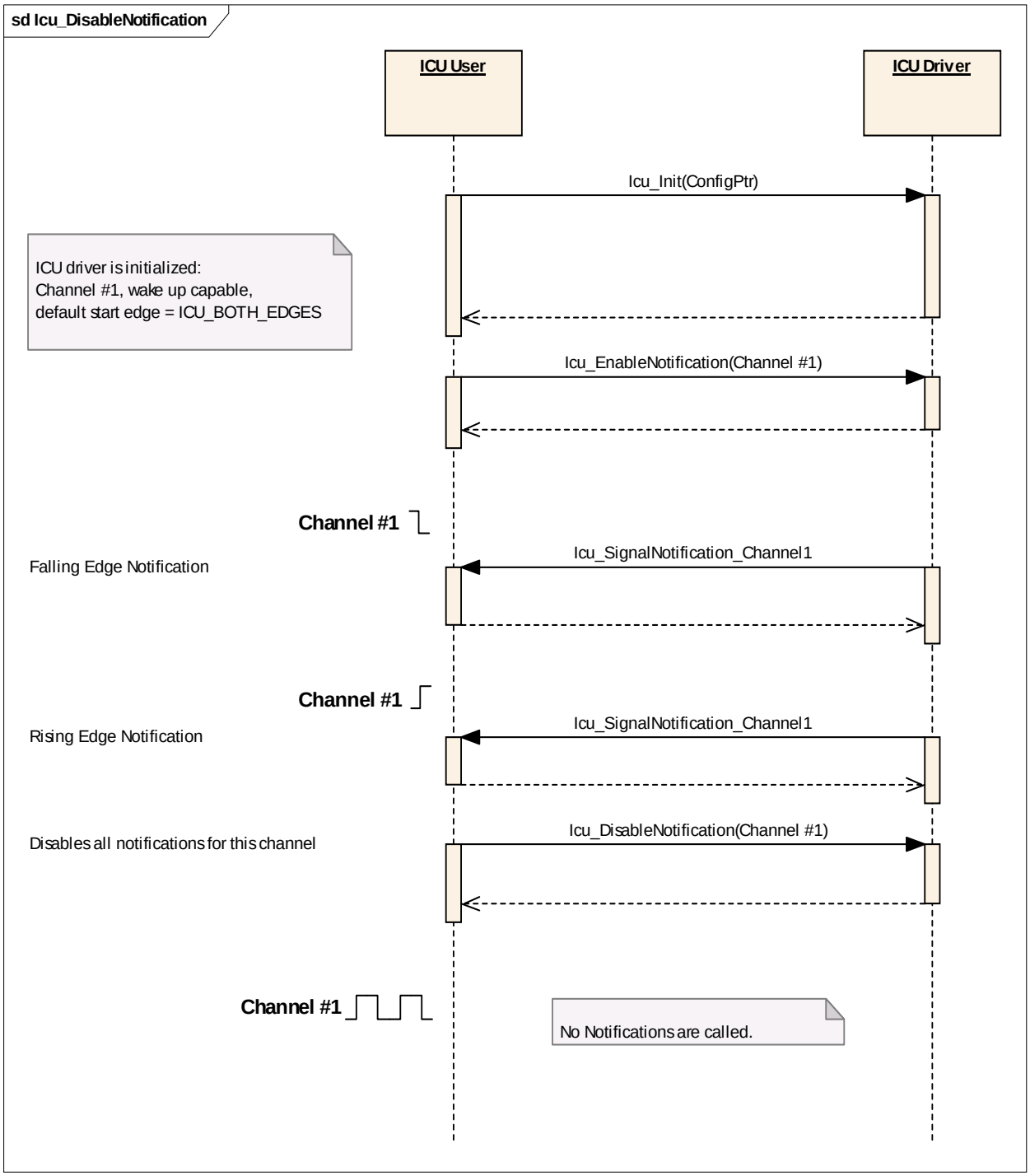


Figure 9.12: Disabling of the notification for a channel

9.9 Icu_EnableNotification

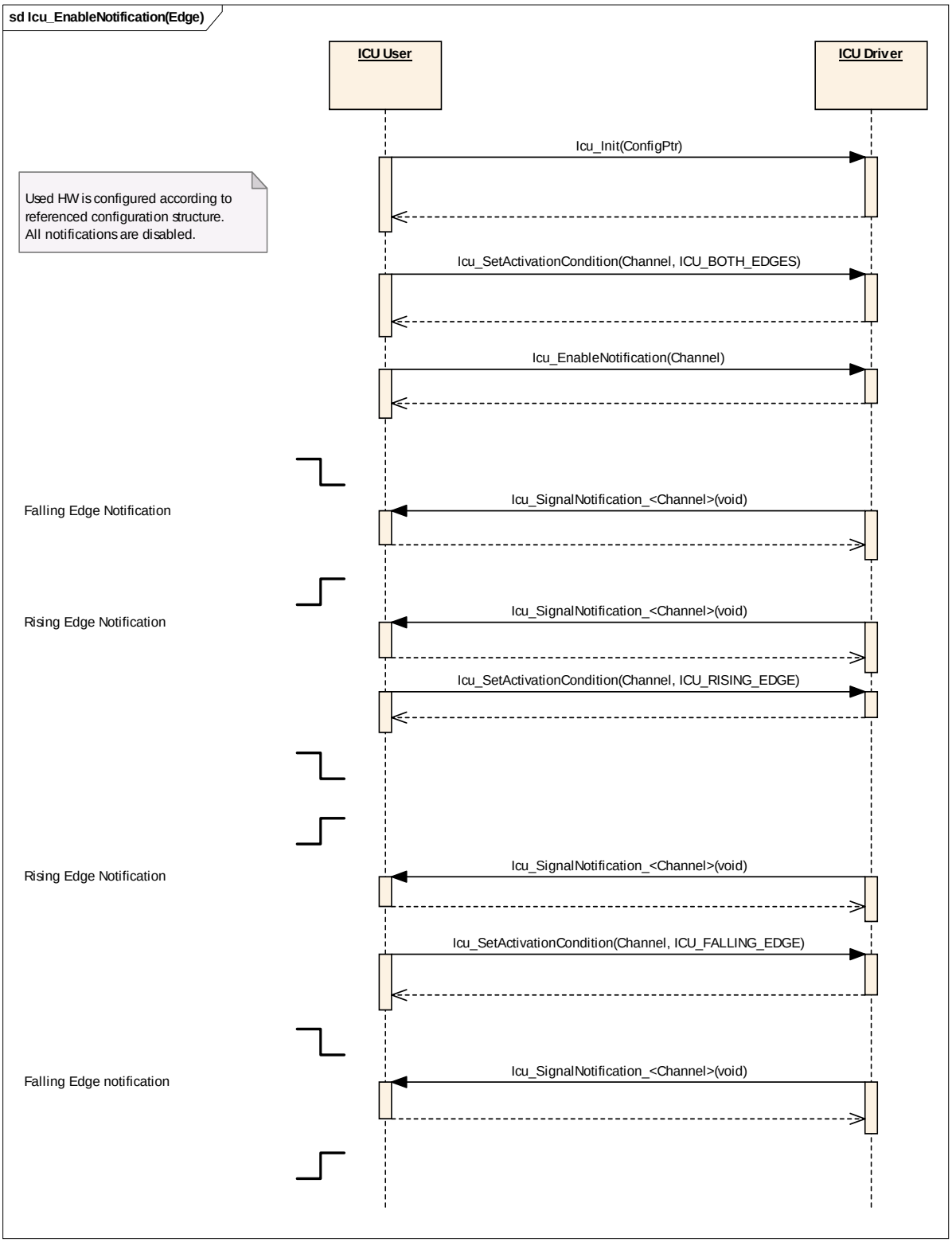


Figure 9.13: Enabling of the edge-notification for a channel

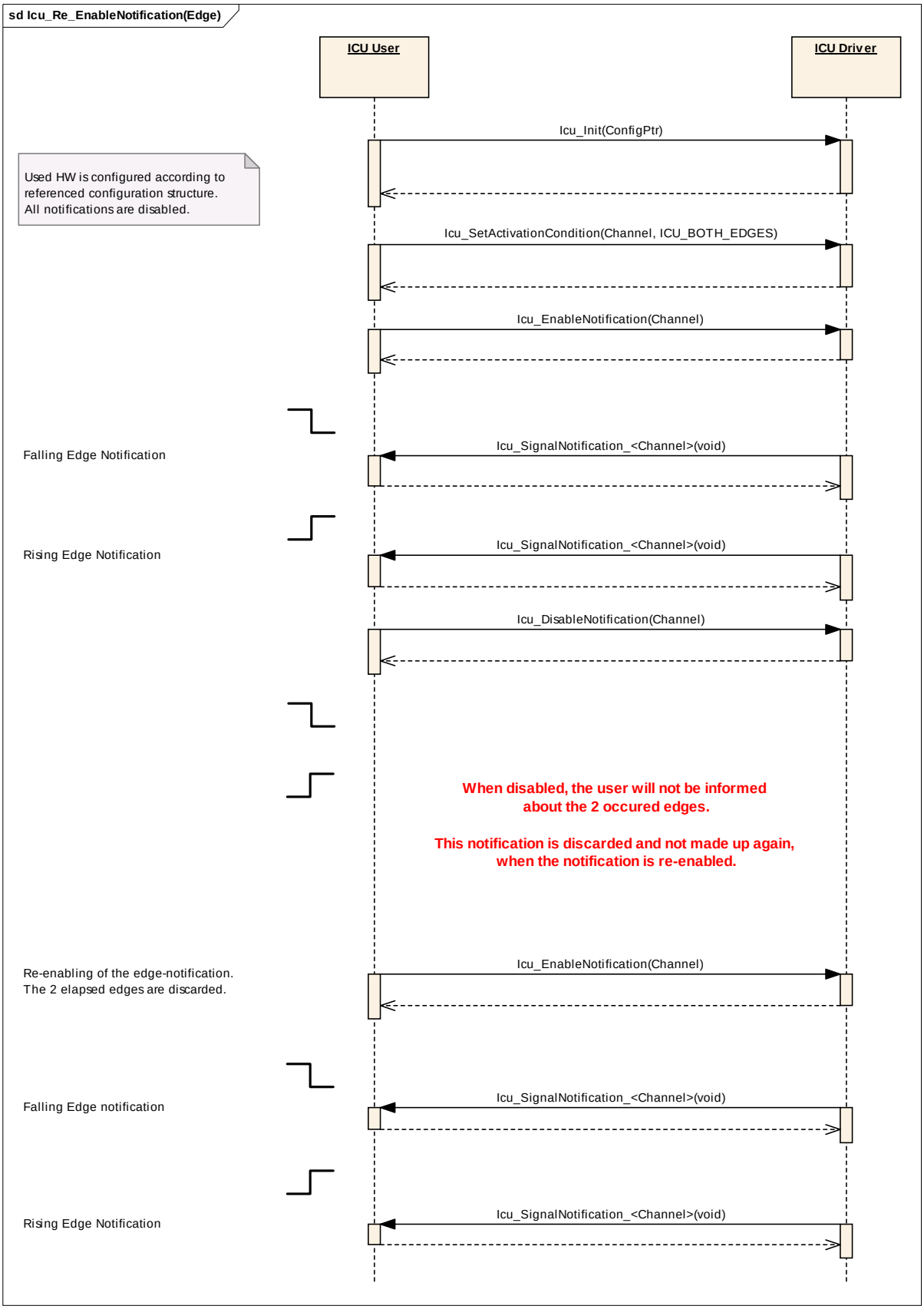


Figure 9.14: Re-enabling of the notification for a channel

9.10 Icu_GetInputState

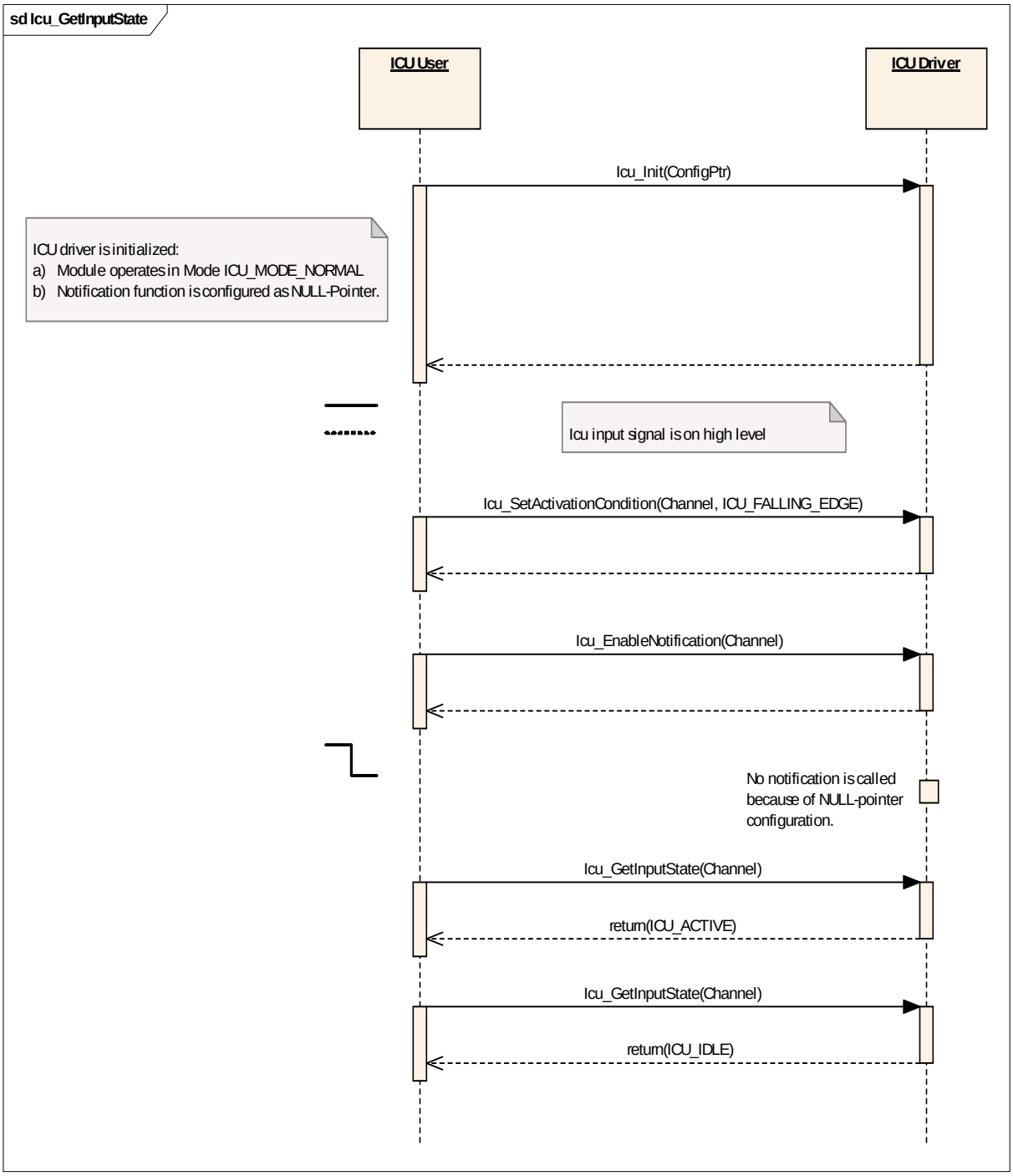


Figure 9.15: Polling of the channel status

9.11 Icu Timestamping

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

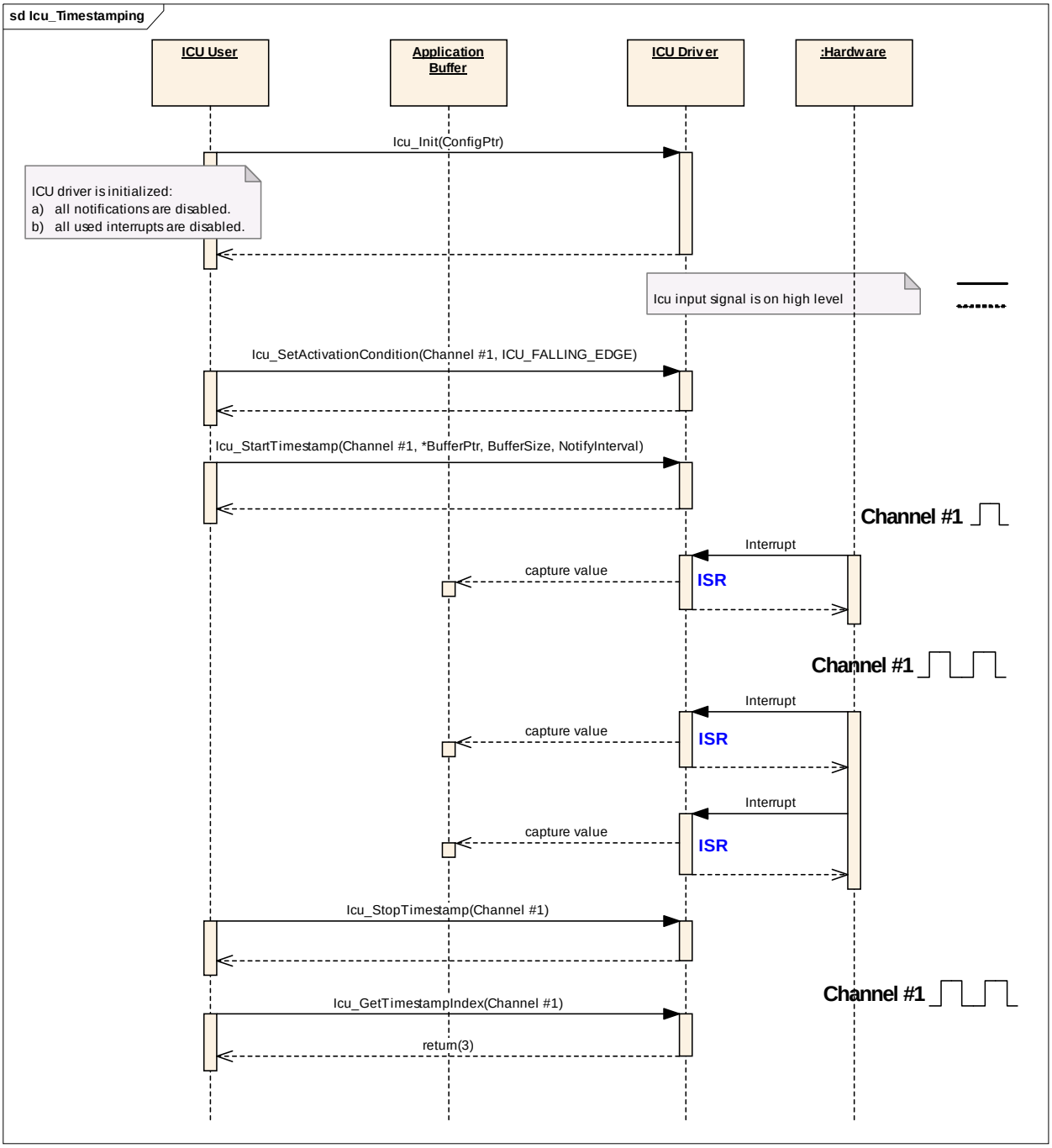
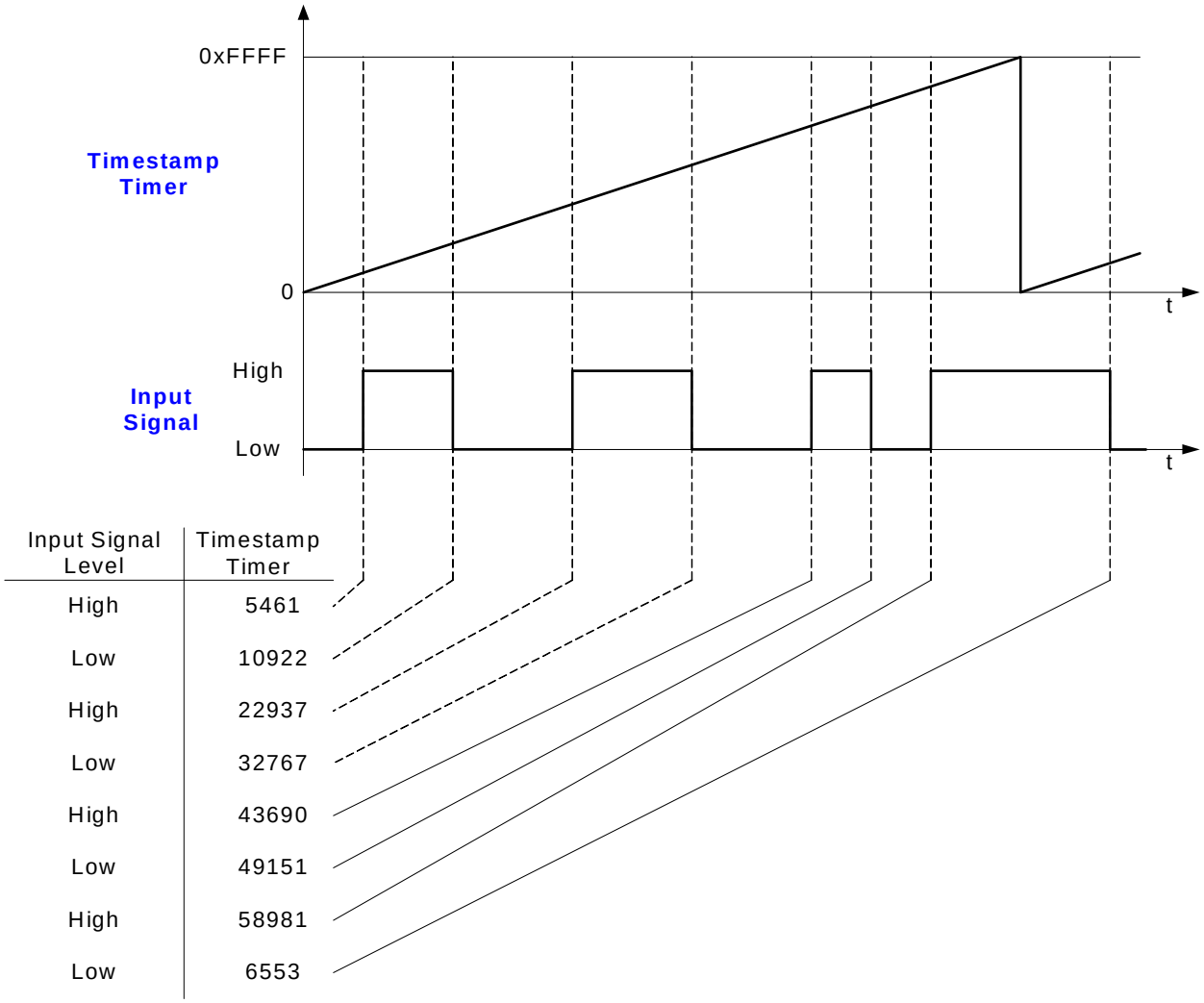


Figure 9.16: Overview of the timestamping functionality of the ICU driver

The Timestamping in general is shown in the following figure:



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

Figure 9.17: Timestamping overview

9.12 Icu Edge Counting

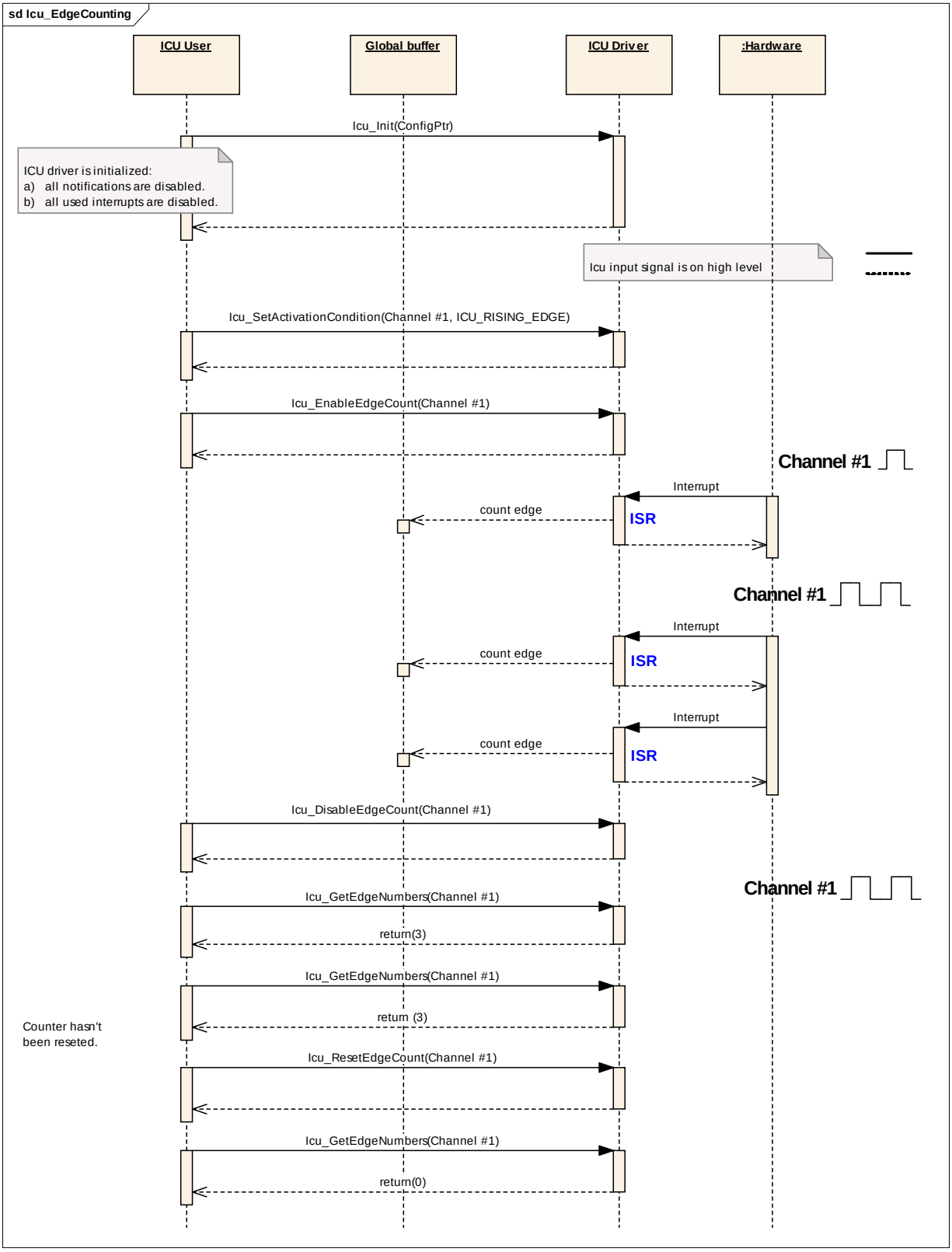


Figure 9.18: Inquire the number of counted edges

9.13 Icu_GetTimeElapsed

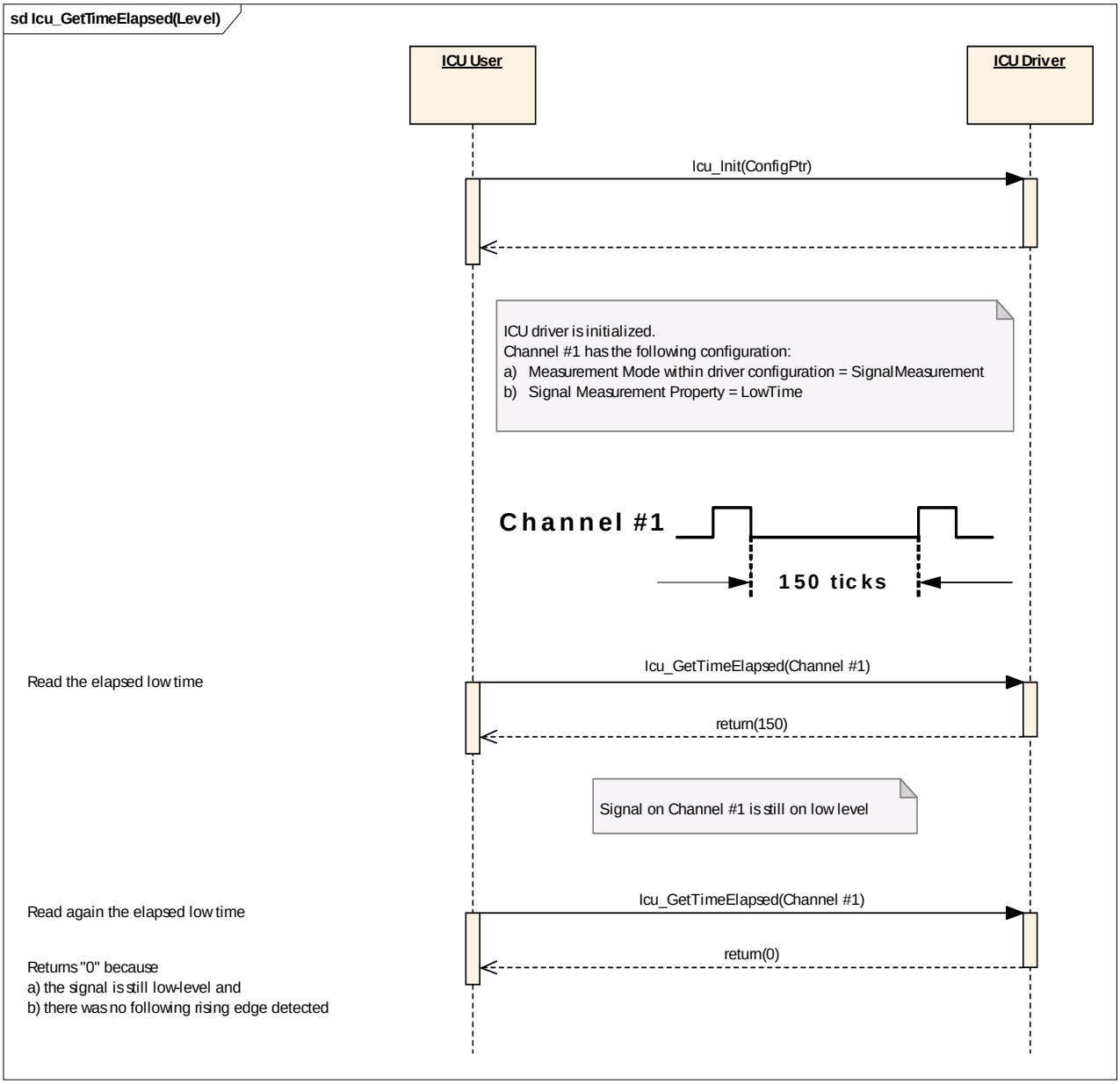


Figure 9.19: Inquire the elapsed level-time of a channel

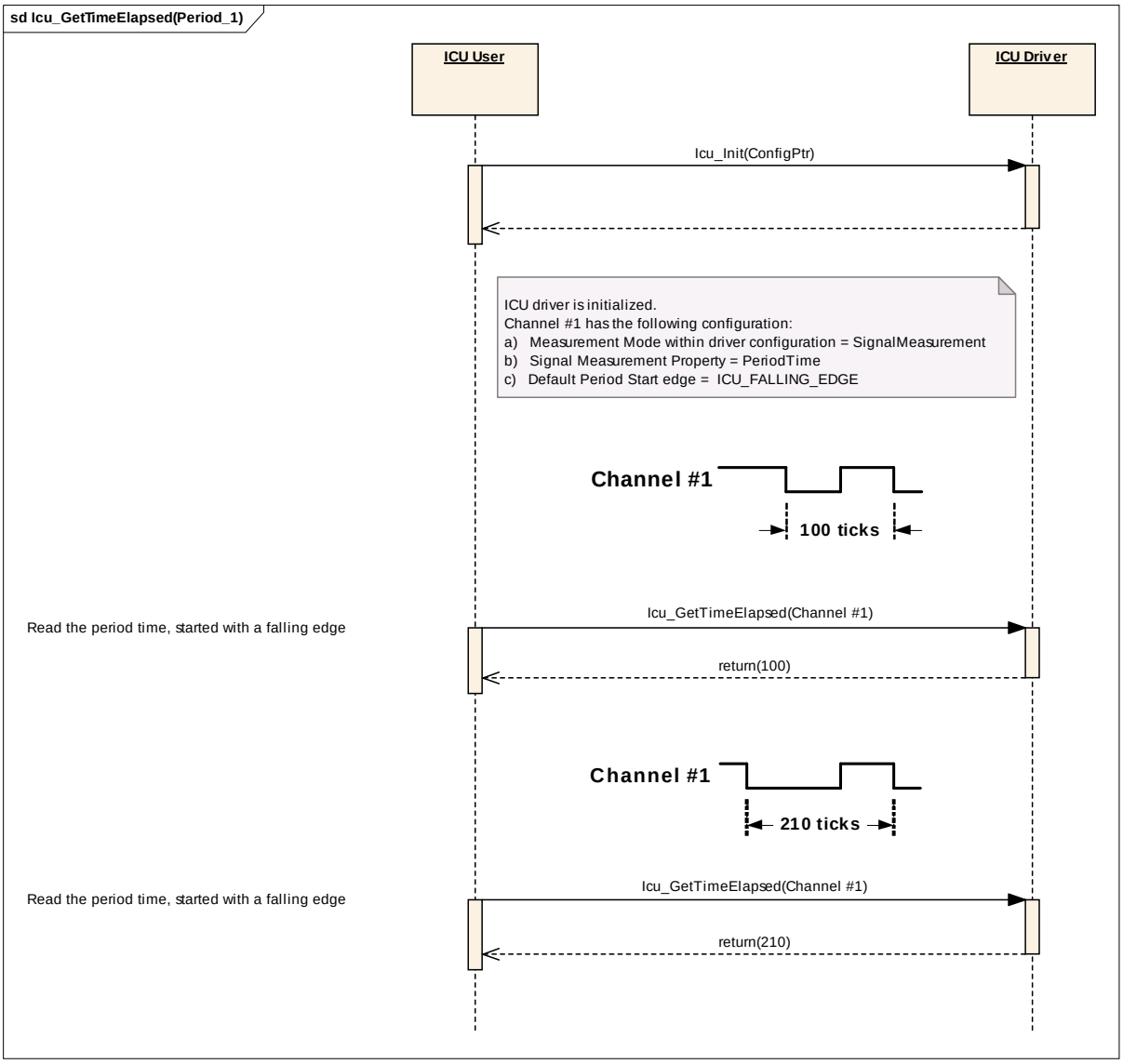


Figure 9.20: Inquire the elapsed period time of a channel

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() !

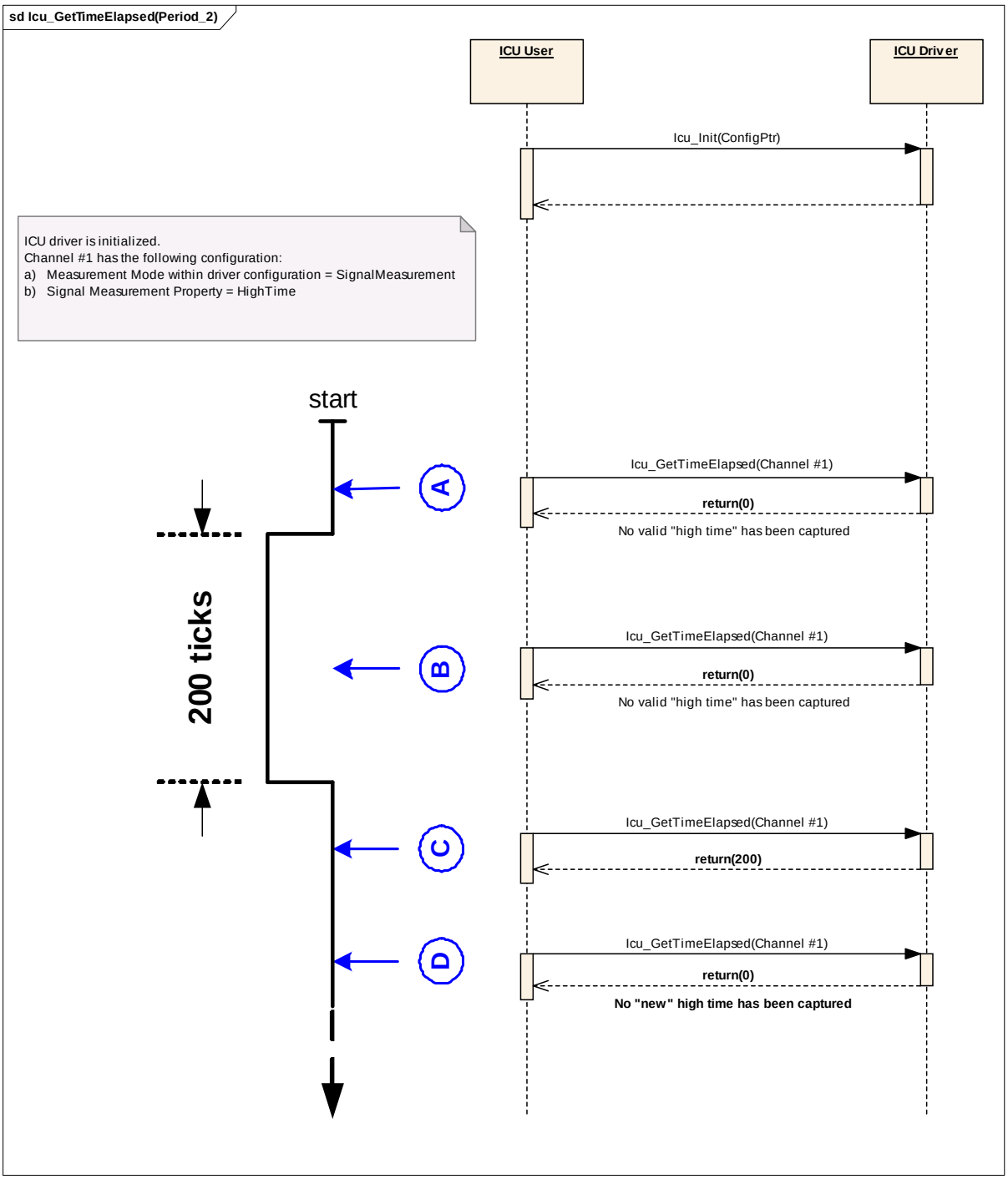


Figure 9.21: Inquire the elapsed high time of a channel

9.14 Icu_GetDutyCycleValues

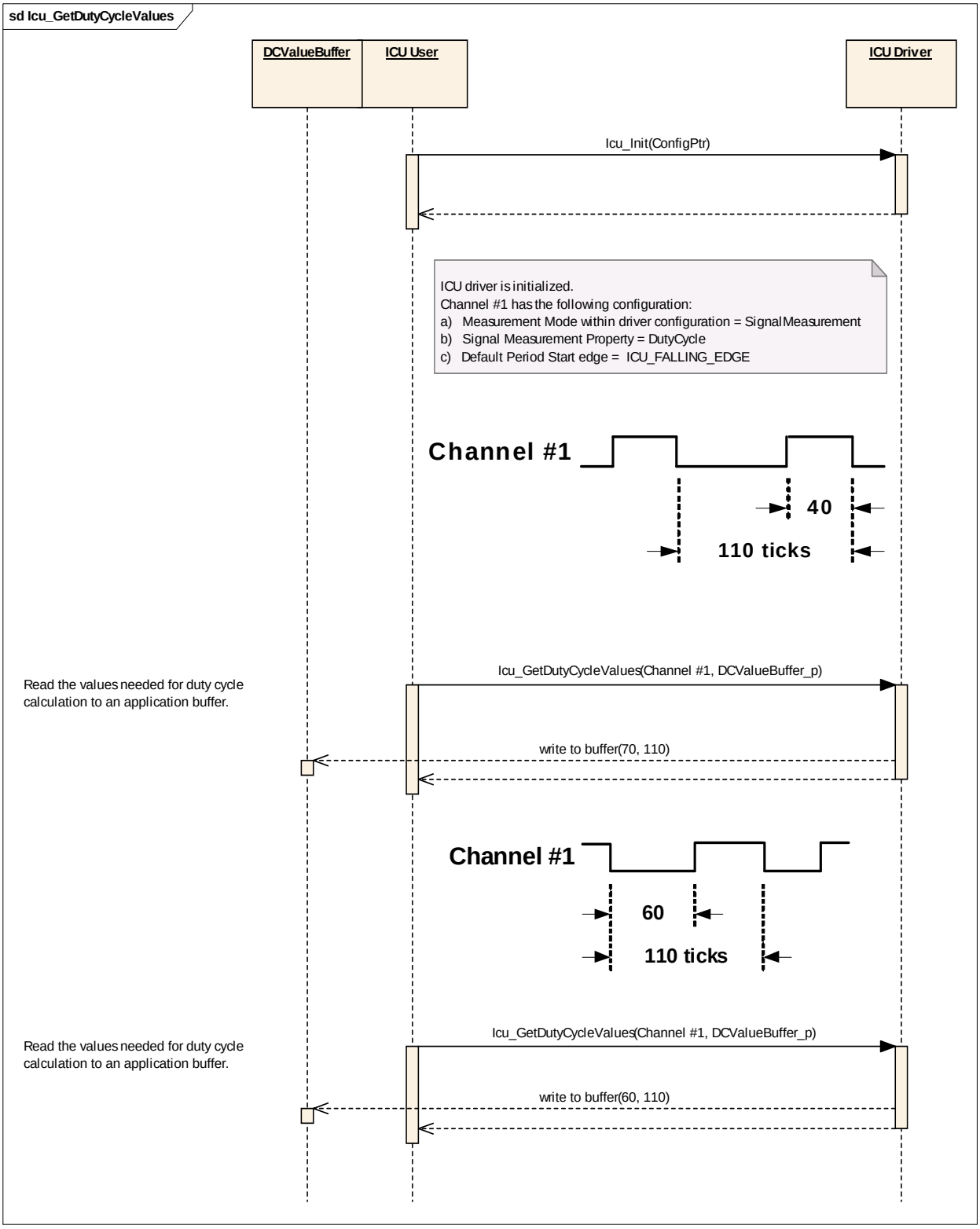


Figure 9.22: Measure the values needed for calculation of duty cycles

9.15 Icu_SignalNotification and Icu_GetInputState

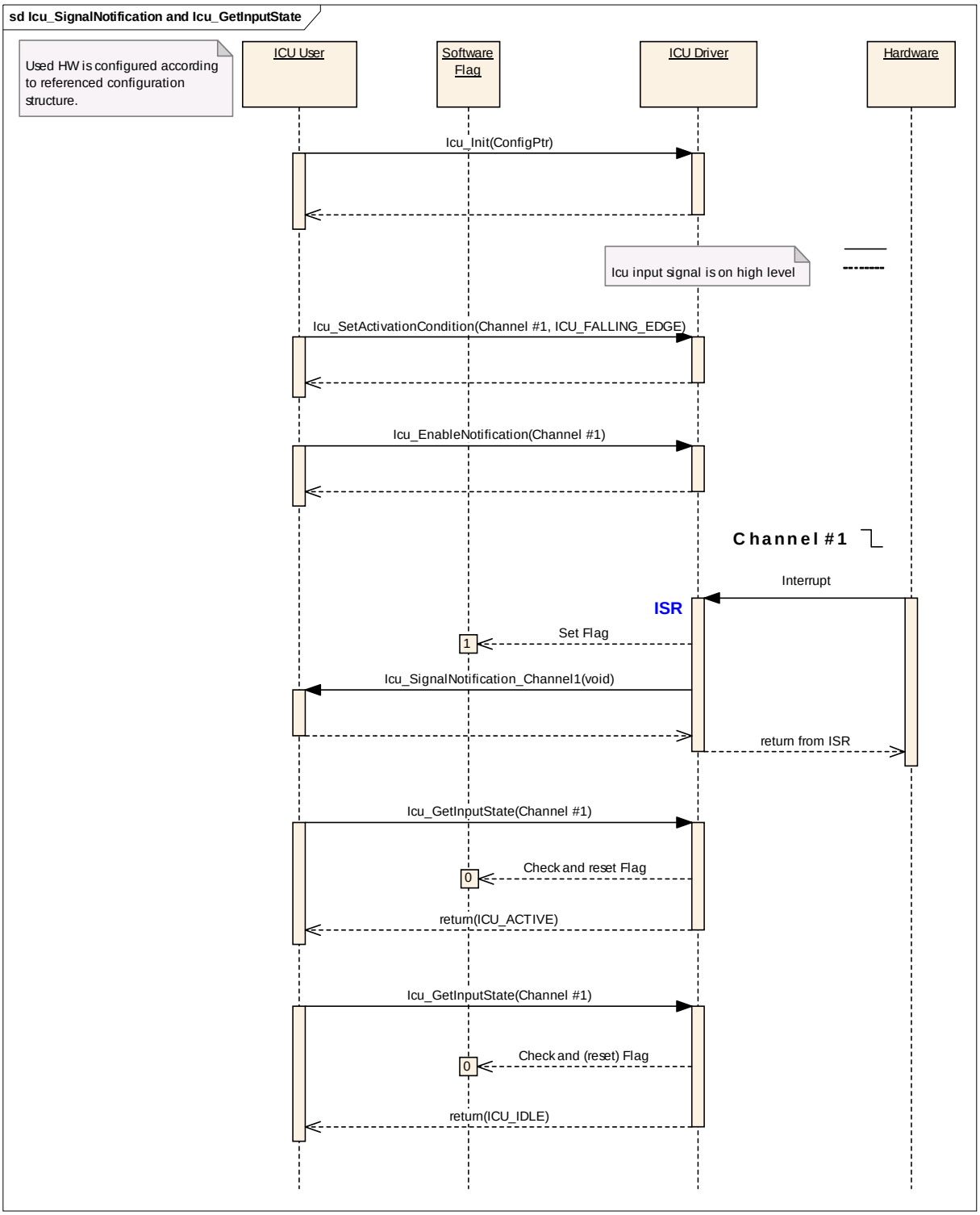


Figure 9.23: Cooperative usage of notification and polling mechanism

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

In addition to this section, it is highly recommended to read the documents:

- AUTOSAR Layered Software Architecture [9]
- AUTOSAR ECU Configuration Specification [8]. This document describes the AUTOSAR configuration methodology and the AUTOSAR configuration metamodel in detail.

The following is only a short survey of the topic and it will not replace the ECU Configuration Specification document.

10.1.1 Configuration and configuration parameters

Configuration parameters define the variability of the generic part(s) of an implementation of a module. This means that only generic or configurable module implementation can be adapted to the environment (software/hardware) in use during system and/or ECU configuration.

The configuration of parameters can be achieved at different times during the software process: before compile time, before link time or after build time. In the following, the term "*configuration class*" (of a parameter) shall be used in order to refer to a *specific configuration point in time*.

10.1.2 Variants

Variants describe sets of configuration parameters. E.g., variant 1: only pre-compile time configuration parameters; variant 2: mix of pre-compile- and post build time-configuration parameters. In one variant a parameter can only be of one configuration class.

Thus describe the possible configuration variants of this module. Each Variant must have a unique name which could be referenced to in later chapters. The maximum number of allowed variants is 3.

10.1.3 Containers

Containers structure the set of configuration parameters. This means:

- all configuration parameters are kept in containers.
- (sub-) containers can reference (sub-) containers. It is possible to assign a multiplicity to these references. The multiplicity then defines the possible number of instances of the contained parameters.

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 Variants

Variant PC (**P**re **C**ompile): Only pre-compile time parameters

Variant PB (**P**ost **B**uild): This variant allows a mix of pre-compile time- and post build time-configuration parameters (multiple-selectable configurable configuration parameter sets).

ICU138: 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. This means that, in contradiction to BSW00414 only one interface for initialization shall be implemented and it shall not depend on the modules configuration, which interface the calling software module shall use.

10.2.2 Driver Configuration

SWS Item	**ICU026**
Container Name	General Configuration
Description	This container contains the configuration (parameters) of the ICU driver.
Configuration Parameters	

Name	`ICU_MAX_CHANNEL`		
Description	This parameter contains the number of Channels configured. It will be gathered by tools during the configuration stage.		
Type	`Icu_ChannelType`		
Unit	`--`		
	`--`		
Configuration Class	***Pre-compile***	x	Variant PC
	Link time	--	--
	Post Build	M	Variant PB
Scope	Module		
Dependency	None		

Name	`ICU_DEV_ERROR_DETECT`		
Description	Switches the Development Error Detection and Notification `ON` or `OFF` (see ICU111).		
Type	`#define`		
Unit	--		
Range	`ON`	Enabled	
	`OFF`	Disabled	
Configuration Class	***Pre-compile***	x	all Variants
	Link time	--	--
	Post Build	--	--

Scope	Module
Dependency	None

Name	ICU_REPORT_WAKEUP_SOURCE		
Description	Pre-processor switch for enabling Wakeup source reporting (see ICU055)		
Type	#define		
Unit	--		
Range	ON	Report Wakeup source	
	OFF	Don't report Wakeup source	
Configuration Class	***Pre-compile***	x	all Variants
	Link time	--	--
	Post Build	--	--
Scope	Module		
Dependency	None		

Name	Hardware Specific Settings per module		
Description	Additional, hardware specific settings (e.g. glitch filters, prescalers) which are valid for the whole driver can be configured here.		
Type	Implementation specific		
Unit	--		
Range	--		
Configuration Class	***Pre-compile***	x	Variant PC
	Link time	--	--
	Post Build	M	Variant PB
Scope	Module		
Dependency	None		

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
Configuration of optional API services (see chapter 10.2.8)	1	The module contains configuration switches for configuring optional API services of the ICU driver.
Channel Configuration (see Chapter 10.2.3)	1 … ICU_MAX_CHANNEL	The module contains 1… channels

10.2.3 Channel Configuration

SWS Item	**ICU027**:
Container Name	Channel configuration
Description	This container contains the configuration (parameters) of an ICU channel.
Configuration Parameters	

Name	Icu_WakeupCapability		
Description	Information about the wakeup-capability of this channel.		
Type	boolean		
Unit	--		
Range	TRUE	Channel **is** wakeup capable	
	FALSE	Channel **is not** wakeup capable	
Configuration Class	***Pre-compile***	x	Variant PC
	Link time	--	--
	Post Build	M	Variant PB

Scope	Module
Dependency	None

Name	Icu_DefaultStartEdge		
Description	Configures the default-activation-edge which shall be used for this channel if there was no activation-edge configured by the call of service `Icu_SetActivationCondition()`. In case the Measurement Mode is `"ICU_MODE_SIGNAL_MEASUREMENT"` and the properties `"DutyCycle"` or `"Period"` are set, the edge configured here is used as Default **Period Start** Edge.		
Type	`Icu_ActivationType`		
Unit	--		
Range	`ICU_RISING_EDGE`	As default, rising edge is the used.	
	`ICU_FALLING_EDGE`	As default, falling edge is the used.	
	`ICU_BOTH_EDGES`	As default, both edges are used.	
Configuration Class	***Pre-compile***	x	Variant PC
	Link time	--	--
	Post Build	M	Variant PB
Scope	Module		
Dependency	None		

Name	Hardware Specific Settings per channel		
Description	Additional, hardware specific channel settings (e.g. glitch filters, prescalers) can be configured here.		
Type	`Implementation specific`		
Unit	--		
Range	--		
Configuration Class	***Pre-compile***	x	Variant PC
	Link time	--	--
	Post Build	M	Variant PB
Scope	Module		
Dependency	None		

Name	Icu_MeasurementMode	
Description	Configures the measurement mode of this channel.	
Type	`Icu_MeasurementModeType`	
Unit	--	

Range	`ICU_MODE_SIGNAL_EDGE_DETECT`	The channel is used for detecting the edges which are configured by the call of the service `Icu_SetActivationCondition()`.
		
		The following API services support this mode:
		`- Icu_EnableNotification()`
		`- Icu_DisableNotification()`
		`- Icu_GetInputState()`

	ICU_MODE_SIGNAL_MEASUREMENT	The channel is used to measure different times between various configurable edges. The configuration of the period-start edges are done by configuration and **cannot** be changed during runtime. The following API services support this mode: - Icu_GetTimeElapsed() - Icu_GetDutyCycleValues() - Icu_GetInputState() This mode can only be configured if at least one of the following switches are set to "on": - ICU_GET_DUTY_CYCLE_VALUES_API - ICU_GET_TIME_ELAPSED_API	
	ICU_MODE_TIMESTAMP	The channel is used to capture timer values on the edges which are configured by the call of the service Icu_SetActivationCondition(). The following API services support this mode: - Icu_StartTimestamp() - Icu_StopTimestamp() - Icu_GetTimestampIndex() - Icu_EnableNotification() - Icu_DisableNotification() This mode can only be configured if ICU_TIMESTAMP_API = ON.	
	ICU_MODE_EDGE_COUNTER	The channel is used to count the edges which are configured by the call of the service Icu_SetActivationCondition(). The following API services support this mode: - Icu_EnableEdgeCount() - Icu_DisableEdgeCount() - Icu_GetEdgeNumbers() - Icu_ResetEdgeCount() This mode can only be configured if ICU_EDGE_COUNT_API= ON.	
Configuration Class	***Pre-compile***	x	Variant PC
	Link time	--	--
	Post Build	M	Variant PB
Scope	Module		
Dependency	The possible measurement modes are depending on the pre-processor switches, which enable/disable optional API services (see 10.2.8).		

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
Configuration of Signal Edge Detection Mode (see chapter 10.2.5)	0 or 1	Icu_MeasurementMode
Configuration of Signal Measurement Mode (see chapter 10.2.6)	0 or 1	Icu_MeasurementMode
Configuration of Timestamp Measurement Mode (see chapter 10.2.7)	0 or 1	Icu_MeasurementMode
Wakeup Configuration (see chapter 10.2.4)	0 or 1	Icu_WakeupCapability and ICU_REPORT_WAKEUP_SOURCE

10.2.4 Wakeup Configuration

SWS Item	**ICU126**:
Container Name	Wakeup Configuration
Description	This container contains the configuration (parameters) needed to configure a wakeup capable channel
Configuration Parameters	

Name	Icu_ChannelWakeupInfo		
Description	If the wakeup-capability is true this value is transmitted to the ECU State Manager (EcuM)		
Type	EcuM_WakeupSourceType		
Unit	--		
Range	--		
Configuration Class	***Pre-compile***	x	Variant PC
	Link time	--	--
	Post Build	M	Variant PB
Scope	ECU		
Dependency	Icu_WakeupCapability and ICU_REPORT_WAKEUP_SOURCE		

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
None	--	--

10.2.5 Configuration of Signal Edge Detection

SWS Item	ICU021
Container Name	Configuration of Signal Edge Detection
Description	This container contains the configuration (parameters) in case the measurement mode is "ICU_MODE_SIGNAL_EDGE_DETECT"
Configuration Parameters	

Name	Icu_SignalNotification_<Channel>
Description	User configurable Notification function for signal notification

	(see ICU021).		
Type	See syntax of Icu_SignalNotification_<Channel> (page 47)		
Unit	Function pointer		
Range	--		
Configuration Class	***Pre-compile***	x	Variant PC
	Link time	--	--
	Post Build	M	Variant PB
Scope	Module		
Dependency	Icu_MeasurementMode		

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
None	--	--

10.2.6 Configuration of Signal Measurement

SWS Item	ICU039
Container Name	Configuration of Signal Measurement
Description	This container contains the configuration (parameters) in case the measurement mode is "ICU_MODE_SIGNAL_MEASUREMENT"
Configuration Parameters	

Name	Icu_SignalMeasurementProperty		
Description	Configures the property that could be measured in case the mode is "ICU_MODE_SIGNAL_MEASUREMENT". This property can not be changed during runtime.		
Type	Icu_SignalMeasurementPropertyType		
Unit	--		
Range	ICU_LOW_TIME	The channel is configured for reading the elapsed Signal Low Time (see ICU081)	
	ICU_HIGH_TIME	The channel is configured for reading the elapsed Signal High Time (see ICU082)	
	ICU_PERIOD_TIME	The channel is configured for reading the elapsed Signal Period Time (see ICU083)	
	ICU_DUTY_CYCLE	The channel is configured to read values which are needed for calculating the duty cycle (coherent Active and Period Time).	
Configuration Class	***Pre-compile***	x	Variant PC
	Link time	--	--
	Post Build	M	Variant PB
Scope	Module		
Dependency	▪ Icu_MeasurementMode ▪ switch ICU_GET_DUTY_CYCLE_VALUES_API ▪ switch ICU_GET_TIME_ELAPSED_API		

Included Containers		
Container Name	**Multiplicity**	**Scope / Dependency**
None	--	--

10.2.7 Configuration of Timestamp Measurement

SWS Item	ICU039
Container Name	Configuration of Timestamp Measurement
Description	This container contains the configuration (parameters) in case the measurement mode is "ICU_MODE_TIMESTAMP"
Configuration Parameters	

Name	Icu_TimestampMeasurementProperty		
Description	Configures the handling of the buffer in case the mode is "ICU_MODE_TIMESTAMP"		
Type	`Icu_TimestampBufferType`		
Unit	--		
Range	`ICU_LINEAR_BUFFER`	The buffer will just be filled once (see ICU065).	
	`ICU_CIRCULAR_BUFFER`	After reaching the end of the buffer, the driver restarts at the beginning of the buffer (see ICU064).	
Configuration Class	***Pre-compile***	x	Variant PC
	Link time	--	--
	Post Build	M	Variant PB
Scope	Module		
Dependency	Icu_MeasurementMode		

Name	Icu_TimestampNotification_<Channel>		
Description	User configurable Notification function if the number of requested timestamps (Notification interval > 0) are acquired.		
Type	See syntax of Icu_TimestampNotification_<Channel> (page 48)		
Unit	Function pointer		
Range	- -		
Configuration Class	***Pre-compile***	x	Variant PC
	Link time	--	--
	Post Build	M	Variant PB
Scope	Module		
Dependency	switch `ICU_TIMESTAMP_API`		

Included Containers		
Container Name	**Multiplicity**	**Scope / Dependency**
None	--	--

10.2.8 Configuration of optional API services

SWS Item	**ICU114:**
Container Name	Configuration of optional API services
Description	This container contains all configuration switches for configuring optional API services of the ICU driver.
Configuration Parameters	

Name	ICU_DE_INIT_API		
Description	Adds / removes the service Icu_DeInit() from the code.		
Type	#define		
Unit	--		
Range	ON	Icu_DeInit () can be used	
	OFF	Icu_DeInit () can not be used	
Configuration Class	***Pre-compile***	x	all Variants
	Link time	--	--
	Post Build	--	--
Scope	Module		
Dependency	None		

Name	ICU_SET_MODE_API		
Description	Adds / removes the service Icu_SetMode() from the code.		
Type	#define		
Unit	--		
Range	ON	Icu_SetMode can be used	
	OFF	Icu_SetMode can not be used	
Configuration Class	***Pre-compile***	x	all Variants
	Link time	--	--
	Post Build	--	--
Scope	Module		
Dependency	None		

Name	ICU_DISABLE_WAKEUP_API		
Description	Adds / removes the service Icu_DisableWakeup() from the code.		
Type	#define		
Unit	--		
Range	ON	Icu_DisableWakeup() can be used	
	OFF	Icu_DisableWakeup() can not be used	
Configuration Class	***Pre-compile***	x	all Variants
	Link time	--	--
	Post Build	--	--
Scope	Module		
Dependency	None		

Name	ICU_ENABLE_WAKEUP_API		
Description	Adds / removes the service Icu_EnableWakeup() from the code.		
Type	#define		
Unit	--		
Range	ON	Icu_EnableWakeup() can be used	
	OFF	Icu_EnableWakeup() can not be used	
Configuration Class	***Pre-compile***	x	all Variants
	Link time	--	--
	Post Build	--	--
Scope	Module		
Dependency	None		

Name	ICU_GET_INPUT_STATE_API		
Description	Adds / removes the service Icu_GetInputState() from the code.		
Type	#define		
Unit	--		
Range	ON	Icu_GetInputState() can be used	
	OFF	Icu_GetInputState() can not be used	
Configuration Class	***Pre-compile***	x	all Variants
	Link time	--	--
	Post Build	--	--
Scope	Module		
Dependency	None		

Name	ICU_SIGNAL_MEASUREMENT_API		
Description	Adds / removes the services Icu_StartSignalMeasurement() and Icu_StopSignalMeasurement() from the code.		
Type	#define		
Unit	--		
Range	ON	Icu_StartSignalMeasurement() and Icu_StopSignalMeasurement() can be used	
	OFF	Icu_StartSignalMeasurement() and Icu_StopSignalMeasurement() can not be used	
Configuration Class	***Pre-compile***	x	all Variants
	Link time	--	--
	Post Build	--	--
Scope	Module		
Dependency	None		

Name	ICU_GET_TIME_ELAPSED_API		
Description	Adds / removes the service Icu_GetTimeElapsed() from the code.		
Type	#define		
Unit	--		
Range	ON	Icu_GetTimeElapsed() can be used	
	OFF	Icu_GetTimeElapsed() can not be used	
Configuration Class	***Pre-compile***	x	all Variants
	Link time	--	--
	Post Build	--	--
Scope	Module		
Dependency	If ICU_SIGNAL_MEASUREMENT_API == OFF this switch is shall also be set to OFF		

Name	ICU_GET_DUTY_CYCLE_VALUES_API		
Description	Adds / removes the service Icu_GetDutyCycleValues() from the code.		
Type	#define		
Unit	--		
Range	ON	Icu_GetDutyCycleValues() can be used	
	OFF	Icu_GetDutyCycleValues() can not be used	
Configuration Class	***Pre-compile***	x	all Variants
	Link time	--	--
	Post Build	--	--
Scope	module		
Dependency	If ICU_SIGNAL_MEASUREMENT_API == OFF this switch is shall also be set to OFF		

Name	ICU_GET_VERSION_INFO_API		
Description	Adds / removes the service Icu_GetVersionInfo() from the code.		
Type	#define		
Unit	--		
Range	ON	Icu_GetVersionInfo() can be used	
	OFF	Icu_GetVersionInfo() can not be used	
Configuration Class	***Pre-compile***	x	all Variants
	Link time	--	--
	Post Build	--	--
Scope	Module		
Dependency	None		

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
Configuration of Timestamping API functionality (see chapter 10.2.8)	None	--
Configuration of Edge-counting API functionality (see chapter 10.2.9)I	None	--

10.2.9 Configuration of Timestamping API functionality

SWS Item	**ICU123**:
Container Name	Configuration of timestamping API functionality.
Description	This container contains all configuration switches for configuring the API of the timestamping functionality.
Configuration Parameters	

Name	ICU_TIMESTAMP_API		
Description	Adds / removes all services related to the timestamping functionality as listed below from the code: ▪ Icu_StartTimestamp() ▪ Icu_StopTimestamp() ▪ Icu_GetTimestampIndex()		
Type	#define		
Unit	--		
Range	ON	The services listed above can be used	
	OFF	The services listed above can not be used	
Configuration Class	***Pre-compile***	x	all Variants
	Link time	--	--
	Post Build	--	--
Scope	Module		
Dependency	None		

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
None	--	--

10.2.10 Configuration of Edge-counting API functionality

SWS Item	**ICU124**:
Container Name	Configuration of edge counting API functionality.
Description	This container contains all configuration switches for configuring the API of the edge counting functionality.
Configuration Parameters	

Name	`ICU_EDGE_COUNT_API`		
Description	Adds / removes all services related to the edge counting functionality as listed below, from the code: ▪ Icu_ResetEdgeCount() ▪ Icu_EnableEdgeCount() ▪ Icu_DisableEdgeCount() ▪ Icu_GetEdgeNumbers()		
Type	`#define`		
Unit	--		
Range	`ON`	The services listed above can be used	
	`OFF`	The services listed above can not be used	
Configuration Class	***Pre-compile***	x	all Variants
	Link time	--	--
	Post Build	--	--
Scope	Module		
Dependency	None		

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
None	--	--

10.2.11 Configuration of Edge-detection functionality

SWS Item	ICU147:
Container Name	Configuration of edge detection functionality.
Description	This container contains all configuration switches for configuring the edge detection functionality.
Configuration Parameters	

Name	`ICU_EDGE_DETECTION_API`		
Description	▪ Adds / removes all services related to the edge detection functionality from the code		
Type	`#define`		
Unit	--		
Range	`ON`	The services listed above can be used	
	`OFF`	The services listed above can not be used	
Configuration Class	***Pre-compile***	x	all Variants
	Link time	--	--
	Post Build	--	--

Scope	Module
Dependency	None

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
None	--	--

10.3 Published Information

Published information contains data defined by the implementer of the SW module that does not change when the module is adapted (i.e. configured) to the actual HW/SW environment. It thus contains version and manufacturer information.

SWS Item	**ICU028**:	
Information elements		
Information element name	***Type / Range***	***Information element description***
`ICU_VENDOR_ID`	`#define`	Vendor ID of the dedicated implementation of this module according to the AUTOSAR vendor list
`ICU_MODULE_ID`	`#define`	Module ID of this module from Module List
`ICU_AR_MAJOR_VERSION`	`#define`	Major version number of AUTOSAR specification on which the appropriate implementation is based on.
`ICU_AR_MINOR_VERSION`	`#define`	Minor version number of AUTOSAR specification on which the appropriate implementation is based on.
`ICU_AR_PATCH_VERSION`	`#define`	Patch level version number of AUTOSAR specification on which the appropriate implementation is based on.
`ICU_SW_MAJOR_VERSION`	`#define`	Major version number of the vendor specific implementation of the module. The numbering is vendor specific.
`ICU_SW_MINOR_VERSION`	`#define`	Minor version number of the vendor specific implementation of the module. The numbering is vendor specific.
`ICU_SW_PATCH_VERSION`	`#define`	Patch level version number of the vendor specific implementation of the module. The numbering is vendor specific.

ICU131:
The ICU driver shall describe which other modules (in which versions) are required. This description shall be done by the implementer.

11 Changes to Release 1

11.1 Deleted SWS Items

SWS Item	*Rationale*
ICU029	Covered by ICU028
ICU047	Chapter 8.3.9: Requirement is in conflict to Requirement ICU049 (See also Bugzilla issues 4092 and 4586)
ICU058	Since the formulation of requirement **BSW12069** was extended, this ICU requirement is invalid and will therefore be removed. Background: The validation of wakeup events shall be done in the EcuM.
ICU041	This requirement was a duplicate of requirement ICU054.
ICU075	The root requirement **BSW12458** has been rejected.
ICU045	This requirement is already covered by ICU020
ICU070	This requirement is already covered by ICU020
ICU076	Deleted due to RfC 12373 (merged Icu_EdgeCounterMeasurementProperty with Icu_DefaultStartEdge)

11.2 Replaced SWS Items

SWS Item of Release 1	*replaced by SWS Item*	*Rationale*
None		

11.3 Changed SWS Items

SWS Item	*Rationale*
ICU005	BSW004 has changed
ICU051	This requirement is described more detailed because of new requirement BSW12461 introduced in [2].
ICU052	This requirement is described more detailed because of new requirement BSW12461 introduced in [2].
ICU053	This requirement is described more detailed because of new requirement BSW12461 introduced in [2].
ICU055	Since the name of EcuM_WakeupsourceType was changed, also the naming of `ICU_REPORT_WAKEUP_REASON` to `ICU_REPORT_WAKEUP_SOURCE` was made.

11.4 Added SWS Items

SWS Item	*Rationale*
ICU063	New service Icu_StartTimestamp() established in Release 2.
ICU064	
ICU065	
ICU066	
ICU067	New service Icu_StopTimestamp() established in Release 2.
ICU068	New notification for timestamping functionality established in Release 2 (see chapter Configurable interfaces).
ICU069	
ICU070	
ICU071	New service Icu_GetTimestampIndex() established in Release 2.

SWS Item	Rationale
ICU072	New service Icu_ResetEdgeCount() established in Release 2.
ICU073	New service Icu_EnableEdgeCount() established in Release 2.
ICU074	
ICU075	
ICU076	Normal edge counting mode established in Release 2.
ICU078	New service Icu_EnableEdgeCount() established in Release 2.
ICU079	New service Icu_DisableEdgeCount() established in Release 2.
ICU080	New service Icu_GetEdgeNumbers() established in Release 2.
ICU081	New service Icu_GetTimeElapsed() established in Release 2.
ICU082	
ICU083	
ICU084	New service Icu_GetDutyCycleValues() established in Release 2.
ICU085	Improvement of Release 1 descriptions.
ICU086	Improvement of Release 1 descriptions.
ICU088	New service Icu_GetDutyCycleValues() established in Release 2.
ICU090	New service Icu_SetActivationCondition() established in Release 2.
ICU091	Clarification due to Bugzilla Bug #4436.
ICU092	Make this service pre compile time configurable On/Off.
ICU093	New service Icu_GetVersionInfo() established in Release 2.
ICU094	Make this service pre compile time configurable On/Off.
ICU095	Make this service pre compile time configurable On/Off.
ICU096	Make this service pre compile time configurable On/Off.
ICU097	Make this service pre compile time configurable On/Off.
ICU098	Make this service pre compile time configurable On/Off.
ICU099	Make this service pre compile time configurable On/Off.
ICU100	Make this service pre compile time configurable On/Off.
ICU101	Make this service pre compile time configurable On/Off.
ICU102	New service Icu_EnableEdgeCount() established in Release 2.
ICU103	New service Icu_DisableEdgeCount() established in Release 2.
ICU104	New service Icu_GetEdgeNumbers() established in Release 2.
ICU105	New service Icu_GetTimeElapsed() established in Release 2.
ICU106	New service Icu_GetDutyCycleValues() established in Release 2.
ICU107	New service Icu_GetTimestampIndex() established in Release 2.
ICU108	New service Icu_StartTimestamp() established in Release 2.
ICU109	New service Icu_StartTimestamp() established in Release 2.
ICU110	New service Icu_GetDutyCycleValues() established in Release 2.
ICU111	Split up Requirement ICU002 into - “Description” of the pre-processor switch (ICU112), - description of the “Detection” of errors (ICU111) - and the description of the “Reporting” of Errors (ICU002).
ICU112	
ICU113	New Requirement of this template.
ICU114	Make this driver more scaleable.
ICU115	New Requirement of this template.
ICU116	New Requirement of this template.
ICU117	New Requirement of this template.
ICU118	New Requirement of this template.
ICU119	Add the missing description for interrupt service routines.
ICU120	New service Icu_StartTimestamp() established in Release 2, introduced also a new checkable parameter.
ICU121	To ascertain the wakeup behavior after Icu_Init()
ICU122	Make this service pre compile time configurable On/Off.
ICU123	Make the timestamping functionality pre compile time configurable On/Off.
ICU124	Make the edge-counting functionality pre compile time configurable On/Off.
ICU125	Check parameter of service Icu_SetMode
ICU126	Configuration Container for wakeup mode configuration

SWS Item	Rationale
ICU127	Clarify the behavior concerning pending interrupts
ICU128	Fully coverage of new introduced requirement BSW12461
ICU129	Fully coverage of new introduced requirement BSW12461
ICU131	New requirement BSW00384 introduced in [1]
ICU132	Fixing a review comment of core partner DC.
ICU133	Introduced to cover BSW12064
ICU134	Clarify description of service Icu_StartTimestamp()
ICU135	Clarify the behaviour of service Icu_GetTimestampIndex()
ICU136	Clarify the behaviour of service Icu_GetTimeElapsed()
ICU137	Clarify the behaviour of service Icu_GetDutyCycleValues()
ICU138	Fulfill the decision of the SPAL team concerning the late introduction of BSW00414 (see SPAL minutes, 42nd meeting, day 2, issue 5)
ICU147	Refer to RfC (Bugzilla 15764)